# Computational Hardness of the Permuted Kernel and Subcode Equivalence Problems*

Paolo Santini, Marco Baldi, and Franco Chiaraluce

Polytechnic University of Marche, Ancona, Italy
{p.santini, m.baldi, f.chiaraluce}@univpm.it

**Abstract.** The Permuted Kernel Problem (PKP) asks to find a permutation which makes an input matrix an element of the kernel of some given vector space. The literature exhibits several works studying its hardness in the case of the input matrix being mono-dimensional (i.e., a vector), while the multi-dimensional case has received much less attention and, de facto, only the case of a binary ambient finite field has been studied. The Subcode Equivalence Problem (SEP), instead, asks to find a permutation so that a given linear code becomes a subcode of another given code. At the best of our knowledge, no algorithm to solve the SEP has ever been proposed. In this paper we study the computational hardness of solving these problems. We first show that, despite going by different names, PKP and SEP are exactly the same problem. Then we consider the state-of-the-art solver for the mono-dimensional PKP (namely, the KMP algorithm), generalize it to the multi-dimensional case and analyze both the finite and the asymptotic regimes. We further propose a new algorithm, which can be thought of as a refinement of KMP. In the asymptotic regime our algorithm becomes slower than existing solutions but, in the finite regime (and for parameters of practical interest), it achieves competitive performances. As an evidence, we show that it is the fastest algorithm to attack several recommended instances of cryptosystems based on PKP. As a side-effect, given the already mentioned equivalence between PKP and SEP, all the algorithms we analyze in this paper can be used to solve instances of this latter problem.

## 1 Introduction

The Permuted Kernel Problem (PKP) is an NP-complete problem [12] which, in its more general formulation, asks to find a permutation of a given $\ell \times n$ matrix $\mathbf{V}$ which belongs to the right kernel of another, given, $m \times n$ matrix $\mathbf{A}$. The most studied case for the PKP is the one in which the input matrix $\mathbf{V}$ has only one row (i.e., $\ell = 1$), hence it is a vector; we will refer to this case as the mono-dimensional PKP. The use of the mono-dimensional PKP in cryptography has been introduced by Shamir in 1989 [26], and recently the problem has been employed to build post-quantum signature schemes [6, 7, 11]. The security of the mono-dimensional variant of PKP has been extensively studied over time [3, 13, 16, 17, 20, 22, 23]. Prior to [23], the solver with the lowest time complexity was the Koussa-Macario-Rat-Patarin (KMP) algorithm [17], which consists in a brute-force search plus standard optimizations, such as reducing to a small instance and employing a meet-in-the-middle approach. In the recent work [23], we have proposed an improvement of the KMP algorithm, describing an approach based on kernel subspaces with small support, i.e., containing vectors that have always zero entries in several positions[1]. These subspaces correspond to kernel equations involving an abnormally small number of coordinates and, consequently, are somewhat easier to solve. Such equations can be used to implement an initial filtering stage, after which the KMP algorithm is executed: this allows excluding some of the candidates which the KMP algorithm would consider and, in several cases, leads to a slight but nontrivial speed-up.

Perhaps surprisingly, the multi-dimensional PKP (i.e., the case in which the input $\mathbf{V}$ has $\ell > 1$ rows) has received much less attention; formally introduced in [18], it has been employed to generalize schemes based on the mono-dimensional version. Note that, in this work, the only recommended instances are for the the binary case (i.e., the ambient finite field is binary). These instances have been cryptanalyzed in [19], using a coding theory approach: roughly, the attack

* Part of the material in this paper has been presented at the 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland [23].

[1] Note that a similar idea was already briefly mentioned in [17], but the authors concluded that finding kernel equations with the desired properties was not feasible.

first searches for low-weight codewords and then uses them to efficiently reconstruct the target permutation. Note that the algorithm in [19] is specific for the binary case.

Another interesting and quite recent problem is the Subcode Equivalence Problem (SEP). On input two linear codes, the SEP asks to find a permutation mapping one into a subcode of the other, and has been proven NP-complete in [4]. There already exist some cryptographic proposals based on SEP: for instance, [14] and the quasi-dyadic specialization in [8]. At the best of our knowledge, currently there is no proposed algorithm to solve SEP; hence, the practical hardness of this problem is currently unknown.

*Our contribution* As we have already highlighted, there exist some connections between the PKP and problems from coding theory. Indeed, both [19] and [23] solve PKP employing coding concepts and algorithms. In this paper, we provide some further contributions along this line of research. We first show that PKP and SEP are actually the very same problem: this somehow justifies the fact that, to solve the PKP, one can borrow concepts from coding theory. As a consequence, all the PKP solvers we discuss in this paper can also be employed to solve SEP.

Then, we delve into the study of techniques for solving the PKP. We first consider the KMP algorithm and study its performance in the asymptotic regime. Namely, we show that, for growing $n$, the time complexity of KMP is a super exponential function of $n$ and is constant in $\ell$. This result is motivated by the fact that the finite field size $q$ decays exponentially with $\ell$, while the number of equations we dispose of increases with $\ell$. Given the operating principle of the KMP algorithm, these two phenomena compensate one each other. However, the asymptotic regime predominates only for very large values of $n$, so that, for parameters of practical interest, the time complexity can vary significantly with $\ell$.

We then improve the analysis of the solver in [23] and generalize it to the multi-dimensional case. We show that such an algorithm is not better than KMP in the asymptotic regime but, in the finite regime, it can achieve non trivial speed-ups with respect to KMP. As concrete examples, we show that for some recommended PKP instances this algorithm has a time complexity which is lower than that of KMP and [19]. Namely, for the mono-dimensional, 128-bits and 192-bits instances of PKP-DSS [6], we show that our algorithm is slightly faster than KMP. For the multi-dimensional instances recommended in [18], we show that our algorithm is faster than both KMP and [19]. Furthermore, our approach is more general since it works regardless of the finite field size.

We also provide an open source proof-of-concept implementation of our algorithm[2], which can be used to validate the theoretical analysis (and some of the employed heuristics) for small parameters, along with an open source software implementation of all the expressions we have used for deriving numerical results.

*Paper organization* In Section 2 we establish the notation and recall some background notions about linear codes. In Section 3 we state useful facts about subcodes, recalling and enriching some notions from [23]. The equivalence between PKP and SEP is presented and discussed in Section 4. The KMP algorithm and its asymptotic behaviour are analyzed in Section 5. The generalization of the algorithm in [23] is presented and discussed in Section 6, which also provides a comparison of the considered PKP solvers. Finally, in Section 7, we briefly comment about PKP and SEP behave in terms of input sizes, and compare them with other well known problems (say, the Syndrome Decoding Problem (SDP)). Section 8 concludes the paper.

## 2 Notation and background

In this section we settle the notation we use in the paper and recall some background concepts about linear codes.

### 2.1 Notation

As usual, $\mathbb{F}_q$ denotes the finite field with $q$ elements, with $q \in \mathbb{N}$ being a prime power. Bold lowercase (resp., uppercase) letters indicate vectors (resp., matrices). Given $\mathbf{a}$ (resp., $\mathbf{A}$), $a_i$ (resp.,

[2] Source code available at `https://github.com/secomms/pkpattack/`

$a_{iw,j}$) denotes the entry in position $i$ (resp., the entry in the $i$-th row and $j$-th column). With some abuse of notation, we use $\mathrm{GL}_{m,n}$ to indicate the set of $m \times n$ matrices over $\mathbb{F}_q$, with $m \leqslant n$, having full rank $m$. The identity matrix of size $n$ is indicated as $\mathbf{I}_n$, while $\mathbf{0}$ denotes the all-zero matrix (the sizes will always be clear from the context). Given a set $A$, $|A|$ denotes its cardinality (i.e., the number of elements). If $a$ is a random variable (or a quantity that depends on some random variables), we write $a \doteq x$ if it has average value $x$. Given a matrix $\mathbf{A}$ and a set $J$, $\mathbf{A}_J$ is the matrix formed by the columns of $\mathbf{A}$ that are indexed by $J$; analogous notation is used for vectors. We denote by $\mathsf{RREF}(\mathbf{A}, J)$ the algorithm that outputs $\mathbf{A}_J^{-1}\mathbf{A}$ if $\mathbf{A}_J$ is square and non singular, otherwise returns a failure. We use $S_n$ to denote the group of length-$n$ permutations. Given $\mathbf{a} = (a_1, \cdots, a_n)$ and $\pi \in S_n$, we write $\pi(\mathbf{a}) = \big(a_{\pi(1)}, \cdots, a_{\pi(n)}\big)$. Analogous notation is employed for matrices, so that $\pi(\mathbf{A})$ is the matrix obtained by permuting the columns of $\mathbf{A}$ according to $\pi$. Given $\mathbf{a}$, $\mathbf{b}$, with some abuse of notations, we define $\mathbf{a} \cap \mathbf{b}$ as the set of entries which appear in both $\mathbf{a}$ and $\mathbf{b}$. We extend the notation to matrices, i.e., $\mathbf{A} \cap \mathbf{B}$ is the set of columns which are in both $\mathbf{A}$ and $\mathbf{B}$. For a matrix $\mathbf{A} \in \mathbb{F}_q^n$ with no repeated column, we define $\mathscr{S}_\ell(\mathbf{A})$ as the set of length-$\ell$ matrices with columns picked from those of $\mathbf{A}$. Notice that $|\mathscr{S}_\ell(\mathbf{A})| = \frac{n!}{(n-\ell)!}$.

### 2.2 Linear codes

A linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with dimension $k$, redundancy $r = n - k$ and rate $R = k/n$ is a linear $k$-dimensional subspace of $\mathbb{F}_q^n$. Any code admits two equivalent representations: a *generator matrix*, that is, any $\mathbf{G} \in \mathrm{GL}_{k,n}$ such that $\mathcal{C} = \{\mathbf{uG} \mid \mathbf{u} \in \mathbb{F}_q^k\}$, or a *parity-check matrix*, that is, any $\mathbf{H} \in \mathrm{GL}_{r,n}$ such that $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n \mid \mathbf{cH}^\top = \mathbf{0}\}$ (where $\top$ denotes transposition). If $\mathbf{G}$ generates $\mathcal{C}$, then any $\mathbf{SG}$, with $\mathbf{S} \in \mathrm{GL}_{k,k}$, is a generator for the same code $\mathcal{C}$; the same holds for parity-check matrices, i.e., $\mathbf{H}$ and $\mathbf{SH}$, with $\mathbf{S} \in \mathrm{GL}_{r,r}$, are parity-check matrices for the same code. Given a vector $\mathbf{x} \in \mathbb{F}_q^n$, its *syndrome* is $\mathbf{s} = \mathbf{xH}^\top$. The dual of $\mathcal{C}$, which we denote by $\mathcal{C}^\perp$, is the space generated by any parity-check matrix $\mathbf{H}$. For any codeword $\mathbf{c} \in \mathcal{C}$ and any $\mathbf{b} \in \mathcal{C}^\perp$, we have $\mathbf{cb}^\top = 0$. The support of $\mathcal{C}$, which we indicate as $\mathrm{Supp}(\mathcal{C})$, is the set of indexes $i$ such that there is at least one codeword $\mathbf{c} \in \mathcal{C}$ with $c_i \neq 0$. A subcode $\mathcal{B} \subseteq \mathcal{C}$ with dimension $k'$ is a $k'$-dimensional linear subspace of $\mathcal{C}$. The number of such subcodes is counted by $\begin{bmatrix} k \\ k' \end{bmatrix}_q = \prod_{i=0}^{k'-1} \frac{1-q^{k-i}}{1-q^{i+1}}$. Any subspace with dimension 1 is the orbit of a codeword, under scalar multiplications by the elements in $\mathbb{F}_q$: the number of such subcodes is given by $\begin{bmatrix} k \\ 1 \end{bmatrix}_q = \frac{q^k-1}{q-1}$.

### 2.3 Useful approximations and asymptotics

In this section we recall some well known asymptotics and approximations for quantities that will arise naturally in our treatment (for more details about these estimates see, for instance, [15]). For two functions $f(n)$ and $g(n)$, we write $f(n) \sim g(n)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. From Stirling's approximation for factorials, we have

$$n! \sim \frac{1}{\sqrt{2\pi n}} \left(\frac{n}{e}\right)^n = 2^{\Big(n\big(\log_2(n) - \log_2(e)\big) + \frac{1}{2}\log_2(n)\Big)\big(1+o(1)\big)}. \tag{1}$$

For any integer $q \geqslant 2$ and constant $x \in [0; 1]$, the q-ary entropy function is defined as

$$h_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x).$$

Let $\omega \in [0; 1]$ be a constant; then

$$\binom{n}{\omega n} = 2^{\big(h_2(\omega)n - \frac{1}{2}\log_2(n)\big)\big(1+o(1)\big)}, \tag{2}$$

$$\frac{n!}{(n-\omega n)!} = 2^{n\big(h_2(\omega) + \omega\log_2(\omega n/e)\big)\big(1+o(1)\big)}. \tag{3}$$

The number of vectors with length $n$, values over $\mathbb{F}_q$ and Hamming weight $\omega n$ is then

$$\binom{n}{\omega n}(q-1)^{\omega n} = 2^{\big(h_q(\omega)\log_2(q)n - \frac{1}{2}\log_2(n)\big)\big(1+o(1)\big)}. \tag{4}$$

Finally, for a constant $d \in \mathbb{N}$, we have

$$\frac{\begin{bmatrix} \omega n \\ d \end{bmatrix}_q}{\begin{bmatrix} n \\ d \end{bmatrix}_q} \sim q^{-d(1-\omega)n}. \tag{5}$$

## 3 Subcodes with small support

In this section we discuss properties of small support subcodes, which will be fundamental to analyze the attacks we describe in this paper.

### 3.1 Number of subcodes with small support

For a random code, the average number of subcodes with dimension $d$ and support size $w$ can be bounded thanks to the following theorem [23].

**Theorem 1** *For a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, we define $\mathcal{A}_{w,d}(\mathcal{C})$ as the set of subcodes of $\mathcal{C}$ with dimension $d$ and support size $w$. Let $N_k(w,d)$ be the average value of $|\mathcal{A}_{w,d}(\mathcal{C})|$, when $\mathcal{C}$ is picked at random among all codes with dimension $k$. Then $\breve{N}_k(w,d) \leqslant N_k(w,d) \leqslant \widehat{N}_k(w,d)$, with*

$$\widehat{N}_k(w,d) = \binom{n}{w} \frac{(q^d-1)^w}{\prod_{i=0}^{d-1}(q^d-q^i)} \frac{\begin{bmatrix} k \\ d \end{bmatrix}_q}{\begin{bmatrix} n \\ d \end{bmatrix}_q},$$

$$\breve{N}_k(w,d) = \binom{n}{w} (q^d-1)^{w-d} \frac{\begin{bmatrix} k \\ d \end{bmatrix}_q}{\begin{bmatrix} n \\ d \end{bmatrix}_q}.$$

*Proof.* See [23].

*Remark 1.* The bounds in the above theorem are tight, up to a factor which tends asymptotically to 1 as $q$ grows. Indeed, consider that

$$\frac{\widehat{N}_k(w,d)}{\breve{N}_k(w,d)} = \frac{(q^d-1)^d}{\prod_{i=0}^{d-1} q^d - q^i} = \frac{(q^d-1)^d}{q^{d^2}\prod_{i=1}^{d}(1-q^{-i})}.$$

Since $e^{-\frac{1}{x}} \approx 1 - \frac{1}{x}$ if $x$ is sufficiently large, we approximate $1 - q^{-i} \approx e^{-\frac{1}{q^i}}$ and get

$$\begin{aligned} \frac{\widehat{N}_k(w,d)}{\breve{N}_k(w,d)} &\approx \frac{(q^d-1)^d}{q^{d^2}\prod_{i=1}^{d} e^{-\frac{1}{q^i}}} = (q^d-1)^d q^{-d^2} e^{\sum_{i=1}^{d}\frac{1}{q^i}} \\ &\leqslant e^{\sum_{i=1}^{d}\frac{1}{q^i}} = e^{\left(\frac{1-q^{-d-1}}{1-q^{-1}}-1\right)} = e^{\frac{q^d-1}{(q-1)q^d}} \leqslant e^{\frac{1}{q-1}}. \end{aligned}$$

The above quantity is constant in both the code length and desired support size and tends to 1 as $q$ increases.

As a consequence of the remark, using the lower bound $\breve{N}_k(w,d)$ instead of $N_k(w,d)$ we obtain a rather tight estimate on the actual number of subcodes. To make an example: for $q = 2$ (i.e., when the error term in the estimate is maximum), we have that $N_k(w,d)$ is less than $e^1 \approx 2.7183$ times greater then the lower bound $\breve{N}_k(w,d)$.

Starting from the above theorem, one can derive the minimum support size we expect to have for $d$-dimensional subcodes, when dealing with random codes. To this end, we consider the following proposition, whose proof is reported in Appendix A.

**Proposition 1** *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a random code with rate $R$. Let $d \in \mathbb{N}$ be constant, and $\omega^* \in [0;1]$ such that $\omega^* = \min\{\omega \in [0;1] \mid N_k(\omega n, d) \geqslant 1\}$. Then*

$$\omega^* \approx h_{q^d}^{-1}\left(1 - R + \frac{d}{n}\right).$$

### 3.2 Finding subcodes with small support

In [23], the authors consider an adaptation of Prange's ISD to find subcodes with small support. In principles, the algorithm may be improved by considering techniques which are normally employed for standard ISD algorithms (e.g., partial gaussian elimination and a meet-in-the-middle search). Yet, for the sake of simplicity, we stick to the algorithm in [23] and, in the next Proposition, recall its time complexity. Note that similar results can be found also in [2,5]. Yet, for the sake of completeness, the algorithmic procedure of ISD, as well as the proof of the proposition, are shown in Appendix B.

**Proposition 2 Time complexity of ISD**
*We consider ISD as an algorithm that, on input $\mathcal{C} \in \mathbb{F}_q^n$ with dimension $k$ and integers $w, d \in \mathbb{N}$ such that $w \leqslant n + d - k$, returns a random element from $\mathcal{A}_{w,d}(\mathcal{C})$ with average running time*

$$T_{\mathsf{ISD}}^{(d)}(n,k,w) = O\left(\frac{k^3 + \binom{k}{d}}{p^{(d)}(n,k,w)}\right),$$

*with* $p^{(d)}(n,k,w) = \min\left\{\frac{\binom{w}{d}\binom{n-w}{k-d}}{\binom{n}{k}}\widetilde{N}_k(w,d)\ ;\ 1\right\}$.

## 4 The equivalence between PKP and SEP

In this section we introduce SEP and see its connections with another code-based problem, namely, the Permutation Equivalence Problem (PEP). We proceed by describing a reduction from SEP to PEP, which gives us a first way to solve SEP. This reduction is interesting since it bridges two different code-based problems but, in practice, is not efficient since it runs in exponential time. Finally, we show that PKP and SEP are equivalent, namely, they are different formulations of the very same problem. This unlocks the possibility of solving SEP with more powerful solvers.

### 4.1 PKP, SEP and PEP

We start by introducing the PKP in its most general formulation.

**Problem 1 Permuted Kernel Problem (PKP)**
*Given $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ and $\mathbf{V} \in \mathbb{F}_q^{\ell \times n}$, with $m, \ell \leqslant n$, find $\pi \in S_n$ such that $\pi(\mathbf{V})\mathbf{A}^\top = \mathbf{0}$.*

When $\ell = 1$ (which, arguably, is the most considered PKP formulation), the above problem corresponds to the one employed, for instance, in [6,26]; we will refer to this case as mono-dimensional. In [18], instead, the authors consider the case $\ell > 1$; we will refer to this case as multi-dimensional.

The subcode equivalence problem is another NP-complete problem introduced in [4] which reads as follows.

**Problem 2 Subcode Equivalence Problem (SEP)**
*Given $\mathcal{C} \subseteq \mathbb{F}_q^n$ with dimension $k$ and $\mathcal{C}' \subseteq \mathbb{F}_q^n$ with dimension $k' < k$, find $\pi \in S_n$ such that $\pi(\mathcal{C}') \subseteq \mathcal{C}$. Equivalently, given $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ and $\mathbf{G}' \in \mathbb{F}_q^{k' \times n}$, find $\mathbf{S} \in \mathrm{GL}_{k',k}$ and $\pi \in S_n$ such that $\mathbf{G}' = \mathbf{S}\pi(\mathbf{G})$.*

Considering the above formulation but requiring that $k' = k$ would correspond to the PEP.

**Problem 3 Permutation Equivalence Problem (PEP)**
*Given $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$, both with dimension $k$, find $\pi \in S_n$ such that $\pi(\mathcal{C}') = \mathcal{C}$. Equivalently, given $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find $\mathbf{S} \in \mathrm{GL}_{k,k}$ and $\pi \in S_n$ such that $\mathbf{G}' = \mathbf{S}\pi(\mathbf{G})$.*

We remember that SEP is NP-complete [4], while a well-known result establishes that the NP-completeness of PEP would imply collapse of the polynomial hierarchy [21]. Also, for codes with small hull (that is, the intersection between a code and its dual), polynomial time algorithms exist [1,25]. These algorithms essentially require some linear algebra computations (e.g., intersecting vector spaces) plus auxiliary operations, such as hull enumeration [25] or reducing to graph isomorphism [1]. The cost of such auxiliary operations can be safely neglected, unless the hull dimension

gets too large: in such a case, the currently known best solvers are different algorithms [2,5] which, however, require to find low weight codewords and consequently take exponential time. Notice that, for random codes, the hull dimension is a small constant with high probability [24]. Hence, for random codes, with overwhelming probability applying the algorithms in [1, 25] results in a polynomial time solver for PEP: for this reason, PEP is considered an easy problem, on average.

### 4.2 Reducing SEP to PEP

We describe a trivial way to solve SEP, which highlights its connection with PEP. Consider the procedure in Algorithm 1 which, basically, does a brute force search over all subcodes of $\mathcal{C}$ and, for each candidate, tries to solve the associated PEP.

**Algorithm 1:** Reduction of SEP to PEP

**Input:** $\mathbf{G} \in \mathbb{F}_q^{k\times n}$, $\mathbf{G}' \in \mathbb{F}_q^{k'\times n}$
**Output:** permutation $\pi \in S_n$ and matrix $\mathbf{S} \in \mathrm{GL}_{k',k}$ such that $\mathbf{G}' = \mathbf{SGP}$

1 Choose uniformly at random $\mathbf{S} \in \mathrm{GL}_{k',k}$;
2 Set $\widetilde{\mathbf{G}} = \mathbf{SG}$;
3 Try to solve PEP on input $\{\widetilde{\mathbf{G}}, \mathbf{G}'\}$;
4 If a solution $\pi \in S_n$ is found, return $\{\mathbf{S}; \pi\}$, else restart from step 1.

The running time of the algorithm is analyzed next.

**Proposition 3** *Algorithm 1 takes average time which is upper bounded from* $O\left(q^{k'(k-k')} T_{\mathsf{PEP}}(q,n,k)\right)$, *where* $T_{\mathsf{PEP}}(q,n,k)$ *is the running time of an algorithm that solves PEP on codes with length* $n$ *and dimension* $k$, *define over a finite field with* $q$ *elements.*

*Proof.* The probability that a choice for $\mathbf{S}$ is valid is given by the reciprocal of $\begin{bmatrix} k \\ k' \end{bmatrix}_q$. Hence, $\begin{bmatrix} k \\ k' \end{bmatrix}_q$ corresponds to the number of times, on average, we call the PEP solver. To conclude the proof, we show that $\begin{bmatrix} k \\ k' \end{bmatrix}_q \leqslant q^{k'(k-k')}$ for any $k'$. Let us write $\begin{bmatrix} k \\ x \end{bmatrix}_q = \prod_{i=0}^{x-1} \frac{q^{k-i}-1}{q^{i+1}-1} = \prod_{i=0}^{x-1} a(i)$. Considering that $\frac{z-1}{y-1} \leqslant \frac{z}{y}$ whenever $z \geqslant y$, we can bound $a(i) \leqslant \frac{q^{k-i}}{q^{i+1}}$ only if $\frac{q^{k-i}}{q^{i+1}} \geqslant 1$ for any index $i \in [0; x-1]$. This holds whenever $k \geqslant 2i+1$ hence, given that $i \leqslant x-1$, it must be $x \leqslant \frac{k+1}{2}$. Consequently, we have

$$\begin{aligned}\begin{bmatrix} k \\ x \end{bmatrix}_q &\leqslant \prod_{i=0}^{x-1} q^{k-2i-1} = q^{x(k-1)} \left(\prod_{i=0}^{x-1} q^{-i}\right)^2 \\ &= q^{x(k-1)} \left(q^{-\frac{(x-1)x}{2}}\right)^2 = q^{x(k-x)}.\end{aligned}$$

If $x > \frac{k+1}{2}$, consider that $\begin{bmatrix} k \\ x \end{bmatrix}_q = \begin{bmatrix} k \\ k-x \end{bmatrix}_q = \begin{bmatrix} k \\ x' \end{bmatrix}_q$ and $x' = k - x < \frac{k-1}{2} < \frac{k+1}{2}$. Hence $\begin{bmatrix} k \\ x' \end{bmatrix}_q \leqslant q^{x'(k-x')} = q^{(k-x)x}$. □

The above reduction is rather simple and, not surprisingly, requires exponential time (regardless of the cost to solve PEP). Yet, it is interesting from a theoretical point of view, since it creates a relation between SEP and PEP. In the next section, we see that SEP and PKP are exactly the same problem: this allows to solve SEP through any known algorithm for PKP.

### 4.3 Equivalence between PKP and SEP

In the following proposition, we show that PKP and SEP are equivalent.

**Proposition 4** *If* $\ell \neq n-m$, *PKP is equivalent to SEP; if* $\ell = n-m$, *PKP is equivalent to PEP.*

*Proof.* In the following, we will denote by $\{\mathbf{A} \in \mathbb{F}_q^{m\times n}, \mathbf{V} \in \mathbb{F}_q^{\ell\times n}\}$ an instance of PKP, and by $\{\mathbf{G} \in \mathbb{F}_q^{k\times n}, \mathbf{G'} \in \mathbb{F}_q^{k'\times n}\}$ an instance of SEP. Also, we denote by $\mathcal{C}$ and $\mathcal{C}'$ the codes generated, respectively, by $\mathbf{G}$ and $\mathbf{G}'$. To avoid burdening the notation and the treatment, we carry out the proof considering that matrices have full rank; in case this is not true, the proof still holds, with the only precaution that one first needs to remove the redundant rows. We start by considering that any SEP instance can be transformed into a PKP instance with some simple linear algebra. Indeed, to solve SEP, we want to find a permutation $\pi$ and a non singular $\mathbf{S} \in \mathrm{GL}_{k',k}$ such that $\mathbf{G}' = \mathbf{S}\pi(\mathbf{G})$. Equivalently, it must be $\pi^{-1}(\mathbf{G}') = \mathbf{SG}$. Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ be a parity-check matrix for $\mathcal{C}$: then, $\pi$ solves SEP if and only if

$$\pi^{-1}(\mathbf{G}')\mathbf{H}^\top = \mathbf{0}.$$

The above corresponds exactly to the requirement for a PKP instance with $m = n - k$ and $\ell = k'$.

The same procedure works in the other way, i.e., in showing how PKP can be seen either as SEP or PEP. We start by considering the case of $\ell < n - m$, and interpret $\mathbf{A}$ as a parity-check matrix for $\mathcal{C}$ and $\mathbf{V}$ as a generator for $\mathcal{C}'$. Repeating the above reasoning, it is immediately seen that the problem corresponds to a SEP instance. If instead $\ell = n - m$, we have that $\mathcal{C}$ and $\mathcal{C}'$ have the same dimension $k = k' = \ell = n - m$, so we end up with a PEP instance. Finally, we notice that if $\ell > n - m$, the code $\mathcal{C}'$ has dimension which is larger then that of $\mathcal{C}$. So, the initial PKP still corresponds to a SEP instance, but the role of the codes are exchanged: $\mathcal{C}$ is the permutation of a subcode of $\mathcal{C}'$. □

A summary of the relations between PKP, SEP and PEP in shown in Table 1. As we have already said, the PEP has already been extensively studied and is considered easy when the input codes are random. Consequently, PKP can be considered easy, on average, if $\ell = n - m$; for this reason, in this paper we will not study this case. Also, from now we set $1 \leqslant \ell < n - m$: indeed, the case of $\ell > n - m$ can be tackled identically, apart from a mere swap in the roles of the parameters $\ell$ and $n - m$.

Table 1: Relations between PKP, SEP and PEP, and corresponding parameters

| PKP | Equivalent problem | Parameters |
|---|---|---|
| $\ell < n - m$ | SEP | $k = n - m$, $k' = \ell$ |
| $\ell = n - m$ | PEP | $k = k' = n - m$ |
| $\ell > n - m$ | SEP | $k = \ell$, $k' = n - m$ |

## 5 General considerations on PKP and asymptotics for the KMP algorithm

In this section we recall general properties about hard instances of PKP. Also, we recall the algorithm in [17], which we refer to as KMP (from the authors initials). We generalize all our considerations, as well as the KMP algorithm, to the multi-dimensional version of PKP. Finally, we derive its running time in the asymptotic regime.

### 5.1 Hardest instances

As in all works regarding PKP, we study the constrains under which hardest to solve instances are obtained; in doing this, we generalize the considerations that one normally has, for the mono-dimensional PKP, to the multi-dimensional case. We generalize the known conditions to the multi-dimensional case. Namely, we want $\mathbf{A} \in \mathbb{F}_q^{m\times n}$ and $\mathbf{V} \in \mathbb{F}_q^{\ell\times n}$ both with full rank, and no repeated columns, and also set the parameters so that, on average, only one solution exists. Formally, these conditions correspond to the following criteria:

i) $\mathrm{Rank}(\mathbf{A}) = m$, $\mathrm{Rank}(\mathbf{V}) = \ell$;

ii) $q^\ell \geqslant n$, $q^m \geqslant n$;
iii) $\mathbf{A}$ and $\mathbf{V}$ have no repeated columns;
iv) $n! \frac{\begin{bmatrix} n-m \\ \ell \end{bmatrix}_q}{\begin{bmatrix} n \\ \ell \end{bmatrix}_q} < 1$.

*Remark 2.* We observe that, whenever $q^{k-\ell} \ll 1$, we can approximate $\frac{\begin{bmatrix} n-m \\ \ell \end{bmatrix}_q}{\begin{bmatrix} n \\ \ell \end{bmatrix}_q} \approx q^{-\ell m}$, so that condition iv becomes $n! q^{-\ell m} < 1$.

Notice that condition ii is simply obtained considering that, if $q^\ell < n$, then for sure $\mathbf{V}$ contains at least two identical columns; the same holds for the condition $q^m < n$. Finally, condition iv is about the uniqueness of the solution. Consider that any permutation $\pi$ of $\mathbf{V}$ yields a vector space with dimension $\ell$. The number of $\ell$-dimensional spaces that are orthogonal to the space generated by $\mathbf{A}$ is counted by $\begin{bmatrix} n-m \\ \ell \end{bmatrix}_q$. Then, we observe that $\frac{\begin{bmatrix} n-m \\ \ell \end{bmatrix}_q}{\begin{bmatrix} n \\ \ell \end{bmatrix}_q}$ is the probability that a random $\ell$-dimensional subspace of $\mathbb{F}_q^n$ is orthogonal to $\mathbf{A}$. Assuming that each $\pi(\mathbf{V})$ behaves as a random subspace of $\mathbb{F}_q^n$, then we can use $n! \frac{\begin{bmatrix} n-m \\ \ell \end{bmatrix}_q}{\begin{bmatrix} n \\ \ell \end{bmatrix}_q} < 1$ as an estimate on the average number of solutions: setting this number to be smaller than 1, we guarantee that, on average, we expect to have no more than one solution. Another common criterion consists in studying instances which are not vacuously hard, i.e., such that at least one solution exists. To do this, we first choose $\widetilde{\mathbf{V}}$ as a random kernel element with full rank and no repeated column, then we select a random permutation $\pi$ and set $\mathbf{V} = \pi(\widetilde{\mathbf{V}})$.

### 5.2 The PKP as a decoding problem

We consider that $\mathbf{A}$ can be enriched with an additional linear equation. Indeed, let $\mathbf{H} = \begin{pmatrix} \mathbf{A} \\ 1, 1, \cdots, 1 \end{pmatrix} \in \mathbb{F}_q^{r \times n}$, with $r = m + 1$, and write

$$\begin{aligned} \mathbf{E} = \widetilde{\mathbf{V}}\mathbf{H}^\top &= \left( \widetilde{\mathbf{V}}\mathbf{A}^\top, \widetilde{\mathbf{V}} \cdot (1, 1, \cdots, 1)^\top \right) \\ &= \left( \mathbf{0}_{\ell \times m}, \sum_{i=1}^{n} \widetilde{\mathbf{v}}_i \right) \\ &= \left( \mathbf{0}_{\ell \times m}, \sum_{i=1}^{n} \mathbf{v}_i \right) \\ &= (\mathbf{0}_{\ell \times m}, \mathbf{e}) \in \mathbb{F}_q^{\ell \times r}, \end{aligned} \tag{6}$$

where $\widetilde{\mathbf{v}}_i$ is the $i$-th column of $\widetilde{\mathbf{V}}$ and $\mathbf{v}_i$ that of $\mathbf{V}$. Notice that $\mathbf{e}$ is a length-$\ell$ column vector. Finally, let

$$\mathbf{S}\mathbf{H} = \mathsf{RREF}(\mathbf{H}, \{n - r + 1, n - r + 2, \cdots, n\}) = (\mathbf{U}, \mathbf{I}_r), \tag{7}$$

with $\mathbf{S} \in \mathrm{GL}_{r,r}$ and $\mathbf{U} \in \mathbb{F}_q^{r \times (n-r)}$. Let $\widetilde{\mathbf{E}} = \mathbf{E}\mathbf{Y}^\top$ and write $\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2)$, and consider that

$$\widetilde{\mathbf{E}} = \mathbf{V}_1\mathbf{U}^\top + \mathbf{V}_2,$$

from which

$$\mathbf{V}_2 = \widetilde{\mathbf{E}} - \mathbf{V}_1\mathbf{U}^\top. \tag{8}$$

Hence, we simply need to correctly guess the action of the permutation for $n-r$ coordinates. Notice that, if $\mathbf{H}$ does not admit a change of basis as in (7), it is enough to row reduce with respect to a different set of $r$ indices. In other words, let $J \subset \{1, 2, \cdots, n\}$ of size $n-r$ and $\neg J = \{1, 2, \cdots, n\} \backslash J$. Also, let $J$ be such that $\mathbf{H}_{\neg J}$ is non singular. Then, it holds that

$$\mathbf{V}_{\neg J} = \big(\mathbf{E} - \mathbf{V}_J\mathbf{H}_J\big)\mathbf{H}_{\neg J}^{-\top}, \tag{9}$$

where the $^{-\top}$ operator denotes the inverse transposal.

**A coding theory perspective** We now reformulate the PKP as a code-based problem. Notice that this is, already, somehow implicit since we have already shown that PKP and SEP are exactly the same problem. Yet, rephrasing everything in terms of codes will be helpful in analyzing the attack we present in the next section.

**Problem 4 PKP as a Decoding Problem**
*Given* $\mathbf{H} \in \mathbb{F}_q^{r\times n}$*, the parity-check matrix of* $\mathcal{C} \subseteq \mathbb{F}_q^{r\times n}$ *with code rate* $R = 1 - \frac{r}{n}$*,* $\mathbf{V} \in \mathbb{F}_q^{\ell\times n}$ *and* $\mathbf{E} \in \mathbb{F}_q^{\ell\times r}$*, find* $\pi \in S_n$ *such that* $\mathbf{E} = \pi(\mathbf{V})\mathbf{H}^\top$*.*

This formulation, which is sometimes referred to as non-homogeneous PKP, highlights the fact that the PKP is, in the end, a decoding problem. Namely, we are given a set of $\ell$ syndromes $\mathbf{e}_i$, where $\mathbf{e}_i$ is the $i$-th row of $\mathbf{E}$, and a set of $\ell$ vectors $\mathbf{v}_i$, where $\mathbf{v}_i$ is the $i$-th row of $\mathbf{V}$. We want to find a permutation $\pi$ so that

$$\mathbf{e}_i = \pi(\mathbf{v}_i)\mathbf{H}^\top, \quad \forall i \in \{1, \cdots, \ell\}.$$

To decode a syndrome into a vector, it is enough to guess the values of the vector in an information set, that is, a set $J \subset \{1, 2, \cdots, n\}$ of size $k$ and such that

$$\{\mathbf{c}_J \neq \mathbf{c}'_J \mid \mathbf{c}, \mathbf{c}' \in \mathcal{C},\ \mathbf{c} \neq \mathbf{c}'\}.$$

It is easy to see that, if $J$ is an information set for $\mathcal{C}$, then $\mathbf{H}_{\neq J}$ is non singular, hence, can be used in (9). In other words, to solve the problem, it is enough to determine the values of $\mathbf{V}$ in the information set $J$: the values outside of the information set can be computed using (9).

### 5.3 KMP algorithm

We here recall the KMP algorithm [17], originally proposed for the mono-dimensional PKP, and generalize it for the multi-dimensional case. The algorithm works with three parameters $u, u_1, u_2 \in \mathbb{N}$, such that $1 \leqslant u \leqslant r$, $u_1, u_2 \geqslant 1$ and $u_1 + u_2 = n - r + u$. The procedure is initialized by choosing a rank-$u$ matrix $\mathbf{S} \in \mathbb{F}^{u\times r}$ such that $\widetilde{\mathbf{H}} = \mathbf{SH}$ has support size $n - r + u$. To do this, we first compute $\mathbf{H}' = \mathsf{RREF}(\mathbf{H}, \{n-r+1, \cdots, n\})$ and then set $\widetilde{\mathbf{H}}$ as the sub-matrix formed by the entries of $\mathbf{H}'$ in the first $u$ rows and the columns in positions $\{1, \cdots, n-r+u\}$. The same transformation is applied to $\mathbf{E}$, obtaining $\widetilde{\mathbf{E}} = \mathbf{ES}^\top \in \mathbb{F}_q^{\ell\times u}$. Then, we partition $\widetilde{\mathbf{H}}$ as $(\widetilde{\mathbf{H}}_1, \widetilde{\mathbf{H}}_2)$, where $\widetilde{\mathbf{H}}_1 \in \mathbb{F}_q^{u\times u_1}$ and $\widetilde{\mathbf{H}}_2 \in \mathbb{F}_q^{u\times u_2}$, and construct two lists

$$\mathcal{L}_1 = \left\{ (\mathbf{X}, \mathbf{X}\widetilde{\mathbf{H}}_1^\top) \middle| \mathbf{X} \in \mathscr{S}_{u_1}(\mathbf{V}) \right\},$$

$$\mathcal{L}_2 = \left\{ (\mathbf{X}, \widetilde{\mathbf{E}} - \mathbf{X}\widetilde{\mathbf{H}}_2^\top) \middle| \mathbf{X} \in \mathscr{S}_{u_2}(\mathbf{V}) \right\}.$$

Let $\mathcal{L} = \mathcal{L}_1 \bowtie \mathcal{L}_2$, where $\bowtie$ is computed as follows:

1. use an efficient search algorithm (e.g., permutation plus binary search) to find collisions, i.e., pairs $(\mathbf{X}, \mathbf{Y}) \in \mathcal{L}_1$ and $(\mathbf{X}', \mathbf{Y}') \in \mathcal{L}_2$ such that $\mathbf{Y} = \mathbf{Y}'$;
2. keep only the collisions for which $\mathbf{X}$ and $\mathbf{X}'$ have no common columns.

By construction, it holds that

$$\mathcal{L} = \left\{ (\mathbf{X}, \mathbf{X}') \in \mathscr{S}_{u_1+u_2}(\mathbf{V}) \mid (\mathbf{X}, \mathbf{X}')\widetilde{\mathbf{H}}^\top = \widetilde{\mathbf{E}} \right\}.$$

Then, we find $J$ of size $n-r$ so that $J \subseteq \{1, \cdots, n-r+u\}$ and $\mathbf{H}_{\{1,\cdots,n\}\setminus J}$ is non singular, compute $\widetilde{\mathbf{H}} = \mathsf{RREF}(\mathbf{H}, \{1, \cdots, n\}\setminus J)$ and use (8) to test each element in $\mathcal{L}$. Namely, for each $\mathbf{X} \in \mathcal{L}$, we use the entries of $\mathbf{X}_J$ as $\widetilde{\mathbf{V}}_J$ and see if the resulting $\widetilde{\mathbf{V}}$ belongs to $\mathscr{S}_n(\mathbf{V})$.

Coherently with all the PKP literature, we measure the running time of the KMP algorithm as the number of matrix-matrix multiplications and list operations.

**Proposition 5 Running time of the KMP algorithm**
*Let* $u_1, u_2 \in \mathbb{N}$ *such that* $k + 1 \leqslant u_1 + u_2 \leqslant n$*. Then, the KMP algorithm runs in time*

$$T_{\mathsf{KMP}}(u_1, u_2) = |\mathcal{L}_1| + |\mathcal{L}_2| + N_{\mathcal{L}_1\bowtie\mathcal{L}_2} + |\mathcal{L}|,$$

*where* $|\mathcal{L}_i| = \frac{n!}{(n-u_i)!}$*,* $N_{\mathcal{L}_1\bowtie\mathcal{L}_2} \doteq \frac{(n!)^2 q^{\ell(n-r-u_1-u_2)}}{(n-u_1)!(n-u_2)!}$ *and* $|\mathcal{L}| \doteq \frac{n!}{(n-u_1-u_2)!} q^{\ell(n-r-u_1-u_2)}$*.*

*Proof.* First, we consider that it must be $u_1 + u_2 > k$ since, putting $\mathbf{H}$ in row reduced echelon form, we obtain parity-check equations with maximum Hamming weight $n - r + 1 = k + 1$. Then, we have that each list $\mathcal{L}_i$ has size $|\mathscr{S}_{u_i}(\mathbf{V})|$: given that $\mathbf{V}$ does not have repeated columns, the number of considered elements in $\mathcal{L}_i$ is $\frac{n!}{(n-u_i)!}$. For the number of collisions $N_{\mathcal{L}_1 \bowtie \mathcal{L}_2}$, we consider that each collision is due to two equal matrices with sizes $\ell \times u$. Since we are dealing with random PKP instances, we can consider that any pair of elements $(\mathbf{X}, \mathbf{Y}) \in \mathcal{L}_1$ and $(\mathbf{X}', \mathbf{Y}') \in \mathcal{L}_2$ collides with probability

$$q^{-\ell u} = q^{-\ell u_1 + u_2 + r - n} = q^{\ell(n-r-u_1-u_2)}.$$

Then, the average value of $N_{\mathcal{L}_1 \bowtie \mathcal{L}_2}$ can be set as

$$|\mathcal{L}_1| \cdot |\mathcal{L}_2| \cdot q^{\ell(n-r-u_1-u_2)}.$$

Finally, we consider that $\mathcal{L}$ contains all the matrices in $\mathscr{S}_{u_1+u_2}(\mathbf{V})$ such that their product by $\widetilde{\mathbf{H}}$ returns $\widetilde{\mathbf{E}}$, having size $\ell \times u$. Hence, the average size of $\mathcal{L}$ can be estimated as

$$|\mathscr{S}_{u_1+u_2}(\mathbf{V})| \cdot q^{\ell(n-r-u_1-u_2)} = \frac{n! q^{\ell(n-r-u_1-u_2)}}{(n-u_1-u_2)!}.$$

□

### 5.4 Asymptotic analysis of KMP

We now derive the asymptotic running time of the KMP algorithm. Let us first consider the following proposition.

**Proposition 6 Asymptotic running time for KMP**
*Let $\mu \in \mathbb{R}_+$ such that $\frac{R}{2} < \mu \leqslant \frac{1}{2}$ and*

$$h_2(\mu) + \mu \log_2\left(\frac{\mu n}{e}\right) + \ell \log_2(q)\big(R - 2\mu\big) = 0.$$

*Then, asymptotically, the running time of the KMP algorithm is minimized with $u_1 = u_2 = \mu n$ and is given by $2^{n c_{\mathsf{KMP}}(\mu)\left(1+o(1)\right)}$, where*

$$c_{\mathsf{KMP}}(\mu) = h_2(\mu) + \mu \log_2\left(\frac{\mu n}{e}\right).$$

*Proof.* See Appendix C.

Notice that the KMP algorithm runs in time which is super exponential in the code length $n$, since the dominant term grows exponentially with $n \log_2(n)$. However, to give a rigorous proof of this fact, we need some further considerations.

First, we express $q$ as a function of the code length. Indeed, we notice that the condition of having no more than one solution can be expressed as $n! \frac{\begin{bmatrix} n-r \\ \ell \end{bmatrix}_q}{\begin{bmatrix} n \\ \ell \end{bmatrix}_q} \approx 1$. Then, taking (1) and (5) into account, we find

$$\begin{aligned} q &\sim \left(\frac{n}{e}\right)^{\frac{1}{\ell(1-R)}} \left(1 + \frac{1}{2n\ell(1-R)} \log_2(\pi n)\right) \\ &= \left(\frac{n}{e}\right)^{\frac{1}{\ell(1-R)}} \big(1 + o(1)\big). \end{aligned} \tag{10}$$

In this regime, we can find an easy, closed-form formula for the running time of KMP. To this end, let us consider the following proposition.

**Proposition 7 Running time of KMP in the asymptotic regime: closed form formula**
*Let $q \sim \left(\frac{n}{e}\right)^{\frac{1}{\ell(1-R)}}$. Let $\mu \in (R/2; 1/2]$ be the value that optimizes the KMP algorithm, as in Proposition 6. Then, $\lim_{n\to\infty} \mu = \mu^* = \frac{R}{1+R}$ and the KMP asymptotically runs in time $2^{n \cdot c^*_{\mathsf{KMP}}}$, where*

$$\begin{aligned} c^*_{\mathsf{KMP}} &= \lim_{n\to\infty} c_{\mathsf{KMP}}(\mu) = c_{\mathsf{KMP}}\left(\frac{R}{1+R}\right) \\ &= \frac{1}{1+R}\left(\log_2(1+R) + R \log_2\left(\frac{n}{e}\right)\right). \end{aligned}$$

*Proof.* See Appendix C.

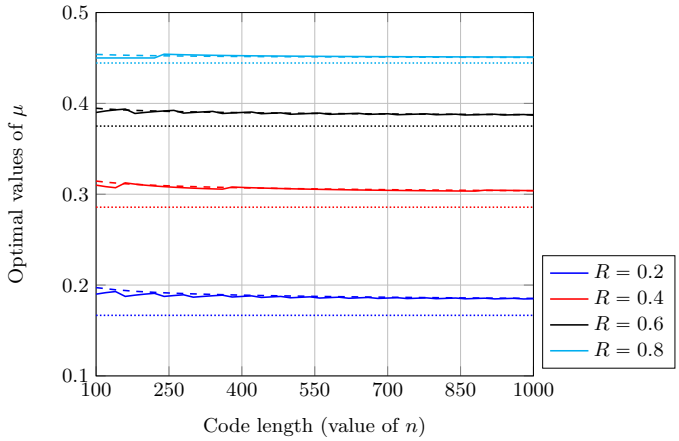



Fig. 1: Values of $\mu$ that optimize the KMP algorithm, for $\ell = 1$. The continuous line has been obtained by selecting, among all possible choices for $(u_1, u_2)$, the ones yielding the smallest value for $T_{\mathsf{KMP}}(u_1, u_2)$. Dashed lines report the optimal values of $\mu$ which we have found solving, numerically, the equation in Proposition 6. Dotted lines correspond to $\mu^* = \frac{R}{1+R}$.

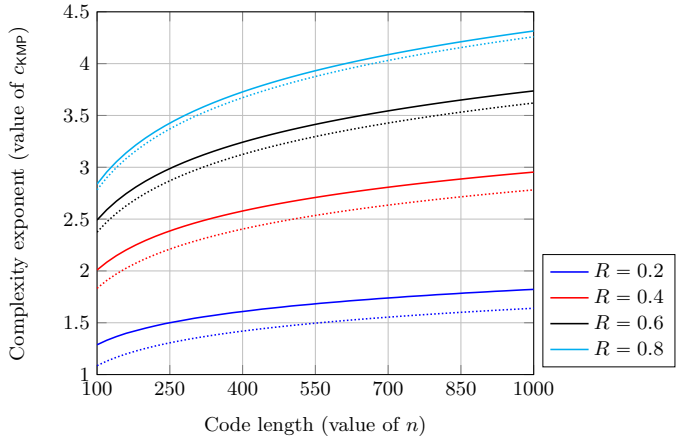



Fig. 2: Complexity exponents for the KMP algorithm, as a function of $n$, for the case of $\ell = 1$. Continuous lines report the values of $c_{\mathsf{KMP}}$ arising from Proposition 6, while dotted lines report the values of $c^*_{\mathsf{KMP}}$.

The above propositions leads to the following three interesting observations, which hold in the asymptotic regime:

- the optimal value for $\mu$ is independent of $\ell$;
- the optimal time complexity is independent of $\ell$;
- the KMP algorithm runs in time which is super exponential in the code length.

We now give more insight about about these facts, with the help of some numerical experiments. We have considered several triplets $(\ell, R, n)$ and, for each one, have found the smallest prime $q$ for which the average number of solutions to PKP is $< 1$. Using these parameters, we have studied the performances of KMP. In Figure 1 we report an example of how the optimal value of $\mu$ behaves, for the case of $\ell = 1$; as we can see from the figure, the optimal value for $\mu$ is already

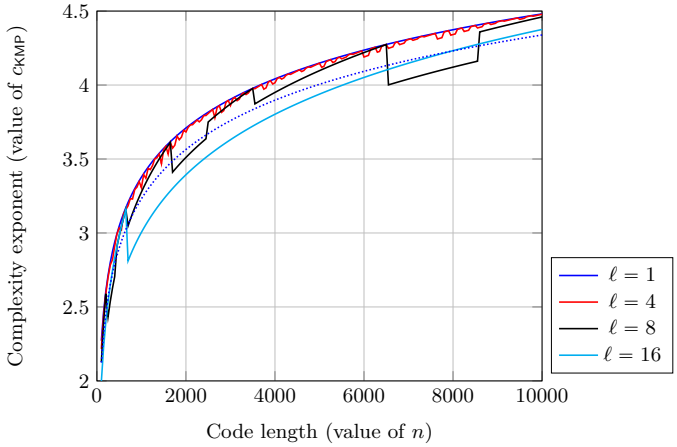



Fig. 3: Complexity exponents for the KMP algorithm, as a function of $n$, for the case of $\ell = 1$. Continuous lines report the values of $c_{\mathsf{KMP}}$ arising from Proposition 6, while dotted lines report the values of $c^*_{\mathsf{KMP}}$.

well approximated by $\mu^*$ for $n \leqslant 1,000$. Figure 2 reports the optimal values of the complexity exponent $c_{\mathsf{KMP}}$, for growing $n$, and a comparison with $c^*_{\mathsf{KMP}}$. We notice that there is a small, but non negligible, difference between the values of $c_{\mathsf{KMP}}$ and $c^*_{\mathsf{KMP}}$. This is due to the fact that $c_{\mathsf{KMP}}$ tends to $c^*_{\mathsf{KMP}}$ very slowly. In any case, as predicted by Proposition 7, the values of $c_{\mathsf{KMP}}$ exhibit a logarithmic dependence in $n$. In Figure 3 we show how the complexity exponent behaves, for the case of $R = 0.5$ and several (constant) values of $\ell$. The fluctuations in the curves are due to the fact that we require $q$ to be a prime, hence, we may need to change the value resulting from (10).

## 6 Improving KMP in the finite regime

In [23] a novel algorithm for the mono-dimensional case of PKP has been proposed. As we have already said, it comes as a refinement of KMP and it has been shown that, in several cases and in the finite regime, it exhibits a (slightly) better running time. In this section we improve the analysis of this algorithm: we first extend it to the multi-dimensional case, provide a more rigorous analysis and finally compare it with other PKP solvers.

### 6.1 The novel attack

The approach we consider is reported in Algorithm 2, while a graphical description is depicted in Figure 4.

In the next Proposition we show that the algorithm is correct, i.e., that in each call it always halts and returns a solution for the PKP instance $\{\mathbf{H}, \mathbf{V}, \mathbf{E}\}$. The proof of the proposition helps in understanding the operating principle.

**Proposition 8** *Algorithm 2 is correct, i.e., it always returns a solution for the input PKP instance.*

*Proof.* In the following, we will indicate by $\pi$ the solution to PKP and $\widetilde{\mathbf{V}} = \pi(\mathbf{V})$. We observe that, since $\pi$ is a solution for the PKP instance $\{\mathbf{H}, \mathbf{V}, \mathbf{E}\}$, that is $\pi(\mathbf{V})\mathbf{H}^\top = \mathbf{E}$, then

$$\mathbf{E}\mathbf{S}^\top = \sigma\big(\pi(\mathbf{V})\big)\big(\mathbf{S}\sigma(\mathbf{H})\big)^\top, \quad \forall \mathbf{S} \in \mathbb{F}_q^{r' \times r}, \ \forall \sigma \in S_n. \tag{11}$$

In line 1 of the algorithm, a subcode of the dual $\mathcal{C}^\perp$ is found. We denote by $\mathbf{H}^* \in \mathbb{F}_q^{d \times n}$ a generator matrix for such a subcode. Observe that $\mathbf{H}^* = \mathbf{S}\mathbf{H}$ for some $\mathbf{S} \in \mathrm{GL}_{d,r}$. The permutation $\sigma$ we compute in line 3 is only considered to simplify the description, since its action on $\mathbf{H}^*$ is such that $\widehat{\mathbf{H}} = \sigma(\mathbf{H}^*)$ has the only non null columns in positions $\{n - r + u - w, \cdots, n - r + u\}$. Thanks to the equality in (11), we have

$$\widehat{\mathbf{E}} = \mathbf{E}\mathbf{S}^\top = \sigma\big(\pi(\mathbf{V})\big)\sigma(\mathbf{S}\mathbf{H})^\top = \widehat{\mathbf{V}}\widehat{\mathbf{H}}^\top.$$

**Algorithm 2:** Combinatorial algorithm to solve PKP

**Data:** $w, w_1, w_2, d, u \in \mathbb{N}$, such that $w \leqslant n$, $w = w_1 + w_2$, $d \leqslant r$, $u \leqslant n - r$
**Input:** $\mathbf{H} \in \mathbb{F}_q^{r \times n}$, $\mathbf{E} \in \mathbb{F}_q^{\ell \times r}$, $\mathbf{V} \in \mathbb{F}_q^{\ell \times n}$
**Output:** $\pi \in S_n$ such that $\pi(\mathbf{V})\mathbf{H}^\top = \mathbf{E}$

/* Find subcode with support size $w$ */
1 Use ISD to find $\mathbf{H}^*$, generator matrix of $\mathcal{B} \subseteq \mathcal{C}^\perp$, with dimension $d$ and support size $w$;

/* Construct the PKP instance $\{\widehat{\mathbf{H}}, \mathbf{V}, \widehat{\mathbf{E}}\}$, and find solutions */
2 Compute $\mathbf{S} \in \mathrm{GL}_{d,r}$ such that $\mathbf{H}^* = \mathbf{S}\mathbf{H}$;
3 Compute $\sigma \in S_n$ such that $\mathrm{Supp}\big(\sigma(\mathbf{H}^*)\big) = \{n - r + u - w + 1, \cdots, n - r + u\}$;
4 Set $\widehat{\mathbf{E}} = \mathbf{E}\mathbf{S}^\top$, $\widehat{\mathbf{H}} = \sigma(\mathbf{H}^*)$;
5 Set $K_1 = \{n - r + u - w + 1, \cdots, n - r + u - w_2\}$, $K_2 = \{n - r + u - w_2, \cdots, n - r + u\}$;
6 Prepare $\mathcal{K}_1 = \left\{ \left(\mathbf{Y}_1, \mathbf{Y}_1 \widehat{\mathbf{H}}_{K_1}^\top\right) \middle| \mathbf{Y}_1 \in \mathscr{S}_{w_1}(\mathbf{V}) \right\}$, $\mathcal{K}_2 = \left\{ \left(\mathbf{Y}_2, \widehat{\mathbf{E}} - \mathbf{Y}_2 \widehat{\mathbf{H}}_{K_2}^\top\right) \middle| \mathbf{Y}_2 \in \mathscr{S}_{w_2}(\mathbf{V}) \right\}$;
7 Compute $\mathcal{K} = \mathcal{K}_1 \bowtie \mathcal{K}_2$;

/* Construct the PKP instance $\{\breve{\mathbf{H}}, \mathbf{V}, \breve{\mathbf{E}}\}$, and find solutions */
8 Compute $\mathbf{M} \in \mathrm{GL}_{r,r}$ such that $\mathbf{M}\sigma(\mathbf{H}) = (\mathbf{U}, \mathbf{I}_r)$;
9 Set $\breve{\mathbf{H}}$ as the matrix formed by the rows of $(\mathbf{U}, \mathbf{I}_r)$ at positions $\{d + 1, \cdots, u\}$ and $\{1, \cdots, n - r + u\}$;
10 Set $\breve{\mathbf{E}}$ as the matrix formed by the columns of $\mathbf{E}\mathbf{M}^\top$ in positions $\{d + 1, \cdots, u\}$;
11 Set $L_1 = \{1, \cdots, n - r + u - w\}$ and $L_2 = \{n - r + u - w + 1, \cdots, n - r + u\}$;
12 Prepare $\mathcal{L}_1 = \left\{ \left(\mathbf{X}_1, \mathbf{X}_1 \breve{\mathbf{H}}_{L_1}^\top\right) \middle| \mathbf{X}_1 \in \mathscr{S}_{n-r+u-w}(\mathbf{V}) \right\}$, $\mathcal{L}_2 = \left\{ \left(\mathbf{X}_2, \breve{\mathbf{V}} - \mathbf{X}_2 \breve{\mathbf{H}}_{L_2}^\top\right) \middle| \mathbf{X}_2 \in \mathcal{K} \right\}$;
13 $\mathcal{L} \leftarrow \mathcal{L}_1 \bowtie \mathcal{L}_2$;

/* Test each produced candidate */
14 Set $\breve{\mathbf{U}}$ as the matrix formed by the last $r - u$ rows of $\mathbf{U}$;
15 **for** $\mathbf{X} \in \mathcal{L}$ **do**
16  Set $\mathbf{X}' = \breve{\mathbf{E}} - \mathbf{X}_{\{1,\cdots,n-r\}} \breve{\mathbf{U}}^\top$;
17  **if** $(\mathbf{X}, \mathbf{X}') \in \mathscr{S}_n(\mathbf{V})$ **then**
18  Compute $\pi$ such that $\sigma\big(\pi(\mathbf{X}, \mathbf{X}')\big) = \mathbf{V}$;
19  Return $\pi$;

Given that $\widehat{\mathbf{H}}$ has only $w$ non null columns, $\{\widehat{\mathbf{H}}, \widehat{\mathbf{V}}, \widehat{\mathbf{E}}\}$ is an easier to solve PKP instance. All the solutions for this PKP instance are found with a meet-in-the-middle approach and are contained in $\mathcal{K}$. Notice that $\sigma(\widetilde{\mathbf{V}})_{K_1 \cup K_2}$ is guaranteed to be in $\mathcal{K}$. Indeed, given that the only non null columns of $\widehat{\mathbf{H}}$ are in positions $K_1 \cup K_2 = \{n - r + u - w + 1, \cdots, n - r + u\}$, we can write

$$(\mathbf{0}_{d \times (n-r+u-w)}, \sigma(\widetilde{\mathbf{V}})_{K_1 \cup K_2}, \mathbf{0}_{d \times (r-u)}) \widehat{\mathbf{H}}^\top = \widehat{\mathbf{E}}.$$

The list $\mathcal{K}$ contains all matrices $(\mathbf{Y}_1, \mathbf{Y}_2) \in \mathscr{S}_w(\widehat{\mathbf{V}})$ such that

$$(\mathbf{0}_{d \times (n-r+u-w)}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{0}_{d \times (r-u)}) \widehat{\mathbf{H}}^\top = \widehat{\mathbf{E}},$$

hence it contains also $\sigma(\widetilde{\mathbf{V}})_{K_1 \cup K_2}$.

For the matrices $\breve{\mathbf{H}}$ and $\breve{\mathbf{E}}$ which are computed in lines 9 and 10 of the algorithm, it holds that $\breve{\mathbf{M}}\sigma(\mathbf{H})$ and $\mathbf{E} = \mathbf{E}\breve{\mathbf{M}}^\top$, where $\breve{\mathbf{M}} \in \mathrm{GL}_{r-d,r}$ is obtained by replacing the first $d$ columns of $\breve{\mathbf{M}}$ with zeros. Hence, (11) continues to hold and, consequently, we have that $\widetilde{\mathbf{V}}_{L_1 \cup L_2}$ must be among the solutions of the PKP instance represented by $\{\breve{\mathbf{H}}, \breve{\mathbf{V}}, \breve{\mathbf{E}}\}$. Notice that $L_1 \cup L_2 = \{1, \cdots, n - (r - u)\}$ and that, to solve the PKP instance, we use again a meet-in-the-middle. In particular, we consider all elements of $\mathscr{S}_{n-r+u-w}(\mathbf{V})$ to build the list $\mathcal{L}_1$ (which is associated to the positions $\{1, \cdots, n - r + u - w\}$), while to build $\mathcal{L}_2$ we use the elements in $\mathcal{K}$. Indeed, we observe that $L_2 = \{n - r + u - w + 1, \cdots, n - (r - u)\} = K_1 \cup K_2$, hence, $\mathcal{L}_2$ contains candidates for $\sigma(\widetilde{\mathbf{V}})_{K_1 \cup K_2}$. Given that the actual $\widetilde{\mathbf{V}}_{K_1 \cup K_2}$ is guaranteed to be in $\mathcal{K}$, we can construct $\mathcal{L}_2$ from $\mathcal{K}$, which avoids to enumerate all elements in $\mathscr{S}_w(\mathbf{V})$.

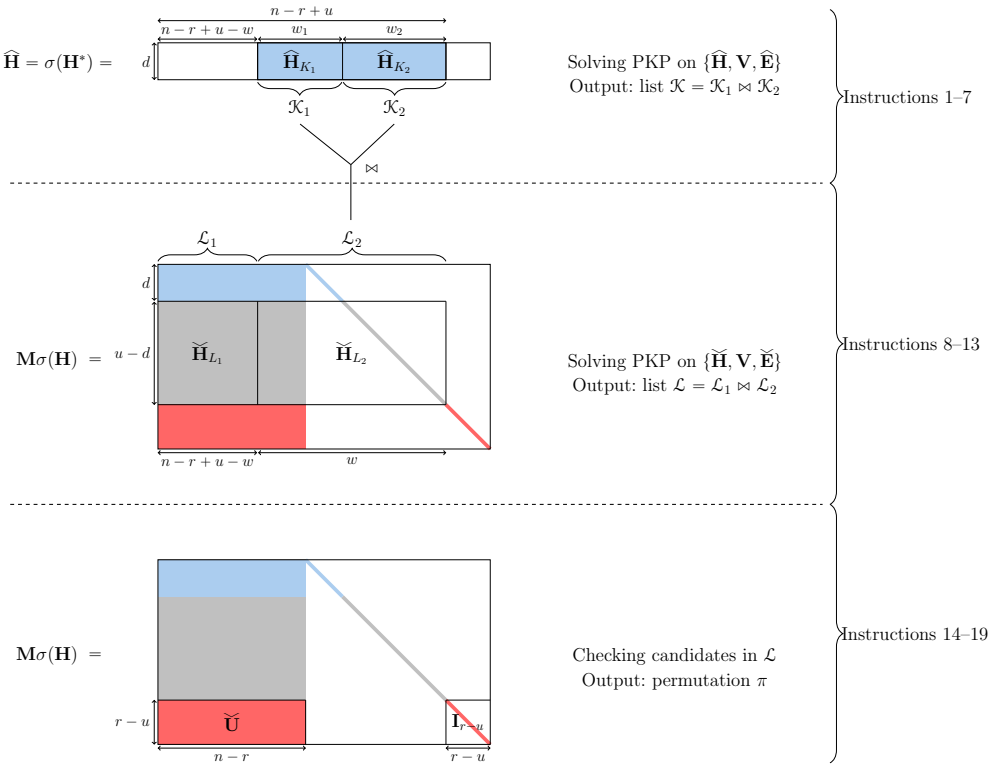



Fig. 4: Representation of the combinatorial attack described in Algorithm 4. Different colors highlight the parity-check equations which are used in each stage.

The list produced in line 13 is guaranteed to contain $\widetilde{\mathbf{V}}_{L_1 \cup L_2}$. Notice that each $\mathbf{X} \in \mathcal{L}$ contains is a matrix with $n-(r-u)$ columns; to compute the remaining $r-u$ columns, which are denoted as $\mathbf{X}'$ in the algorithm, we we exploit the systematic form $(\mathbf{U}, \mathbf{I}_r)$. Notice that, if $(\mathbf{X}, \mathbf{X}') \in \mathscr{S}_n(\mathbf{V})$, then this implies that $\sigma(\widetilde{\mathbf{V}}) = (\mathbf{X}, \mathbf{X}') = \sigma\big(\pi(\mathbf{V})\big)$. From this relation, $\pi$ can be easily found. □

In the following Proposition we derive the time complexity of Algorithm 2. Notice that, as for KMP, we express the cost in terms of linear algebra and lists operations.

**Proposition 9** *Let $d, w_1, w_2$ such that $\breve{N}_{w_1+w_2,d} > 1$. Then, Algorithm 2 runs in average time*

$$T_{\mathsf{ISD}}^{(d)}(n, r, w) + T_{\mathcal{K}} + T_{\mathcal{L}} + |\mathcal{L}|,$$

*where $w = w_1 + w_2$, $|\mathcal{K}| \doteq \max\left\{\frac{n!}{(n-w)!} q^{-d\ell}\ ,\ 1\right\}$ and*

$$T_{\mathcal{K}} \doteq \frac{n!}{(n-w_1)!} + \frac{n!}{(n-w_2)!} + \frac{(n!)^2 q^{-d\ell}}{(n-w_1)!(n-w_2)!},$$

$$T_{\mathcal{L}} \doteq \frac{n!}{(r+w-u)!} + |\mathcal{K}| + \frac{n!}{(r+w-u)!} \cdot |\mathcal{K}| \cdot q^{-\ell(u-d)},$$

$$|\mathcal{L}| \doteq \frac{(n-w)! q^{-(u-d)\ell}}{(r-u)!} \cdot |\mathcal{K}|$$

*Proof.* Since $\breve{N}_k(w, d) > 1$, we expect $\mathcal{C}^\perp$ to contain at least a subcode with dimension $d$ and support size $w$. To find such a subcode, we have a cost given by $T_{\mathsf{ISD}}^{(d)}(n, r, w)$.

Steps 2–4 are obtained with some basic linear algebra, so we omit their cost. The cost of building the lists $\mathcal{K}_1$ and $\mathcal{K}_2$ is given by

$$|\mathcal{K}_1| + |\mathcal{K}_2| = \frac{n!}{(n-w_1)!} + \frac{n!}{(n-w_2)!}.$$

Using an efficient strategy to find collisions (e.g., permutation plus binary search), searching for collisions would take time $|\mathcal{K}_1| + |\mathcal{K}_2|$. Notice that every time we find a collision between two elements $(\mathbf{Y}_1, \mathbf{Y}_2)$, we need to check if there are not repeated columns. The basic schoolbook algorithm would take time $O(\ell w^2)$. Given that the cost is already polynomial and that it can be significantly reduced with more clever approaches, we will omit it. Assuming that each term $\mathbf{Y}_1\breve{\mathbf{H}}_{K_1}^\top$ and $\breve{\mathbf{E}} - \mathbf{Y}_2\breve{\mathbf{H}}_{K_2}^\top$ is a random $\ell \times d$ matrix over $\mathbb{F}_q$, we can estimate the average number of collisions as

$$\frac{|\mathcal{K}_1| \cdot |\mathcal{K}_2|}{q^{d\ell}} = \frac{(n!)^2 q^{-d\ell}}{(n-w_1)!(n-w_2)!}$$

Consequently, we estimate the cost of instructions 2–7 as

$$T_{\mathcal{K}} \doteq \frac{n!}{(n-w_1)!} + \frac{n!}{(n-w_2)!} + \frac{(n!)^2 q^{-d\ell}}{(n-w_1)!(n-w_2)!}.$$

After collisions are filtered, $\mathcal{K}$ contains, on average, $\max\left\{\frac{n!}{(n-w)!}q^{-d\ell}\ ,\ 1\right\}$ elements. Notice that we need to use the max operator since, knowing that the algorithm is correct, existence of at least one solution is guaranteed. Hence, we set $|\mathcal{K}| = \max\left\{\frac{n!}{(n-w)!}q^{-d\ell}\ ,\ 1\right\}$.

Steps 8–11 involve only linear algebra, hence their cost will be neglected. The cost to build $\mathcal{L}_1$ and $\mathcal{L}_2$ is, again, estimated with the number of elements. Notice that $|\mathcal{L}_1| = \frac{n!}{n-(n-r+u-w)!} = \frac{n!}{(r+w-u)!}$ while $\mathcal{L}_2$ has average number of elements given by $|\mathcal{K}|$. Given that $\mathbf{X}_1\breve{\mathbf{H}}_{L_1}^\top$ and $\breve{\mathbf{V}} - \mathbf{X}_2\breve{\mathbf{H}}_{L_2}^\top$ are $\ell \times (u-d)$ matrices, we assume that each pair of list elements collides with probability $q^{-\ell(u-d)}$. Hence, the average number of collisions is

$$\begin{aligned}\frac{|\mathcal{L}_1| \cdot |\mathcal{L}_2|}{q^{\ell(u-d)}} &= \frac{n!}{(r+w-u)!} \cdot |\mathcal{K}| \cdot q^{-\ell(u-d)} \\ &= \max\left\{\frac{(n!)^2 q^{-u\ell}}{(r+w-u)!(n-w)!}\ ,\ \frac{n! q^{-\ell(u-d)}}{(r+w-u)!}\right\}.\end{aligned}$$

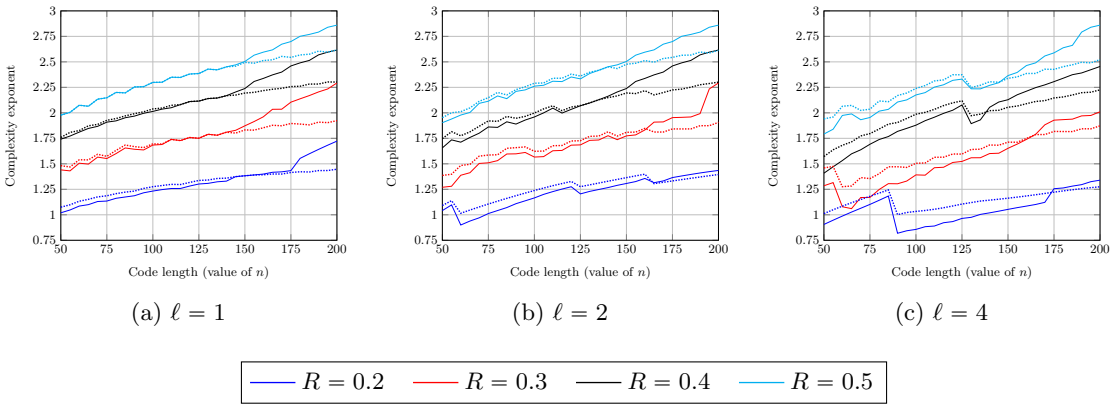
Consequently, the cost of executing steps 12–13 is

$$T_{\mathcal{L}} \doteq \frac{n!}{(r+w-u)!} + |\mathcal{K}| + \frac{n!}{(r+w-u)!} \cdot |\mathcal{K}| \cdot q^{-\ell(u-d)}.$$

To conclude the proof, we need to estimate the cost of going through instructions 14-19, i.e., to test each element of $\mathcal{L}$. Each test requires basic matrix operations, hence we can use $|\mathcal{L}|$ as an estimate for the running time. We consider that for each matrix $\mathbf{X}_2$ in $\mathcal{L}_2$, the number of matrices $\mathbf{X}_1$ that i) do not have columns identical to those of $\mathbf{X}_2$, and ii) provide a collision can be obtained as $\frac{(n-w)!q^{-(u-d)\ell}}{\left(n-w-(n-r+u-w)\right)!} = \frac{(n-w)!q^{-(u-d)\ell}}{(r-u)!}$. Multiplying this quantity by the number of elements in $\mathcal{L}_2$, we obtain that the average size of $\mathcal{L}$ is $\frac{(n-w)!q^{-(u-d)\ell}}{(r-u)!} \cdot |\mathcal{K}|$. □

### 6.2 Comparison with KMP

In this section we compare the performances of our algorithm with those of KMP. For both algorithms, we have considered the finite regime, that is: for KMP, we have employed the estimate of $T_{\mathsf{KMP}}$ resulting from Proposition 5, while for our algorithm we have referred to Proposition 9. In Figures 5a, 5b, 5c we report the running times of the two attacks for different code rates $R$, several (but not asymptotically large) code lengths $n$ and the cases of $\ell = 1$, 2 and 4. The figures report the complexity exponent, that is: for a running time equal to $T$, we have displayed $\frac{1}{n}\log_2\left(T\right)$. As it can be seen, our algorithm can indeed be faster than KMP in several occasions. Indeed, regardless of $\ell$ and unless $n$ gets too large, our algorithm performs better than KMP, since the associated complexity exponents are lower. We also notice that, for larger value of $\ell$, the improvement becomes more significant. When approaching the asymptotic regime (i.e., when $n$ gets larger), our algorithm becomes slower.




(a) $\ell = 1$ (b) $\ell = 2$ (c) $\ell = 4$

Fig. 5: Complexity exponents for our algorithm and KMP, for $\ell = 1$. Continuous lines are referred to our algorithm, dotted lines are referred to the KMP algorithm.

To have more insight on how our algorithm compares with KMP, in Table 2 we have considered some other examples. To analyze parameters with practical interest, we have focused on the PKP-DSS instances recommended in [6]; the rows associated to these parameters are highlighted in grey. For each instance, we have frozen the values of $n$ and $r$, increased the values of $\ell$ and have, consequently, recomputed $q$. For our algorithm, the table also contains the parameters which optimize the attack. We can see that, in several cases, our algorithm can be significantly faster than KMP.

In particular, the speed-up gets more relevant for large values of $\ell$. This is due to the fact that, when $\ell$ increases, the required value of $q$ gets lower. Recalling Proposition 1, this implies that we can run our algorithm with more aggressive parameters, that is, larger values of $d$. This makes the initial filtering stage more powerful, since when $d$ gets larger, the elements in $\mathcal{K}_1$ and $\mathcal{K}_2$ must collide on a larger number of equations. Notice that our algorithm makes sense when $w$ and $d$ are such that $w < n - r + d$. Indeed, $d$-dimensional subcodes with support size $n - r + d$ can be found with a basic Gaussian elimination. Such subcodes are already, implicitly, employed in the KMP algorithm. To do better than KMP, we need to use subcodes with the same dimension but a lower support size: this happens only if $w < n - r + d$. However, the existence of $d$-dimensional subcodes with support size lower than $n - r + d$ is not guaranteed. If such subcodes do not exist, then our algorithm should not outperform KMP. For example, consider the first two instances with $(n, r) = (106, 48)$. Our algorithm is optimized by choosing $d = 21$ and $w = 79$, but $n - r + d = 79$: as we can see from the table, our algorithm has essentially the very same running time of KMP.

Finally, we observe that the relevant term in the complexity of our algorithm is never $T_{\mathsf{ISD}}^{(d)}(n, r, w)$. In other words, this implies that the cost of ISD is always much smaller than the cost of creating and merging the lists.

### 6.3 Comparison with the attack in [19]

To make another practical example, we consider the PKP instances recommended in [18], for which $(n, r, q, \ell)$ correspond to $(38, 16, 2, 10)$ and $(42, 16, 2, 11)$. Notice that we are already referring to the non homogeneous version, i.e., we are considering $r = m + 1$. In Table 3 we consider the performances of Algorithm 2, and compare them with those of the attack in [19] and with KMP. There instances have originally been proposed for security levels of 79 and 89 bits, but [19] shows how they can be attacked with costs approximately given by $2^{63}$ and $2^{77}$, respectively. Coherently with the literature on PKP (and with the analysis we performed in this paper), the estimates in [19] count the number of matrix by matrix multiplications. Notice that, interestingly, the attack in [19] has some similarities with our approach: for instance, it starts by repeatedly calling ISD to find low weight codewords.

As we can see from the table, the running time of our algorithm is lower than that of [19]. Namely, the gain is approximately 5 bits for the first instance, and 8 bits for the second one. Also, our algorithm works regardless of the considered finite field, while the one in [19] is specific to the binary case.

Table 2: Comparison between the running times (in $\log_2$ units) of KMP and Algorithm 2

| $(n, r, q, \ell)$ | KMP | $(d, w, w_1, u)$ | Algorithm 2 |
|---|---|---|---|
| $(69, 42, 251, 1)$ | 127.37 | $(1, 22, 2, 16)$ | 125.47 |
| $(69, 42, 17, 2)$ | 125.26 | $(14, 40, 20, 27)$ | 121.34 |
| $(69, 42, 7, 3)$ | 124.32 | $(3, 24, 5, 16)$ | 118.45 |
| $(69, 42, 5, 4)$ | 118.85 | $(11, 36, 18, 22)$ | 110.87 |
| $(69, 42, 3, 5)$ | 128.00 | $(8, 30, 12, 22)$ | 115.16 |
| $(69, 42, 3, 6)$ | 118.10 | $(10, 33, 16, 20)$ | 101.72 |
| $(69, 42, 3, 7)$ | 112.20 | $(8, 30, 15, 16)$ | 91.04 |
| $(69, 42, 2, 8)$ | 127.00 | $(12, 34, 17, 24)$ | 105.77 |
| $(69, 42, 2, 9)$ | 119.64 | $(10, 31, 15, 20)$ | 96.52 |
| $(94, 55, 509, 1)$ | 190.82 | $(1, 31, 2, 22)$ | 189.77 |
| $(94, 55, 23, 2)$ | 190.48 | $(2, 32, 3, 22)$ | 186.43 |
| $(94, 55, 11, 3)$ | 178.89 | $(2, 30, 4, 18)$ | 173.75 |
| $(94, 55, 5, 4)$ | 189.36 | $(4, 34, 6, 23)$ | 179.85 |
| $(94, 55, 5, 5)$ | 172.07 | $(3, 31, 6, 17)$ | 163.40 |
| $(94, 55, 3, 6)$ | 185.98 | $(18, 54, 27, 34)$ | 172.52 |
| $(94, 55, 3, 7)$ | 175.68 | $(14, 49, 24, 27)$ | 159.03 |
| $(94, 55, 3, 8)$ | 166.51 | $(11, 45, 22, 22)$ | 148.03 |
| $(94, 55, 2, 9)$ | 190.76 | $(11, 42, 16, 29)$ | 169.04 |
| $(106, 48, 4093, 1)$ | 257.40 | $(21, 79, 39, 23)$ | 257.40 |
| $(106, 48, 67, 2)$ | 256.97 | $(21, 79, 39, 23)$ | 256.97 |
| $(106, 48, 17, 3)$ | 256.41 | $(1, 40, 2, 21)$ | 251.56 |
| $(106, 48, 11, 18)$ | 245.74 | $(1, 39, 3, 18)$ | 241.13 |
| $(106, 48, 7, 5)$ | 245.71 | $(1, 39, 3, 18)$ | 239.95 |
| $(106, 48, 5, 6)$ | 245.72 | $(2, 41, 5, 19)$ | 238.33 |
| $(106, 48, 5, 7)$ | 237.49 | $(2, 40, 6, 16)$ | 227.60 |
| $(106, 48, 3, 8)$ | 251.90 | $(3, 43, 6, 22)$ | 241.35 |
| $(106, 48, 3, 9)$ | 244.82 | $(3, 42, 7, 19)$ | 230.59 |

For a completely fair comparison, we observe that our algorithm requires an exponential amount of memory, while the one in [19] has a much lower space complexity. In principles, an algorithm running in time $T$ and using a large memory $M$ should be somehow penalized, in the sense that its overall cost should be larger than $T$. This additional cost is normally neglected and only the running time is considered. Yet, for the sake of completeness, we briefly comment about this fact. Establishing how a large memory usage affects the performances of an algorithm is a rather involved task. We will stick to the analysis in [10], in which the authors conclude that the logarithmic cost seems to be the most appropriate one, for the case of ISD algorithms. Given that the large space complexity of advanced ISD algorithms is essentially due to operations with lists, it makes sense to extrapolate this result and apply it to also to our case. Hence, we consider an overall cost of $T \cdot \log_2(M)$. Given that, for Algorithm 2, we have $M \approx T$, we can use $T \log_2(T)$ as an estimate of its cost.

Even if we assume no penalty for the attack in [19], our algorithm remains competitive. Indeed, the costs would become approximately 63 and 75, which are in the same ballpark of [19]. Under other models (for instance, the cubic root model which would lead to a cost of $T\sqrt[3]{T} = T^{\frac{4}{3}}$), the cost of our algorithm would become much larger.

Table 3: Comparison between the running times (in $\log_2$ units) of KMP, the attack in [19] and algorithm 2, for the instances recommended in [18]

| $(n, r, q, \ell)$ | KMP | [19] | $(d, w, w_1, u)$ | Algorithm 2 |
|---|---|---|---|---|
| $(38, 16, 2, 10)$ | 74.9 | 63 | $(5, 21, 10, 10)$ | 57.7 |
| $(42, 16, 2, 11)$ | 87.4 | 77 | $(6, 26, 13, 11)$ | 69.2 |

## 7 Further considerations

Arguably, the interest in PKP and SEP is mainly due to their cryptographic applications. At the best of our knowledge, the problems have been used only in the design of signature schemes in the Fiat-Shamir paradigm. Yet, we cannot exclude that, in the future, other applications appear, e.g., signatures in the hash&sign paradigm or encryption schemes. Analogously, most of the attention has been dedicated to the mono-dimensional PKP. The use of multi-dimensional PKP has only been considered in [18], but [19] and this paper show that the recommended instances have a security level which is significantly below the claimed one. Yet, this is not enough to conclude that multi-dimensional PKP is less useful, with respect to mono-dimensional PKP. Following this line of reasoning, some questions arise naturally; for instance:

*Can a PKP-based encryption scheme, or a hash&sign signature scheme, be competitive?*

*Can multi-dimensional PKP be preferable than mono-dimensional PKP, in some cases?*

In the remainder of this section, we argue why these questions may admit, in principles, a positive answer. Namely, we show that to achieve a running time of at least $2^{128}$ operations (corresponding to a security level of 128 bits) the required input size for PKP (equivalently, SEP) can be rather small, when compared to other problems. This implies that, potentially, PKP and SEP may be employed to design cryptographic schemes with competitive performances. Notice that we adopt a purely speculative point of view, that is, we do not propose any specific construction but only consider the performances that hypothetical such schemes may achieve. Yet, these results hint at the fact that these problems are worth looking into.

### 7.1 Relevant quantities and scenarios

As it is common when studying hard problems, we first focus on the input size. For PKP, the input is constituted by $\mathbf{H}$, of size $r \times n$, and $\mathbf{V}$, of size $\ell \times n$; both matrices take values in $\mathbb{F}_q$. Notice that, for both matrices and without loss of generality, we can employ a convenient representation and consider the RREF form. Indeed, this allows to save some space, since the identity matrices can be excluded from the input. Consequently, to represent such matrices, the number of bits we need is

$$\text{Size}(\mathbf{H}) = r(n-r)\log_2(q),$$
$$\text{Size}(\mathbf{V}) = \ell(n-\ell)\log_2(q).$$

For the overall input size, we can consequently consider $\text{Size}(\mathbf{H}) + \text{Size}(\mathbf{V})$. Notice that, under a cryptographic point of view, the input size can be interpreted as the public key plus ciphertext size of an hypotethical encryption scheme in which $\mathbf{H}$ constitutes the public key and $\mathbf{V}$ is the ciphertext.

Notice that, when considering signature schemes in the Fiat-Shamir paradigm, $\mathbf{H}$ can be fixed or generated at random starting from some seed. Instead, the size of (some) exchanged messages is, more or less, equal to that of $\mathbf{V}$. Consequently, in this scenario, it makes sense to neglect the size of $\mathbf{H}$ and consider only that of $\mathbf{V}$.

### 7.2 The case study of 128 bits of security

As in several other works, we focus on the case of 128 bits of classical security, which is the minimum security level which is required, nowadays, for cryptographic schemes. We first consider the minimum input size we need, for PKP, to have that all known attacks have a running time not lower than $2^{128}$. In other words, we consider several code rates and, for each value of $R$, find the values of $n$, $q$ and $\ell$ such that all known attacks run in time $\geqslant 2^{128}$ and the value of $\text{Size}(\mathbf{H})+\text{Size}(\mathbf{V})$ is minimized. We first focus on the mono-dimensional case. For the corresponding parameters, see Table 4, where we additionally compare with the results in [9], in which the authors derive the minimum input size for the Syndrome Decoding Problem (SDP) with the low Hamming weight and high Hamming weight requirements. As we can see from the Table, the PKP can achieve the same complexity with a much smaller input. We have also considered how the problem behaves, when switching to the multi-dimensional version. Interestingly, we found that the input size can

be reduced significantly. Indeed, when $\ell$ gets larger, we can use smaller values for $q$, and this has a positive impact on Size($\mathbf{V}$). For instance, considering the same rate $R \approx 0.51$, we can choose $n = 70$, $r = 32$, $q = 2$ and $\ell = 10$, yielding to an input size of approximately 0.23 kB. Notice that this reduction is mostly due to the fact that representing $\mathbf{H}$ becomes significantly less costly: instead of the 1.12 kB required for the mono-dimensional case, we here need only 0.15 kB.

We now focus on minimizing just the size of $\mathbf{V}$. For $\ell = 1$, we found that the optimal choice is $n = 69$, $q = 239$ and $r = 41$ (the code rate is approximately 0.4), yielding to Size($\mathbf{V}$) $\approx 67$ B. When switching to the multi-dimensional case, we can obtain approximately the same sizes with essentially the same $n$ but a much smaller $q$. For instance, choosing $n = 68$, $q = 17$ and $\ell = 2$, we obtain, in practice, the same value for Size($\mathbf{V}$). We observe that, using the same $n$, the solution to PKP has the same size; however, we can use a much smaller value for $q$, and this should make the arithmetic faster.

Table 4: Minimum input size, in kB, and corresponding parameters, for different problems and 128 bits of complexity

| Problem | Parameters | Min. Input Size |
|---|---|---|
| SDP, Low Weight | $q = 2$, $R = 0.326$ | 46.75 |
| SDP, High Weight | $q = 3$, $R = 0.369$ | 12.38 |
| PKP, $\ell = 1$ | $n = 62$, $q = 653$, $r = 30$ | 1.19 |

## 8 Conclusions

We have studied the hardness to solve the PKP and the SEP. First, we have shown that the two problems are actually equivalent, hence, all solvers for the former can also be used to solve the latter. Despite this result is based on a very simple reduction, to the best of our knowledge, this is the first time it is made explicit. Then, we have deeply studied the performance of state-of-the-art solvers. For what concerns the KMP algorithm, we have generalized it to the multi-dimensional case (i.e., when $\ell > 1$) and have derived its complexity in the asymptotic regime. Our analysis shows that, perhaps surprisingly, the running time does not depend on the value of $\ell$. Also, the algorithm runs in time which is super-exponential in the code length. We have then thoroughly analyzed the algorithm we introduced in [23], extended it to the multi-dimensional case (regardless of the finite field size) and compared it with KMP and the attack in [19], which is specific to the binary field. Our analysis shows that, in the finite regime, our algorithm is in several cases faster than other approaches; instead, in the asymptotic regime, KMP has better performance. Finally, we have considered how PKP and SEP behave in terms of input size. We have shown that they can achieve practical security levels with rather compact inputs, when compared to other problems (say, SDP). Also, switching to the multi-dimensional version has no practical impact on the security, while it can lead to a significantly reduced input size. This analysis hints at the fact that, potentially, PKP and SEP can be used to design very promising quantum-safe cryptographic primitives.

## References

1. Bardet, M., Otmani, A., Saeed-Taha, M.: Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In: 2019 IEEE International Symposium on Information Theory (ISIT). pp. 2464–2468. IEEE (2019)
2. Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: On the computational hardness of the code equivalence problem in cryptography. Advances in Mathematics of Communications **0**, – (2022)
3. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H.: On the security of the permuted kernel identification scheme. In: Brickell, E.F. (ed.) Advances in Cryptology — CRYPTO' 92. Lecture Notes in Computer Science, vol. 740, pp. 305–311. Springer (1992)
4. Berger, T.P., Gueye, C.T., Klamti, J.B.: A NP-complete problem in coding theory with application to code based cryptography. In: International Conference on Codes, Cryptology, and Information Security. pp. 230–237. Springer (2017)

5. Beullens, W.: Not Enough LESS: An Improved Algorithm for Solving Code Equivalence Problems over $\mathbb{F}_q$. In: International Conference on Selected Areas in Cryptography. pp. 387–403. Springer (2020)
6. Beullens, W., Faugère, J.C., Koussa, E., Macario-Rat, G., Patarin, J., Perret, L.: PKP-based signature scheme. In: F. Hao, S.R., Gupta, S.S. (eds.) Progress in Cryptology – INDOCRYPT 2019. Lecture Notes in Computer Science, vol. 11898, pp. 3–22. Springer, Cham (2019)
7. Bidoux, L., Gaborit, P.: Shorter Signatures from Proofs of Knowledge for the SD, MQ, PKP and RSD Problems. arXiv preprint arXiv:2204.02915 (2022)
8. Boidje, B.O., Gueye, C.T., Dione, G.N., Klamti, J.B.: Quasi-Dyadic Girault Identification Scheme. In: International Conference on Codes, Cryptology, and Information Security. pp. 307–321. Springer (2019)
9. Bricout, R., Chailloux, A., Debris-Alazard, T., Lequesne, M.: Ternary syndrome decoding with large weight. In: International Conference on Selected Areas in Cryptography. pp. 437–466. Springer (2020)
10. Esser, A., May, A., Zweydinger, F.: McEliece needs a break–solving McEliece-1284 and quasi-cyclic-2918 with modern ISD. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 433–457. Springer (2022)
11. Feneuil, T.: Building MPCitH-based Signatures from MQ, MinRank, Rank SD and PKP. Cryptology ePrint Archive (2022)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco (1979)
13. Georgiades, J.: Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology **5**(2), 133–137 (1992)
14. Girault, M.: A (non-practical) three-pass identification protocol using coding theory. In: International Conference on Cryptology. pp. 265–272. Springer (1990)
15. Guruswami, V.: Introduction to coding theory, lecture 2: Gilbert-varshamov bound. University Lecture (2010)
16. Jaulmes, É., Joux, A.: Cryptanalysis of PKP: a new approach. In: K., K. (ed.) Public Key Cryptography. PKC 2001. Lecture Notes in Computer Science, vol. 1992, pp. 165–172. Springer, Berlin, Heidelberg (2001)
17. Koussa, E., Macario-Rat, G., Patarin, J.: On the complexity of the Permuted Kernel Problem. Cryptology ePrint Archive, Report 2019/412 (2019), `https://ia.cr/2019/412`
18. Lampe, R., Patarin, J.: Analysis of some natural variants of the PKP algorithm. Cryptology ePrint Archive (2011)
19. Paiva, T.B., Terada, R.: Cryptanalysis of the binary permuted kernel problem. In: International Conference on Applied Cryptography and Network Security. pp. 396–423. Springer (2021)
20. Patarin, J., Chauvaud, P.: Improved algorithms for the permuted kernel problem. In: Stinson, D.R. (ed.) Cryptology — CRYPTO' 93 Proceedings. CRYPTO 1993. Lecture Notes in Computer Science, vol. 773, pp. 391–402. Springer, Berlin, Heidelberg (1993)
21. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Transactions on Information Theory **43**(5), 1602–1604 (1997)
22. Poupard, G.: A realistic security analysis of identification schemes based on combinatorial problems. European Transactions on Telecommunications **8**(5), 471–480 (1997)
23. Santini, P., Baldi, M., Chiaraluce, F.: A Novel Attack to the Permuted Kernel Problem. In: 2022 IEEE International Symposium on Information Theory (ISIT). pp. 1441–1446 (2022). https://doi.org/10.1109/ISIT50566.2022.9834867
24. Sendrier, N.: On the dimension of the hull. SIAM Journal on Discrete Mathematics **10**(2), 282–293 (1997)
25. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Transactions on Information Theory **46**(4), 1193–1203 (2000)
26. Shamir, A.: An efficient identification scheme based on permuted kernels. In: Brassard, G. (ed.) Advances in Cryptology — CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 606–609. Springer (1989)

## Appendix A - Proof of Proposition 1

We have already observed that the bounds $\breve{N}_k(w,d)$ and $\hat{N}_k(w,d)$ are tight, up to a factor which is not greater than $e^1 \approx 2.7183$: so, we can safely use $\breve{N}_k(w,d)$ in place of $N_k(w,d)$. Let $k = Rn$, for a constant $R \in [0;1]$; since $d$ is constant, from (5) we have

$$\frac{\begin{bmatrix} k \\ d \end{bmatrix}_q}{\begin{bmatrix} n \\ d \end{bmatrix}_q} \sim 2^{d(k-n)\log_2(q)} = 2^{dn(R-1)\log_2(q)}.$$

From (4), and neglecting the $o(1)$ (since they vanish for growing $n$), we have that

$$\begin{aligned}\binom{n}{w}(q^d-1)^{w-d} &= \binom{n}{w}(q^d-1)^w(q^d-1)^{-d}\\ &= 2^{nh_{q^d}(\omega)\log_2(q^d)-d\log_2(q^d-1)}\\ &= 2^{d\left(nh_{q^d}(\omega)\log_2(q)-d\log_2(q^d-1)\right)}.\end{aligned}$$

Then, we can write

$$\widetilde{N}_k(w,d) = 2^{nd\log_2(q)h_{q^d}(\omega)-d\log_2(q^d-1)-dn(1-R)\log_2(q)}.$$

Let $\omega^*$ be the minimum $\omega \in [0;1]$ so that $\widetilde{N}_k(\omega n, d) \geqslant 1$. From the above equation, we obtain

$$\omega^* = h_{q^d}^{-1}\left(1-R+\frac{\log_2(q^d-1)}{n\log_2(q)}\right).$$

We further notice that, whenever $q^d \gg 1$, we can further simplify $\log_2(q^d-1) \approx d\log_2(q)$, from which

$$\omega^* \approx h_{q^d}^{-1}\left(1-R+\frac{d}{n}\right).$$

## Appendix B - Proof of Proposition 2

ISD is a randomized, iterative procedure in which the steps in Algorithm 3 are continuously executed until the algorithm successes.

**Algorithm 3:** One iteration of ISD for $d > 1$

**Input:** Code $\mathcal{C}$ generated by $\mathbf{G} \in \mathrm{GL}_{k,n}$, $w, d \in \mathbb{N}$ such that $w \leqslant n-k+d$
**Output:** failure, or generator matrix for $\mathcal{B} \subseteq \mathcal{C}$ with dimension $d$ and support size $w$

Choose uniformly at random $\sigma \in S_n$;
**if** $\mathsf{RREF}\big(\sigma(\mathbf{G}), \{1,\cdots,k\}\big)$ fails **then**
    Report failure;
**else**
    $(\mathbf{I}_d, \mathbf{M}) \leftarrow \mathsf{RREF}\big(\sigma(\mathbf{G}), \{1,\cdots,k\}\big)$
**for** $U \subseteq \{1,\cdots,k\}$ with size $d$ **do**
    $\mathbf{B} \leftarrow$ matrix formed by rows of $\mathbf{M}$ indexed by $U$;
    **if** $\mathbf{B}$ has support size $w-d$ **then**
        Return $\sigma^{-1}\big((\mathbf{I}_d, \mathbf{B})\big)$
Report failure;

Consider the running time in Proposition 2. Notice that the enumerator of the formula corresponds to the cost of each iteration, while the denominator is the success probability of each iteration. For this probability, we consider that the code contains $N_k(w,d)$ subcodes with dimension $d$ and support size $w$. The probability that a chosen permutation is valid for one of them is given by $\frac{\binom{w}{d}\binom{n-w}{k-d}}{\binom{n}{k}}$ and, if we multiply it by the expected number of subcodes, we obtain the average number of subcodes each ISD iteration is able to find. Now: if this product is smaller than 1, then we can deem it as the success probability. Instead, if it is close to (or greater) than 1, then we can assume that every ISD iteration returns a subcode, so that the success probability of every iteration is basically 1. This reasoning explains why we can set the success probability of each iteration as $p^{(d)}(n,k,w) = \min\left\{\frac{\binom{w}{d}\binom{n-w}{k-d}}{\binom{n}{k}}N_k(w,d)\ ;\ 1\right\}$. Finally, we replace $N_k(w,d)$ with the (tight) lower bound $\widehat{N}_k(w,d)$.

## Appendix C - Asymptotics of KMP algorithm

### Proof of Proposition 6

We first notice that the number of elements in $\mathcal{L}$ is always not larger than the number of collisions between $\mathcal{L}_1$ and $\mathcal{L}_2$, that is, $N_{\mathcal{L}_1 \bowtie \mathcal{L}_2}$. Hence, asymptotically, the cost of the algorithm can be assumed as $\max\{|\mathcal{L}_1|\,,\,|\mathcal{L}_2|\,,\,N_{\mathcal{L}_1 \bowtie \mathcal{L}_2}\}$, and the algorithm is optimized when the three quantities are identical. To achieve this, we choose $u_1 = u_2 = \mu n$, where $\mu \in [0;1]$. This guarantees that $|\mathcal{L}_1| = |\mathcal{L}_2|$ and, recalling the asymptotics in Section 2.3, we have[3]

$$L = |\mathcal{L}_1| = |\mathcal{L}_2| = 2^{n\left(h_2(\mu)+\mu\log_2\left(\frac{\mu n}{e}\right)\right)}.$$

Furthermore, it holds that

$$\begin{aligned} N_{\mathcal{L}_1 \bowtie \mathcal{L}_2} &= \frac{L^2}{q^{\ell(n-r-2\mu n)}} \\ &= 2^{n\left(2h_2(\mu)+2\mu\log_2\left(\frac{\mu n}{e}\right)-\ell\log_2(q)(R-2mu)\right)}. \end{aligned}$$

Then, it is easy to see that $L = N_{\mathcal{L}_1 \bowtie \mathcal{L}_2}$ happens when $\mu$ is such that

$$h_2(\mu) + \mu\log_2\left(\frac{\mu n}{e}\right) + \ell\log_2(q)(R-2\mu) = 0. \tag{12}$$

This proves the proposition.

### Proof of Proposition 7

In the asymptotic regime we have $q \sim \left(\frac{n}{e}\right)^{\frac{1}{\ell(1-R)}}$. Consequently, we rewrite (12) as

$$h_2(\mu) + \mu\log_2\left(\frac{\mu n}{e}\right) + \frac{1}{1-R}\log_2\left(\frac{n}{e}\right)(R-2\mu) = 0. \tag{13}$$

It is easy to see that, for any sufficiently large $n$, the above equation always admits a root $\mu$ in the range $\mu^* \in (R/2;1/2]$. Indeed, the function on the left side of the equation is continuous, is positive for $\mu = R/2$ and negative for $\mu = 1/2$: consequently, it must have a root in the range $(R/2;1/2]$. Let $\omega^*$ be the limit of the root of (13), for $n$ going at infinity, and consider that

$$\begin{aligned} &\lim_{n\to\infty} h_2(\mu^*) + \mu^*\log_2\left(\frac{\mu^* n}{e}\right) + \frac{1}{1-R}\log_2\left(\frac{n}{e}\right)(R-2\mu^*) \\ &\quad = \mu^*\log_2\left(\frac{n}{e}\right) + \frac{1}{1-R}\log_2\left(\frac{n}{e}\right)(R-2\mu^*). \end{aligned}$$

Requiring the above limit to be equal to 0, we find

$$\mu^* = \frac{R}{1+R}.$$

Then, we consider that

$$\begin{aligned} c_{\mathsf{KMP}}(\mu^*) &= h_2(\mu^*) + \mu^*\log_2\left(\frac{\mu^* n}{e}\right) \\ &= -(1-\mu^*)\log_2(1-\mu^*) + \mu^*\log_2\left(\frac{n}{e}\right) \\ &= -\frac{1}{1+R}\log_2\left(\frac{1}{1+R}\right) + \frac{R}{1+R}\log_2\left(\frac{n}{e}\right) \\ &= \frac{1}{1+R}\log_2(1+R) + \frac{R}{1+R}\log_2\left(\frac{n}{e}\right). \end{aligned}$$

[3] To ease notation, we here neglect the $o(1)$ terms.