algorithms

MDPI

Article

A Finite Regime Analysis of Information Set Decoding Algorithms

Marco Baldi [1,*], Alessandro Barenghi [2], Franco Chiaraluce [1], Gerardo Pelosi [2] and Paolo Santini [1]

1 Department of Information Engineering (DII), Università Politecnica delle Marche, 60131 Ancona, Italy; f.chiaraluce@univpm.it (F.C.); p.santini@pm.univpm.it (P.S.)
2 Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, 20133 Milano, Italy; alessandro.barenghi@polimi.it (A.B.); gerardo.pelosi@polimi.it (G.P.)
* Correspondence: m.baldi@univpm.it; Tel.: +39-071-2204894

Received: 20 June 2019; Accepted: 25 September 2019; Published: 1 October 2019

Abstract: Decoding of random linear block codes has been long exploited as a computationally hard problem on which it is possible to build secure asymmetric cryptosystems. In particular, both correcting an error-affected codeword, and deriving the error vector corresponding to a given syndrome were proven to be equally difficult tasks. Since the pioneering work of Eugene Prange in the early 1960s, a significant research effort has been put into finding more efficient methods to solve the random code decoding problem through a family of algorithms known as information set decoding. The obtained improvements effectively reduce the overall complexity, which was shown to decrease asymptotically at each optimization, while remaining substantially exponential in the number of errors to be either found or corrected. In this work, we provide a comprehensive survey of the information set decoding techniques, providing finite regime temporal and spatial complexities for them. We exploit these formulas to assess the effectiveness of the asymptotic speedups obtained by the improved information set decoding techniques when working with code parameters relevant for cryptographic purposes. We also delineate computational complexities taking into account the achievable speedup via quantum computers and similarly assess such speedups in the finite regime. To provide practical grounding to the choice of cryptographically relevant parameters, we employ as our validation suite the ones chosen by cryptosystems admitted to the second round of the ongoing standardization initiative promoted by the US National Institute of Standards and Technology.

Keywords: asymmetric cryptosystems; code-based cryptosystems; information set decoding

1. Introduction

Asymmetric cryptosystems are traditionally built on a mathematical function which is hard to compute unless the knowledge of a special parameter is available. Typically, such a function is known as a mathematical trapdoor, and the parameter acts as the private key of the asymmetric cryptosystem. Decoding a random linear block code was first proven to be equivalent to solve an instance of the three dimensional matching by Elwyn Berlekamp et al. in 1978 [1]. By contrast, efficient decoding algorithms for well structured codes have a long history of being available. Therefore, McEliece himself proposed to disguise an efficiently decodable code as a random code and employ the knowledge of the efficiently decodable representation as the private key of an asymmetric cryptosystem. In this way, a legitimate user of the cryptosystem would be able to employ an efficient decoder for the chosen hidden code, while an attacker would be forced to resort to decoding techniques for a generic linear code.

Since the original proposal, a significant amount of variants of the McEliece cryptosystem were proposed swapping the original decodable code choice (Goppa codes [2]) with other efficiently

decodable codes with the intent of enhancing computational performances of reducing the key size. The first attempt in this direction was the Niederreiter cryptosystem [3] using generalized Reed–Solomon (GRS) codes. While the original proposal by Niederreiter was broken by Sidel'nikov and Shestakov in [4], replacing the hidden code with a Goppa code in Niederreiter's proposal yields a cryptosystem which is currently unbroken. More recently, other families of structured codes have been considered in this framework, such as Quasi Cyclic (QC) codes [5], Low Density Parity Check (LDPC) codes [6], Quasi Dyadic (QD) codes [7], Quasi Cyclic Low Density Parity Check (QC-LDPC) codes [8] and Quasi Cyclic Moderate Density Parity Check (QC-MDPC) codes [9].

The significant push in the development of code-based cryptosystems was also accompanied by a comparably sized research effort in their cryptanalysis. In particular, the best attack technique that does not rely on the underlying hidden code structure, and thus is applicable to all the variants, is known as Information Set Decoding (ISD). In a nutshell, ISD attempts at finding enough error-free locations in a codeword to be able to decode it regardless of the errors which affect the codeword itself. Such a technique was first proposed by Prange [10] as a more efficient alternative to decode a general linear block code, with respect to a straightforward guess on the error affected locations. Since then, a significant amount of improvements to Prange's original technique were proposed [11–16], effectively providing significant polynomial speedups on the exponential-time decoding task. In addition to the former works, where the focus is to propose an operational description of a general decoding technique based on information set decoding, the works by the authors of [17–19] provide a more general view on general decoding techniques, including split syndrome decoding and supercode decoding, and report proven bounds on the complexities of the said approaches. Finally, we report the work of Bassalygo et al. [20] as the first tackling formally the complexity of decoding linear codes. For a more comprehensive survey of hard problems in coding theory, we refer the interested reader to [21–23].

The common praxis in the literature concerning ISD improvements is to evaluate the code parameters for the worst-case-scenario of the ISD, effectively binding together the code rate and number of corrected errors to the code length. Subsequently, the works analyze the asymptotic speedup as a function of the code length alone. While this approach is effective in showing an improvement in the running time of the ISD in principle, the practical relevance of the improvement when considering useful parameter sizes in cryptography may be less significant.

We note that, in addition to being the most efficient strategy to perform general random linear code decoding, ISD techniques can also be employed to recover the structure of the efficiently decodable code from its obfuscated version for the LDPC, QC-LDPC and QC-MDPC code families.

Recently, the National Institute of Standards and Technology (NIST) has started a selection process to standardize asymmetric cryptosystems resistant to attacks with quantum computers. Since decoding a random code is widely believed to require an exponential amount of time in the number of errors, even in presence of quantum computers, code-based cryptosystems are prominent candidate in the NIST selection process [24]. Hence, having accurate and shared expressions in the finite length regime, both in the classical and in the quantum computing setting, for the work factor of attacks targeting such schemes, it is important to define a common basis for their security assessment. A work sharing our intent is [25], where a non-asymptotic analysis of some ISD techniques is performed. However, a comprehensive source of this type is not available in the literature, to the best of our knowledge.

1.1. Contributions

In this work, we provide a survey of the existing ISD algorithms, with explicit finite regime expressions for their spatial and temporal complexities. We also detail which free parameters have to be optimized for each of the ISD algorithms, and provide a software tool implementing the said optimization procedure on a given set of code parameters in [26].

1.2. Paper Organization

This work is organized as follows. Section 2 states the required notation and recollects the code-based cryptography background required to understand ISD algorithms. Section 3 surveys the existing ISD algorithms, providing complexity estimates for both the space they require and their execution time. Section 4 contains a critical discussion of the results obtained in the finite regime in comparison to the ones available via asymptotic estimates, while Section 5 summarizes our conclusions.

2. Background on Computationally Intractable Coding Theory Problems

In this section, we introduce the notation and background on error correcting codes, and state which hard problems we focus on, for which best solvers are ISD algorithms.

In the following, we consider the case of binary linear block codes, denoting as $\mathcal{C}(n,k,d)$ a code of length n, dimension k and minimum distance d. This code is thus a subspace of $\{0,1\}^n$ containing 2^k distinct vectors, and can be represented by a $k \times n$ binary matrix G known as the code generator matrix. It is commonplace to indicate with $r = n - k$ the amount of redundant bits in an element of the code, i.e., a codeword. The minimum distance of a linear block code corresponds to the minimum weight of its codewords, apart from the null one (which, clearly, has null weight). An alternative representation is the one provided by the so-called parity check matrix H, which is obtained through the algebraic constraint $HG^T = 0_{r\times k}$. The parity check is thus an $r \times n$ binary matrix for which it is easy to show that the product of a codeword c by H is a null r-bit-long vector. The quantity obtained multiplying a generic n-bit vector $\tilde{c}$ by H is called the syndrome $s = H\tilde{c}^T$ of $\tilde{c}$ through H. Note that, if the binary vector $\tilde{c}$ is not a codeword, the syndrome of $\tilde{c}$ through H is not null; in other words, we can write $\tilde{c} = c + e$, with c being a codeword, and thus have $s = He^T$.

Decoding of $\tilde{c}$ through $\mathcal{C}(n,k,d)$ consists in finding the codeword c whose distance from $\tilde{c}$ is minimum. The procedure of finding a length-n binary vector e, with Hamming weight smaller than or equal to some integer t, given a non-null syndrome s and a parity matrix H is known as syndrome decoding. The name stems from the fact that decoding a given $\tilde{c}$ becomes possible by computing the syndrome s of $\tilde{c}$, solving the syndrome decoding, and adding the obtained vector e to $\tilde{c}$, and corresponds to finding the codeword which is at minimum distance from $\tilde{c}$.

We can now recall the two decisional problems in coding theory which were proven to be NP-Complete by Berlekamp et al. in [1], and from which the main building blocks of the cryptographic trapdoors of code-based cryptosystems are derived.

Statement 1 (Coset weights problem)**.** *Given a random, $r \times n$ binary matrix H, an r bit vector s and a positive integer t, determine if an n bit vector e with $wt(e) \leq t$, where $wt(\cdot)$ is the Hamming weight function, such that $He^T = s$ exists.*

The Coset weights problem is also known as the Decisional Syndrome Decoding Problem (DSDP).

Statement 2 (Subspace weights problem)**.** *Given a random, $r \times n$ binary matrix H, and a positive integer w, determine if an n bit vector c with $wt(c) = t$, where $wt(\cdot)$ is the Hamming weight function, such that $Hc^T = 0_{r\times 1}$ exists.*

The Subspace weights problem is also known as the Decisional Codeword Finding Problem (DCFP).

The typical hard problems which are employed to build code-based cryptographic trapdoors are the search variants of the aforementioned two problems. Specifically, the Syndrome Decoding Problem (SDP) asks to *find* an error vector of weight $\leq t$, while the Codeword Finding Problem (CFP) asks to *find* a codeword with a given weight w. A well known result in computational complexity theory (e.g., [27] Chap. 2.5) states that any decision problem belonging to the NP-Complete class has a search-to-decision reduction. In other words, it is possible to solve an instance of the search problem

with a polynomial amount of calls to an oracle for the corresponding decision problem. This, in turn, states that the difficulty of Syndrome Decoding Problem (SDP) and Codeword Finding Problem (CFP) is the same of solving their decisional variants Decisional Syndrome Decoding Problem (DSDP) and Decisional Codeword Finding Problem (DCFP), i.e., they are as hard as an NP-Complete problem. We note that, in the light of such a reduction, and despite it is a notation abuse, it is commonplace to state that Syndrome Decoding Problem (SDP) and Codeword Finding Problem (CFP) are NP-Complete, although only decisional problems belong to the NP-complete class.

2.1. Applications to Cryptography

The class of NP-Complete problems is of particular interest to design cryptosystems, as it is widely believed that problems contained in such a class cannot be solved in polynomial time by a quantum computer. Indeed, the best known approaches to solve both the Codeword Finding Problem (CFP) and the Syndrome Decoding Problem (SDP) have a computational complexity which is exponential in the weight of the codeword or error vector to be found. A notable example of code-based cryptosystem relying on the hardness of the Syndrome Decoding Problem (SDP) is the one proposed by Niederreiter in [3].

Niederreiter cryptosystem generates a public–private key-pair selecting as the private key an instance of a code from a family for which efficient decoding algorithms are available. The code is then represented by its parity check matrix H_{priv}, which is multiplied by a rank-r random square binary matrix S obtaining the public key of the cryptosystem $H_{pub} = SH_{priv}$. The assumption made by Niederreiter is that the multiplication by the random, full rank matrix S makes H_{pub} essentially indistinguishable from a random parity matrix. While the original choice to employ a Reed–Solomon code as private code was found to falsify this assumption and lead to a practical attack, other code families have proven to be good candidates (e.g., Goppa codes, Low/Medium Density Parity Check codes [28–31]). A message is encrypted in the Niederreiter cryptosystem encoding it as a fixed weight error vector e and computing its syndrome through H_{pub}, $s = H_{pub}e^T$, which acts as the ciphertext. The owner of the private key (S, H_{priv}) is able to decipher the ciphertext through first obtaining $s' = S^{-1}s$ and subsequently performing the syndrome decoding of s' employing H_{priv}.

It is easy to note that, under the assumption that H_{pub} is indistinguishable from a random parity check matrix, an attacker willing to perform a Message Recovery Attack (MRA) must solve an instance of the Syndrome Decoding Problem (SDP). We note that, as proven by Niederreiter [3], the Syndrome Decoding Problem (SDP) is computationally equivalent to the problem of correcting a bounded amount of errors affecting a codeword, when given a random generator matrix of the code, G. Such a problem goes by the name of Decoding Problem, and is the mainstay of the original cryptosystem proposal by McEliece [32]. In such a scheme, the ciphertext thus corresponds to the sum between a codeword of the public code, obtained as mG, with m being a length-k vector, and a vector e of weight t. The message can either be encoded into e or into m; in the latter case, Message Recovery Attack (MRA) is performed by searching for the error vector e and, subsequently, by adding it to the intercepted ciphertext. We point out that this search can automatically turned into the formulation of Syndrome Decoding Problem (SDP), by first computing a valid H_{pub} from G and then by trying to solve Syndrome Decoding Problem (SDP) on the syndrome of the intercepted ciphertext through H_{pub}.

One of the most prominent cases where Codeword Finding Problem (CFP) appears in code-based cryptosystems is represented by a Key Recovery Attack (KRA) against Niederreiter cryptosystems where the private parity check matrix H_{priv} contains rows with a known low weight w. Indeed, in such a case, considering H_{pub} as the generator matrix of the dual code, solving the Codeword Finding Problem (CFP) for such a code reveals the low weight rows of H_{priv}. We note that such a Key Recovery Attack (KRA) is in the same computational complexity class as Syndrome Decoding Problem (SDP), assuming that the obfuscation of H_{pub} makes it indistinguishable from a random one.

Two notable cases where solving the Codeword Finding Problem (CFP) is the currently best known method to perform a Key Recovery Attack (KRA) are the LEDAcrypt [33] and BIKE [34]

proposals to the mentioned NIST standardization effort for post-quantum cryptosystems. Since such a Codeword Finding Problem (CFP) can also be seen as the problem of finding a binary vector c with weight w such that $H_{pub}c^T = 0$, the problem is also known as the *Homogeneous Syndrome Decoding Problem (SDP)*, as it implies the solution of a simultaneous set of linear equations similar to the Syndrome Decoding Problem (SDP), save for the syndrome being set to zero.

2.2. Strategies to Perform MRA

As described in the previous section, security of code-based cryptosystems relies on the hardness of solving Syndrome Decoding Problem (SDP) or Codeword Finding Problem (CFP) instances. In this section, we analyze the case of Syndrome Decoding Problem (SDP) and show that the optimal strategy to perform Message Recovery Attack (MRA) depends on the code parameters. The optimal strategy for solving Syndrome Decoding Problem (SDP) depends on the relation between the actual parameters of the instance under analysis. In particular, in the cases where t is above the Gilbert–Varshamov (GV) distance [35], Generalized Birthday Algorithm (GBA) is the best currently known algorithm for solving Syndrome Decoding Problem (SDP) [36,37]. However, for the cases we consider in this paper, practical values of t are significantly smaller than the GV distance; in such cases, the best known methods to solve Syndrome Decoding Problem (SDP) go by the name of Information Set Decoding (ISD) algorithms. Such algorithms are aimed at lessening the computational effort required in the guesswork of an exhaustive search for the unknown error vector e of weight t, given a syndrome and a parity check matrix. We point out that it is also possible to adapt all Information Set Decoding (ISD) algorithms, save for the first one proposed by Prange [10], to solve the Codeword Finding Problem (CFP), as a consequence of the structural similarity of the two problems.

All Information Set Decoding (ISD) algorithms share a common structure where an attempt at retrieving the error vector corresponding to a given syndrome is repeated, for a number of times whose average value depends on the success probability of the single attempt itself. The complexity of all Information Set Decoding (ISD) variants can be expressed as the product between the complexity of each attempt, which we denote as C_{iter}, and the average number of required attempts. In particular, such a value can be obtained as the reciprocal of the success probability of each attempt, which we denote as Pr_{succ}; thus, when considering a code with length n, redundancy $r = n - k$ and Hamming weight of the sought error bounded to t, we generically denote the time complexity of obtaining one solution of Syndrome Decoding Problem (SDP) employing the Information Set Decoding (ISD) variant at hand, i.e.,

$$C_{\text{ISD}}(n, r, t) = \frac{C_{\text{iter}}}{\text{Pr}_{\text{succ}}}. \tag{1}$$

As we show in the following, the work factor of a Message Recovery Attack (MRA) performed through Information Set Decoding (ISD) may actually depend on the system parameters; to this end, we first exploit the following well-known result. Let $\mathcal{C}(n, k, d)$ be a linear binary code with length n, dimension k and minimum distance d, and let H be a parity-check matrix for $\mathcal{C}(n, k, d)$. Let s be a length-r binary vector and t be an integer $\leq n$; then, if $t < \left\lceil \frac{d}{2} \right\rceil$, there is at maximum one vector e of weight t such that $s = He^T$.

Thus, when H_{pub} is the parity-check matrix of a code with minimum distance $d > 2t$, then solving Syndrome Decoding Problem (SDP) guarantees that the found error vector corresponds to the one that was used to encrypt the message. In this case, the attack work factor corresponds to $C_{\text{ISD}}(n, r, t)$.

However, when $d \leq 2t$, the time complexity of a Message Recovery Attack (MRA) needs to be derived through a different approach. Indeed, in such a case, the adversary has no guarantee that the output of Information Set Decoding (ISD) corresponds to the error vector that was actually used in the encryption phase. Thus, the work factor of a Message Recovery Attack (MRA) cannot be simply taken as the time complexity of the chosen Information Set Decoding (ISD) algorithm.

Let s be the syndrome corresponding to the intercepted ciphertext and e be the searched error vector, i.e., $s = H_{pub}e^T$. We define

$$N(e) = \left| \left\{ e' \in \mathbb{F}_2^n \text{ s.t. } wt(e') = t \text{ and } H_{pub}e'^T = s \right\} \right|. \tag{2}$$

Clearly, $N(e)$ corresponds to the number of valid outputs that Information Set Decoding (ISD) can produce, when applied on the syndrome s corresponding to e. In such a case, the probability that an Information Set Decoding (ISD) iteration will not find any valid error vector can be estimated as $(1 - \mathrm{Pr}_{\mathrm{succ}})^{N(e)}$. Thus, in such a case, one attempt of Information Set Decoding (ISD) will succeed with probability $\mathrm{Pr}'_{\mathrm{succ}}(e) = 1 - (1 - \mathrm{Pr}_{\mathrm{succ}})^{N(e)}$. In particular, the algorithm will randomly return a vector among the set of the $N(e)$ admissible ones: thus, the probability that the obtained vector corresponds to e is $1/N(e)$.

To obtain a closed-form expression for the attack work factor, we can consider the average value of $N(e)$, which we obtain by averaging over all the possible vectors e of weight t and length n, and denote it with N. Then, the attack work factor can be computed as

$$WF_{\mathrm{MRA}} \approx N \frac{C_{\mathrm{iter}}}{\mathrm{Pr}'_{\mathrm{succ}}} = N \frac{C_{\mathrm{iter}}}{1 - (1 - \mathrm{Pr}_{\mathrm{succ}})^N}. \tag{3}$$

In particular, it can be shown that $NP \geq 1 - (1 - P)^N$, so that

$$WF_{\mathrm{MRA}} = \alpha C_{\mathrm{ISD}}(n, k, t), \tag{4}$$

with

$$\alpha = \frac{N\mathrm{Pr}_{\mathrm{succ}}}{1 - (1 - \mathrm{Pr}_{\mathrm{succ}})^N} \geq 1. \tag{5}$$

We point out that, for the cases we analyze in this paper, we have $N\mathrm{Pr}_{\mathrm{succ}} \ll 1$, so that $\alpha \approx 1$. Thus, from now on, we assume $\alpha = 1$, i.e., that the time complexity of performing a Message Recovery Attack (MRA) is equal to that of running an Information Set Decoding (ISD) algorithm in the case in which a unique solution exists.

3. A Finite Regime Analysis of Information Set Decoding Techniques

In the following, we report an analysis of the best known variants of Information Set Decodings (ISDs) and their execution on a classical computer, namely the ones proposed by Prange [10], Lee and Brickell [11], Leon [12], Stern [13], Finiasz and Sendrier [14], May, Meurer and Thomae [15], and Becker, Joux, May and Meurer [16]. For the sake of clarity, we describe the Information Set Decoding (ISD) algorithms in their syndrome decoding formulation, highlighting for the first variant amenable to dual-use, i.e., Lee and Brickell's, how to adapt the technique to the Codeword Finding Problem (CFP). For all these algorithms, we provide finite-regime time complexities and space complexities, with the aim to analyze the actual computational effort and memory resources needed to solve both the Syndrome Decoding Problem (SDP) and the Codeword Finding Problem (CFP) on instances with cryptographically sized parameters. We also report lower bounds on the complexities of the execution of Prange, Lee and Brickell's and Stern's variants of the Information Set Decoding (ISD) on a quantum computer, allowing an evaluation of the corresponding computational efforts.

We provide the exact formulas for the time complexity of ISD variants as a function of the code length n, the code dimension k and the number of errors t. We note that the ISD algorithms having the best asymptotic time complexity are also characterized by an exponential space complexity, which may significantly hinder their efficiency or make their implementation unpractical. In particular, we also analyze the computational cost of such algorithms with a logarithmic memory access cost criterion. Indeed, the logarithmic access cost criterion is the one which fits better scenarios where the

spatial complexity of an algorithm is more than polynomial in its input size, therefore resulting in a non-negligible cost for the memory accesses.

In the reported formulas, we employ the $\mathcal{O}$-notation simply to remove the need to specify the computing architecture- or implementation-dependant constants.

3.1. Prange

Prange's algorithm [10] is the first known variant of ISD, based on the idea of guessing a set $\mathcal{I}$ of k error-free positions in the error vector e to be found in the Syndrome Decoding Problem (SDP). For this purpose, the columns of H are permuted so that those indexed by $\mathcal{I}$ are packed to the left. This operation is equivalent to the multiplication of H by an appropriately sized permutation matrix P. The column-reordered matrix $\widehat{H} = HP$ is hence obtained, which can be put in Reduced Row Echelon Form (RREF), with the identity matrix I_r placed to the right, i.e., $[V\ I_r] = U\widehat{H}$. If turning $\widehat{H}$ in Reduced Row Echelon Form (RREF) is not possible as the $r \times r$ rightmost submatrix is not full-rank, a different permutation is picked. The same transformation U required to bring H in Reduced Row Echelon Form (RREF) is then applied to the single-bit rows of the column syndrome vector s, obtaining $\bar{s} = Us$. If the weight of the permuted error vector $\widehat{e}$ obtained as $\widehat{e} = eP = [0_{1\times k}\ \bar{s}^T]$, where $0_{1\times k}$ is the all-zero error vector of length k, matches the expected error weight t, then the algorithm succeeds and the non-permuted error vector $\widehat{e}P^T$ is returned. A pseudo-code description of Prange's ISD algorithm is provided in Algorithm 1.

Algorithm 1: Syndrome decoding formulation of Prange's ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered

Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$

Data: P: an $n \times n$ permutation matrix
$\bar{s}$ an r-bit long binary column vector
V: an $r \times k$ binary matrix $V = [v_0, \ldots, v_{k-1}]$

```
repeat
    repeat
        P ← RandomPermutationGen(n)
        Ĥ ← HP                                  // the corresponding error vector is ê = eP
        ⟨ U, [V|W] ⟩ ← RedRowEchelonForm(Ĥ)                    // UH = [V|W_{r×r}]
    until W ≠ I_r
    s̄ ← Us
    ê ← [0_{1×k} s̄^T]
until weight(ê) = t
return êP^T
```

Proposition 1 (Computational complexity of Algorithm 1)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, the complexity for finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 1 can be computed starting from the probability*

of success $\mathrm{Pr}_{\mathrm{succ}}$ *of a single iteration of the loop at Lines* 1–9 *and the computational requirements of executing the loop body* c_{iter}. *In particular, the time complexity is* $C_{\mathrm{ISD}}(n,r,t) = \frac{1}{\mathrm{Pr}_{\mathrm{succ}}} c_{\mathrm{iter}} = \frac{\binom{n}{t}}{\binom{r}{t}} (C_{\mathrm{IS}}(n,r) + n)$, *with*

$$\begin{aligned} C_{\mathrm{IS}}(n,r) &= \frac{1}{\prod_{i=1}^{r}(1-2^{-i})} C_{\mathrm{RREF}}(n,r) + r^2 + n, \\ C_{\mathrm{RREF}}(n,r) &= \mathcal{O}\left(\frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1\right). \end{aligned} \tag{6}$$

The spatial complexity is $S_{\mathrm{ISD}}(n,r,t) = \mathcal{O}(rn)$.

Proof. The loop body of Algorithm 1 is dominated by the cost of finding an information set and validating it through checking that the matrix W is indeed an identity, i.e., that the corresponding submatrix of $\widehat{H}$ indeed has full rank.

Note that, in an $r \times r$ binary matrix, the first row has a probability of $\frac{1}{2^r}$ of being linearly dependent from itself (i.e., zero); the second row has a probability of $\frac{2}{2^r}$ of being linearly dependent (i.e., zero or equal to the first). With an inductive argument, we obtain that the rth row has a probability of $\frac{2^{r-1}}{2^r}$ of being linearly dependent from the previous ones. We thus have that the probability of having all the rows independent from one another is $\prod_{i=1}^{r}(1 - \frac{2^{i-1}}{2^r}) = \prod_{i=1}^{r}(1 - \frac{1}{2^i})$.

We thus have that the column permutation (Line 4), with computational complexity rn (which can be lowered to n keeping only the permuted column positions) and the Reduced Row Echelon Form (RREF) transformation (Line 5), with cost $\mathcal{O}\left(\frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1\right)$ have to be repeated $\frac{1}{\prod_{i=1}^{r}(1-2^{-i})}$ times, yielding the first addend of the computational cost $C_{\mathrm{IS}}(n,r)$. The cost $C_{\mathrm{RREF}}(n,r)$ is derived considering the Reduced Row Echelon Form (RREF) as an iterative algorithm performing as many iterations as the rank of the identity matrix in the result (i.e., r in this case). Each iteration $0 \le i \le r-2$ proceeds to find a pivot, taking $\mathcal{O}(r-i)$, swaps it with the $(r-i)$th row in $\mathcal{O}(n)$ and proceeds to add the pivot to all the remaining $r-i-1$ rows which have a one in the $(n-i)$th column. The total cost is

$$C_{\mathrm{RREF}}(n,r) = \mathcal{O}\left(\sum_{i=0}^{r-2}(r-i) + rn + \sum_{i=0}^{r-2}(r-1-i)(n-i)\right) = \mathcal{O}\left(\frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1\right).$$

The second addend of the cost $C_{\mathrm{IS}}(n,r)$ is constituted by the computational complexity of computing $\bar{s} = Us$, which is r^2. The total cost of computing an iteration c_{iter} is the sum of $C_{\mathrm{IS}}(n,r)$ and the cost of building $\widehat{e}$, i.e., $\mathcal{O}(n)$.

$\mathrm{Pr}_{\mathrm{succ}}$ is obtained as the number of permuted error vectors with the error-affected positions such that they are fitting the hypotheses made by the algorithm, divided by the number of all the possible error vectors. This fact holds for all ISD algorithms. In the case of Prange's ISD, the permuted error vectors admissible by the hypotheses are $\binom{r}{t}$, as all the error-affected positions should be within the last r bits of the permuted error vectors, while the number of error vectors is $\binom{n}{t}$. □

For the sake of clarity, from now on, we denote as $\mathrm{ISEXTRACT}(H,s)$ the procedure computing $\langle P, [V\ I_r], \bar{s}\rangle$, performed on Lines 2–7 of Algorithm 1, with computational time complexity $C_{\mathrm{IS}}(n,r)$ and space complexity $\mathcal{O}(rn)$.

3.2. Lee–Brickell

The ISD algorithm introduced by Lee and Brickell in [11] starts with the same initial operations as in Prange's, i.e., the computation of the Reduced Row Echelon Form (RREF) of $\widehat{H}$ and the derivation of the corresponding syndrome $\bar{s}$. However, Lee and Brickell improved Prange's original idea by allowing p positions in the k selected in the error vector to be error-affected. These p remaining error positions are guessed. To verify the guess, Lee and Brickell exploit the identity $[V\ I_r]\,\widehat{e}^T = \bar{s}$, where $\widehat{e}$

is split in two parts, $\widehat{e} = [\widehat{e}_1 \widehat{e}_2]$, with $\widehat{e}_1$ being k bit long and with weight p, and $\widehat{e}_2$ being r bits long and with weight $t - p$. The identity is rewritten as $V\widehat{e}_1^T + I_r\widehat{e}_2^T = V\widehat{e}_1^T + \widehat{e}_2^T = \bar{s}$, from which follows the fact that $V\widehat{e}_1^T + \bar{s} = \widehat{e}_2^T$ must have weight $t - p$. Indeed, this condition is employed by the algorithm to check if the guess of p positions is correct. The procedure is summarized in Algorithm 2.

Algorithm 2: Syndrome decoding formulation of Lee and Brickell's ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered

Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$

Data: P: an $n \times n$ permutation matrix
$\widehat{e} = eP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{e}$, $0 \leq p \leq t$, $p = 2$ proven optimal
$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $[V\ I_r]$
V: an $r \times k$ binary matrix $V = [v_0, \ldots, v_{k-1}]$

1 **repeat**
2 $\quad \langle P, [V\ I_r], \bar{s} \rangle \leftarrow \textsc{ISExtract}(H, s)$
3 $\quad$ **for** $j \leftarrow 1$ **to** $\binom{k}{p}$ **do**
4 $\quad\quad \mathcal{I} \leftarrow \textsc{IntegerToCombination}(j)$ `// ` $\mathcal{I}$ `is a set of` p `distinct integers in` $\{0, \ldots, k-1\}$
5 $\quad\quad$ **if** $\textsc{weight}(\bar{s} + \sum_{i \in \mathcal{I}} v_i) = t - p$ **then**
6 $\quad\quad\quad \widehat{e} \leftarrow [0_{1\times k}\ (\bar{s} + \sum_{i \in \mathcal{I}} v_i)^T]$
7 $\quad\quad\quad$ **foreach** $i \in \mathcal{I}$ **do**
8 $\quad\quad\quad\quad \widehat{e} \leftarrow \widehat{e} + [0_{1\times i}\ 1\ 0_{1\times(r+k-1-i)}]$
9 $\quad\quad\quad$ **break**
10 **until** $\textsc{weight}(\widehat{e}) = t$
11 **return** $\widehat{e}P^T$

Proposition 2 (Computational complexity of Algorithm 2)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 2 requires an additional parameter $0 \leq p \leq t$.*

The time complexity of Algorithm 2 can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–10 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is

$$C_{\mathrm{ISD}}(n, r, t, p) = \frac{1}{\mathrm{Pr}_{\mathrm{succ}}} c_{\mathrm{iter}} = \frac{\binom{n}{t}}{\binom{k}{p}\binom{r}{t-p}} \left(C_{\mathrm{IS}}(n, r) + \binom{k}{p} (C_{\mathrm{IntToComb}} + pr) \right),$$

where $C_{\mathrm{IS}}(n, r)$ is as in Equation (6) and $C_{\mathrm{IntToComb}} = \mathcal{O}((2k - p)(\log \binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e, finding the corresponding combination among all the $\binom{k}{p}$ ones. The spatial complexity is $S_{\mathrm{ISD}}(r, n) = \mathcal{O}(rn)$.

Proof. The probability of success of Lee and Brickell's ISD is obtained following the same line of reasoning employed for Prange's, thus dividing the number of admissible permuted error vectors, $\binom{k}{p}\binom{r}{t-p}$ by the number of the possible error vectors $\binom{n}{t}$.

The cost of an iteration of Lee and Brickell's algorithm can be obtained as the cost of adding together $p + 1$ bit vectors of length r, i.e., pr (Line 6), multiplied by the number of such additions,

i.e., $\binom{k}{p}$ as they constitute the body of the loop at Lines 4–9. Note that, in a practical implementation where the value of p is fixed, it is possible to avoid $\mathrm{C}_{\mathrm{IntToComb}}$ altogether, specializing the algorithm with a p-deep loop nest to enumerate all the weight p, length k binary vectors. □

3.3. Adapting Lee and Brickell to Solve CFP

The structure of Lee and Brickell's Information Set Decoding (ISD) allows employing substantially the same algorithm to solve the Codeword Finding Problem (CFP), given a parity matrix H as the representation of the code where a weight w codeword c should be found. The line of reasoning to employ Lee and Brickell's Information Set Decoding (ISD) to solve the Codeword Finding Problem (CFP) is to note that, by definition, for any codeword c of the code represented by H we have that $Hc^T = 0_{1\times r}$, i.e., a codeword multiplied by the parity check matrix yields a null syndrome. As a consequence, we have that $0_{1\times r} = Hc^T = HPP^Tc^T = \hat{H}P^Tc^T = \hat{H}\hat{c}^T = V\hat{c}_1^T + \hat{c}_2^T$. This implies that $V\hat{c}_1^T = \hat{c}_2^T$, which can be exploited as an alternative stopping condition to the one of Algorithm 2, yielding in turn Algorithm 3. The only remaining difference between the Syndrome Decoding Problem (SDP) solving Lee and Brickell's ISD and the Codeword Finding Problem (CFP) is represented by the ISEXTRACT primitive, which no longer needs to compute a transformed syndrome $\bar{s} = Us$ as it is null. We thus have a small reduction in $C_{\mathrm{IS}}(n,r)$, which becomes $C_{\mathrm{IS}}(n,r) = \frac{1}{\prod_{i=1}^{r}(1-2^{-i})}\, C_{\mathrm{RREF}}(n,r) + n$, losing an additive r^2 term. We note that such a reduction is expected to have little impact in practice as the dominant portion of the ISEXTRACT function is represented by the Reduced Row Echelon Form (RREF) computation. This in turn implies that solving the Syndrome Decoding Problem (SDP) on a code $\mathcal{C}$ has practically the same complexity of finding a codeword with weight $w = t$ in the same code. Therefore, finding low-weight codewords in the code defined by a Niederreiter cryptosystem public key H_{pub} has an effort comparable to the one of performing syndrome decoding assuming an error with the same weight as the codeword to be found. Two families of codes which may be vulnerable to such an attack unless the parameters are designed taking into account a Codeword Finding Problem (CFP) ISD are the Low Density Parity Check (LDPC) and Moderate Density Parity Check (MDPC) codes. Indeed, such code families can be represented with a parity check matrix with low-weight rows, and such a low-weight representation can be relied upon to perform efficient decoding, leading to an effective cryptosystem break. Indeed, we now show that, if a code $\mathcal{C}(n,k,d)$ can be represented by a low-weight parity matrix H_{priv}, the code will contain low weight codewords. Without loss of generality, consider $k > r$. Moreover, consider H_{priv} as split in three portions $[A_l A_r B]$ of size $r \times (k-r)$, $r \times r$ and $r \times r$, respectively, with B non-singular. We derive the corresponding generator matrix as

$$G = \begin{bmatrix} I_{(k-r)\times(k-r)} & 0_{(k-r)\times r} & (B^{-1}A_l)^T \\ 0_{r\times(k-r)} & I_{r\times r} & (B^{-1}A_r)^T \end{bmatrix}$$

and consider the bottom r rows $[0_{r\times(k-r)} \;\; I_{r\times r} \;\; (B^{-1}A_r)^T]$. Consider the product of such bottom rows by B^T, yielding $[0_{r\times(k-r)} \;\; I_{r\times r} \;\; (B^{-1}A_r)^T]B^T = [0_{r\times(k-r)} \;\; B^T \;\; A_r^T]$ and note that all the rows of this product are valid codewords, as they are the result of a linear combination of rows of the generator matrix G. Moreover, given that the private parity-check matrix H_{priv} has low row and column weight by construction, we have that the aforementioned codewords, i.e., the rows of $[0_{r\times(k-r)} \;\; B^T \;\; A_r^T]$, also have a low weight. This fact may thus allow an attacker to perform a Key Recovery Attack (KRA) retrieving the low-weight codewords and rebuilding H_{priv}.

A different attack strategy for the same code families is to try and find codewords in the dual code with respect to the one represented by the parity check matrix H_{priv}. Such a code, by definition, sees H_{priv} as a valid generator matrix, and thus makes it possible to directly reconstruct H_{priv} solving r instances of Codeword Finding Problem (CFP) to obtain the r instances of H_{priv}. Solving the Codeword Finding Problem (CFP) on the dual code implies that Algorithm 3 is called considering the aforementioned G matrix as a parity check matrix. Thus, if we denote with $C_{ISD}(n,r,w)$ the complexity of solving Codeword Finding Problem (CFP) on the code described by H_{priv}, solving the Codeword

Finding Problem (CFP) on the dual code, will have a complexity of $C_{ISD}(n, k, w')$, where w' is the weight of the codeword of the dual code. Whether this strategy or the one of solving the Codeword Finding Problem (CFP) on the primal code is more advantageous depending on the code rate and the values of w and w'.

Algorithm 3: Codeword finding formulation of Lee and Brickell's ISD.

Input: H: an $r \times n$ binary parity-check matrix
w: the weight of the codeword to be found
Output: c: an n-bit codeword with $\textsc{weight}(c) = w$
Data: P: an $n \times n$ permutation matrix
$\widehat{c} = cP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{c}$, $0 \leq p \leq w$,
V: an $r \times k$ binary matrix $V = [v_0, \ldots, v_{k-1}]$

```
1  repeat
2      ⟨P, [V I_r]⟩ ← ISEXTRACT(H)
3      for j ← 1 to (k choose p) do
4          I ← INTEGERTOCOMBINATION(j) // I is a set of p distinct integers in {0,...,k−1}
5          if WEIGHT(Σ_{i∈I} v_i) = w − p then
6              ĉ ← [0_{1×k} (Σ_{i∈I} v_i)^T]
7              foreach i ∈ I do
8                  ĉ ← ĉ + [0_{1×i} 1 0_{1×(r+k−1−i)}]
9              break
10 until WEIGHT(ĉ) = w
11 return ĉP^T
```

3.4. Leon

The algorithm proposed by Leon in [12], reported in Algorithm 4, improves the Lee and Brickell's Information Set Decoding (ISD) assuming that the contribution to the value of the first ℓ bits of the syndrome $\bar{s}$, $\bar{s}_{\text{up}}$, comes only from columns in V, i.e., there is a run of zeroes of length ℓ leading the final r bits of the permuted error vector $\widehat{e}$, i.e., $\widehat{e} = [\widehat{e}_1 0_{1\times\ell} \widehat{e}_2]$, where $\widehat{e}_1$ is k bits long and $\widehat{e}_2$ is $r - \ell$ bits long. We thus have that the expected situation after the permutation and RREF computation is

$$\hat{H}\hat{e}^T = \begin{bmatrix} V_{\text{up}} & I_{\text{up}} \\ V_{\text{down}} & I_{\text{down}} \end{bmatrix} \begin{bmatrix} \widehat{e}_1^T \\ 0_{1\times\ell}^T \\ \widehat{e}_{\text{down}}^T \end{bmatrix} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix} = \bar{s}$$

where $\widehat{e}_{\text{down}}$ is assumed to have a run of ℓ zeroes in its first bits. Such an assumption will clearly reduce the success rate of an iteration, as not all the randomly chosen permutations will select columns having this property. However, making such an assumption allows performing a preliminary check of the value of the sum of the ℓ topmost bits only of each column of V. Indeed, such a sum should match the value of the corresponding ℓ topmost bits of $\bar{s}$, $\bar{s}_{\text{up}}$, because the ℓ leading null bits in $\widehat{e}_{\text{down}}$ in turn nullify the contribution of the columns in the topmost ℓ rows I_{up} of the identity matrix. Such a check (Line 5 in Algorithm 4) allows discarding a selection of the p columns from the ones of V, earlier, saving addition instructions with respect to a full column check. The length ℓ of the run of zeroes should be picked so that the trade-off between the reduction in success probability is compensated by the gain in the speed of a single iteration.

Algorithm 4: Syndrome decoding formulation of Leon's ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered

Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$

Data: P: an $n \times n$ permutation matrix
$\widehat{e} = eP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{e}$, $0 \leq p \leq t$,
ℓ: length of the run of zeroes in $\widehat{e} = [\widehat{e'}_{1\times k}\ 0_{1\times \ell}\ \widehat{e''}_{1\times r-\ell}]$
$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $[V\ I_r]$,
$\bar{s} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix}$
V: an $r \times k$ binary matrix
$V = \begin{bmatrix} v_0 & \dots & v_{k-1} \end{bmatrix} = \begin{bmatrix} V_{\text{up}} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up } 0} & \dots & v_{\text{up } k-1} \\ v_{\text{down } 0} & \dots & v_{\text{down } k-1} \end{bmatrix}$

1 **repeat**
2 $\quad \langle P, [V\ I_r], \bar{s} \rangle \leftarrow \textsc{ISExtract}(H, s)$
3 $\quad$ **for** $j \leftarrow 1$ **to** $\binom{k}{p}$ **do**
4 $\quad\quad \mathcal{I} \leftarrow \textsc{IntegerToCombination}(j)$ `//` $\mathcal{I}$ `is a set of` p `distinct integers in` $\{0, \dots, k-1\}$
5 $\quad\quad$ **if** $\textsc{weight}(\bar{s}_{\text{up}} + \sum_{i \in \mathcal{I}} v_{\text{up } i}) = 0$ **then**
6 $\quad\quad\quad$ **if** $\textsc{weight}(\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I}} v_{\text{down } i}) = t - p$ **then**
7 $\quad\quad\quad\quad \widehat{e} \leftarrow [0_{1\times(k+\ell)}\ (\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I}} v_{\text{down } i})^T]$
8 $\quad\quad\quad\quad$ **foreach** $i \in \mathcal{I}$ **do**
9 $\quad\quad\quad\quad\quad \widehat{e} \leftarrow \widehat{e} + [0_{1\times i}\ 1\ 0_{1\times(r+k-1-i)}]$
10 $\quad\quad\quad\quad$ **break**
11 **until** $\textsc{weight}(\widehat{e}) = t$
12 **return** $\widehat{e}P^T$

Proposition 3 (Computational complexity of Algorithm 4)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 4 requires two additional parameters $0 \leq p \leq t$, $0 \leq \ell \leq r$. The time complexity of Algorithm 4 can be computed starting from the probability of success Pr_{succ} of a single iteration of the loop at Lines 1–11 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is*

$$\text{C}_{\text{ISD}}(n, r, t, p, \ell) = \frac{1}{\text{Pr}_{\text{succ}}} c_{\text{iter}} = \frac{\binom{n}{t}}{\binom{k}{p}\binom{r-\ell}{t-p}} \left(\text{C}_{\text{IS}}(n, r) + \binom{k}{p} \left(\text{C}_{\text{IntToComb}} + p\ell + \frac{\binom{k}{p}}{2^\ell} p(r-\ell) \right) \right),$$

where $\text{C}_{\text{IS}}(n, r)$ is as in Equation (6) and $\text{C}_{\text{IntToComb}} = \mathcal{O}((2k-p)(\log \binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the $\binom{k}{p}$ ones. Note that, if the value of p is fixed, it is possible to avoid $\text{C}_{\text{IntToComb}}$, specializing the algorithm with a p-deep loop nest to generate the combinations. The spatial complexity is $\text{S}_{\text{ISD}}(r, n) = \mathcal{O}(rn)$.

Proof. The success probability of an iteration of Leon's algorithm follows the same line of reasoning of Prange's and Lee and Brickell's, dividing the number of admissible permuted error vectors $\binom{k}{p}\binom{r-\ell}{t-p}$ by the total one $\binom{n}{t}$. The complexity of a single iteration is obtained considering that the loop at Lines 4–10 will perform $\binom{k}{p}$ iterations, where $p+1$ vectors of length ℓ are added together (complexity $p\ell$), and, if the result is zero, a further addition of $p+1$ bit vectors, each one of length $r-\ell$ has to be performed (complexity $p(r-\ell)$). This further addition takes place with a probability of $\frac{\binom{k}{p}}{2^\ell}$, as all possible values for $\bar{s}_{\text{up}}$ are 2^ℓ, and only $\binom{k}{p}$ attempts at hitting the correct one are made, thus yielding the correct complexity, under the assumption that the sums of ℓ bit vectors are independent and uniformly distributed over all the ℓ bit strings. □

3.5. Stern

Stern's algorithm, introduced in [13], improves Leon's (Algorithm 4) by employing a meet-in-the-middle strategy to find which set of size p, containing ℓ bit portions of columns of V, adds up to the first ℓ bits of the syndrome $\bar{s}$. For the sake of clarity, consider V as $V = \begin{bmatrix} V_{\text{up}} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up } 0} & \cdots & v_{\text{up } k-1} \\ v_{\text{down } 0} & \cdots & v_{\text{down } k-1} \end{bmatrix}$ where $v_{\text{up i}}$ are ℓ-bit column vectors, and $v_{\text{down } i}$ are $(r-\ell)$-bit column vectors, and the transformed syndrome $\bar{s}$ as $\bar{s} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix}$.

Stern's strategy splits the p-sized set $\mathcal{I}'$ of indexes of the columns of V_{up}, which should add up to $\bar{s}_{\text{up}}$, into two $\frac{p}{2}$ sized ones $\mathcal{I}$ and $\mathcal{J}$ ($\mathcal{I}' = \mathcal{I} \cup \mathcal{J}$). Stern's strategy mandates that all columns indexed by $\mathcal{I}$ should be within the leftmost $\frac{k}{2}$ ones of V, while the ones indexed by $\mathcal{J}$ should be within the rightmost $\frac{k}{2}$ ones. It then exploits the following equation

$$\sum_{i' \in \mathcal{I}'} v_{\text{up } i'} = \bar{s}_{\text{up}} \Leftrightarrow \bar{s}_{\text{up}} = \sum_{i \in \mathcal{I}} v_{\text{up } i} + \sum_{j \in \mathcal{J}} v_{\text{up } j} \Leftrightarrow \bar{s}_{\text{up}} + \sum_{i \in \mathcal{I}} v_{\text{up } i} = \sum_{j \in \mathcal{J}} v_{\text{up } j}$$

to precompute the value of $\bar{s}_{\text{up}} + \sum_{i \in \mathcal{I}} v_{\text{up } i}$ for all possible $\binom{k/2}{p/2}$ choices of $\mathcal{I}$, and store them into a lookup table $\mathcal{L}$, together with the corresponding choice of $\mathcal{I}$. The algorithm then enumerates all possible $\frac{p}{2}$ sized sets of indexes $\mathcal{J}$, computing for each one $\sum_{j \in \mathcal{J}} v_{\text{up } j}$, and checking if the result is present in $\mathcal{L}$. If this is the case, the algorithm has found a candidate pair $(\mathcal{I}, \mathcal{J})$ for which $\sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{up } i} = \bar{s}_{\text{up}}$ holds, and thus proceeds to check if $\sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i} = \bar{s}_{\text{down}}$. This strategy reduces the cost of computing an iteration quadratically at the price of increasing the number of iterations with respect to Lee and Brickell's approach, and taking a significant amount of space to store the lookup table $\mathcal{L}$ which contains $\binom{k/2}{p/2}$ elements. We note that Stern's variant of the ISD is the first one to exhibit non-polynomial memory requirements, due to the size of the set $\mathcal{I}$, which should be memorized and looked up. Stern's algorithm is summarized in Algorithm 5.

Algorithm 5: Syndrome decoding formulation of Stern's ISD.

Input: s: an r-bit long syndrome (column vector)

H: an $r \times n$ binary parity-check matrix

t: the weight of the error vector to be recovered

Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$

Data: P: an $n \times n$ permutation matrix

$\widehat{e} = eP$: the error vector permuted by P

p: the weight of the first k bits of $\widehat{e}$, $0 \leq p \leq t$,

ℓ: length of the run of zeroes in $\widehat{e} = [\widehat{e'}_{1\times k}\ 0_{1\times \ell}\ \widehat{e''}_{1\times r-\ell}]$

$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $[V\ I_r]$,

$\bar{s} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix}$

V: an $r \times k$ binary matrix

$V = \begin{bmatrix} v_0 & \dots & v_{k-1} \end{bmatrix} = \begin{bmatrix} V_{\text{up}} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up } 0} & \cdots & v_{\text{up } k-1} \\ v_{\text{down } 0} & \cdots & v_{\text{down } k-1} \end{bmatrix}$

$\mathcal{L}$: list of pairs $(\mathcal{I}, a_{\mathcal{I}})$, with $\mathcal{I}$ a set of integer indexes between 0 and $\frac{k}{2} - 1$, and $a_{\mathcal{I}}$ an ℓ-bit binary column vector

repeat
  $\langle P, [V\ I_r], \bar{s} \rangle \leftarrow \textsc{ISextract}(H, s)$
  $\mathcal{L} \leftarrow \varnothing$
  for $j \leftarrow 1$ **to** $\binom{k/2}{p/2}$ **do**
    $\mathcal{I} \leftarrow \textsc{IntegerToCombination}(j)$ `//` $\mathcal{I}$ `is a set of` $p/2$ `distinct integers in` $\{0, \dots, \frac{k}{2} - 1\}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathcal{I}, \bar{s}_{\text{up}} + \sum_{i \in \mathcal{I}} v_{\text{up } i})\}$
  for $j \leftarrow 1$ **to** $\binom{k/2}{p/2}$ **do**
    $\mathcal{J} \leftarrow \textsc{IntegerToCombination}(j)$ `//` $\mathcal{J}$ `is a set of` $p/2$ `distinct integers in` $\{0, \dots, \frac{k}{2} - 1\}$
    if $(\mathcal{I}, \sum_{i \in \mathcal{J}} v_{\text{up } i+k/2}) \in \mathcal{L}$ **then**
      if $\textsc{weight}(\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i}) = t - p$ **then**
        $\widehat{e} \leftarrow [0_{1\times(k+\ell)}\ (\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i})^T]$
        foreach $i \in \mathcal{I}$ **do**
          $\widehat{e} \leftarrow \widehat{e} + [0_{1\times i}\ 1\ 0_{1\times(r+k-1-i)}]$
        foreach $i \in \mathcal{J}$ **do**
          $\widehat{e} \leftarrow \widehat{e} + [0_{1\times i+k/2}\ 1\ 0_{1\times(r+k/2-1-i)}]$
        break
until $\textsc{weight}(\widehat{e}) = t$
return $\widehat{e}P^T$

Proposition 4 (Computational complexity of Algorithm 5)**.** *As for Algorithm 4, given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 5 requires two additional parameters $0 \leq p \leq t$, $0 \leq \ell \leq (k - p)$.*

The time complexity of Algorithm 5 can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ *of a single iteration of the loop at Lines* 1–17 *and the computational requirements of executing the loop body* c_{iter}. *In particular, the time complexity is*

$$\begin{aligned} C_{\mathrm{ISD}}(n,r,t,p,\ell) &= \frac{1}{\mathrm{Pr}_{\mathrm{succ}}} c_{\mathrm{iter}} \\ &= \frac{\binom{n}{t}}{\binom{k/2}{p/2}^2 \binom{r-\ell}{t-p}} \left(C_{\mathrm{IS}}(n,r) + \binom{k/2}{p/2} \frac{p}{2}\ell + \binom{k/2}{p/2} (C_{\mathrm{IntToComb}} + \frac{p}{2}\ell + \frac{\binom{k/2}{p/2}}{2^\ell} p(r-\ell)) \right) \end{aligned}$$

where $C_{\mathrm{IS}}(n,r)$ *is as in Equation* (6) *and* $C_{\mathrm{IntToComb}} = \mathcal{O}((2k-p)(\log \binom{k}{p})^2)$ *is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the* $\binom{k/2}{p/2}$ *ones. Note that, if the value of* p *is fixed, it is possible to avoid* $C_{\mathrm{IntToComb}}$, *specializing the algorithm with a* p*-deep loop nest to generate the combinations. The spatial complexity is* $S_{\mathrm{ISD}}(n,r,t,p,\ell) = \mathcal{O}\left(rn + \binom{k/2}{p/2}(\frac{p}{2}\log_2(\frac{k}{2}) + \ell)\right)$.

Proof. The success probability of an iteration of Stern's algorithm follows the same line of reasoning of the previous ones, dividing the number of admissible permuted error vectors $\binom{k/2}{p/2}^2 \binom{r-\ell}{t-p}$ by the total one $\binom{n}{t}$. The complexity of a single iteration is obtained considering that the loop at Lines 5–7 will compute $\binom{k/2}{p/2}$ additions of $\frac{p}{2}+1$ vectors, each one ℓ bits in length (complexity $\binom{k/2}{p/2}\frac{p}{2}\ell$). The loop at Lines 8–16 performs $\binom{k/2}{p/2}$ iterations, where $\frac{p}{2}+1$ vectors of length ℓ are added together (complexity $p\ell$), and the result is looked up in table $\mathcal{L}$. If the result is found, a further addition of $p+1$ bit vectors, each one $r-\ell$ bits long is performed (complexity $p(r-\ell)$). This further addition takes place with a probability of $\frac{\binom{k/2}{p/2}}{2^\ell}$, as all the possible values for the computed ℓ bit sum are 2^ℓ, and only $\binom{k/2}{p/2}$ are present in $\mathcal{L}$.

The spatial complexity of Stern's algorithm is the result of adding together the space required for the operations on the H matrix (i.e., rn) with the amount of space required by the list $\mathcal{L}$, which is $\binom{k/2}{p/2}$ elements long. Each element of the list takes $\frac{p}{2}\log_2(\frac{k}{2})$ bits to store the set of indexes, and ℓ bits to store the partial sum, yielding a total spatial cost for the list of $\binom{k/2}{p/2}(\frac{p}{2}\log_2(\frac{k}{2}) + \ell)$ bits. □

The aforementioned temporal complexity is obtained assuming a constant memory access cost which, given the exponential amount of memory required is likely to be ignoring a non-negligible amount of time spent to perform memory accesses. Indeed, it is possible to take into account such a time employing a logarithmic memory access cost model. Recalling that the address decoding logic for an n element digital memory of any kind cannot have a circuit depth smaller than $\log_2(n)$, we consider that the operations involved in the computation of an iteration will require such an access, in turn obtaining a cost per iteration equal to $c_{iter-logcost} = c_{iter}\log_2(S_{\mathrm{ISD}}(n,r,t,p,\ell))$.

3.6. Finiasz–Sendrier

Finiasz and Sendrier in [14] proposed two improvements on Stern's Information Set Decoding (ISD) algorithm, obtaining Algorithm 6. The first improvement is represented by removing the requirement for the presence of a run of ℓ zeroes in the permuted error vector $\widehat{e}$ and allowing the p error bits to be guessed to be present also in that region of $\widehat{e}$. Such an approach raises the success probability of an iteration. Following the fact that the p positions which should be guessed are picked among the first $k+\ell$ ones of the error vector, Finiasz and Sendrier computed only a partial Reduced Row Echelon Form (RREF) transformation obtaining a smaller, $(r-\ell)\times(r-\ell)$, identity matrix in the lower rightmost portion of $U\widehat{H} = \begin{bmatrix} V_{\mathrm{up}} & 0_{\ell\times(r-\ell)} \\ V_{\mathrm{down}} & I_{r-\ell} \end{bmatrix}$, and leaving a zero submatrix on top of the

identity. As a consequence, the cost of computing such an Reduced Row Echelon Form (RREF) is reduced to

$$C_{\text{RREF}-\text{FS}}(n,r,\ell) = -\frac{\ell^3}{3} - \frac{\ell^2 n}{2} + \frac{\ell^2 r}{2} - \frac{3\ell^2}{2} - \frac{3\ell n}{2} + \frac{\ell r}{2} - \frac{13\ell}{6} + \frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1.$$

Algorithm 6: Syndrome decoding formulation of Finiasz–Sendrier ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered

Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$

Data: P: an $n \times n$ permutation matrix
$\widehat{e} = eP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{e}$, $0 \le p \le t$,
ℓ: a free parameter $0 \le \ell \le r - (t - p)$
$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $\begin{bmatrix} V_{\text{up}} & 0_{\ell \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}$,
$\bar{s} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix}$
V: an $r \times (k+\ell)$ binary matrix $V = \begin{bmatrix} v_0 & \dots & v_{k-1} \end{bmatrix} = \begin{bmatrix} V_{\text{up}} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up } 0} & \cdots & v_{\text{up } k+\ell-1} \\ v_{\text{down } 0} & \cdots & v_{\text{down } k+\ell-1} \end{bmatrix}$
$\mathcal{L}$: list of pairs $(\mathcal{I}, a_{\mathcal{I}})$, with $\mathcal{I}$ a set of integer indexes between 0 and $\frac{k+\ell}{2} - 1$, and $a_{\mathcal{I}}$ an ℓ-bit binary column vector

repeat
  $\langle P, \begin{bmatrix} V_{\text{up}} & 0_{\ell \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}, \bar{s} \rangle \leftarrow \textsc{ISextract-FS}(H, s, \ell)$
  $\mathcal{L} \leftarrow \varnothing$
  for $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/2}$ **do**
    $\mathcal{I} \leftarrow \textsc{IntegerToCombination}(j)$ `// ` $\mathcal{I}$ `is a set of` $p/2$ `distinct int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathcal{I}, \bar{s}_{\text{up}} + \sum_{i \in \mathcal{I}} v_{\text{up i}})\}$
  for $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/2}$ **do**
    $\mathcal{J} \leftarrow \textsc{IntegerToCombination}(j)$ `// ` $\mathcal{J}$ `is a set of` $p/2$ `distinct int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    if $(\mathcal{I}, \sum_{i \in \mathcal{J}} v_{\text{up } i+(k+\ell)/2}) \in \mathcal{L}$ **then**
      if $\textsc{weight}(\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i}) = t - p$ **then**
        $\widehat{e} \leftarrow [0_{1 \times (k+\ell)} \ (\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i})^T]$
        foreach $i \in \mathcal{I}$ **do**
          $\widehat{e} \leftarrow \widehat{e} + [0_{1 \times i} \ 1 \ 0_{1 \times (r+k-1-i)}]$
        foreach $i \in \mathcal{J}$ **do**
          $\widehat{e} \leftarrow \widehat{e} + [0_{1 \times i+(k+\ell)/2} \ 1 \ 0_{1 \times (r+(k+\ell)/2-1-i)}]$
        break
until $\textsc{weight}(\widehat{e}) = t$
return $\widehat{e}P^T$

Considering that the invertibility condition is required only for an $(r-\ell) \times (r-\ell)$ submatrix, we have that

$$C_{\text{IS}-\text{FS}}(n,r,\ell) = \frac{1}{\prod_{i=1}^{r-\ell}(1 - 2^{-i})} C_{\text{RREF}}(n,r,\ell) + r^2$$

for a use of the ISD to solve Syndrome Decoding Problem (SDP), while the last r^2 term is not present in case the method is employed to solve the Codeword Finding Problem (CFP).

Proposition 5 (Computational complexity of Algorithm 6)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 6 also requires two additional parameters $0 \leq p \leq t$, $0 \leq \ell \leq (k-p)$.*

The time complexity of Algorithm 6 can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–17 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is

$$\begin{aligned} C_{\mathrm{ISD}}(n,r,t,p,\ell) = \frac{1}{\mathrm{Pr}_{\mathrm{succ}}} c_{\mathrm{iter}} &= \frac{\binom{n}{t}}{\binom{(k+\ell)/2}{p/2}^2 \binom{r-\ell}{t-p}} \Big(C_{\mathrm{IS-FS}}(n,r,\ell) \\ &+ \binom{(k+\ell)/2}{p/2} \frac{p}{2}\ell + \binom{(k+\ell)/2}{p/2} \left(C_{\mathrm{IntToComb}} + \frac{p}{2}\ell + \frac{\binom{(k+\ell)/2}{p/2}}{2^\ell} p(r-\ell) \right) \Big), \end{aligned}$$

where $C_{\mathrm{IntToComb}} = \mathcal{O}((2k-p)(\log \binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the $\binom{(k+\ell)/2}{p/2}$ ones. Note that, if the value of p is fixed, it is possible to avoid $C_{\mathrm{IntToComb}}$, specializing the algorithm with a p-deep loop nest to generate the combinations. The spatial complexity is $S_{\mathrm{ISD}}(n,r,t,p,\ell) = \mathcal{O}\left(rn + \binom{(k+\ell)/2}{p/2} (\frac{p}{2} \log_2(\frac{k+l}{2}) + \ell) \right)$.

With a line of reasoning analogous to Stern's ISD, we consider the complexity of Finiasz and Sendrier's ISD with a logarithmic memory access cost, multiplying the cost of the iteration by the binary logarithm of the size of the required memory.

3.7. May–Meurer–Thomae

The variant of ISD proposed by May, Meurer and Thomae in [15] improves Finiasz and Sendrier's variant by introducing two tweaks, resulting in Algorithm 7. The first tweak changes the way in which the p error positions in the permuted error vector $\widehat{e}$ are chosen. Instead of splitting them equally as $\frac{p}{2}$ in the leftmost $\frac{k+\ell}{2}$ columns and $\frac{p}{2}$ in the subsequent $\frac{k+\ell}{2}$ ones, the selection is made picking two disjoint sets of indexes $\mathcal{I}, \mathcal{J} \subset \{0, \ldots, k+\ell-1\}$. Such an approach increases the number of possible permutations which respect this constraint.

The second tweak considers V_{up} as logically split into two submatrices $V_{\mathrm{up}} = \begin{bmatrix} V_{\mathrm{up}-\ell_1} \\ V_{\mathrm{up}-\ell_2} \end{bmatrix}$ dividing it row-wise into two parts, the first one with ℓ_1 rows and the second one with $\ell_2 = \ell - \ell_1$ rows. After performing the same partial RREF, as it is done in the Finiasz and Sendrier's ISD, we obtain $U\widehat{H} = \begin{bmatrix} V_{\mathrm{up}-\ell_1} & 0_{\ell_1 \times (r-\ell)} \\ V_{\mathrm{up}-\ell_2} & 0_{\ell_2 \times (r-\ell)} \\ V_{\mathrm{down}} & I_{r-\ell} \end{bmatrix}$ and the corresponding $\bar{s} = \begin{bmatrix} \bar{s}_{\mathrm{up}-\ell_1} \\ \bar{s}_{\mathrm{up}-\ell_2} \\ \bar{s}_{\mathrm{down}} \end{bmatrix}$.

Such a subdivision is employed to further enhance the efficiency of the checks on the columns of V with respect to the idea of matching bit strings with a precomputed set introduced by Stern. To this end, the sets $\mathcal{I}$ and $\mathcal{J}$ are in turn obtained as the disjoint union of a pair subsets with cardinality $\frac{p}{4}$. Let $\mathcal{I}$ be $\mathcal{I}_1 \cup \mathcal{I}_2$ and $\mathcal{J}$ be $\mathcal{J}_1 \cup \mathcal{J}_2$. For the sake of simplicity, the disjoint union is realized picking the elements of $\mathcal{I}_1, \mathcal{J}_1$ in $\{0, \ldots, \frac{k+\ell}{2} - 1\}$ and the ones of $\mathcal{I}_2, \mathcal{J}_2$ in $\{\frac{k+\ell}{2}, \ldots, k+\ell-1\}$.

The May–Meurer–Thomae (MMT) algorithm thus proceeds to exploit a time to memory trade-off deriving from the same equation employed by Stern, applying twice the precomputation strategy. The derivation from the test equality on the syndrome is done as follows

$$\bar{s}_{\mathrm{up}} = \sum_{i \in \mathcal{I}} v_{\mathrm{up}\ i} + \sum_{j \in \mathcal{J}} v_{\mathrm{up}\ j} \Leftrightarrow$$

$$\begin{bmatrix}\bar{s}_{\text{up}-\ell_1}\\ \bar{s}_{\text{up}-\ell_2}\end{bmatrix} = \begin{bmatrix}a_{\text{up}-\ell_1}\\ \bar{s}_{\text{up}-\ell_2}\end{bmatrix} + \begin{bmatrix}b_{\text{up}-\ell_1}\\ 0_{\ell_2\times 1}\end{bmatrix}, \quad \begin{bmatrix}a_{\text{up}-\ell_1}\\ \bar{s}_{\text{up}-\ell_2}\end{bmatrix} = \sum_{j\in\mathcal{J}} \begin{bmatrix}v_{\text{up}-\ell_1\, j}\\ \bar{v}_{\text{up}-\ell_2\, j}\end{bmatrix}, \quad \begin{bmatrix}b_{\text{up}-\ell_1}\\ 0_{\ell_2\times 1}\end{bmatrix} = \sum_{i\in\mathcal{I}} \begin{bmatrix}v_{\text{up}-\ell_1\, i}\\ \bar{v}_{\text{up}-\ell_2\, i}\end{bmatrix}.$$

May, Meurer and Thomae exploited the strategy described by Stern to derive candidate values for the elements of $\mathcal{I}$ and $\mathcal{J}$, rewriting the last two equalities as

$$\begin{bmatrix}a_{\text{up}-\ell_1}\\ \bar{s}_{\text{up}-\ell_2}\end{bmatrix} + \sum_{j\in\mathcal{J}_1} \begin{bmatrix}v_{\text{up}-\ell_1\, j}\\ \bar{v}_{\text{up}-\ell_2\, j}\end{bmatrix} = \sum_{j\in\mathcal{J}_2} \begin{bmatrix}v_{\text{up}-\ell_1\, j}\\ \bar{v}_{\text{up}-\ell_2\, j}\end{bmatrix}, \quad \begin{bmatrix}b_{\text{up}-\ell_1}\\ 0_{\ell_2\times 1}\end{bmatrix} + \sum_{i\in\mathcal{I}_1} \begin{bmatrix}v_{\text{up}-\ell_1\, i}\\ \bar{v}_{\text{up}-\ell_2\, i}\end{bmatrix} = \sum_{i\in\mathcal{I}_2} \begin{bmatrix}v_{\text{up}-\ell_1\, i}\\ \bar{v}_{\text{up}-\ell_2\, i}\end{bmatrix}$$

and exploiting their form to build two lists of candidate values for $\mathcal{I}$ and $\mathcal{J}$, $\bar{\mathcal{I}}$ and $\bar{\mathcal{J}}$, such that for the elements of the first list, it holds that $0_{\ell_2\times 1} + \sum_{i\in\mathcal{I}_1} \bar{v}_{\text{up}-\ell_2\, i} = \sum_{i\in\mathcal{I}_2} \bar{v}_{\text{up}-\ell_2\, i}$, and for the elements of the second list it holds that $\bar{s}_{\text{up}-\ell_2} + \sum_{j\in\mathcal{J}_1} \bar{v}_{\text{up}-\ell_2\, j} = \sum_{j\in\mathcal{J}_2} \bar{v}_{\text{up}-\ell_2\, j}$. We note that, through a straightforward implementation optimization, matching the one employed in Stern's algorithm, only the first list needs to be materialized (appearing as $\mathcal{L}$ in Algorithm 7).

The second observation made in the MMT algorithm relates to the possibility of reducing the size of the list involved in Stern's algorithm to compute the value of the sought error vector via a meet-in-the-middle strategy. The observation relies on the fact that it is possible to obtain the first $k+\ell$ bits of the permuted error vector $\widehat{e}$, $\widehat{e}[k+\ell]$ as the sums of two $k+\ell$ long bit vectors, $\widehat{e}_1, \widehat{e}_2$ with weight $\frac{p}{2}$ each. Stern's algorithm limits the positions of the ones in $\widehat{e}_1, \widehat{e}_2$ to be in the first half of the bits for $\widehat{e}_1$ and in the second half for $\widehat{e}_2$, yielding a single valid pair $\widehat{e}_1, \widehat{e}_2$ for a given $\widehat{e}$. By contrast May–Meurer–Thomae does not constrain the positions of the two sets of $\frac{p}{2}$ positions to be picked from different halves of the $k+\ell$ region of the error vector, but instead it only constrains the choice to non-overlapping positions. In such a fashion, considering the correct guess of p positions, we have that they can be split into the two $\frac{p}{2}$ sets in $\binom{p}{p/2}$ possible valid ways, in turn increasing the likelihood of a correct guess. If this choice is made, it is possible to reduce the size of the lists employed to compute the man in the middle approach by a factor of $\binom{p}{p/2}$, while retaining (on average), at least a solution. To this end, the authors suggested picking the value of l_2 as $\binom{\frac{k+\ell}{2}}{\frac{p}{4}}$, in a way to reduce the size of the lists by the proper factor.

Proposition 6 (Computational complexity of Algorithm 7)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 7 requires three additional parameters $0 \le p \le t$, ℓ_1 and ℓ_2 such that $\ell_1 + \ell_2 = \ell, 0 \le \ell \le (k-p)$.*

The time complexity of May–Meurer–Thomae is

$$\begin{aligned}&C_{\text{ISD}}(n,r,t,p,\ell_1,\ell_2) = \\ &\frac{\binom{n}{t}}{\binom{k+\ell}{p}\binom{r-\ell}{t-p}}\left(C_{\text{IS-FS}}(n,r,\ell) + \binom{(k+\ell)/2}{p/4}\frac{p}{4}\ell_2 + \min\left(\binom{(k+\ell)/2}{p/4}, \frac{\binom{(k+\ell)/2}{p/2}}{\binom{p}{p/2}}\right)\cdot\left(\frac{p}{4}\ell_2 + \frac{\binom{(k+\ell)/2}{p/4}}{2^{\ell_2}}\frac{p}{2}\ell_1\right) + \right.\\ &\left. +\binom{(k+\ell)/2}{p/4}\frac{p}{4}\ell_2 + \binom{(k+\ell)/2}{p/4}\left(\frac{p}{4}\ell_2 + \frac{\binom{(k+\ell)/2}{p/4}}{2^{\ell_2}}\left(\frac{p}{2}\ell_1 + \frac{\binom{(k+\ell)}{p/2}\binom{p}{p/2}}{2^{\ell_1}}p(r-\ell)\right)\right)\right).\end{aligned}$$

The spatial complexity of May–Meurer–Thomae is

$$\begin{aligned}&S_{\text{ISD}}(n,r,t,p,\ell) = \\ &\mathcal{O}\left(rn + \binom{(k+\ell)/2}{p/4}\left(\frac{p}{4}\log_2\left(\frac{k+l}{2}\right) + \ell_2\right) + \min\left(\binom{(k+\ell)/2}{p/4}, \frac{\binom{(k+\ell)/2}{p/2}}{\binom{p}{p/2}}\right)\left(\frac{p}{2}\log_2(k+\ell) + \ell\right)\right).\end{aligned}$$

Algorithm 7: Syndrome decoding formulation of May–Meurer–Thomae ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered
Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\text{WEIGHT}(e) = t$
Data: P: an $n \times n$ permutation matrix
$\widehat{e} = eP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{e}$, $0 \leq p \leq t$, $p = 2$ proven optimal
ℓ_1, ℓ_2: two parameters with $\ell_1 + \ell_2 = \ell$, $0 \leq \ell \leq r - (t - p)$
$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $\begin{bmatrix} V_{\text{up}-\ell_1} & 0_{\ell_1 \times (r-\ell)} \\ V_{\text{up}-\ell_2} & 0_{\ell_2 \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}, \bar{s} = \begin{bmatrix} \bar{s}_{\text{up}-\ell_1} \\ \bar{s}_{\text{up}-\ell_2} \\ \bar{s}_{\text{down}} \end{bmatrix}$
V: an $r \times (k + \ell)$ binary matrix $V = \begin{bmatrix} v_0 & \dots & v_{k-1} \end{bmatrix} = \begin{bmatrix} V_{\text{up}-\ell_1} \\ V_{\text{up}-\ell_2} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up}-\ell_1\,0} & \dots & v_{\text{up}-\ell_1\,k+\ell-1} \\ v_{\text{up}-\ell_2\,0} & \dots & v_{\text{up}-\ell_2\,k+\ell-1} \\ v_{\text{down}\,0} & \dots & v_{\text{down}\,k+\ell-1} \end{bmatrix}$
$\mathcal{L}$: list of pairs $(\mathcal{I}, a_{\mathcal{I}})$, with $\mathcal{I}$ a set of integer indexes between 0 and $k + \ell - 1$, and $a_{\mathcal{I}}$ an ℓ-bit binary column vector; length of $\mathcal{L}$ is kept at most $\binom{(k+\ell)}{p/2} / \binom{p}{p/2}$
$\mathcal{L}_1, \mathcal{L}_2$: lists of pairs $(\mathcal{I}_1, a_{\mathcal{I}_1})$, and $(\mathcal{J}_1, a_{\mathcal{J}_1})$ with $\mathcal{I}_1, \mathcal{J}_1$ sets of integer indexes between 0 and $\frac{k+\ell}{2} - 1$, and $a_{\mathcal{I}_1}, a_{\mathcal{J}_1}, \ell_1$ and ℓ_2 bit binary column vectors

repeat
  $\langle P, \begin{bmatrix} V_{\text{up}-\ell_1} & 0_{\ell_1 \times (r-\ell)} \\ V_{\text{up}-\ell_2} & 0_{\ell_2 \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}, \bar{s} \rangle \leftarrow \text{ISEXTRACT-FS}(H, s, \ell)$
  $\mathcal{L}_1 \leftarrow \varnothing$
  for $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/4}$ **do**
    $\mathcal{I}_1 \leftarrow \text{INTEGERTOCOMBINATION}(j)$ `//` $\mathcal{I}_1$ `is a set of` $p/4$ `int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{(\mathcal{I}_1, \bar{s}_{\text{up}-\ell_2} + \sum_{i \in \mathcal{I}_1} v_{\text{up}-\ell_2\,i})\}$
  $\mathcal{L} \leftarrow \varnothing$
  for $i \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/4}$ **do**
    $\mathcal{I}_2 \leftarrow \text{INTEGERTOCOMBINATION}(i)$ `//` $\mathcal{I}_2$ `is a set of` $p/4$ `int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    if $(\mathcal{I}_1, \sum_{i \in \mathcal{I}_2} v_{\text{up}-\ell_2\,i+(k+\ell)/2}) \in \mathcal{L}_1$ **then**
      $\mathcal{I} \leftarrow \mathcal{I}_1$
      for $j \in \mathcal{I}_2$ **do**
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{j + \frac{k+\ell}{2}\}$
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathcal{I}, \sum_{i \in \mathcal{I}} v_{\text{up}-\ell_1\,i})\}$
      if $|\mathcal{L}| \geq \binom{(k+\ell)}{p/2} / \binom{p}{p/2}$ **then**
        break
  `//` $\mathcal{L}_1$ `is no longer needed from here onwards`
  $\mathcal{L}_2 \leftarrow \varnothing$
  for $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/4}$ **do**
    $\mathcal{J}_1 \leftarrow \text{INTEGERTOCOMBINATION}(j)$ `//` $\mathcal{J}_1$ `is a set of` $p/4$ `int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{(\mathcal{J}_1, \sum_{i \in \mathcal{J}_1} v_{\text{up}-\ell_2\,i})\}$
  for $a \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p/4}$ **do**
    $\mathcal{J}_2 \leftarrow \text{INTEGERTOCOMBINATION}(a)$ `//` $\mathcal{J}_2$ `is a set of` $p/4$ `int.s in` $\{0, \dots, \frac{k+\ell}{2} - 1\}$
    if $(\mathcal{J}_1, \bar{s}_{\text{up}-\ell_2} + \sum_{i \in \mathcal{J}_2} v_{\text{up}-\ell_2\,i+(k+\ell)/2}) \in \mathcal{L}_2$ **then**
      $\mathcal{J} \leftarrow \mathcal{J}_1$
      for $j \in \mathcal{J}_2$ **do**
        $\mathcal{J} \leftarrow \mathcal{J} \cup \{j + \frac{k+\ell}{2}\}$
      if $(\mathcal{I}, \bar{s}_{\text{up}-\ell_1} + \sum_{j \in \mathcal{J}} v_{\text{up}-\ell_1\,\text{j}}) \in \mathcal{L}$ **then**
        if $\text{WEIGHT}(\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down}\,i}) = t - p$ **then**
          $\widehat{e} \leftarrow 0_{1 \times (k+\ell)}\ (\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down}\,i})^T]$
          foreach $i \in \mathcal{I} \cup \mathcal{J}$ **do**
            $\widehat{e} \leftarrow \widehat{e} + [0_{1 \times i}\ 1\ 0_{1 \times (r+k-1-i)}]$
          break
until $\text{WEIGHT}(\widehat{e}) = t$
return $\widehat{e}P^T$

Proof. The computational complexity is derived considering the number of iterations of the loops in the algorithm, taking into account the probability of the checks being taken. The spatial complexity is obtained as the sum of the size of the matrix H, the size requirements of $\mathcal{L}_1$ (as $\mathcal{L}_2$ has the same size and can reuse its space) and the expected size of $\mathcal{L}$ considering how many pairs may survive the check performed when building it. □

3.8. Becker–Joux–May–Meurer

The Becker–Joux–May–Meurer (BJMM) algorithm introduced in [16] improves the MMT algorithm in two ways: the former is a recursive application of the list building strategy, and the latter is a change to the list element generation employed. We discuss the latter first, forsaking for the sake of clarity its recursive application at first. We then describe the adaptations needed to adopt the recursive splitting strategy without issues.

The BJMM algorithm considers that it is possible to represent a vector e of weight p, length $k+\ell$, as the sum of two vectors e_1, e_2 of weight $\frac{p}{2}+\varepsilon$, and with the same length, under the assumption that the ε extra ones cancel out during the addition. We recall that the MMT approach demands that both e_1 and e_2 have weight strictly equal to $\frac{p}{2}$. The BJMM algorithm thus raises the number of valid pairs e_1, e_2 employed to represent e by a factor equal to $\binom{k+\ell-p}{\varepsilon}$. Such an improvement is employed to further reduce the size of the lists of values which need to be computed to find e with a meet-in-the-middle approach on checking that the condition $Ve_1 + Ve_2 = \bar{s}_{\text{up}}$. Indeed, since $R = \binom{p}{\frac{p}{2}}\binom{k+\ell-p}{\varepsilon}$ pairs (e_1, e_2) which respect $e = e_1 + e_2$ exist, searching a $\frac{1}{R}$ fraction of the $\binom{k+l}{\frac{p}{2}+\epsilon}^2$ exhaustively will yield (on average) a solution, assuming that the solution pairs are uniformly distributed over all the $\binom{k+l}{\frac{p}{2}+\epsilon}^2$ ones.

Willing to employ a strategy to enumerate only a $\frac{1}{R}$ fraction of the pairs, while doing useful computation instead of a simple $\frac{1}{R}$ sub-sampling of the space of the pairs, the BJMM algorithm opts for performing a partial check of the $Ve_1 + Ve_2 = \bar{s}_{\text{up}}$ equation, on a smaller number of bits than ℓ and discarding the pairs which do not pass the check.

Let us denote with $V_{\text{up}[\rho]}$ the first ρ rows of V_{up} and with $\bar{s}_{\text{up}[\rho]}$ the first ρ bits of the syndrome $\bar{s}_{\text{up}}$ The BJMM algorithm thus employs the test $V_{\text{up}[\rho]}e_1 + V_{\text{up}[\rho]}e_2 = \bar{s}_{\text{up}[\rho]}$ to obtain a twofold objective: discard a fraction of the (e_1, e_2) pairs, and select the portion to be kept among the pairs which at least have the first ρ bits of the sum of the columns of $V_{\text{up}[\rho]}$ matching the value of the corresponding syndrome bits. Such an approach has the advantage over a random sampling that the pairs which are selected have already been checked for compliance on a part of the $Ve_1 + Ve_2 = \bar{s}_{\text{up}}$ equation, i.e., it performs a random subsampling while doing useful computation. The BJMM paper suggests that the value of ρ should be $\rho \approx \log_2(R)$: under the assumption that the r bit sums being performed are made of random values, and that the sum should match a given r bit value, only a fraction equal to $\frac{1}{2^\rho}$, i.e., $\approx \frac{1}{R}$ survives. Note that, regardless of the choice of ρ, a selection of the correct positions will always survive the preliminary checks on ρ bits, while wrong solutions are filtered out on the basis that they will not match on the first ρ bits. In the complete algorithm, the aforementioned line of reasoning is applied twice, as the list-building strategy is recursively applied, leading to two different reduction factors depending on the recursion depth itself.

We now come to the second improvement of the BJMM algorithm, the recursive application of the meet-in-the-middle strategy employed by all the ISDs since Stern's. Stern's meet-in-the-middle approach starts from rewriting $V_{\text{up}[l]}e_1 + V_{\text{up}[l]}e_2 = \bar{s}_{\text{up}}$ as $V_{\text{up}[l]}e_1 = V_{\text{up}[l]}e_2 + \bar{s}_{\text{up}}$. The original BJMM proceeds to build two lists of pairs. The first list contains, for all possible e_1, the pairs $(V_{\text{up}[l]}e_1, e_1)$. The second list contains, for all possible e_2, the pairs $(V_{\text{up}[l]}e_2 + \bar{s}_{\text{up}}, e_2)$. The BJMM algorithm sorts lexicographically the two lists on the first elements of the pairs and then checks for the matching pairs in essentially linear time in the length of the lists.

We note that a more efficient (memory saving) way of performing the same computation involves inserting in a (open hash) hashmap which employs as the key $V_{\text{up}[l]}e_1$ for the value e_1. Subsequently, computing on the fly $V_{\text{up}[l]}e_2 + \bar{s}_{\text{up}}$, and looking it up in the hashmap, yields all the matching pairs for e_2. Let N be the number of possible pairs and M the number of matching pairs, the original strategy requires $\mathcal{O}(2\sqrt{N} + 2\sqrt{N}\log_2 2\sqrt{N} + M)$ vector sized operations, while the modified one requires $\mathcal{O}(\sqrt{N} + \sqrt{N} + M)$.

The BJMM algorithm employs the meet-in-the-middle strategy to generate the values for the candidate vectors e more than once. In particular, a candidate for e, $e^{(0)}$, weight p, length $k+\ell$,

is generated from two vectors $e_1^{(1)}, e_2^{(1)}$, weight $p_1 = \frac{p}{2} + \varepsilon_1$, length $k+\ell$. The $e_1^{(1)}, e_2^{(1)}$ vectors are in turn generated by pairs $e_1^{(2)}, e_2^{(2)}$ and $e_3^{(2)}, e_4^{(2)}$, which all have weight $p_2 = p_1 + \varepsilon_2$. Finally, $e_1^{(2)}, e_2^{(2)}, e_3^{(2)}, e_4^{(2)}$, are generated by pairs $e_{1+2i}^{(3)}, e_{2+2i}^{(3)}, i \in \{0,1,2,3\}$, which have length $k+\ell$, and weight $p_3 = \frac{p_2}{2} + 0$, i.e., no extra ones which will cancel out are allowed when generating $e^{(2)}$'s.

In adopting this approach, two issues must be coped with: no overlapping positions for the ones should be present between any $e_{1+2i}^{(3)}, e_{2+2i}^{(3)}$ pair, and the values all the $V_{\mathrm{up}[l]} e_i^{(1)}$, $V_{\mathrm{up}[l]} e_i^{(2)}$, $V_{\mathrm{up}[l]} e^{(1)}$ are matched against should be unrelated, so that the sampling of pairs during the merge action of two lists is indeed picking items independently from another list merger on the same level. This allows a list merger at a lower level to consider the elements from above to be picked at random. The first issue is solved picking the positions of the ones for a $e_{1+2i}^{(3)}, e_{2+2i}^{(3)}$ pair from disjoint sets. Since the independence among the inputs of two level-3 list mergers should still hold, a pair of disjoint sets is generated for each level-3 list pair. In other words, for a given $e_{1+2i}^{(3)}, e_{2+2i}^{(3)}$ pair, the position of the ones of $e_{1+2i}^{(3)}$, belong to a $\frac{k+\ell}{2}$ sized set which has null intersection with the set of positions from which the ones of $e_{2+2i}^{(3)}$ are picked. This constraint may cause a given target permuted error not to be representable as a combination of the aforementioned pairs. Indeed, consider the fact that the aforementioned strategy implies that there are possible $\binom{\frac{k+\ell}{2}}{p_3}^2$ pairs, while the vectors to be represented are $\binom{k+\ell}{2p_3}$.

The second issue is solved slightly modifying the matching equations as follows

$$
\begin{array}{lcl}
V_{\mathrm{up}[\ell]} e_1^{(1)} = V_{\mathrm{up}[\ell]} e_2^{(1)} + \bar{s}_{\mathrm{up}[\ell]} & \rightarrow & \text{no change} \\
\\
V_{\mathrm{up}[r_1]} e_1^{(2)} = V_{\mathrm{up}[r_1]} e_2^{(2)} + \bar{s}_{\mathrm{up}[r_1]} & \rightarrow & V_{\mathrm{up}[r_1]} e_1^{(2)} = V_{\mathrm{up}[r_1]} e_2^{(2)} + \bar{s}_{\mathrm{up}[r_1]} + \mathtt{rnd}_1 \\
V_{\mathrm{up}[r_1]} e_3^{(2)} = V_{\mathrm{up}[r_1]} e_4^{(2)} + 0_{[r_1]} & \rightarrow & V_{\mathrm{up}[r_1]} e_3^{(2)} = V_{\mathrm{up}[r_1]} e_4^{(2)} + \mathtt{rnd}_1 \\
\\
V_{\mathrm{up}[r_2]} e_1^{(3)} = V_{\mathrm{up}[r_2]} e_2^{(3)} + \bar{s}_{\mathrm{up}[r_2]} & \rightarrow & V_{\mathrm{up}[r_2]} e_1^{(3)} = V_{\mathrm{up}[r_2]} e_2^{(3)} + \bar{s}_{\mathrm{up}[r_2]} + \mathtt{rnd}_1 + \mathtt{rnd}_2 \\
V_{\mathrm{up}[r_2]} e_3^{(3)} = V_{\mathrm{up}[r_2]} e_4^{(3)} + 0_{r_2} & \rightarrow & V_{\mathrm{up}[r_2]} e_3^{(3)} = V_{\mathrm{up}[r_2]} e_4^{(3)} + \mathtt{rnd}_2 \\
V_{\mathrm{up}[r_2]} e_5^{(3)} = V_{\mathrm{up}[r_2]} e_6^{(3)} + 0_{r_2} & \rightarrow & V_{\mathrm{up}[r_2]} e_5^{(3)} = V_{\mathrm{up}[r_2]} e_5^{(3)} + \mathtt{rnd}_1 + \mathtt{rnd}_2 \\
V_{\mathrm{up}[r_2]} e_7^{(3)} = V_{\mathrm{up}[r_2]} e_8^{(3)} + 0_{r_2} & \rightarrow & V_{\mathrm{up}[r_2]} e_7^{(3)} = V_{\mathrm{up}[r_2]} e_7^{(3)} + \mathtt{rnd}_2
\end{array}
$$

so that the checks at each level also act in such a fashion that the total sum over r_2, r_1, ℓ matches the syndrome already, while retaining the desired randomness.

Proposition 7 (Computational complexity of Algorithm 8)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 8 requires three additional parameters $0 \le p \le t$, ℓ_1 and ℓ_2 such that $\ell_1 + \ell_2 = \ell, 0 \le \ell \le (k-p)$.*

The time complexity of the algorithm is

$$C_{\mathrm{ISD}}(n, r, t, p, \ell_1, \ell_2) = P_{success-BJMM}^{-1} C_{iter}$$

where the probability of an iteration to succeed, $P_{success-BJMM}$, is equal to the one in May–Meurer–Thomae (i.e., $\frac{\binom{n}{t}}{\binom{k+\ell}{p}\binom{r-\ell}{t-p}}$), multiplied by a factor which quantifies the fact that it is possible, picking the two disjoint sets over the subsets of p_i positions from the error vector are selected may result in a set which does not contain enough

positions. Such a factor, considered in all the splits in the BJMM is $\left(\frac{\binom{(k+\ell)/2}{p_3}^2}{\binom{k+\ell}{p_2}}\right)^4$*. The cost of an iteration of the loop at Lines 1–27 of the BJMM is as follows*

$$4\left(k+\ell+2\binom{\frac{k+\ell}{2}}{p_3}+\ell_2+\binom{\frac{k+\ell}{2}}{p_3}^2(2p_3\ell_2)\right)+2\left(\left(\frac{\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}}{2^{\ell_2}}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2(2p_2\ell_1)\right)$$
$$+\left(\frac{\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon_1}}{2^{\ell_1}}\left(\frac{\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}}{2^{\ell_2}}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2\right)^2(2p_1\ell)+\frac{\left(\frac{\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon_1}}{2^{\ell_1}}\left(\frac{\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}}{2^{\ell_2}}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2\right)^2}{2^{\ell}}(p(r-\ell)).$$

The first line constitutes the cost of the loop at Lines 4–16, the second line is the cost of the loop at Lines 17–23, the third line is the cost of the loop at Lines 24–27 save for the portion related to the branch at Line 25 being taken. The last line is the cost of computing the body of the taken branch, multiplied by the probability of such a branch being taken.

The BJMM variant of the ISD shares with the Stern, Finiasz and Sendrier, and May–Meurer–Thomae ISDs the fact that the exponential memory requirements should be taken into account by a logarithmic access cost, instead of a constant one. We do so following the same method employed for the aforementioned variants, i.e., augmenting the cost of the iteration accordingly.

3.9. Speedups in ISD Algorithms Due to Quasi-Cyclic Codes

A common choice to reduce the size of the keys in a McEliece or Niederreiter cryptosystem is to employ a so-called *quasi-cyclic* code. Such a code is characterized by a parity-check matrix composed by circulant block matrices, i.e., matrices where all the rows are obtained as a cyclic shift of the first one.

It is possible to exploit such a structure to provide a polynomial speedup factor to both the solution of Codeword Finding Problem (CFP) and Syndrome Decoding Problem (SDP). The speedup in the solution of the Codeword Finding Problem (CFP) can be derived in a straightforward fashion observing that, in the case of both the Codeword Finding Problem (CFP) against the primal and the one against the dual code, for each codeword to be found in a quasi cyclic code with p sized circulant blocks, $p-1$ more codewords can be derived simply as a block-wise circulant shift of the first one. As a consequence, for a given codeword with weight w sought by the algorithm, it is guaranteed that at least p many of them are present in the code. Thus, in this case. the success probability of each iteration can be obtained as $1-(1-\Pr_{\text{succ}})^p$; when $p\Pr_{\text{succ}} \ll 1$, this in turn implies that the probability of success approximately grows by a factor p, in turn speeding up any ISD by the same factor.

An analogous line of reasoning leads to exploit the Decoding One Out of Many (DOOM) algorithm proposed by Sendrier in [38] to speed up the solution of the Syndrome Decoding Problem (SDP). Decoding One Out of Many (DOOM) relies on the fact that a set of syndromes S through the same parity check matrix are provided to the attacker, and he attempts at decoding at least one of them. In case of a quasi cyclic code, cyclically shifting the syndrome yields a different, valid syndrome, and a predictable cyclic shift on the corresponding (unknown) error vector. It is therefore possible for an attacker to derive p different syndromes, starting from one and, in case one of them is successfully decoded, no matter which one, he will be able to reconstruct the sought error vector. Essentially, Decoding One Out of Many (DOOM) performs multiple ISD instances, taking care of duplicating only the checks which involve the syndrome, thus pooling the considerable amount of effort required in the rest of the iteration. The overall speedup achieved by Decoding One Out of Many (DOOM) for a quasi cyclic code with block size p is $\sqrt{p}$.

Algorithm 8: Syndrome decoding formulation of Becker–Joux–May–Meurer ISD.

Input: s: an r-bit long syndrome (column vector)
H: an $r \times n$ binary parity-check matrix
t: the weight of the error vector to be recovered
Output: e: an n-bit binary row error vector s.t. $He^T = s$, with $\textsc{weight}(e) = t$
Data: P: an $n \times n$ permutation matrix
$\widehat{e} = eP$: the error vector permuted by P
p: the weight of the first k bits of $\widehat{e}$, $0 \le p \le t$
ℓ: a free parameter $0 \le \ell \le r - (t - p)$
$\bar{s}$: an r-bit long binary column vector, equal to the syndrome of e through $\begin{bmatrix} V_{\text{up}} & 0_{\ell \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}$, $\bar{s} = \begin{bmatrix} \bar{s}_{\text{up}} \\ \bar{s}_{\text{down}} \end{bmatrix}$
V: an $r \times (k+\ell)$ binary matrix $V = \begin{bmatrix} v_0 & \dots & v_{k-1} \end{bmatrix} = \begin{bmatrix} V_{\text{up}} \\ V_{\text{down}} \end{bmatrix} = \begin{bmatrix} v_{\text{up } 0} & \dots & v_{\text{up } k+\ell-1} \\ v_{\text{down } 0} & \dots & v_{\text{down } k+\ell-1} \end{bmatrix}$
$\mathcal{L}_b^{(a)}$: list of set of indexes $\mathcal{I}^{(a)}$, at layer a, (root layer: $a = 0$, leaf layer $a = 3$)
$|\mathcal{I}^{(a)}| = p_a = p_{a-1}/2 + \varepsilon_a, a \in 1, 2, 3, p_0 = p, \varepsilon_3 = 0$; the elements of $\mathcal{I}^{(a)}$ are integers in $\{0, \dots, k+\ell-1\}$
ℓ_1, ℓ_2: parameters respecting $0 \le \ell_2 \le \ell_1 \le \ell$, stated as optimized for $l_1 \approx \log_2 \binom{p}{p/2}\binom{k+l-p}{\varepsilon_1}$,
$l_2 \approx \log_2 \binom{p_1}{p_1/2}\binom{k+l-p_1}{\varepsilon_2}$

repeat
  $\langle P, \begin{bmatrix} V_{\text{up}} & 0_{\ell \times (r-\ell)} \\ V_{\text{down}} & I_{r-\ell} \end{bmatrix}, \bar{s} \rangle \leftarrow \textsc{ISExtract-FS}(H, s, \ell)$
  for $i \leftarrow 0$ **to** 3 **do**
    $\mathcal{L}_{2i}^{(3)} \leftarrow \varnothing$
    $\mathfrak{I}_{2i} \leftarrow \textsc{RandomExtract}(k+\ell)$ `// populates` $\mathfrak{I}_{2i}$ `with` $\frac{k+\ell}{2}$ `values in` $\{0, \dots, k+\ell-1\}$
    $\mathfrak{I}_{2i+1} \leftarrow \{0, \dots, k+\ell-1\} \setminus \mathfrak{I}_{2i}$
    for $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p_3}$ **do**
      $\mathcal{I} \leftarrow \textsc{EnumerateComb}(\mathfrak{I}_{2i}, p_3, j)$ `// Picks the` j`th comb. of` p_3 `items of` $\mathfrak{I}_{2i}$
      $\mathcal{L}_{2i}^{(3)} \leftarrow \mathcal{L}_{2i}^{(3)} \cup \{(\mathcal{I}, \sum_{i \in \mathcal{I}} v_{\text{up}-\ell_2\, i})\}$
  for $i \leftarrow 0$ **to** 1 **do**
    $x \leftarrow \textsc{RandBitString}(\ell_2) + \bar{s}_{\text{up}-\ell_2}$
    $\mathcal{L}_{2i}^{(2)} \leftarrow \varnothing$
    foreach $j \leftarrow 1$ **to** $\binom{(k+\ell)/2}{p_3}$ **do**
      $\mathcal{J} \leftarrow \textsc{EnumerateComb}(\mathfrak{I}_{2i+1}, p_3, j)$
      $a \leftarrow x + \sum_{j \in \mathcal{J}} v_{\text{up}-\ell_2\, j}$
      if $(\mathcal{I}, a) \in \mathcal{L}_{4i}^{(3)}$ **then**
        $\mathcal{L}_{2i}^{(2)} \leftarrow \mathcal{L}_{2i}^{(2)} \cup \{(\mathcal{I} \cup \mathcal{J}, x + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{up}-\ell_1\, i})\}$
      if $|\mathcal{L}_{2i}^{(2)}| > \binom{k+\ell}{p_2} / \left(\binom{p_1}{p_1/2}\binom{k+l-p_1}{\epsilon_2}\right)$ **then**
        break
  foreach $(\mathcal{I}, \mathcal{J}) \in \mathcal{L}_0^{(1)} \times \mathcal{L}_1^{(1)}$ **do**
    if $\sum_{i \in \mathcal{I}} v_{\text{up } i} + \sum_{j \in \mathcal{J}} v_{\text{up } j} = \bar{s}^{[\ell]}$ **then**
      if $\textsc{weight}(\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i}) = t - p$ **then**
        $\widehat{e} \leftarrow [0_{1 \times (k+\ell)}\ (\bar{s}_{\text{down}} + \sum_{i \in \mathcal{I} \cup \mathcal{J}} v_{\text{down } i})^T]$
        foreach $i \in \mathcal{I} \cup \mathcal{J}$ **do**
          $\widehat{e} \leftarrow \widehat{e} + [0_{1 \times i}\ 1\ 0_{1 \times (r+k-1-i)}]$
        break
until $\textsc{weight}(\widehat{e}) = t$
return $\widehat{e}P^T$

3.10. Speedups from Quantum Computing

While there is no known polynomial time algorithm running on a quantum computer able to solve either Syndrome Decoding Problem (SDP) or Codeword Finding Problem (CFP), it is still possible

to achieve a significant speedup in the attacks exploiting Grover's zero-finding algorithm. Grover's algorithm [39] finds a zero of an n-input Boolean function with a computational cost of $\sqrt{2^n}$ function computations, instead of the 2^n required with a classical computer. The first instance of a proposed exploitation of Grover's algorithm to speed up ISDs was made by Bernstein in [40], observing that one iteration of Prange's algorithm can be rewritten as a Boolean function having a zero iff the iteration is successful in finding a valid error vector. The essence of the observation is that the Reduced Row Echelon Form (RREF) computation, and the weight check on the resulting syndrome can be expressed as Boolean functions, and it is straightforward to extend them so that a single bit output indicating the success of the iteration is added. Such an approach allows reducing the number of iterations to be performed to the square root of the one for the classical algorithm, since each iteration of Prange's algorithm is essentially trying (exhaustively) to find a zero of the aforementioned Boolean function. We therefore rephrase the computational complexity of Prange's algorithm on a quantum computer. For the sake of simplicity in the analysis, we forgo the overhead of implementing the Boolean function as a reversible circuit, obtaining a conservative estimate of the actual complexity.

Proposition 8 (Quantum computational complexity of Algorithm 1)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 1 running on a quantum computer can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–7 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is $C_{\mathrm{ISD}}(n,r,t) = \sqrt{\frac{1}{\mathrm{Pr}_{\mathrm{succ}}}} c_{\mathrm{iter}} = \sqrt{\frac{\binom{n}{t}}{\binom{r}{t}}} \left(C_{\mathrm{IS}}(n,r) + \mathcal{O}(n)\right)$, with*

$$C_{\mathrm{IS}}(n,r) = \frac{1}{\prod_{i=1}^{r}(1-2^{-i})} C_{\mathrm{RREF}}(n,r) + r^2,$$

$$C_{\mathrm{RREF}}(n,r) = \mathcal{O}\left(\frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1\right).$$

The spatial complexity is $S_{\mathrm{ISD}}(n,r,t) = \mathcal{O}(rn)$.

Following an argument similar to the one for Prange, we note that there is essentially no difficulty in interpreting Lee and Brickell's variant of the ISD as computing a slightly more complex Boolean function at each iteration, allowing to reformulate its complexity (Proposition 2) for the quantum case as follows.

Proposition 9 (Quantum computational complexity of Algorithm 2)**.** *Given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 2 requires an additional parameter $0 \leq p \leq t$.*

The time complexity of Algorithm 2 running on a quantum computer can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–14 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is

$$C_{\mathrm{ISD}}(n,r,t,p) = \sqrt{\frac{1}{\mathrm{Pr}_{\mathrm{succ}}}} c_{\mathrm{iter}} = \sqrt{\frac{\binom{n}{t}}{\binom{k}{p}\binom{r}{t-p}}} \left(C_{\mathrm{IS}}(n,r) + \binom{k}{p}(C_{\mathrm{IntToComb}} + pr)\right),$$

where $C_{\mathrm{IntToComb}} = \mathcal{O}((2k-p)(\log\binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the $\binom{k}{p}$ ones. The spatial complexity is $S_{\mathrm{ISD}}(r,n) = \mathcal{O}(rn)$.

Similarly, we also reformulate Leon's Algorithm 4 as follows.

Proposition 10 (Quantum computational complexity of Algorithm 4)**.** *Given H, an $r \times n$ binary parity-check matrix, s, an r-bit long syndrome (column vector) obtained through H, and the two parameters $0 \leq p \leq t$, $0 \leq \ell \leq (k-p)$, the complexity of finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 4 running on a quantum computer can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–10 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is $\mathrm{C}_{\mathrm{ISD}}(n,r,t,p,\ell) = \sqrt{\frac{1}{\mathrm{Pr}_{\mathrm{succ}}}}c_{\mathrm{iter}} = \sqrt{\frac{\binom{n}{t}}{\binom{k}{p}\binom{r-\ell}{t-p}}}\left(\mathrm{C}_{\mathrm{IS}}(n,r) + \binom{k}{p}\left(\mathrm{C}_{\mathrm{IntToComb}} + p\ell + \frac{\binom{k}{p}}{2^\ell}p(r-\ell)\right)\right)$, where $\mathrm{C}_{\mathrm{IntToComb}} = \mathcal{O}((2k-p)(\log\binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the $\binom{k}{p}$ ones. Note that, if the value of p is fixed, it is possible to avoid $\mathrm{C}_{\mathrm{IntToComb}}$, specializing the algorithm with a p-deep loop nest to generate the combinations. The spatial complexity is $S_{\mathrm{ISD}}(r,n) = \mathcal{O}(rn)$.*

Finally, we tackle the reformulation of Stern's ISD for the quantum case.

Proposition 11 (Quantum computational complexity of Algorithm 5)**.** *As for Algorithm 4, given H, an $r \times n$ binary parity-check matrix and s, an r-bit long syndrome (column vector) obtained through H, finding the row error vector e with length n and weight t such that $s = He^T$ with Algorithm 5 requires two additional parameters $0 \leq p \leq t$, $0 \leq \ell \leq (k-p)$.*

The time complexity of Algorithm 5 running on a quantum computer can be computed starting from the probability of success $\mathrm{Pr}_{\mathrm{succ}}$ of a single iteration of the loop at Lines 1–14 and the computational requirements of executing the loop body c_{iter}. In particular, the time complexity is

$$\mathrm{C}_{\mathrm{ISD}}(n,r,t,p,\ell) = \sqrt{\frac{1}{\mathrm{Pr}_{\mathrm{succ}}}}c_{\mathrm{iter}} = \sqrt{\frac{\binom{n}{t}}{\binom{k/2}{p/2}^2\binom{r-\ell}{t-p}}}\left(\mathrm{C}_{\mathrm{IS}}(n,r) + \binom{k/2}{p/2}\frac{p}{2}\ell + \binom{k/2}{p/2}(\mathrm{C}_{\mathrm{IntToComb}} + \frac{p}{2}\ell + \frac{\binom{k/2}{p/2}}{2^\ell}p(r-\ell))\right)$$

where $\mathrm{C}_{\mathrm{IntToComb}} = \mathcal{O}((2k-p)(\log\binom{k}{p})^2)$ is the cost of decoding an integer into its combinadics representation, i.e., finding the corresponding combination among all the $\binom{k/2}{p/2}$ ones. Note that, if the value of p is fixed, it is possible to avoid $\mathrm{C}_{\mathrm{IntToComb}}$, specializing the algorithm with a p-deep loop nest to generate the combinations. The spatial complexity is $S_{\mathrm{ISD}}(n,r,t,p,\ell) = \mathcal{O}\left(rn + \binom{k/2}{p/2}(\frac{p}{2}\log_2(\frac{k}{2}) + \ell)\right)$.

Since advanced ISD algorithms reduce the overall complexity by reducing the number of iterations at the cost of an increased complexity per iteration, the speedup due to Grover's algorithm is less evident for modern variants than for classical forms of ISD. Indeed, for all cases where the trade-off on a classical computer reduces by a factor α the number of iterations, at the cost of raising by α the cost of the iteration itself, we have that the trade-off turns out to be disadvantageous on a quantum computer where the speedup factor α is cut down to $\sqrt{\alpha}$ by Grover, while the single iteration slowdown stays the same. This was already observed in [41], where it was found that the quantum variant of Stern's algorithm does not achieve smaller work factors than the quantum variant of Lee and Brickell's algorithm. Indeed, in the same work, it is noted that the complexity of the quantum-computer variant of Stern's algorithm achieves a smaller complexity than the straightforward quantized version of the BJMM ISD. Finally, we note that the authors of [42] reported a re-elaboration of the MMT and BJMM ISDs for quantum computers which succeeds in effectively lowering their asymptotic complexities. We however do not take them into account in this work, as no finite regime complexity formulas are provided in [42] for the proposed algorithms. We report, in Table 1, a summary of all the computational complexities of the examined Information Set Decoding (ISD) algorithms, together with the parameters which should be optimized to achieve the best running time for each one of them.

Table 1. Overview of ISDs costs in terms of execution time complexities, considering a constant access memory cost model, expressed as: $\texttt{Cost} = \texttt{NumberOfIterations} \times \texttt{Cost}_{\texttt{single iteration}}$.

Name	No.of Iterations (i.e., $1/\Pr_{\text{succ}}$)	Parameters to Optimize
Prange [10]	$\frac{\binom{n}{t}}{\binom{r}{t}}$	none
Lee–Brickell [11]	$\frac{\binom{n}{t}}{\binom{k}{p}\binom{r}{t-p}}$	$0 \le p \le t$ ($p = 2$ is asymp. optimal)
Leon [12]	$\frac{\binom{n}{t}}{\binom{k}{p}\binom{r-\ell}{t-p}}$	$0 \le p \le t, 0 \le \ell \le r-(t-p)$
Stern [13]	$\frac{\binom{n}{t}}{\binom{k/2}{p/2}^2\binom{r-\ell}{t-p}}$	$0 \le p \le t, 0 \le \ell \le r-(t-p)$
Finiasz–Sendrier [14]	$\frac{\binom{n}{t}}{\binom{(k+\ell)/2}{p/2}^2\binom{r-\ell}{t-p}}$	$0 \le p \le t, 0 \le \ell \le r-(t-p)$
MMT [15]	$\frac{\binom{n}{t}}{\binom{k+\ell}{p}\binom{r-\ell}{t-p}}$,	$0 \le p \le t, \ell_1+\ell_2 = \ell, 0 \le \ell \le r-(t-p), \ell_2 \approx \log_2\binom{\frac{k+\ell}{2}}{p/4}$
BJMM [16]	$\frac{\binom{n}{t}}{\binom{k+\ell}{p}\binom{r-\ell}{t-p}}\left(\frac{\binom{\frac{k+\ell}{2}}{p_3}^2}{\binom{k+\ell}{p_2}}\right)^{-4}$	$\ell_2 \approx \log_2\left(\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\epsilon_2}\right)$, $\ell_1 \approx \log_2\left(\binom{p}{p/2}\binom{k+\ell-p}{\epsilon_1}\right)$, $0 \le p \le t$, $0 \le \ell \le r$, $0 \le \ell_2 \le \ell_1 \le \ell$, $0 \le \epsilon_1 \le t-p$, $0 \le \epsilon_2 \le t-p_1$, $p_3 = \frac{p_2}{2}$, $p_2 = \frac{p_1}{2}+\varepsilon_2$, $p_1 = \frac{p}{2}+\varepsilon_1$

Name	Single Iteration Cost
Prange [10]	$C_{\text{IS}}(n,r) + \mathcal{O}(n)$
Lee–Brickell [11]	$C_{\text{IS}}(n,r) + \binom{k}{p}(pr)$
Leon [12]	$C_{\text{IS}}(n,r) + \binom{k}{p}(pr)$
Stern [13]	$C_{\text{IS}}(n,r) + \binom{k}{p}\left(p\ell + \frac{\binom{\ell}{p}}{2^\ell}p(r-\ell)\right)$
Finasz-Sendrier [14]	$C_{\text{IS}}(n,r) + \binom{k/2}{p/2}\left(p\ell + \frac{\binom{k/2}{p/2}}{2^\ell}p(r-\ell)\right)$
MMT [15]	$\begin{cases} C_{\text{IS-FS}}(n,r,\ell) + \min\left(\binom{(k+\ell)/2}{p/4}, \frac{\binom{(k+\ell)/2}{p/2}}{\binom{p}{p/2}}\right)\cdot\left(\frac{p}{4}\ell_2 + \frac{\binom{(k+\ell)/2}{p/4}}{2^{\ell_2}}\frac{p}{2}\ell_1\right) \\ +\binom{(k+\ell)/2}{p/4}\frac{p}{2}\ell_2 + \binom{(k+\ell)/2}{p/4}\left(\frac{p}{4}\ell_2 + \frac{\binom{(k+\ell)/2}{p/4}}{2^{\ell_2}}\frac{p}{2}\ell_1 + \frac{\binom{(k+\ell)/2}{p/4}\binom{k+\ell}{p/2}}{2^{\ell_1+\ell_2}\binom{p}{p/2}}p(r-\ell)\right) \end{cases}$
BJMM [16]	$\begin{cases} C_{\text{IS-FS}}(n,r,\ell) \\ +4\left(k+\ell+2\binom{\frac{k+\ell}{2}}{p_3}+\ell_2+\binom{\frac{k+\ell}{2}}{p_3}^2(2p_3\ell_2)\right)+2\left(\left(\frac{\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}}{2^{\ell_2}}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2(2p_2\ell_1)\right) \\ +\left(\frac{1}{2^{\ell_1}}\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon_1}\left(\frac{1}{2^{\ell_2}}\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2\right)^2(2p_1\ell) \\ +\frac{1}{2^\ell}\left(\frac{1}{2^{\ell_1}}\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon_1}\left(\frac{1}{2^{\ell_2}}\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\varepsilon_2}\binom{\frac{k+\ell}{2}}{p_3}^2\right)^2\right)^2(p(r-\ell)) \end{cases}$

4. Quantitative Assessment of ISD Complexities

In this section, we analyze the computational complexities to solve the Syndrome Decoding Problem (SDP) and the Codeword Finding Problem (CFP) for sets of code parameters relevant to post quantum cryptosystems. To this end, we select the proposals which were admitted to the second round of the NIST post quantum cryptography standardization effort [24] relying on a Niederreiter cryptosystem, or its McEliece variant.

In particular, we consider Classic McEliece [43] and NTS-KEM [44], which employ Goppa codes, and BIKE [34] and LEDAcrypt [31], which employ quasi cyclic codes, to assess our finite domain approach on both quasi-cyclic codes and non-quasi-cyclic codes. The parameters for the reported cryptosystems are designed to match the computation effort required to break them to the one required to break AES-128 (Category 1), AES-192 (Category-3), and AES-256 (Category 5). We report the code length n, code dimension k, number of errors to be corrected t and size of the circulant block p for

all the aforementioned candidates in Table 2. Furthermore, for the cryptosystems relying on low- or moderate-density parity check codes, we also report the weight of the codeword to be found, w, in the case of a Codeword Finding Problem (CFP).

We implemented all the complexity formulas from Section 3 employing Victor Shoup's NTL library, representing the intermediate values either as arbitrary precision integers, or as `NTL::RR` selectable precision floating point numbers. We chose to employ a 128 bit mantissa and the default 64 bit exponent for the selectable precision.

To minimize the error in computing a large amount of binomial coefficients, while retaining acceptable performance, we precompute the exact values for all the $\binom{n}{k}$ binomials for all pairs n, k up to $\binom{3{,}000}{200}$. Furthermore, to minimize the error of Stirling's approximation whenever $n \gg k$ we also precompute all the exact values for the binomials up to $\binom{10{,}000}{10}$, and compute the exact value whenever $k < 10$. To provide a fast approximated computation for all the binomials which do not fall in the aforementioned intervals, we compute the binomials employing the logarithm of Stirling's series approximated to the fourth term.

Table 2. Summary of the code parameters for the second round submissions: code length n, code redundancy $n - k = r$, and number of errors expected to be corrected t.

Category	Cipher	Variant	n	k	t	w	p
1	Classic McEliece		3488	2720	64	-	1
	NTS-KEM		4096	3328	64	-	1
	BIKE-2	KEM-CPA	20,326	10,163	134	142	10,163
		KEM-CCA	23,558	11,779	134	142	11,779
	LEDAcrypt	KEM, $n_0 = 2$	29,878	14,939	136	154	14,939
		KEM, $n_0 = 3$	24,807	16,538	86	243	8269
		KEM, $n_0 = 4$	30,188	22,641	69	364	7547
		KEM-LT, $n_0 = 2$	71,798	35,899	136	154	35,899
		KEM-XT, $n_0 = 2$	104,294	52,147	136	154	52,147
3	Classic McEliece		4608	3360	96	-	1
	NTS-KEM		8192	7152	80	-	1
	BIKE-2	KEM-CPA	39,706	19,583	199	206	19,583
		KEM-CCA	49,642	24,821	199	206	24,821
	LEDAcrypt	KEM, $n_0 = 2$	51,386	25,693	199	210	25,693
		KEM, $n_0 = 3$	48,201	32,134	127	363	16,067
		KEM, $n_0 = 4$	57,364	43,023	101	540	14,341
		KEM-LT, $n_0 = 2$	115,798	57,899	199	210	57,899
		KEM-XT, $n_0 = 2$	192,442	96,221	199	210	96,221
5	Classic McEliece	$n = 6688$	6688	5024	128	-	1
		$n = 6960$	6960	5413	119	-	1
		$n = 8192$	8192	6528	128	-	1
	NTS-KEM		8192	6424	136	-	1
	BIKE-2	KEM-CPA	65,498	32,749	264	274	32,749
		KEM-CCA	81,194	40,597	264	274	40,597
	LEDAcrypt	KEM, $n_0 = 2$	73,754	36,877	267	286	36,877
		KEM, $n_0 = 3$	82,311	54,874	169	495	27,437
		KEM, $n_0 = 4$	90,764	68,073	134	676	22,691
		KEM-LT, $n_0 = 2$	178,102	89,051	267	286	89,051
		KEM-XT, $n_0 = 2$	304,534	152,267	267	286	152,267

We explored the parameter space of each algorithm considering the fact that the different parameters drive different trade-off points in each algorithm. To this end, we explored an appropriately sized region of the parameter space, which we report in Table 3. To determine the explored region, we started from a reasonable choice and enlarged the region until the value of the parameters minimizing the attack complexity was no longer on the region edge for all the involved parameters. We employed, for the choice of the ℓ_2 parameter in the MMT ISD variant and the ℓ_1 and ℓ_2 parameters in the BJMM variant, the choices which were advised in the respective works.

Table 3. Explored parameter range for the different ISD variants.

ISD Variant	Parameter Range
Prange [10]	none
Lee–Brickell [11]	$1 \leq p \leq 12$ ($p = 2$ is asymp. optimal)
Leon [12]	$1 \leq p \leq 12, 0 \leq \ell \leq \min(100, r - (t - p))$
Stern [13]	$2 \leq p \leq 18, 0 \leq \ell \leq \min(100, r - (t - p))$
Finasz-Sendrier [14]	$2 \leq p \leq 18, 0 \leq \ell \leq \min(100, r - (t - p))$
MMT [15]	$4 \leq p \leq 34,\ 110 \leq \ell \leq \min(350, r - (t - p))$ $\ell_1 + \ell_2 = \ell, \ell_2 = \lfloor \log_2 \binom{\frac{k+\ell}{2}}{p/4} \rceil$
BJMM [16]	$10 \leq p \leq 24, 90 \leq \ell \leq \min(330, r - (t - p))$, $\ell_2 = \log_2 \lfloor (\binom{p_1}{p_1/2}\binom{k+\ell-p_1}{\epsilon_2}) \rceil, \ell_1 = \log_2 \lfloor (\binom{p}{p/2}\binom{k+\ell-p}{\epsilon_1}) \rceil$, $0 \leq \epsilon_1 \leq 4,\ 0 \leq \epsilon_2 \leq 4$,

We took into account the advantage provided by a quasi cyclic code in both the Syndrome Decoding Problem (SDP) and the Codeword Finding Problem (CFP) solution complexity, reducing it by a factor equal to $\sqrt{p}$, the square root of the cyclic block size for the Syndrome Decoding Problem (SDP), and p for the Codeword Finding Problem (CFP), in accordance with the point raised in Section 3.9.

Table 4 reports the computational costs of solving both Syndrome Decoding Problem (SDP) and Codeword Finding Problem (CFP) by means of the described variants of the Information Set Decoding (ISD). In addition to the computational complexities obtained, the value of a simple asymptotic cost criterion for the Information Set Decoding (ISD)s, described in [34], is reported. Such a criterion states that the asymptotic complexity of an ISD is $2^{-\log_2(1-\frac{k}{n})t}$, for the case of the use in solving a Syndrome Decoding Problem (SDP). A noteworthy point to observe is that, considering the finite regime value of the complexities, the May–Meurer–Thomae algorithm attains a lower time complexity than the Becker–Joux–May–Meurer algorithm in most cases. Indeed, while the Becker–Joux–May–Meurer Information Set Decoding (ISD) variant has a lower asymptotic cost, considering a worst-case-scenario for the solution of the Syndrome Decoding Problem (SDP), i.e., code rate close to 0.5, and a large enough value for n, a finite regime estimate of its cost reports that employing the May–Meurer–Thomae approach should result in a faster computation. Concerning the space complexities of the approaches with exponential (in the code length n) space requirements, we report the obtained values in Table 5. We note that the Information Set Decoding (ISD) variants proposed by Stern and Finiasz and Sendrier have an overall lower memory consumption that their more advanced counterparts. In particular, the space complexities of the aforementioned variants start as low as 16 Gib for Category 1 parameters, and are thus amenable to an implementation which keeps the entire lists in main memory on a modern desktop. By contrast, the May–Meurer–Thomae and Becker–Joux–May–Meurer Information Set Decoding (ISD) variants require a significantly higher amount of memory, with the latter being less demanding than the former in all cases but the one of LEDAcrypt in its $n_0 = 2$ parameterization for extremely long term keys (LEDAcrypt-XT). In all cases, the space complexities of May–Meurer–Thomae and Becker–Joux–May–Meurer exceed 2^{50}, pointing strongly to the need of a significant amount of mass storage to implement a practical attack. Such a requirement is even more stringent in the case of higher security levels, where the memory requirements exceed 2^{100} for most parameter choices.

Table 4. Computational costs expressed as $\log_2(\texttt{bit_operations})$, for each ISDs applied to NIST `round-2` code-based cryptosystems. Information Set Decoding (ISD) variants are labeled as: Prange (Pr), Lee and Brickell (LB), Leon (Le), Stern (St), Finiasz and Sendrier (FS), May–Meurer–Thomae (MMT), and Becker–Joux–May–Meurer (BJMM); quantum accelerated variants prefixed by "Q-". The quantum complexities are expressed in $\log_2(\texttt{bit_operations})$ for the corresponding non-reversible circuit, and thus provide a lower bound on the actual quantum circuit complexity. The table also reports a simple approximation for the asymptotic cost of classical ISDs, $-\log_2(1-\frac{k}{n})t$, computed for all parameters

Category 1—Security Level Equivalent to Break AES-128 on a Classical Machine												
Problem	**Cipher**	**Variant**	**Pr**	**LB**	**Le**	**St**	**FS**	**MMT**	**BJMM**	**Q-LB**	**Q-St**	$-\log_2(1-\frac{k}{n})t$
Codeword Finding Problem (CFP)	BIKE-2	CPA	167.28	156.31	154.66	146.56	146.54	123.92	151.01	92.28	93.696	142.00
		CCA	167.61	156.65	154.91	146.84	146.82	124.29	151.38	92.66	94.074	142.00
	LEDAcrypt	$n_0 = 2$	180.25	169.06	167.24	158.76	158.75	135.21	163.79	99.21	100.62	154.00
		$n_0 = 3$	169.46	157.89	156.60	147.79	147.76	126.45	151.12	94.06	95.469	142.15
		$n_0 = 4$	179.86	167.79	165.69	157.13	157.10	135.17	160.35	99.71	101.12	151.07
		LT, $n_0 = 2$	182.44	171.27	169.16	160.88	160.88	137.96	169.02	101.5	102.98	154.00
		XT, $n_0 = 2$	183.44	172.28	170.09	161.91	161.91	138.90	172.14	102.6	104.02	154.00
Syndrome Decoding Problem (SDP)	Classic McEliece		170.82	160.38	160.43	152.51	152.46	118.61	149.91	96.41	97.15	139.73
	NTS-KEM		186.03	175.33	175.33	166.76	166.72	127.52	162.54	104.11	104.81	154.56
	BIKE-2	CPA	165.86	155.06	153.37	145.60	145.58	123.78	150.36	94.98	96.39	134.00
		CCA	166.30	155.51	153.74	146.00	145.99	124.28	150.86	95.48	96.88	134.00
	LEDAcrypt	$n_0 = 2$	169.05	158.23	156.35	148.59	148.57	126.83	154.30	97.26	98.67	136.00
		$n_0 = 3$	167.89	157.53	154.65	147.82	147.81	123.66	152.31	96.31	97.72	136.31
		$n_0 = 4$	169.62	159.40	155.86	149.32	149.31	123.17	155.49	97.47	98.22	138.00
		LT, $n_0 = 2$	171.95	161.15	159.00	151.46	151.45	129.63	160.32	100.30	101.70	136.00
		XT, $n_0 = 2$	173.24	162.44	160.23	152.78	152.78	130.88	163.75	101.62	103.02	136.00
Category 3—Security Level Equivalent to Break AES-192 on a Classical Machine												
Problem	**Cipher**	**Variant**	**Pr**	**LB**	**Le**	**St**	**FS**	**MMT**	**BJMM**	**Q-LB**	**Q-St**	$-\log_2(1-\frac{k}{n})t$
Codeword Finding Problem (CFP)	BIKE-2	CPA	229.29	217.29	215.52	205.50	205.49	176.73	210.18	123.7	125.17	201.99
		CCA	233.76	221.73	219.83	209.79	209.78	181.12	215.22	126.2	127.68	206.00
	LEDAcrypt	$n_0 = 2$	237.86	225.78	223.87	213.72	213.72	184.67	219.33	128.3	129.76	210.00
		$n_0 = 3$	241.70	228.98	227.03	216.59	216.57	188.66	220.93	130.5	131.96	212.34
		$n_0 = 4$	254.92	241.73	238.97	228.80	228.78	200.04	232.76	137.6	139.01	224.12
		LT, $n_0 = 2$	239.86	227.79	225.65	215.68	215.68	188.81	224.99	130.5	131.93	210.00
		XT, $n_0 = 2$	241.21	229.15	226.93	217.08	217.07	190.88	227.63	131.9	133.34	210.00

Table 4. *Cont.*

Problem	Cipher	Variant	Pr	LB	Le	St	FS	MMT	BJMM	Q-LB	Q-St	$-\log_2(1-\frac{k}{n})t$
Category 3—Security Level Equivalent to Break AES-192 on a Classical Machine												
Syndrome Decoding Problem (SDP)	Classic McEliece		214.70	203.45	203.36	194.36	194.32	155.09	190.53	118.71	120.18	180.91
	NTS-KEM		272.34	260.40	259.95	248.01	247.91	190.18	240.63	147.85	148.46	238.21
	BIKE-2	CPA	229.50	217.61	215.82	205.99	205.98	178.12	210.87	127.49	128.89	195.12
		CCA	234.01	222.09	220.17	210.33	210.32	182.57	215.95	130.11	131.51	199.00
	LEDAcrypt	$n_0 = 2$	234.12	222.20	220.26	210.43	210.42	182.79	216.33	130.23	131.63	199.00
		$n_0 = 3$	235.32	223.84	220.82	211.91	211.90	180.13	217.56	130.67	132.07	201.29
		$n_0 = 4$	235.98	224.66	220.97	212.39	212.39	177.94	218.56	131.26	132.66	202.00
		LT, $n_0 = 2$	236.74	224.83	222.67	213.02	213.02	187.02	222.64	133.01	134.41	199.00
		XT, $n_0 = 2$	238.47	226.57	224.33	214.80	214.80	189.40	225.67	134.79	136.19	199.00
Category 5—Security Level Equivalent to Break AES-256 on a Classical Machine												
Problem	**Cipher**	**Variant**	**Pr**	**LB**	**Le**	**St**	**FS**	**MMT**	**BJMM**	**Q-LB**	**Q-St**	$-\log_2(1-\frac{k}{n})t$
Codeword Finding Problem (CFP)	BIKE-2	CPA	302.77	289.92	288.03	276.41	276.40	239.96	281.57	160.7	162.18	274.00
		CCA	303.23	290.38	288.41	276.83	276.83	241.65	282.72	161.3	162.72	274.00
	LEDAcrypt	$n_0 = 2$	315.08	302.11	300.19	288.35	288.34	250.97	293.22	167.0	168.44	286.00
		$n_0 = 3$	320.55	306.93	304.48	292.78	292.77	257.64	297.70	170.3	171.71	289.56
		$n_0 = 4$	312.68	298.84	295.66	284.59	284.58	250.96	289.06	166.8	168.22	280.57
		LT, $n_0 = 2$	317.16	304.20	302.03	290.38	290.37	257.62	299.52	169.3	170.76	286.00
		XT, $n_0 = 2$	318.57	305.61	303.37	291.83	291.83	261.01	302.47	170.8	172.24	286.00
Syndrome Decoding Problem (SDP)	Classic McEliece	$n = 6688$	293.55	281.32	281.17	270.46	270.42	219.80	263.74	158.38	159.84	256.89
	Classic McEliece	$n = 6960$	294.49	282.29	282.05	271.18	271.13	218.79	264.39	158.87	160.33	258.18
	Classic McEliece	$n = 8192$	331.64	319.08	318.77	306.63	306.54	249.74	299.89	177.55	179.01	294.34
	NTS-KEM		338.58	325.94	325.68	313.52	313.44	256.88	306.72	181.02	182.48	300.85
	BIKE-2	CPA	300.21	287.47	285.56	274.15	274.15	239.12	279.52	163.30	164.70	264.00
		CCA	300.83	288.10	286.11	274.75	274.74	240.81	280.84	164.00	165.40	264.00
	LEDAcrypt	$n_0 = 2$	303.56	290.79	288.84	277.40	277.39	242.63	282.64	165.18	166.58	267.00
		$n_0 = 3$	303.84	291.53	288.42	277.98	277.98	238.73	284.34	165.48	166.88	267.86
		$n_0 = 4$	303.68	291.54	287.78	277.67	277.67	234.66	282.96	165.52	166.92	268.00
		LT, $n_0 = 2$	306.33	293.58	291.40	280.14	280.14	249.21	289.67	168.16	169.55	267.00
		XT, $n_0 = 2$	308.15	295.39	293.14	282.00	282.00	252.61	293.04	170.03	171.43	267.00

Table 5. Memory costs expressed as $\log_2(\texttt{memory_size_in_bits})$, for each list-based ISDs (i.e., St [13], FS [14], MMT [15], and BJMM [16]) applied to NIST `round-2` code-based cryptosystems. Information Set Decoding (ISD) variants are labeled as: Stern (ST), Finiasz and Sendrier (FS), May–Meurer–Thomae (MMT), and Becker–Joux–May–Meurer (BJMM)

Category 1—Security Level Equivalent to Break AES-128 on a Classical Machine						
Problem	**Cipher**	**Variant**	**St**	**FS**	**MMT**	**BJMM**
Codeword Finding Problem (CFP)	BIKE-2	CPA	40.72	40.74	53.47	52.59
		CCA	41.38	41.40	54.34	53.08
	LEDAcrypt	$n_0 = 2$	42.46	42.48	55.74	55.48
		$n_0 = 3$	39.81	39.83	52.26	50.51
		$n_0 = 4$	39.40	39.43	51.73	50.59
		LT, $n_0 = 2$	46.40	46.40	48.41	61.44
		XT, $n_0 = 2$	48.06	48.07	50.08	64.26
Syndrome Decoding Problem (SDP)	Classic McEliece		34.68	34.76	78.03	68.19
	NTS-KEM		35.60	35.65	80.29	70.36
	BIKE-2	CPA	40.72	40.74	53.47	52.59
		CCA	41.38	41.40	54.34	53.08
	LEDAcrypt	$n_0 = 2$	42.46	42.48	55.74	55.48
		$n_0 = 3$	42.89	42.91	56.33	55.51
		$n_0 = 4$	44.31	44.32	69.63	58.72
		LT, $n_0 = 2$	46.40	46.40	48.41	61.44
		XT, $n_0 = 2$	48.06	48.07	50.08	64.26
Category 3—Security Level Equivalent to Break AES-192 on a Classical Machine						
Problem	**Cipher**	**Variant**	**St**	**FS**	**MMT**	**BJMM**
Codeword Finding Problem (CFP)	BIKE-2	CPA	43.67	43.68	79.51	57.23
		CCA	44.74	44.75	81.58	58.66
	LEDAcrypt	$n_0 = 2$	44.90	44.90	81.89	59.01
		$n_0 = 3$	42.80	42.81	77.78	56.22
		$n_0 = 4$	42.30	42.31	76.79	55.07
		LT, $n_0 = 2$	48.54	48.54	76.54	65.32
		XT, $n_0 = 2$	50.81	50.81	66.74	68.52
Syndrome Decoding Problem (SDP)	Classic McEliece		35.66	35.71	80.41	70.51
	NTS-KEM		77.63	77.74	88.99	80.42
	BIKE-2	CPA	43.67	43.68	79.51	57.23
		CCA	44.74	44.75	81.58	58.66
	LEDAcrypt	$n_0 = 2$	44.90	44.90	81.89	59.01
		$n_0 = 3$	45.88	45.89	83.84	61.30
		$n_0 = 4$	47.18	47.18	98.20	63.29
		LT, $n_0 = 2$	48.54	48.54	63.74	65.32
		XT, $n_0 = 2$	50.81	50.81	66.74	68.52
Category 5—Security Level Equivalent to Break AES-256 on a Classical Machine						
Problem	**Cipher**	**Variant**	**St**	**FS**	**MMT**	**BJMM**
Codeword Finding Problem (CFP)	BIKE-2	CPA	45.98	45.98	106.61	61.50
		CCA	46.95	46.96	109.11	62.69
	LEDAcrypt	$n_0 = 2$	46.51	46.52	108.00	61.71
		$n_0 = 3$	45.20	45.21	104.56	59.68
		$n_0 = 4$	44.35	44.36	102.36	58.74
		LT, $n_0 = 2$	50.46	50.46	92.80	68.73
		XT, $n_0 = 2$	52.86	52.86	83.64	72.22
Syndrome Decoding Problem (SDP)	Classic McEliece	$n = 6688$	37.48	47.18	84.97	75.83
	Classic McEliece	$n = 6960$	47.58	57.02	85.81	77.02
	Classic McEliece	$n = 8192$	67.64	76.82	87.95	79.97
	NTS-KEM		67.50	67.59	87.77	79.72
	BIKE-2	CPA	45.98	45.98	106.62	61.50
		CCA	46.95	46.96	97.62	62.69
	LEDAcrypt	$n_0 = 2$	46.51	46.52	108.00	61.71
		$n_0 = 3$	48.27	48.28	112.62	64.77
		$n_0 = 4$	49.23	49.24	115.13	65.98
		LT, $n_0 = 2$	50.46	50.46	92.80	68.73
		XT, $n_0 = 2$	52.86	52.86	83.64	72.22

5. Conclusions

In this work, we survey the current approaches to solve efficiently the Information Set Decoding (ISD) problem, providing a complete procedural description of the algorithms at hand. We provide finite regime expressions to estimate both the computational demand and the space requirements of the different Information Set Decoding (ISD) alternatives. To provide insights on the effectiveness of asymptotic estimates for the Information Set Decoding (ISD) complexities, we evaluate them on a set of parameters chosen by code-based cryptosystems submitted to the current NIST standardization effort for post quantum cryptography. Our results show that the May–Meurer–Thomae variant of the Information Set Decoding (ISD) may be preferable in the case an actual attack is implemented against the cryptosystems at hand. Moreover, we also report that the simple approximation provided in [34] for the asymptotic cost of an Information Set Decoding (ISD) appears to be overestimating the complexity of the attack in the finite regime, by a small but significant amount. In light of the obtained results, we deem a practical evaluation of the complexities of the described algorithms on reduced security instances of the cryptosystems at hand an interesting topic for further investigation.

Author Contributions: Conceptualization, M.B., A.B., F.C., G.P. and P.S.; Writing – original draft, M.B., A.B., F.C., G.P. and P.S.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berlekamp, E.R.; McEliece, R.J.; van Tilborg, H.C.A. On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theory* **1978**, *24*, 384–386. [CrossRef]
2. Goppa, V.D. A new class of linear correcting codes. *Probl. Pered. Inf.* **1970**, *6*, 24–30.
3. Niederreiter, H. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Contr. Inf. Theory* **1986**, *15*, 159–166.
4. Sidel'nikov, V.M.; Shestakov, S. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discret. Math. Appl.* **1978**, *2*, 439–444. [CrossRef]
5. Gaborit, P. Shorter Keys for Code Based Cryptography. In Proceedings of the International Workshop on Coding and Cryptography 2015, Bergen, Norway, 14–18 March 2005; pp. 81–90.
6. Monico, C.; Rosenthal, J.; Shokrollahi, A. Using Low Density Parity Check Codes in the McEliece Cryptosystem. In Proceedings of the IEEE International Symposium on Information Theory (ISIT 2000), Sorrento, Italy, 25–30 June 2000; p. 215.
7. Misoczki, R.; Barreto, P.S.L.M. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5867, pp. 376–392.
8. Baldi, M.; Bodrato, M.; Chiaraluce, F. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Security and Cryptography for Networks*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5229, pp. 246–262.
9. Misoczki, R.; Tillich, J.P.; Sendrier, N.; Barreto, P.S.L.M. MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes. In Proceedings of the IEEE International Symposium on Information Theory (ISIT 2013), Istanbul, Turkey, 7–12 July 2013; pp. 2069–2073.
10. Prange, E. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory* **1962**, *8*, 5–9. [CrossRef]
11. Lee, P.J.; Brickell, E.F. An Observation on the Security of McEliece's Public-Key Cryptosystem. In Proceedings of the Advances in Cryptology—EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, 25–27 May 1988; pp. 275–280.
12. Leon, J.S. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Inf. Theory* **1988**, *34*, 1354–1359. [CrossRef]
13. Stern, J. A Method for Finding Codewords of Small Weight. In Proceedings of the Coding Theory and Applications, 3rd International Colloquium, Toulon, France, 2–4 November 1988; pp. 106–113.

14. Finiasz, M.; Sendrier, N. Security Bounds for the Design of Code-Based Cryptosystems. In Proceedings of the Advances in Cryptology—ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, 6–10 December 2009; pp. 88–105.
15. May, A.; Meurer, A.; Thomae, E. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Proceedings of the Advances in Cryptology—ASIACRYPT 2011—17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul South Korea, 4–8 December 2011; pp. 107–124.
16. Becker, A.; Joux, A.; May, A.; Meurer, A. Decoding Random Binary Linear Codes in $2^n/20$: How $1 + 1 = 0$ Improves Information Set Decoding. In Proceedings of the Advances in Cryptology—EUROCRYPT 2012—31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012; pp. 520–536.
17. Coffey, J.T.; Goodman, R.M. The complexity of information set decoding. *IEEE Trans. Inf. Theory* **1990**, *36*, 1031–1037. [CrossRef]
18. Kruk, E. Decoding complexity bound for linear block codes. *Probl. Peredachi Inf.* **1989**, *25*, 103–107.
19. Barg, A.; Kruk, E.; van Tilborg, H. On the complexity of minimum distance decoding of long linear codes. *IEEE Trans. Inf. Theory* **1999**, *45*, 1392–1405. [CrossRef]
20. Bassalygo, L.; Zyablov, V.; Pinsker, M. Problems of complexity in the theory of correcting codes. *Probl. Peredachi Inf.* **1977**, *13*, 5–17.
21. Barg, A. Complexity Issues in Coding Theory. *Electronic Colloquium on Computational Complexity (ECCC) - Reports Series 1997*. 1997, TR97-096. Available online: https://eccc.weizmann.ac.il/eccc-reports/1997/TR97-046/Paper.pdf (accessed on 27 September 2019).
22. Pless, V.S. (Ed.) *Handbook of Coding Theory*; Elsevier Science Inc.: New York, NY, USA, 1998.
23. Kabatiansky, G.; Krouk, E.; Semenov, S. *Error Correcting Coding and Security for Data Networks: Analysis of the Superchannel Concept*; Wiley Inc.: New York, NY, USA, 2005.
24. USA National Institute of Standards and Technology. *Post-Quantum Crypto Project*; USA National Institute of Standards and Technology: Gaithersburg, MA, USA, 2016.
25. Hamdaoui, Y.; Sendrier, N. A non asymptotic analysis of information set decoding. *IACR Cryptol. EPrint Arch.* **2013**, *2013*, 162.
26. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Santini, P. LEDAtools. 2019. Available online: https://github.com/LEDAcrypt/LEDAtools (accessed on 27 September 2019).
27. Arora, S.; Barak, B. *Computational Complexity—A Modern Approach*; Cambridge University Press: Cambridge, UK, 2009.
28. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Rosenthal, J.; Santini, P.; Schipani, D. Design and implementation of a digital signature scheme based on low-density generator matrix codes. *arXiv* **2018**, arXiv:1807.06127.
29. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Santini, P. LEDAkem: A post-quantum key encapsulation mechanism based on QC-LDPC codes. *arXiv* **2018**, arXiv:1801.08867.
30. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Santini, P. LEDAkem: A Post-Quantum Key Encapsulation Mechanism Based on QC-LDPC Codes. In Proceedings of the Post-Quantum Cryptography—9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, 9–11 April 2018; Volume 10786, pp. 3–24.
31. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Santini, P. LEDAcrypt: QC-LDPC Code-Based Cryptosystems with Bounded Decryption Failure Rate. In Proceedings of the Code-Based Cryptography, 7th International Workshop, CBC 2019, Darmstadt, Germany, 18–19 May 2019.
32. McEliece, R.J. *A Public-Key Cryptosystem Based on Algebraic Coding Theory*; DSN Progress Report; 1978; pp. 114–116. Available online: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19780016269.pdf (accessed on 27 September 2019)
33. Baldi, M.; Barenghi, A.; Chiaraluce, F.; Pelosi, G.; Santini, P. LEDAcrypt Website. 2019. Available online: https://www.ledacrypt.org/ (accessed on 27 September 2019).
34. Aragon, N.; Barreto, P.S.L.M.; Bettaieb, S.; Bidoux, L.; Blazy, O.; Deneuville, J.C.; Gaborit, P.; Gueron, S.; Guneysu, T.; Aguilar Melchor, C.; et al. BIKE: Bit Flipping Key Encapsulation, 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms. Available online: https://bikesuite.org/ (accessed on 27 September 2019).

35. MacKay, D.J.C. *Information Theory, Inference and Learning Algorithms*, 1st ed.; Cambridge University Press: Cambridge, UK, 2003.
36. Kirchner, P. Improved generalized birthday attack. *IACR Cryptol. EPrint Arch.* **2011**, *2011*, 377.
37. Niebuhr, R.; Cayrel, P.L.; Buchmann, J. Improving the Efficiency of Generalized Birthday Attacks against Certain Structured Cryptosystems. In Proceedings of the Workshop on Coding And Cryptography (WCC 2011), Paris, France, 11–15 April 2011; pp. 163–172.
38. Sendrier, N. Decoding One Out of Many. In Proceedings of the Post-Quantum Cryptography—4th International Workshop, PQCrypto 2011, Taipei, Taiwan, 29 November–2 December 2011; pp. 51–67.
39. Grover, L.K. A Fast Quantum Mechanical Algorithm for Database Search. In Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, Philadephia, PA, USA, 22–24 May 1996; pp. 212–219.
40. Bernstein, D.J. Grover vs. McEliece. In *Post-Quantum Cryptography*; Sendrier, N., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 73–80.
41. de Vries, S. Achieving 128-Bit Security Against Quantum Attacks in OpenVPN. Master's Thesis, University of Twente, Enschede, The Netherlands, 2016.
42. Kachigar, G.; Tillich, J. Quantum Information Set Decoding Algorithms. In Proceedings of the Post-Quantum Cryptography—8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, 26–28 June 2017; Volume 10346; pp. 69–89.
43. Bernstein, D.J.; Chou, T.; Lange, T.; Maurich, I.V.; Misoczki, R.; Niederhagen, R.; Persichetti, E.; Peters, C.; Schwabe, P.; Sendrier, N.; et al. Classic McEliece: Conservative code-based cryptography, 2019. NIST Post-Quantum Cryptography Project: Second Round Candidate Algorithms. Available online: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions (accessed on 27 September 2019).
44. Albrecht, M.; Cid, C.; Paterson, K.G.; Tjhai, C.J.; Tomlinson, M. NTS-KEM, 2019. NIST Post-Quantum Cryptography Project: Second Round Candidate Algorithms. Available online: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions (accessed on 27 September 2019).

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).