Pseudorandomness of Decoding, Revisited: Adapting OHCP to Code-Based Cryptography

Maxime Bombar, Alain Couvreur, Thomas Debris-Alazard

PSEUDORANDOMNESS OF DECODING, REVISITED: ADAPTING OHCP TO CODE-BASED CRYPTOGRAPHY

MAXIME BOMBAR [1,2,3], ALAIN COUVREUR [2,1], AND THOMAS DEBRIS–ALAZARD [2,1]

Abstract. Recent code-based cryptosystems rely, among other things, on the hardness of the decisional decoding problem. If the search version is well understood, both from practical and theoretical standpoints, the decision version has been less studied in the literature, and little is known about its relationships with the search version, especially for structured variants. On the other hand, in the world of Euclidean lattices, the situation is rather different, and many reductions exist, both for unstructured and structured versions of the underlying problems. For the latter versions, a powerful tool called the OHCP framework (for Oracle with Hidden Center Problem), which appears to be very general, has been introduced by Peikert *et al.* (STOC 2017) and has proved to be very useful as a black box inside reductions.

In this work, we revisit this technique and extract the very essence of this framework, namely the Oracle Comparison Problem (OCP), to show how to recover the support of the error, solving an Oracle with Hidden Support Problem (OHSP), more suitable for code-based cryptography. This yields a new worst-case to average-case search-to-decision reduction for the Decoding Problem, as well as a new average-case to average-case reduction. We then turn to the structured versions and explain why this is not as straightforward as for Euclidean lattices. If we fail to give a search-to-decision reduction for structured codes, we believe that our work opens the way towards new reductions for structured codes, given that the OHCP framework proved to be so powerful in lattice-based cryptography. Furthermore, we also believe that this technique could be extended to codes endowed with other metrics, such as the rank metric, for which no reduction is known.

1. Introduction

Security reductions in post–quantum cryptography. In the last two decades, there has been a longstanding trend to develop reductions between *generic* or even *worst-case* problems, in view to provide security guarantees of some encryption schemes and digital signatures. The most significant part of the known reductions concern lattice–based cryptography. In particular, the worst-case to average-case reductions between various lattice problems ([Reg05, LPR10, SS11, LS15, PRS17, RSW18, PMS21]) provide a very convincing argument to assert that the security of cryptographic primitives rest only on the worst-case hardness of well-studied problems such as SVP or SIS. The recent conclusion of the third round of NIST standardisation process testifies from this trust: among the four selected schemes for standardisation, three of them are based on lattices.

In comparison, code–based cryptography appears to lag behind from the point of view of security reductions, despite being very promising in terms of simplicity of the designs, short length of ciphertexts, efficiency of encryption and decryption; and even short key sizes, for instance with

[1] Laboratoire LIX, École Polytechnique, Institut Polytechnique de Paris, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau Cedex

[2] Inria

[3] CWI, Cryptology Group, Amsterdam, The Netherlands

E-mail addresses: maxime.bombar@cwi.nl, alain.couvreur@inria.fr, thomas.debris@inria.fr.

Date: October 27, 2023.

The authors would like to thank Damien Stehlé for his helpful insights on the OHCP technique for lattices, and Jean-Pierre Tillich for pointing us toward this framework. MB and TDA would also like to thank Eleni Pavlakis and Théo Gantzer for sharing their flat in San Francisco which permitted this work to be conducted in good conditions.

This work was funded by the French Agence Nationale de la Recherche through ANR JCJC COLA (ANR-21-CE39-0011), ANR BARRACUDA (ANR-21-CE39-0009-BARRACUDA) and *Plan France 2030* ANR-22-PETQ-0008.

BIKE [AAB+21a] and HQC [AAB+21b]. Indeed, in the last decades, a recurrent argument to claim the security of code–based cryptosystems was the NP–completeness of the so-called Decoding Problem [BMvT78]. This NP–completeness argument is only partially convincing since it is well–known that some NP–hard problems turn out to be easy for a large density of instances. In short, cryptographers are much more interested by problems which are hard *on average*, while NP–completeness only guarantees a worst-case hardness.

The Decoding Problem. A random $[n, k]-$code is the row–space of a uniformly random matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ (called a *generator matrix* of the code)[1]:

$$\mathcal{C} = \{\mathbf{mG} \mid \mathbf{m} \in \mathbb{F}_2^k\} \subset \mathbb{F}_2^n.$$

The (average-case) Decoding Problem can then be defined as follows:

Definition 1 ((Average-case) Search Decoding Problem)**.** *Given a random code $\mathcal{C}$, a vector $\mathbf{y} \in \mathbb{F}_2^n$ and a target distance $t \in \mathbb{N}$, the goal is to find a codeword (if exists) $\mathbf{c} \in \mathcal{C}$ and an error vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight $|\mathbf{e}| = t$ such that $\mathbf{y} = \mathbf{c} + \mathbf{e}$.*

Alternatively, this problem can be seen as solving a linear system with a non linear constraint given by the targetted Hamming weight. Indeed, a code $\mathcal{C}$ can also be defined by a *parity-check* matrix, that is to say a matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ such that

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_2^n \mid \mathbf{xH}^\top = \mathbf{0}\}.$$

The above decoding problem is then equivalent to finding a word $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight t such that $\mathbf{eH}^\top = \mathbf{s}$ for a given *syndrome* $\mathbf{s} \stackrel{\text{def}}{=} \mathbf{yH}^\top \in \mathbb{F}_2^{n-k}$.

Note that for solving the Decoding Problem, it is enough to recover the positions i such that $\mathbf{e}_i \neq 0$, *i.e.* the support of the error. This is even true for larger fields $\mathbb{F}_q$, at the cost of solving an additional linear system to recover the exact coefficients. In other words, decoding is equivalent to recovering the *support* of the error.

This computational problem has been studied for over sixty years [Pra62, Ste88, Dum91, BJMM12, MO15, BM17, CDMT22], and is widely considered to be hard to solve, even with the help of a putative quantum computer. Moreover, it benefits from a search-to-decision reduction, due to Fischer and Stern [FS96], which asserts the hardness of the following *decisional* version:

Definition 2 ((Average-case) Decision Decoding Problem)**.** *Given a random code $\mathcal{C}$ defined by a uniformly random generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$, a target distance $t \in \mathbb{N}$ and a vector $\mathbf{y} \in \mathbb{F}_2^n$, decide whether $\mathbf{y}$ is uniformly distributed over $\mathbb{F}_2^n$, or of the form $\mathbf{mG} + \mathbf{e}$ for some $\mathbf{m} \in \mathbb{F}_2^k$ and $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight $|\mathbf{e}| = t$.*

Such a reduction is very useful for cryptographic applications, since various cryptosystems, such as Alekhnovich cryptosystem [Ale03], rely on the hardness of the decisional version. When the length n of the code is *a priori* unbounded, this problem is also known as LPN (Learning Parity with Noise) in the literature.

If the hypothesis that the Decoding Problem is hard on average is widely accepted by the community, we lack theoretical results to corroborate that, since the literature on security reductions for codes remains very limited. The first worst-case to average-case reduction for the Decoding Problem is due to Brakerski, Lyubashevsky, Vaikuntanathan and Wichs in the recent breakthrough work [BLVW19] (and subsequently Yu and Zhang [YZ21]). This limited number of reductions is probably one of the reasons why NIST did not yet select any code–based submission. On the other hand, three of the four submissions selected to advance to the fourth round are based on binary error correcting codes. Moreover, with the recent attacks on SIDH ([CD23, MMP+23, Rob23]), NIST announced its will to standardise at least one code-based candidate, which increases the importance of theoretical studies of the underlying assumptions.

[1]Note that such a code has a dimension less than k when $\mathbf{G}$ has not full rank but this happens only with a negligible probability.

Structured variants. The plain Decoding Problem often leads to cryptosystems with large key sizes. In order to improve on that, it has been proposed to use codes with an additional structure, such as a large automorphism group $\mathrm{Aut}(\mathcal{C})$. For example, quasi-cyclic codes, introduced in cryptography by Gaborit in [Gab05], are very appealing since they offer a very good efficiency, while keeping the same security parameter as for truly random codes. Indeed, the best approach for solving the Decoding Problem of such structured codes remains the DOOM attack by Sendrier [Sen11], which only allows a $\sqrt{\sharp\mathrm{Aut}(\mathcal{C})}$ speed–up. Quasi–cyclic codes are in particular used in BIKE [AAB+22a] and HQC [AAB+22b] which are two of the three code–based proposals remaining in the fourth round of NIST competition. However, such structured codes are not restricted to encryption schemes. In particular, Bombar, Couteau, Couvreur and Ducros ([BCCD23]) have recently used the decision version of the Decoding Problem of random quasi-abelian codes, which generalise both random linear codes and quasi-cyclic codes, to build an efficient pseudorandom correlations generator for the OLE correlation (Oblivious Linear Evaluation) over any field $\mathbb{F}_q$ with $q > 2$. This allows to design the first efficient silent (*i.e.* which requires almost no communication in the preprocessing phase) N-party secure computation protocols for computing arbitrary arithmetic circuits over $\mathbb{F}_q$ for $q > 2$.

However, on the security reductions point of view, the situation is even worse than that of the plain Decoding Problem; there is even no complete search-to-decision reduction. The only known reduction for structured variants is the recent work of Bombar, Couvreur and Debris-Alazard [BCD22], via the introduction of the new problem called *Function Field Decoding Problem* (FFDP), which yield a search-to-decision reduction for some quasi-cyclic codes. This reduction has been extended in [BCCD23] to slightly more general quasi-abelian codes, but the question remains fully open for the codes and parameter sets used in NIST submissions BIKE and HQC.

From lattices to codes. Motivated by this state-of-affairs, a recent trend of research in code–based cryptography has been to take inspiration from the literature on Euclidean lattices to provide new reductions for codes: [DRT21] gives a quantum reduction from the Decoding Problem to the problem of finding a short codeword, in the way of [Reg05, SSTX09]; the reduction of [BCD22] is an average-case to average-case search-to-decision reduction for structured variants of the Decoding Problem, in the spirit of [LPR10], replacing number fields used in the lattice setting by function fields (somehow the analogue in positive characteristics). However, their reduction only works when the irreducible modulus splits completely in the underlying ring of integers, which is not the case with the parameter choice of BIKE and HQC. In the latter situation, the problem remains fully open. Our motivation with this work was to advance towards a general search-to-decision reduction. In this context, one may wonder if all the tools used to design reductions for lattices have been translated in the context of error correcting codes.

The answer to this question is *negative*. Indeed, in the breakthrough paper [PRS17], Peikert, Regev and Stephens-Davidowitz introduced a powerful tool for reductions called the OHCP framework (for Oracle with Hidden Center Problem). Until the aforementioned work, search-to-decision reductions for lattices had arithmetic and algebraic limitations in the choice of the modulus and the number field of the considered structured lattice problem: [LPR10] required the modulus to split completely, and the chosen number field to be Galois. The arithmetic hypothesis on the modulus was removed in [LS15] with the use of the modulus switching technique. The work of [PRS17] allows to completely get rid of such algebraic and arithmetic hypotheses, and Rosca, Stehlé and Wallet later used it in [RSW18] to design a complete search-to-decision reduction. This OHCP technique proved itself extremely useful as a black box inside the latest reductions in the context of structured lattice problems such as ring-LWE [PRS17, RSW18], polynomial-LWE [RSW18], module-LWE [BJRW20] or NTRU [PMS21]. On the other hand, even if this technique is considered to be very general, it has never been used outside of the lattice world.

Contributions. In this article, we revisit the OHCP framework from [PRS17] and adapt it to the coding theoretic setting (in Section 3). More precisely, we extract the very essence of this technique which appears to be the OCP technique (for Oracle Comparison Problem) ([PRS17, Definition 4.1]) and was overlooked before as a mere technical step. Building on top of OCP, we

show how given an algorithm solving the decisional Decoding Problem, it is possible to recover the support of the error, and hence to decode, solving the computational Decoding Problem. In other words, we show how to solve a problem which may be called OHSP for *Oracle with Hidden Support Problem*, and which is more suitable for code-based cryptography (see Figure 1)[2]

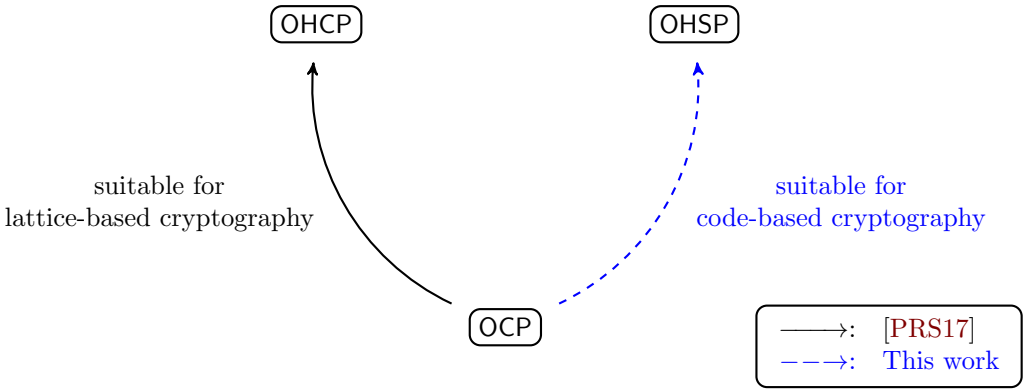

FIGURE 1. Relationships between OCP, OHCP and OHSP.

Combining this framework with a recent result of Debris-Alazard and Resch ([DR22]) on smoothing bounds for codes which applies to any *radial* smoothing distribution (in particular, it applies to the Bernoulli noise, which was not captured before), we derive a new reduction from the worst-case search Decoding Problem to the average-case decision Decoding Problem, in the spirit of what has been done in the lattice-based setting.

In Section 4, we discuss instantiations and parameters for which our reduction holds for relevant parameters. It turns out that with this completely different approach, we recover the same parameters and noise ratio than the worst-case to average-case search-to-search reduction of Brakerski *et al.* [BLVW19]. In particular, we reduce a *worst-case* search decoding problem whose hardness is superpolynomial, to an average-case decisional problem, to get the following (informal) theorem.

Theorem (Informal)**.** *Let $n, k, t \in \mathbb{N}$, $D < 1/2$ be such that*

$$\frac{k}{n} = \frac{1}{n^D} \quad \text{and} \quad \frac{t}{n} = \frac{\log_2(n)^2}{n^{1-D}}.$$

Suppose that there exists an algorithm which distinguishes with polynomial advantage between $(\mathbf{A}, \mathbf{sA}+\mathbf{t})$ and $(\mathbf{A}, \mathbf{y})$ where $\mathbf{A}$ is a random binary $k \times n$ matrix, $\mathbf{y}$ is a random binary vector, and $\mathbf{t}$ is a random binary vector of Hamming weight $\frac{n}{2}\left(1 - 1/n^{D(1+o(1))}\right)$.

Then there exists an algorithm which solves the worst-case decoding problem for input codes[3] of length n, dimension k and at decoding distance t.

Note that since the search-to-decision reduction of [FS96] is very tight, it can be composed with the reduction of [BLVW19] to yield a worst-to-average case search-to-decision reduction with the same parameters.

Finally, in Section 5, we discuss our attempt to give a reduction in the structured case, such as quasi-cyclic codes. In particular, we single out a difficulty that arises with codes but was not present in the lattice world, due to the choice of the error distribution. More precisely, in the lattice setting, the error distribution is taken through the Minkowski embedding, which transforms an actual product of polynomials (convolution) into a Shur product (coordinate-wise multiplication). The error then affects each component *independently*. In the Hamming world, this *independence* is not respected, and there seems to need a new idea to derive the reduction.

[2]Note that it is possible to give a formal definitions of all the problems we mention, but instead, we choose to put forth the intuition (as well as rigourous proofs on how to solve them), in order to avoid superfluous technicalities which would only obfuscate the speech.

[3]Input codes are supposed to be *balanced* as in the reduction of [BLVW19].

However, we believe that our OHSP technique can be seen as a first step towards more general reductions for structured codes, in the same manner that OHCP had a huge impact for reductions in lattice–based cryptography. We also believe that this paves the way for reductions for other metrics used in cryptography, such as the rank metric, for which no search-to-decision reduction is known.

The diagram in Figure 2 represents the relationships between problems in code–based cryptography. The black arrows represent previously known reductions.

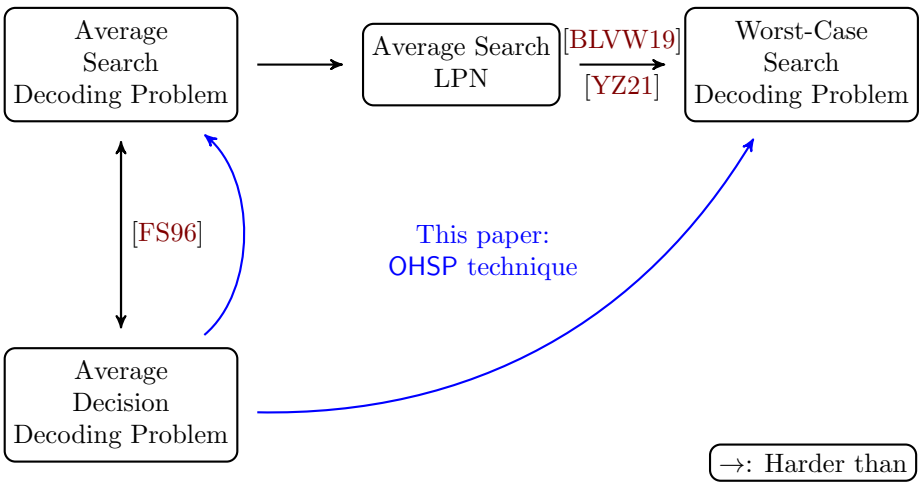

FIGURE 2. Known reductions for the decoding problem used in code-based cryptography.

Outline of the article. The present article is organized as follows: In Section 2 we recall the notations and some elementary notions. Then we start Section 3 by giving formally our search-to-decision reduction in Theorem 1. It is followed by a high-level description of how this theorem is obtained. In Subsections 3.1 and 3.2 we prove formally Theorem 1. In Section 4 we discuss instantiations of our search-to-decision reduction, first as an average-to-average reduction and ultimately as a worst-to-average reduction, in the context of the plain decoding problem. In section 5 we describe our failed attempt to apply our reduction template to quasi-cyclic codes.

2. Preliminaries

Notation. When a and b are two integers, $[\![a, b]\!]$ denotes the set of integers $\{a, a+1, \ldots, b\}$, and we denote by $\mathsf{poly}(n)$ any quantity which is an $O\left(n^{\alpha}\right)$ for some constant α. Vectors are in *row notation* and they will be written with bold letters, such as $\mathbf{e}$. Uppercase bold letters are used to denote matrices (such as $\mathbf{G}$). The canonical inner product $\sum_{i=1}^{n} x_i y_i$ between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. The *support* $\mathrm{Supp}(\mathbf{x})$ of $\mathbf{x}$ is the positions of its non-zero coordinates

$$\mathrm{Supp}(\mathbf{x}) \stackrel{\text{def}}{=} \{i \in [\![1, n]\!] \colon x_i \neq 0\}$$

and its Hamming weight $|\mathbf{x}|$ is the cardinality of its support

$$|\mathbf{x}| \stackrel{\text{def}}{=} \sharp \mathrm{Supp}(\mathbf{x}).$$

The sphere in $\mathbb{F}_2^n$ centered at $\mathbf{0}$ and of radius t (for the Hamming metric $|\cdot|$) will be denoted by $\mathcal{S}_t^n$ (or simply $\mathcal{S}_t$ when the ambient space is clear).

In this article, we wish to emphasize on which probability space the probabilities or the expectations are taken. We will denote by a subscript the random variable specifying the associated

probability space over which the probabilities or expectations are taken. For instance the probability $\mathbb{P}_X(E)$ of the event E is taken over the probability space Ω over which the random variable X is defined.

The *statistical distance* between two random variables X and Y taking their values in a same finite space $\mathcal{E}$ is defined as

$$\Delta(X,Y) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{a\in\mathcal{E}} \left|\mathbb{P}\left(X=a\right) - \mathbb{P}\left(Y=a\right)\right|. \tag{1}$$

The statistical distance between two random variables depends only on their distributions. Recall that for any event E, we have $|\mathbb{P}_X(E) - \mathbb{P}_Y(E)| \leqslant \Delta(X,Y)$. Therefore, computing probabilities over X or Y will differ by at most $\Delta(X,Y)$. The statistical distance enjoys many interesting properties. Among other things, it cannot increase by applying a function f,

$$\Delta(f(X), f(Y)) \leqslant \Delta(X,Y) \quad (\textit{data processing inequality}). \tag{2}$$

For the data processing inequality to hold, the function f may be randomized as soon as its internal randomness is independent from X and Y. In particular, it implies that the "success" probability of any algorithm $\mathcal{A}$ for inputs distributed according to X or Y, can only differ by at most $\Delta(X,Y)$. Furthermore, when $(X_1,\ldots,X_r)$ and $(Y_1,\ldots,Y_r)$ are two sequences of random variables such that the X_i's (respectively the Y_i's) are pairwise independent, then

$$\Delta((X_1,\ldots,X_r),(Y_1,\ldots,Y_r)) \leqslant \sum_{i=1}^{r} \Delta(X_i,Y_i). \tag{3}$$

In the sequel, we denote by $X \leftarrow \mathcal{D}$ when X is a random variable following distribution $\mathcal{D}$. In addition when $\mathcal{E}$ is a finite set, we allow ourselves to denote $X \leftarrow \mathcal{E}$ when X is uniformly distributed over $\mathcal{E}$. A Bernoulli random variable $X \leftarrow \mathrm{Ber}(\omega)$ *of parameter* $\omega \in \mathbb{R}_+$ is any binary random variable $X \in \mathbb{F}_2$ such that

$$\mathbb{P}\left(X=1\right) = \frac{1}{2}\left(1-2^{-\omega}\right).$$

Remark 1. *This notation may seem surprising. It is however more comfortable to use in our setting. The rationale behind this choice is that in our reduction we strongly need to "focus" in the neighbourhood of* $1/2$. *This notation has also the following advantage: a simple calculation shows that given two independent random variables* $X \leftarrow \mathrm{Ber}(\omega_1)$ *and* $Y \leftarrow \mathrm{Ber}(\omega_2)$, *then* $X+Y \leftarrow \mathrm{Ber}(\omega_1+\omega_2)$. *This will be comfortable in the sequel.*

Finally, $X \leftarrow \mathrm{Ber}(\omega)^{\otimes N}$ means that $X \stackrel{\text{def}}{=} (X_1,\ldots,X_N)$ where the X_i's are independent and identically distributed Bernoulli random variables of parameter ω.

3. Search-to-decision reduction in the Oracle Comparison Problem (OCP) framework

Let us assume that we have a probabilistic algorithm $\mathcal{A}$ running in time T that can distinguish noisy codewords at some Hamming distance and uniform random vectors over the ambient space. Its inputs are $\mathbf{A} \in \mathbb{F}_2^{k\times N}$ and $\mathbf{y} \in \mathbb{F}_2^N$. The aim of $\mathcal{A}$ is to output "1" if and only if $\mathbf{y} = \mathbf{sA} + \mathbf{e}$ for some $\mathbf{s} \in \mathbb{F}_2^k$ and the bits of $\mathbf{e}$ are independent and identically distributed Bernoulli random variables of parameter ω. Namely, $\mathbf{y}$ is a noisy codeword $\mathbf{c}+\mathbf{e}$ where $\mathbf{c} \in \mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{mA} \ : \ \mathbf{m} \in \mathbb{F}_2^k\right\}$ and $|\mathbf{e}| \approx \frac{N}{2}\left(1-2^{-\omega}\right)$. Otherwise, $\mathcal{A}$ has to output "0". In addition, to be relevant in a cryptographic context, we suppose that $\mathcal{A}$ may give false answers. In that case we are interested in its *advantage* $\varepsilon(k,N,\omega)$ which is defined as follows

$$\varepsilon(k,N,\omega) \stackrel{\text{def}}{=} \frac{1}{2}\left(\mathbb{P}_{\mathbf{A},\mathbf{s},\mathbf{e}}\left(\mathcal{A}\left(\mathbf{A},\mathbf{sA}+\mathbf{e}\right)=1\right) - \mathbb{P}_{\mathbf{A},\mathbf{y}}\left(\mathcal{A}\left(\mathbf{A},\mathbf{y}\right)=1\right)\right), \tag{4}$$

where the random variables satisfy

$$(i)\ \mathbf{A} \leftarrow \mathbb{F}_2^{k\times N}, \quad (ii)\ \mathbf{s} \leftarrow \mathbb{F}_2^k, \quad (iii)\ \mathbf{y} \leftarrow \mathbb{F}_2^N \quad \text{and} \quad (iv)\ \mathbf{e} \leftarrow \mathrm{Ber}(\omega)^{\otimes N}. \tag{5}$$

We say that $\mathcal{A}$ distinguishes between distributions $(\mathbf{A}, \mathbf{sA}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with advantage $\varepsilon(k, N, \omega)$. It may happen that we omit the dependence in (k, N, w) and simply write ε (that will be clear from the context). The following general theorem shows that from any such putative "distinguishing" algorithm $\mathcal{A}$, we can build an algorithm solving a fixed decoding problem, namely recovering $\mathbf{t}$ from $(\mathbf{G}, \mathbf{mG} + \mathbf{t})$.

Theorem 1. *Let $N, n \in \mathbb{N}$ and $k \in [\![0, \min(N, n)]\!]$. Let $(\mathbf{G}, \mathbf{mG} + \mathbf{t})$ with $\mathbf{G} \in \mathbb{F}_2^{k \times n}$, $\mathbf{m} \in \mathbb{F}_2^k$, and $|\mathbf{t}| = t \in [\![0, n]\!]$. Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes in time T distributions $(\mathbf{A}, \mathbf{sA} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with advantage $\varepsilon(k, N, \omega)$ where $\mathbf{A}, \mathbf{s}, \mathbf{e}$, $\mathbf{y}$ satisfy (5) and $\omega \in \mathbb{R}_+$ verifying*

$$\omega = \Omega(1) \quad \text{and} \quad \omega = O(n). \tag{6}$$

Let $\omega_0, \alpha \in \mathbb{R}_+$ be such that

$$t\, \omega_0 = \omega \quad \text{and} \quad \alpha \stackrel{def}{=} \max\left(\frac{1}{\varepsilon(k, N, \omega)}, N, n\right). \tag{7}$$

Then, there exists an algorithm which takes as input $(\mathbf{G}, \mathbf{mG} + \mathbf{t})$ and which outputs $\mathbf{t}$ in time $T\ \mathsf{poly}(\alpha)$ with probability (over its internal randomness and **not** *the choice of $\mathbf{G}$, $\mathbf{m}$ and $\mathbf{t}$ which are fixed) bigger than*

$$1 - 2^{-\Omega(n)} - N\ \mathsf{poly}(\alpha) \max_{x \geqslant 0}\ \Delta\Big(\big(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\big), (\mathbf{a}, e(x))\Big), \tag{8}$$

where $\mathbf{a} \leftarrow \mathbb{F}_2^k$, $\mathbf{r}(x) \leftarrow \mathrm{Ber}(2^x\omega_0)^{\otimes n}$ and $e(x) \leftarrow \mathrm{Ber}(2^x\omega_0 t)$ with $x \geqslant 0$.

This theorem will follow from a sequence of lemmas. Before providing a rigorous demonstration, let us give an informal sketch of the proof.

Remark 2. *Similarly to* [PRS17]*, our algorithm rests on a distinguishing process between two distinct* oracles*. Informally, by* oracle *we mean a black box that we can query arbitrarily many times and whose outputs are independent random elements following a given distribution. Formally, an oracle $\mathcal{O}(x)$ can be modelised by a sequence $(X_i)_{i \in \mathbb{N}}$ of independent identically distributed random variables whose distribution may depend from some parameter x.*

Step 1. (*From distinguishing* LPN *samples to distinguishing noisy codewords*). We start from an algorithm $\mathcal{A}$ that distinguishes, with advantage ε, between a noisy codeword $\mathbf{c} + \mathbf{e}$ (by outputting 1) and a uniform $\mathbf{y} \in \mathbb{F}_2^N$ (by outputting 0) with $\mathbf{c}$ drawn uniformly at random from some random binary $[N, k]$-code $\mathcal{C}$, and $\mathbf{e} \leftarrow \mathrm{Ber}(\omega)^{\otimes N}$. This algorithm can easily be turned into an algorithm $\mathcal{A}'$ distinguishing (with the same advantage ε) oracles

$$\mathcal{O}(\omega) : (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e) \quad \text{and} \quad \mathcal{O}(\infty) : (\mathbf{a}, u) \tag{9}$$

where $\mathbf{s} \in \mathbb{F}_2^k$, $e \leftarrow \mathrm{Ber}(\omega)$, $\mathbf{a} \leftarrow \mathbb{F}_2^k$ and $u \leftarrow \mathbb{F}_2$[4].Indeed, given one of the above oracles $\mathcal{O}$, in order to design $\mathcal{A}'$, it is enough to perform N queries $(\mathbf{a}_i, b_i)$ to $\mathcal{O}$ and gather them to generate the pair $(\mathbf{A}, \mathbf{b})$ where the columns of $\mathbf{A}$ are the $\mathbf{a}_i^\top$'s and $\mathbf{b} = (b_1, \ldots, b_N)$. Then, we feed $\mathcal{A}$ with the generated pair $(\mathbf{A}, \mathbf{b})$ to make our decision. Defining such an algorithm $\mathcal{A}'$ solving the above LPN-decisional problem with at most N queries may seem at first sight tautological, but for our reduction it is more convenient to emphasize this point.

This is why, for proving Theorem 1, we will suppose that we directly have an algorithm $\mathcal{A}'$ distinguishing LPN-oracles $\mathcal{O}(\omega)$ and $\mathcal{O}(\infty)$ with some advantage ε *and* querying at most N times the input oracle.

Step 2. (*From a noisy codeword to* LPN*-samples*). The starting point of the reduction consists in noticing that, from any input of a decoding problem, we can build some LPN-oracle. Given

$$\Big(\mathbf{G}, \mathbf{y} \stackrel{\text{def}}{=} \mathbf{mG} + \mathbf{t}\Big) \in \mathbb{F}_2^{k \times n} \times \mathbb{F}_2^n,$$

[4]A sample from $\mathcal{O}(\cdot)$ is called an LPN sample.

we can design the following oracle $\mathcal{O}_0$. Sample $\mathbf{r}$ according to $\mathrm{Ber}(\omega_0)^{\otimes n}$, then compute $\mathbf{r}\mathbf{G}^\top$ and

$$\langle \mathbf{y}, \mathbf{r} \rangle = \langle \mathbf{m}\mathbf{G} + \mathbf{t}, \mathbf{r} \rangle = \langle \mathbf{m}, \mathbf{r}\mathbf{G}^\top \rangle + \langle \mathbf{t}, \mathbf{r} \rangle. \tag{10}$$

The oracle $\mathcal{O}_0$ outputs LPN–like samples of the form:

$$\mathcal{O}_0 : \ (\mathbf{a}', \langle \mathbf{s}, \mathbf{a}' \rangle + e') \quad \text{where} \quad \begin{cases} \mathbf{s} \stackrel{\text{def}}{=} \mathbf{m} \\ \mathbf{a}' \stackrel{\text{def}}{=} \mathbf{r}\mathbf{G}^\top \\ e' \stackrel{\text{def}}{=} \langle \mathbf{t}, \mathbf{r} \rangle. \end{cases} \tag{11}$$

The random variable e' follows a Bernoulli distribution of parameter $\omega_0 |\mathbf{t}| = \omega_0 t$ (see Lemma 1 further) which equals ω (under the notation of Theorem 1, Equation (7)). However, one can notice that our above sample is not a valid LPN instance since $\mathbf{a}' = \mathbf{r}\mathbf{G}^\top$ is *a priori* not uniformly distributed and is correlated to e'. Nonetheless, thanks to the data processing inequality (see Equation (2)), replacing the sample $(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{m}, \mathbf{r}\mathbf{G}^\top \rangle + \langle \mathbf{t}, \mathbf{r} \rangle)$ by a genuine LPN sample $(\mathbf{a}, \langle \mathbf{a}, \mathbf{m} \rangle + e)$ changes the probabilities by at most the additive term

$$\Delta\left(\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{r}, \mathbf{t} \rangle\right), (\mathbf{a}, e)\right), \quad \text{where} \quad e \leftarrow \mathrm{Ber}(\omega) \ \text{ and is independent from } \mathbf{a}.$$

Further, in § 4, when we instantiate Theorem 1, parameters are chosen so that this statistical distance is negligible. This is obtained by carefully choosing ω_0. In particular, we use smoothing bounds as given in [BLVW19, DST19, DDRT22, DR22].

Now, one may wonder how we can use $\mathcal{O}_0$ with our algorithm $\mathcal{A}'$ distinguishing between LPN-distributions to solve our underlying decoding problem. It is the aim of the next step.

Step 3. (*Applying the Oracle Comparison Problem* OCP *framework*). For a formal definition of OCP, the interested reader can refer to [PRS17, Definition 4.1]. Intuitively, given access to two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ whose acceptance probability are just a "shift" of one another, the goal of OCP is to tell which one is in advance, and which one lags behind (see Figure 3).

The first core idea of the reduction is to notice that in order to build the oracle $\mathcal{O}_0$ of (11), we have computed $\langle \mathbf{y}, \mathbf{r} \rangle$ (see (10)), leading to an LPN sample with parameter $\omega_0 |\mathbf{t}| = \omega$ (see Lemma 1 further). One could have done the same thing but this time by computing $\langle \mathbf{y}+\mathbf{z}, \mathbf{r} \rangle$ for some fixed $\mathbf{z} \in \mathbb{F}_2^n$ instead. This has the following consequence: our new oracle provides LPN-samples with Bernoulli noise of parameter $\omega_0 |\mathbf{t} + \mathbf{z}|$. Arguably this innocent looking fact is the key of our reduction, which follows the approach of [PRS17, RSW18]. Let us define the oracle $\mathcal{O}^{\mathbf{v}_i}$ as $\mathcal{O}_0$, but instead of outputting $\langle \mathbf{y}, \mathbf{r} \rangle$ it outputs $\langle \mathbf{y}+\mathbf{v}_i, \mathbf{r} \rangle$ where $(\mathbf{v}_j)_{1 \leqslant j \leqslant n}$ is the canonical basis of $\mathbb{F}_2^n$. Then we feed $\mathcal{A}'$ with $\mathcal{O}^{\mathbf{v}_i}$. By assumption, $\mathcal{A}'$ distinguishes between an LPN oracle with noise $\mathrm{Ber}(\omega)$ and a uniform noise $\mathrm{Ber}(\infty)$ with advantage ε. Therefore, the probability that $\mathcal{A}'$ outputs 1 when fed with $\mathcal{O}_0$ is roughly $1/2 + \varepsilon$. On the other hand, $\mathcal{O}^{\mathbf{v}_i}$ defines an LPN oracle with Bernoulli parameter $\omega_0 |\mathbf{t} + \mathbf{v}_i|$, where $|\mathbf{t}| = t$ and $|\mathbf{v}_i| = 1$. Therefore, the noise is distributed either as $\mathrm{Ber}(\omega_0(t-1))$ or $\mathrm{Ber}(\omega_0(t+1))$ depending on whether $t_i = 1$ *or* not, that is to say on whether i belongs to the support of $\mathbf{t}$ *or* not. In other words, the behaviour of $\mathcal{O}^{\mathbf{v}_i}$ depends on the *hidden support* of $\mathbf{t}$. From then on, one may prematurely conclude that the acceptance probability of $\mathcal{A}'$ when fed with $\mathcal{O}^{\mathbf{v}_i}$ slightly differs from the one when fed with $\mathcal{O}_0$; a behaviour that could be detected. Unfortunately the success probability, $1/2 + \varepsilon$, may be the same in all these cases. This brings us to the second core idea of the reduction. Instead of defining $\mathcal{O}_0$ and $\mathcal{O}^{\mathbf{v}_i}$ by sampling $\mathbf{r}$ according to $\mathrm{Ber}(\omega_0)$, we choose $\mathbf{r} \leftarrow \mathrm{Ber}(2^x \omega_0)$ for $x \in \mathbb{R}_+$. The LPN-noise now follows the following distributions

$$\mathrm{Ber}(2^x \omega_0 t) \text{ in } \mathcal{O}_0 \quad \text{and} \quad \begin{cases} \mathrm{Ber}\left(2^x \omega_0 (t-1)\right) & \text{if } t_i = 1 \\ \mathrm{Ber}\left(2^x \omega_0 (t+1)\right) & \text{if } t_i = 0 \end{cases} \text{ in } \mathcal{O}^{\mathbf{v}_i}. \tag{12}$$

We can notice that by letting $x \to \infty$, above distributions go to $\mathrm{Ber}(\infty)$. However, the fundamental remark is not here. By definition, our distinguishing algorithm $\mathcal{A}'$ does not

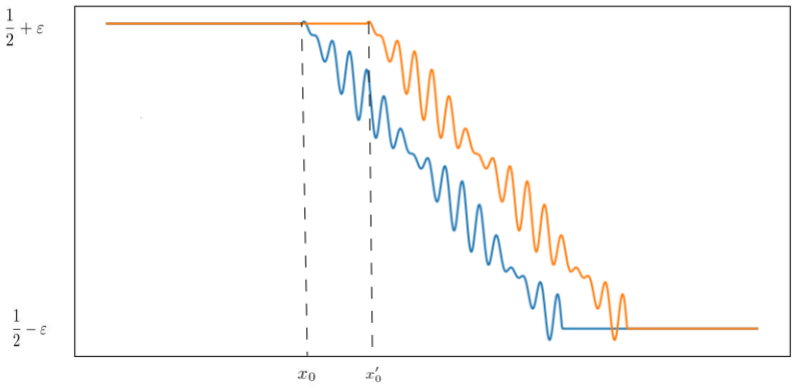

FIGURE 3. Illustration of Step 3 (in the case $t_i = 1$).

behave "in the same way" when is given as input $\mathcal{O}(\omega_0 t)$ or $\mathcal{O}(\infty)$; fact which is quantified by its advantage ε. Therefore, if one feeds $\mathcal{A}'$ with the oracle $\mathcal{O}_0$ (which outputs LPN samples with Bernoulli noise of parameter $\omega_0 t$ when $x = 0$), then playing on $x \geqslant 0$ one can detect a difference in its probability to output 1. Let us say that the change of behaviour happens at some x_0, namely for a noise $\mathrm{Ber}(2^{x_0}\omega_0 t)$. Let us suppose that now we feed $\mathcal{O}^{\mathbf{v}_i}$ to $\mathcal{A}'$. One can also choose different values x and look at the probability that $\mathcal{A}'$ outputs 1. But, we know that this change of behaviour will happen when the noise follows some Bernoulli distribution of parameter $2^{x_0}\omega_0 t$. Therefore, in that case, we will observe a difference at some $x_0' \geqslant 0$ when (according to Equation (12); see also Figure 3)

$$\begin{cases} 2^{x_0'}\omega_0(t-1) = 2^{x_0}\omega_0 t \iff x_0' = x_0 + \log\left(\frac{t}{t-1}\right) > x_0 & \text{if } t_i = 1, \\ \\ 2^{x_0'}\omega_0(t+1) = 2^{x_0}\omega_0 t \iff x_0' = x_0 + \log\left(\frac{t}{t+1}\right) < x_0 & \text{if } t_i = 0. \end{cases}$$

It turns out that with classical statistical methods, we can now detect this difference in the acceptance probability of $\mathcal{A}'$. The idea is just to estimate when $\mathcal{A}'$ changes its behaviour given as input $\mathcal{O}_0$ and $\mathcal{O}^{\mathbf{v}_i}$. Depending whether $t_i = 1$ or not, this change of behaviour will happen for a smaller x with input $\mathcal{O}_0$, or a bigger x. This yields the claimed reduction: we are able to decide whether $t_i = 1$ or 0 for any $i \in [\![1, n]\!]$, *i.e.* we are able to recover the *hidden support* of the error, and hence to solve the decoding problem. In other words, we turned a "distinguishing decoding" algorithm into a "search decoding" algorithm.

From now on, $\mathcal{A}$ denotes an algorithm running in time T and taking as input an oracle $\mathcal{O}$ which can be queried at most N times and outputting vectors in $\mathbb{F}_2^k \times \mathbb{F}_2$. Furthermore, its advantage to distinguish between $\mathcal{O}(\omega)$ and $\mathcal{O}(\infty)$ (defined in (9)) is given by

$$\varepsilon = \frac{1}{2}\Big(\mathbb{P}\left(\mathcal{A}\left(\mathcal{O}(\omega)\right) = 1\right) - \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}(\infty)\right) = 1\right)\Big) > 0. \tag{13}$$

Remark 3. *After possibly replacing $\mathcal{A}(\mathcal{O})$ by $1 - \mathcal{A}(\mathcal{O})$, one can always suppose the advantage to be positive.*

3.1. **Building LPN-oracles from a decoding instance: Step 2.** Our aim in this step is to study oracles $\mathcal{O}^{\mathbf{z}}(x)$ and $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$ which are given in Figure 4, where $\mathbf{z} \in \mathbb{F}_2^n$ is a parameter, $x \in \mathbb{R}$ an input and $\mathbf{y} = \mathbf{mG} + \mathbf{t}$. Notice that $(\mathbf{G}, \mathbf{y})$ is known while $(\mathbf{m}, \mathbf{t})$ are unknown; preventing us from being able to run $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$. However, as we will explain below, $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$ is an ideal version of $\mathcal{O}^{\mathbf{z}}(x)$ that we "only" use to analyse the success probability of the reduction.

Oracle $\mathcal{O}^{\mathbf{z}}(x)$:
  Input: $x \in \mathbb{R}$
    Sample: $\mathbf{r} \leftarrow \mathrm{Ber}(2^x\omega_0)^{\otimes n}$
  Return: $\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{y}+\mathbf{z}, \mathbf{r}\rangle\right)$

Oracle $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$:
  Input: $x \in \mathbb{R}$
    Sample $\mathbf{r} \leftarrow \mathrm{Ber}(2^x\omega_0)^{\otimes n}$
      and $\mathbf{a} \leftarrow \mathbb{F}_2^k$
  Return: $(\mathbf{a}, \langle \mathbf{a}, \mathbf{m}\rangle + \langle \mathbf{z}+\mathbf{t}, \mathbf{r}\rangle)$

FIGURE 4. Oracles $\mathcal{O}^{\mathbf{z}}(x)$ and $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$

Oracle $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$ is an ideal version of $\mathcal{O}^{\mathbf{z}}(x)$. It follows from the fact that $\mathcal{O}^{\mathbf{z}}$ outputs LPN-like samples

$$\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{m}, \mathbf{r}\mathbf{G}^\top\rangle + \langle \mathbf{z}+\mathbf{t}, \mathbf{r}\rangle\right).$$

Notice that, contrary to $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}$, oracle $\mathcal{O}^{\mathbf{z}}$ does not provide genuine LPN samples (that is the reason why we said LPN *like*) since $\mathbf{r}\mathbf{G}^\top$ is not uniformly distributed and is correlated to $\langle \mathbf{z}+\mathbf{t}, \mathbf{r}\rangle$. However, in both oracles the noise term is the same. In the following lemma (often called the piling-up lemma) we show how it behaves.

Lemma 1. *Let* $\mathbf{r} \leftarrow \mathrm{Ber}(\alpha)^{\otimes n}$*, then for any* $\mathbf{z} \in \mathbb{F}_2^n$ *we have*

$$\langle \mathbf{z}, \mathbf{r}\rangle \leftarrow \mathrm{Ber}(|\mathbf{z}|\alpha).$$

Proof. Let $z \stackrel{\text{def}}{=} |\mathbf{z}|$ and $p \stackrel{\text{def}}{=} \frac{1}{2}\left(1-2^{-\alpha}\right)$. By definition of $\mathbf{r} \leftarrow \mathrm{Ber}(\alpha)^{\otimes n}$ we have the following computation

$$\begin{aligned}
\mathbb{P}_{\mathbf{r}}\left(\langle \mathbf{z}, \mathbf{r}\rangle = 1\right) &= \sum_{j \text{ odd}} \binom{z}{j} p^j (1-p)^{z-j} \\
&= \frac{1}{2}\left(\sum_j \binom{z}{j} p^j (1-p)^{z-j} - \sum_j (-1)^j \binom{z}{j} p^j (1-p)^{z-j}\right) \\
&= \frac{1}{2}\left(1-(1-2p)^z\right) \\
&= \frac{1}{2}\left(1-2^{-z\alpha}\right),
\end{aligned}$$

which concludes the proof. □

3.2. **Oracle Comparison Problem technique: Step 3.** Let us introduce the following function

$$p : x \in \mathbb{R} \longmapsto \mathbb{P}\left(\mathcal{A}(\mathcal{O}^{\mathbf{0}}_{\mathrm{ideal}}(x)) = 1\right) \tag{14}$$

where we feed to $\mathcal{A}$ the ideal oracle. Recall that $(\mathbf{v}_i)_{1\leqslant i\leqslant n}$ denotes the canonical basis of $\mathbb{F}_2^n$; from Lemma 1, we notice that

$$\begin{aligned}
p\left(x+\log\frac{|\mathbf{t}+\mathbf{v}_i|}{|\mathbf{t}|}\right) &= \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}^{\mathbf{0}}_{\mathrm{ideal}}\left(x+\log\frac{|\mathbf{t}+\mathbf{v}_i|}{|\mathbf{t}|}\right)\right)=1\right) \\
&= \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}_{\mathrm{ideal}}(x)\right)=1\right)
\end{aligned} \tag{15}$$

where the last equality follows from the fact that $\mathcal{O}^{\mathbf{0}}_{\mathrm{ideal}}\left(x+\log\frac{|\mathbf{t}+\mathbf{v}_i|}{|\mathbf{t}|})\right)$ outputs proper LPN samples with Bernoulli noise of parameter

$$2^{x+\log\frac{|\mathbf{t}+\mathbf{v}_i|}{|\mathbf{t}|}}\omega_0|\mathbf{t}| = 2^x\omega_0\left|\mathbf{t}+\mathbf{v}_i\right|.$$

In other words, the probability that $\mathcal{A}$ outputs 1 when fed with $\mathcal{O}^{\mathbf{v}_i}_{\mathrm{ideal}}(x)$ is the probability that $\mathcal{A}$ outputs 1 when fed with $\mathcal{O}^{\mathbf{0}}_{\mathrm{ideal}}$ on x shifted by

$$\log\left(\frac{|\mathbf{t}+\mathbf{v}_i|}{|\mathbf{t}|}\right) = \begin{cases} \log(1-1/t) & \text{if } t_i = 1 \\ \log(1+1/t) & \text{otherwise.} \end{cases}$$

Let us stress that (15) would not hold if one had defined p in (14) by feeding $\mathcal{A}$ with $\mathcal{O}^{\mathbf{0}}$ instead of $\mathcal{O}^{\mathbf{0}}_{\mathrm{ideal}}$. Indeed, notice that outputs $(\mathbf{a}, b)$ of $\mathcal{O}^{\mathbf{0}}(x)$ are such that *both* the distributions of b

and $\mathbf{a}$ are functions of x. Hence, changing x in the non–ideal oracle $\mathcal{O}^{\mathbf{0}}(x)$ might change the distribution of the first component of the output and (15) would no longer hold. We crucially used that changing x in $\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)$ only modifies the noise term.

As roughly described below Theorem 1, the core idea of the reduction is to feed to $\mathcal{A}$ oracles $\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)$ and $\mathcal{O}^{\mathbf{v}_i}_{\text{ideal}}(x)$ and then to draw the probability to output 1 when x ranges over $[0, +\infty[$. Practically, we compute statistical estimates of this probability when x ranges over a discretisation of $[0, X_{\max}]$, for some $X_{\max}$. In the choice of $X_{\max}$ and the discretisation step, a trade-off should be made. On the one hand, for the empirical estimates to be close enough to the actual probability function p of (14), the upper bound $X_{\max}$ should be large enough and the discretisation step should be small enough. On the other hand for the statistical estimator to run in polynomial time, $X_{\max}$ should not be too large and the discretisation step should not be too small.

Then, using that $\mathcal{A}$ discriminates oracles $\mathcal{O}(\omega_0 t) = \mathcal{O}^{\mathbf{0}}_{\text{ideal}}(0)$ and $\mathcal{O}(\infty) = \mathcal{O}^{\mathbf{0}}_{\text{ideal}}(\infty)$, we will be able for both oracles to determine the first input x that induces a change in the behaviour of $\mathcal{A}$. We will compare both values and depending on which one is the biggest, we will decide if $t_i = 1$, or not. The correction of this procedure relies on (15) showing that one distribution is the shift of the other one. However, one may note that we cannot run $\mathcal{A}\left(\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)\right)$ and $\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}_{\text{ideal}}(x)\right)$ as $\mathbf{m}, \mathbf{t}$ are unknown. We have instead access to $\mathcal{A}\left(\mathcal{O}^{\mathbf{0}}(x)\right)$ and $\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}(x)\right)$ for which we do not know if their probabilities to output 1 are a shift of the other one. In order to be able to analyse our procedure, we will use the following remark: the probability of success when given the real oracles only differs by at most an additive term $\Delta\left(\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{r}, \mathbf{t}\rangle\right), (\mathbf{a}, e)\right)$ (multiplied by the number of queries to the oracles) to the case where it is given the ideal oracles. Therefore, as soon as we can estimate the aforementioned statistical distance, it is enough to perform the analysis when given ideal oracles.

The following technical lemma (whose proof is essentially the same as the one of [PRS17, Lemma 4.2]) shows how two oracles depending on a parameter x can be distinguished if the distribution of one is the shift of the other one. This statement was initially used to solve the Oracle Comparison Problem (OCP) problem introduced in [PRS17, §4]. Think that we will instantiate this lemma using oracles $\mathcal{O}_{s_1}(x) = \mathcal{A}\left(\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)\right)$ with $s_1 = 0$ and $\mathcal{O}_{s_2}(x) = \mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}_{\text{ideal}}(x)\right)$ with $s_2 = \log\left(1 \pm \frac{1}{t}\right)$.

Lemma 2. *Let $s_1, s_2 \in \mathbb{R}$ and $p : \mathbb{R} \to [0,1]$. We suppose that there exists $\alpha > 0$ and $p_\infty \in [0,1]$ such that p verifies the following assumptions*

(i) $p(s_1) - p_\infty \geqslant \frac{1}{\alpha}$;
(ii) $\forall x \in \mathbb{R}_+$, $|p(x) - p_\infty| \leqslant \alpha 2^{-\frac{x}{\alpha}}$;
(iii) p *is α-lipschitz.*

Let $\mathcal{O}_{s_1}$ and $\mathcal{O}_{s_2}$ be two oracles that output 0 or 1 and such that

$$\forall x \in \mathbb{R}, \quad \mathbb{P}\left(\mathcal{O}_{s_1}(x) = 1\right) = p(s_1 + x) \quad \text{and} \quad \mathbb{P}\left(\mathcal{O}_{s_2}(x) = 1\right) = p(s_2 + x).$$

We suppose that a call to one of the above oracle costs a time T. Furthermore, s_1 and s_2 are such that

$$\text{either} \quad \text{(I)} \quad s_1 \leqslant s_2 \quad \text{or} \quad \text{(II)} \quad s_1 \geqslant s_2 + \frac{1}{\alpha}.$$

Then, there exists an algorithm, running in time $T\,\mathsf{poly}(\alpha)$, taking as inputs $(\mathcal{O}_{s_1}, \mathcal{O}_{s_2})$, querying them $\mathsf{poly}(\alpha)$ times and which can decide whether (I) *or* (II) *holds, with a success probability $\geqslant 1 - e^{-\alpha}$ (over the outputs of the oracles $\mathcal{O}_{s_i}$'s).*

Proof. The fundamental idea of the proof is to introduce the following function

$$h(s) \stackrel{\text{def}}{=} \max_{x \geqslant 0} (1 + x)\,|p(s + x) - p_\infty|\,.$$

Estimating this function thanks to the oracles $\mathcal{O}_{s_1}$ and $\mathcal{O}_{s_2}$ (by using classical statistical methods) will discriminate both considered cases, namely if $s_1 \leqslant s_2$ or $s_1 \geqslant s_2 + \frac{1}{\alpha}$. We will show in the second part of the proof how to estimate h evaluated at s_1 and s_2. Let us first show that h discriminates cases $s_1 \leqslant s_2$ or $s_1 \geqslant s_2 + \frac{1}{\alpha}$.

- Case $s_1 \leqslant s_2$: we have
$$h(s_2) \leqslant h(s_1). \tag{16}$$
By definition,
$$\begin{aligned} h(s_2) &= \max_{x \geqslant 0} (1+x)\,|p(s_2+x) - p_\infty| \\ &= \max_{x \geqslant 0} (1+x)\,|p\left(s_1 + (x + s_2 - s_1)\right) - p_\infty| \\ &\leqslant \max_{x \geqslant 0} \left(1 + (x + s_2 - s_1)\right)|p\left(s_1 + (x + s_2 - s_1)\right) - p_\infty| \\ &= \max_{y \geqslant s_2 - s_1 \geqslant 0} (1+y)\,|p\left(s_1 + y\right) - p_\infty|\,, \end{aligned}$$
which shows Equation (16).

- Case $s_1 \geqslant s_2 + \frac{1}{\alpha}$: we have
$$h(s_1) < h(s_2) - P\left(\frac{1}{\alpha}\right). \tag{17}$$
for some polynomial P. For this case let us define
$$\widehat{x}(s) \stackrel{\text{def}}{=} \min \operatorname*{argmax}_{x \geqslant 0} (1+x)\,|p(s+x) - p_\infty|\,.$$
It is the smallest value $x \in [0, +\infty)$ at which h reaches its maximum. Notice that
$$\begin{aligned} h(s_1) &= (1 + \widehat{x}(s_1))\,|p(s_1 + \widehat{x}(s_1)) - p_\infty| \\ &\leqslant (1 + \widehat{x}(s_1) + s_1 - s_2)\,|p(s_1 + \widehat{x}(s_1)) - p_\infty| \\ &= (1 + (\widehat{x}(s_1) + s_1 - s_2))\,|p\left(s_2 + (\widehat{x}(s_1) + s_1 - s_2)\right) - p_\infty| \\ &\leqslant h(s_2), \end{aligned}$$
where in the last line we used that $\widehat{x}(s_1) + s_1 - s_2 \geqslant 0$ as $\widehat{x}(s_1) \geqslant 0$ and $s_1 - s_2 \geqslant 0$ by assumption. Therefore, from the above last inequality,
$$\begin{aligned} h(s_2) &\geqslant (1 + (\widehat{x}(s_1) + s_1 - s_2))\,|p\left(s_2 + (\widehat{x}(s_1) + s_1 - s_2)\right) - p_\infty| \\ &= \left(1 + \frac{s_1 - s_2}{1 + \widehat{x}(s_1)}\right) h(s_1), \end{aligned}$$
which shows that
$$h(s_2) - h(s_1) \geqslant \frac{s_1 - s_2}{1 + \widehat{x}(s_1)} h(s_1) \geqslant \frac{1}{\alpha}\,\frac{h(s_1)}{1 + \widehat{x}(s_1)}. \tag{18}$$
But,
$$h(s_1) = \max_{x \geqslant 0} (1+x)\,|p(s_1+x) - p_\infty| \geqslant |p(s_1) - p_\infty| \geqslant \frac{1}{\alpha}, \tag{19}$$
where in the last inequality we used assumption *(i)* on p. Plugging this in (18) leads to
$$h(s_2) - h(s_1) \geqslant \frac{1}{\alpha^2}\,\frac{1}{1 + \widehat{x}(s_1)}. \tag{20}$$
Let us now bound $\widehat{x}(s_1)$ from above. We have by assumption *(ii)* about p,
$$\begin{aligned} h(s_1) &= (1 + \widehat{x}(s_1))\,|p(s_1 + \widehat{x}(s_1)) - p_\infty| \\ &\leqslant \alpha(1 + \widehat{x}(s_1)) 2^{-\frac{s_1 + \widehat{x}(s_1)}{\alpha}} \\ &\leqslant \alpha(1 + \widehat{x}(s_1)) 2^{-\frac{\widehat{x}(s_1)}{\alpha}} \end{aligned}$$
but $h(s_1) \geqslant \frac{1}{\alpha}$ according to (19). Therefore,
$$-\frac{\widehat{x}(s_1)}{\alpha} + \log\left(1 + \widehat{x}(s_1)\right) \geqslant \log\left(\frac{1}{\alpha^2}\right) \Longrightarrow \widehat{x}(s_1) \leqslant 2\alpha \log(\alpha) + \alpha \log\left(1 + \widehat{x}(s_1)\right).$$

Consequently,

$$\widehat{x}(s_1) \leqslant C\alpha \log(\alpha),$$

for some constant C. Plugging this in (20) shows (17).

To summarize, considering cases $s_1 \leqslant s_2$ or $s_1 \geqslant s_2 + \frac{1}{\alpha}$, we have $h(s_2) \leqslant h(s_1)$ or $h(s_2) > h(s_1) + P\left(\frac{1}{\alpha}\right)$. Therefore, in order to distinguish both cases it is enough to find an approximation of $h(s_1)$ and $h(s_2)$ (by at most a $P\left(\frac{1}{\alpha}\right)/2$ factor). However, one may wonder how to find these estimations, since s_1 and s_2 are unknown. Recall that we have access to the oracles $\mathcal{O}_{s_1}$ and $\mathcal{O}_{s_2}$ which are such that

$$\mathbb{P}\left(\mathcal{O}_{s_i}(x) = 1\right) = p(s_i + x).$$

The idea of the proof is then to estimate $\mathbb{P}\left(\mathcal{O}_{s_i}(x) = 1\right)$ by running $\mathcal{O}_{s_i}(x)$ many times (one call to it costs a time T) and repeating this process for many different values of x. It will give an approximation of the graph of the map $x \mapsto p(s_i + x)$ and therefore an estimate of $h(s_i)$. All of this can be achieved by using the most basic statistical tool: empirical estimators of the expectation. The procedure is described in Algorithm 1.

Algorithm 1: Estimator of $h(s_i)$

Parameters: N_{iter}, $x_{\max}$ and δ
Input : $\mathcal{O}_{s_i}$
Output : $\overline{h}(s_i) \in \mathbb{R}_+$ be the estimation of $h(s_i)$
for $j = 0, \dots, X_{\max} \overset{def}{=} \lfloor \frac{x_{\max}}{\delta} \rfloor$ **do**
  $\overline{p}(j) = 0$
  for $\ell = 0, \dots, N_{\text{iter}} - 1$ **do**
    $b = \mathcal{O}_{s_i}(\delta j)$
    $\overline{p}(j) = \overline{p}(j) + \frac{b}{N_{\text{iter}}}$ ▷ We compute here the empiric value of $\mathbb{P}\left(\mathcal{O}_{s_i}(\delta j) = 1\right)$
  end
end
return $\overline{h}(s_i) = \max_j (1 + \delta j)\left|\overline{p}(j) - \overline{p}(X_{\max})\right|$

Parameters N_{iter}, $x_{\max}$ and δ of Algorithm 1 will be chosen later. Notice that one call to this algorithm costs a time given by

$$X_{\max}\, N_{\text{iter}}\, T, \text{ where, } \left(X_{\max} \overset{\text{def}}{=} \left\lfloor \frac{x_{\max}}{\delta} \right\rfloor\right),$$

as one call to $\mathcal{O}_{s_i}$ costs T. Furthermore, $X_{\max} N_{\text{iter}}$ is the number of queries to the oracle $\mathcal{O}_{s_i}$. Our aim is to show that for well chosen parameters

$$\mathbb{P}\left(\left|\overline{h}(s_i) - h(s_i)\right| \geqslant \frac{P\left(\frac{1}{\alpha}\right)}{2}\right) \leqslant e^{-\alpha}, \tag{21}$$

where the probability is computed over $\mathcal{O}_{s_i}$ which is itself used to compute $\overline{h}(s_i)$. This will conclude the proof, since in the case $s_1 \leqslant s_2$ we will have $\overline{h}(s_1) - \overline{h}(s_2) \geqslant -P\left(\frac{1}{\alpha}\right)$ and in the other case $\overline{h}(s_1) - \overline{h}(s_2) < -P\left(\frac{1}{\alpha}\right)$ with probability $\geqslant 1 - e^{-\alpha}$.

To prove this statement, let us first show that for all $\chi \in [0, 1]$ and all $j \in [\![0, X_{\max}]\!]$,

$$\left| \left|\overline{p}(j) - \overline{p}(X_{\max})\right| - \left|(p(s_i + (j + \chi)\delta) - p_\infty)\right| \right| \leqslant 2Y + \alpha 2^{-\frac{s_i + X_{\max}\delta}{\alpha}} \tag{22}$$

holds with probability larger than $1 - 3X_{\max} e^{-2N_{\text{iter}}Y^2}$.

Notice that $\overline{p}(j)$ is the empirical expectation of $\mathcal{O}_{s_i}(\delta j)$ where $\mathbb{E}\left(\mathcal{O}_{s_i}(\delta j)\right) = \mathbb{P}\left(\mathcal{O}_{s_i}(\delta j) = 1\right) = p(s_i + \delta j)$ (the oracle only outputs 1 or 0). Therefore, by Chernoff's bound, for some $Y \geqslant 0$, we

have

$$\mathbb{P}\left(|\overline{p}(j) - p(s_i + \delta j)| \geqslant Y\right) = \mathbb{P}\left(|\overline{p}(j) - \mathbb{P}\left(\mathcal{O}_{s_i}(\delta j) = 1\right)| \geqslant Y\right) \leqslant 2e^{-2N_{\text{iter}}Y^2}. \tag{23}$$

Next, from the union bound,

$$\mathbb{P}\left(\forall j \in [\![0, X_{\max}]\!],\ \ |\overline{p}(j) - p(s_i + \delta j)| \leqslant Y\right) \geqslant 1 - 2X_{\max}\, e^{-2N_{\text{iter}}Y^2}. \tag{24}$$

Let us now make the following computation for $\chi \in [0,1]$,

$$\begin{aligned}|\overline{p}(j) - p(s_i + (j+\chi)\delta)| &\leqslant |\overline{p}(j) - p(s_i + j\delta)| + |p(s_i + j\delta) - p(s_i + (j+\chi)\delta)| \\ &\leqslant |\overline{p}(j) - p(s_i + j\delta)| + \alpha\delta,\end{aligned}$$

where in the last line we used assumption *(iii)* that p is α-lipschitz together with $\chi \in [0,1]$. According to Equation (24),

$$\mathbb{P}\Big(\forall j \in [\![0, X_{\max}]\!],\ |\overline{p}(X_{\max}) - p(s_i + (j+\chi)\delta)| \leqslant Y + \alpha\delta\Big) \geqslant 1 - 2X_{\max}\, e^{-2N_{\text{iter}}Y^2}. \tag{25}$$

Furthermore,

$$\begin{aligned}|\overline{p}(X_{\max}) - p_\infty| &\leqslant |\overline{p}(X_{\max}) - p(s_i + X_{\max}\delta)| + |p(s_i + X_{\max}\delta) - p_\infty| \\ &\leqslant |\overline{p}(X_{\max}) - p(s_i + X_{\max}\delta)| + \alpha 2^{-\frac{s_i + x_{\max}}{\alpha}},\end{aligned}$$

where we used assumption *(ii)* on p. According to (23),

$$\mathbb{P}\left(|\overline{p}(X_{\max}) - p_\infty| \leqslant Y + \alpha 2^{-\frac{s_i + x_{\max}}{\alpha}}\right) \geqslant 1 - 2e^{-2N_{\text{iter}}Y^2}. \tag{26}$$

Notice now by triangle inequalities,

$$\begin{aligned}&\Big|\ |\overline{p}(j) - \overline{p}(X_{\max})| - |(p(s + (j+\chi)\delta) - p_\infty)|\ \Big| \\ &\qquad\leqslant |\overline{p}(j) - \overline{p}(X_{\max}) - (p(s + (j+\chi)\delta) - p_\infty)| \\ &\qquad\leqslant |\overline{p}(j) - p(s + (j+\chi)\delta)| + |\overline{p}(X_{\max}) - p_\infty|\,.\end{aligned}$$

Therefore, combining the union bound with (25) and (26) leads to our claim given in (22). Let us define now,

$$\overline{q}(j) \overset{\text{def}}{=} (1 + j\delta)\,|\overline{p}(j) - \overline{p}(X_{\max})| \quad \text{and} \quad q(x) \overset{\text{def}}{=} (1+x)\,|p(s_i + x) - p_\infty|\,.$$

We have the following computation,

$$\begin{aligned}&|\overline{q}(j) - q((j+\chi)\delta)| \\ &\quad= \Big|(1 + j\delta)\,|\overline{p}(j) - \overline{p}(X_{\max})| - (1 + (j+\chi)\delta)\,|p(s_i + (j+\chi)\delta) - p_\infty|\,\Big| \\ &\quad\leqslant (1 + j\delta)\Big|\ |\overline{p}(j) - \overline{p}(X_{\max})| - |p(s_i + (j+\chi)\delta) - p_\infty|\ \Big| \\ &\qquad+ \chi\delta\,|p(s_i + (j+\chi)\delta) - p_\infty| \\ &\quad\leqslant (1 + x_{\max})\Big|\ |\overline{p}(j) - \overline{p}(X_{\max})| - |p(s_i + (j+\chi)\delta) - p_\infty|\ \Big| + \alpha\delta 2^{-\frac{s_i + (j+\chi)\delta}{\alpha}},\end{aligned}$$

where in the last line, we used assumption *(ii)* on p together with $\chi \leqslant 1$. Therefore, according to (22), for all j and $\chi \in [0,1]$,

$$|\overline{q}(j) - q((j+\chi)\delta)| \leqslant 2(1 + x_{\max})Y + \alpha 2^{-\frac{s_i + x_{\max}}{\alpha}} + \alpha\delta \tag{27}$$

with probability $\geqslant 1 - 3X_{\max} e^{-2N_{\text{iter}}Y^2}$. Notice now that, by definition of $\overline{h}(s_i)$ and $h(s)$, we have

$$\begin{aligned}\left|\overline{h}(s_i) - h(s_i)\right| &\leqslant \left| \max_j \overline{q}(j) - \max_{\substack{j\in]\!]0,X_{\max}]\!] \\ \chi\in[0,1] \\ (j+\chi)\delta\leqslant x_{\max}}} q((j+\chi)\delta)\right| + \max_{t\geqslant x_{\max}} q(t) \\ &\leqslant \max_{j,\chi} |\overline{q}(j) - q((j+\chi)\delta)| + \alpha(1+x_{\max})2^{-\frac{x_{\max}}{\alpha}},\end{aligned}$$

where in the last line, we used assumption *(ii)*. Therefore, by plugging (27) in the above equations, we have with probability $\geqslant 1 - 3X_{\max}e^{-2NY^2}$,

$$\begin{aligned}\left|\overline{h}(s_i) - h(s_i)\right| &\leqslant 2Y + \alpha 2^{-\frac{s_i+x_{\max}}{\alpha}} + \alpha\delta + \alpha(1+x_{\max})2^{-\frac{x_{\max}}{\alpha}} \\ &\leqslant 2Y + 2\alpha(1+x_{\max})2^{-\frac{x_{\max}}{\alpha}} + \alpha\delta.\end{aligned}$$

Now, let us choose parameters such that

$$x_{\max} = -\alpha \log \frac{P\left(\frac{1}{\alpha}\right)}{6\alpha(1+x_{\max})}, \quad Y = \frac{P\left(\frac{1}{\alpha}\right)}{12} \quad \text{and} \quad \delta = \frac{P\left(\frac{1}{\alpha}\right)}{6\alpha}.$$

Plugging this in Equation leads to

$$\left|\overline{h}(s_i) - h(s_i)\right| \leqslant \frac{P\left(\frac{1}{\alpha}\right)}{2}.$$

Furthermore, by choosing N_{iter} as

$$-2N_{\text{iter}}Y^2 = -\alpha + \log\left(\frac{1}{3X_{\max}}\right) \iff N_{\text{iter}} = \frac{\alpha + \log(3X_{\max})}{2Y^2}.$$

Then the above inequality is true with probability $\geqslant 1 - e^{-\alpha}$. Recall that the cost of our algorithm is given by

$$X_{\max}\, N_{\text{iter}}\, T = X_{\max} \left(\frac{\alpha + \log(3X_{\max})}{P\left(\frac{1}{\alpha}\right)^2/72}\right) T = \mathsf{poly}(\alpha)T$$

as $X_{\max} = \left\lfloor \frac{x_{\max}}{\delta} \right\rfloor = \mathsf{poly}(\alpha)$. This concludes the proof. □

Equipped with this statement, we are almost ready to prove Theorem 1. However, it still remains to verify that the function p given in (14) satisfies the assumption of the lemma for some parameters α and p_∞.

Lemma 3. *We use the notation of Theorem 1. Let p be the function defined in (14), and let*

$$p_\infty \overset{def}{=} \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(\infty)\right) = 1\right) \qquad (\mathcal{O}^{\mathbf{0}}_{\text{ideal}} \text{ is defined in Figure 4}). \tag{28}$$

Then, we have

(i) $p(0) - p_\infty \geqslant \frac{1}{\alpha}$;
(ii) $|p(x) - p_\infty| \leqslant \alpha 2^{-\frac{x}{\alpha}}$;
(iii) p *is* α*-lipschitz;*

for some α satisfying

$$\alpha = C \max\left(\frac{1}{\varepsilon}, N, n\right) \tag{29}$$

for some large enough constant C and where ε is the distinguishing advantage of $\mathcal{A}$ given in Equation (13).

Proof. Let us first prove *(i)*. Following the discussion in Step 1, let $\mathcal{O}(\omega) = \mathcal{O}^{\mathbf{0}}_{\text{ideal}}(0)$ and $\mathcal{O}(\infty) = \mathcal{O}^{\mathbf{0}}_{\text{ideal}}(\infty)$ (defined in (9)). By definition of p,

$$\begin{aligned}p(0) - p_\infty &= \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}(\omega)\right) = 1\right) - \mathbb{P}\left(\mathcal{A}\left(\mathcal{O}(\infty)\right) = 1\right) \\ &= 2\varepsilon \\ &\geqslant \frac{1}{\alpha},\end{aligned}$$

where in the last line we used the assumption on α given in Equation (29).

Let us prove *(ii)*. Using the data processing inequality (2) together with (3), for $X \leftarrow \mathrm{Ber}(2^x\omega_0 t)$ and $Y \leftarrow \mathrm{Ber}(\infty)$, we have

$$\begin{aligned}|p(x) - p(\infty)| &\leqslant N\,\Delta\left(X, Y\right)\\ &= N\,2^{-2^x\omega_0 t}.\end{aligned}$$

Notice now that

$$N2^{-2^x\omega_0 t} \leqslant \alpha 2^{-\frac{x}{\alpha}} \iff \log(N) - 2^x\omega_0 t \leqslant -\frac{x}{\alpha} + \log(\alpha),$$

and the last equality is verified for all $x \geqslant 0$ since, from (6), we know that $\omega_0 t = \omega = \Omega(1)$ and $\alpha \geqslant CN$ for some large enough constant C. It proves item *(ii)*.

We are now ready to finish the proof by proving item *(iii)*. In the same manner as before, for $X \leftarrow \mathrm{Ber}(2^x\omega_0 t)$ and $Y \leftarrow \mathrm{Ber}(2^y\omega_0 y)$ and for all $x, y \geqslant 0$, we have

$$\begin{aligned}|p(x) - p(y)| &\leqslant N\Delta(X, Y)\\ &= N\left|2^{-2^x\omega_0 t} - 2^{-2^y\omega_0 t}\right|\\ &\leqslant N\omega_0 t\,|x - y|\,,\end{aligned}$$

where the last inequality follows from the mean value theorem. Notice now that $N\omega_0 t \leqslant \alpha$ as $N\omega_0 t = N\omega = O(Nn)$. It concludes the proof. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. The algorithm recovering $\mathbf{t}$ from $\mathbf{mG} + \mathbf{t}$ simply runs for any $i \in [\![1, n]\!]$ the procedure of Lemma 2 with oracles $\mathcal{A}(\mathcal{O}^{\mathbf{0}}(x))$ and $\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}(x)\right)$. However to see why it works, let us make the analyse of the success probability as if the following oracles were given

$$\mathcal{O}_{s_1}(x) \stackrel{\text{def}}{=} \mathcal{A}\left(\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)\right) \quad \text{and} \quad \mathcal{O}_{s_2}(x) \stackrel{\text{def}}{=} \mathcal{A}\left(\mathcal{O}^{\mathbf{v_i}}_{\text{ideal}}(x)\right).$$

with $s_1 = 0$ and s_2 chosen later. Notice that according to the definition of p given in Equation (14) we have

$$\mathbb{P}\left(\mathcal{O}_{s_1}(x) = 1\right) = p(s_1 + x) \quad \text{and} \quad \mathbb{P}\left(\mathcal{O}_{s_2}(x) = 1\right) = p(s_2 + x)$$

where s_2 is such that (see Equation (15))

$$s_2 = \begin{cases} \log\left(1 - \frac{1}{t}\right) & \text{if } t_i = 1\\ \log\left(1 + \frac{1}{t}\right) & \text{otherwise.}\end{cases}$$

Therefore, for $t \geqslant 1$, either

$$s_2 > s_1 = 0 \quad \text{if} \quad t_i = 0$$

or,

$$s_2 + \frac{1}{t} = \log\left(1 - \frac{1}{t}\right) + \frac{1}{t} \leqslant 0 = s_1 \quad \text{if} \quad t_i = 1.$$

Consequently, to apply Lemma 2 we need to have $\alpha \geqslant t$. But the function p has also to verify items *(i)*, *(ii)* and *(iii)* of the lemma. According to Lemma 3, all these assumptions are met if we choose α as a $\Theta\left(\max\left(\frac{1}{\varepsilon}, N, n\right)\right)$ (recall that $t \leqslant n$). Notice that $\mathsf{poly}(\alpha) = \mathsf{poly}\left(\max\left(\frac{1}{\varepsilon}, N, n\right)\right)$.

Running the procedure of Lemma 2 for any $i \in [\![1, n]\!]$ will output the support of $\mathbf{t}$, namely $\{i \in [\![1, n]\!], t_i \neq 0\}$, with probability

$$\geqslant \left(1 - e^{-\alpha}\right)^n = \left(1 - e^{-\Omega(n)}\right)^n = 1 - 2^{-\Omega(n)}.$$

and in time $T\mathsf{poly}(\alpha)$.

However, in reality it is not possible to run the procedure with the ideal oracles. Instead, we need to use oracles $\mathcal{A}(\mathcal{O}^{\mathbf{0}}(x))$ and $\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}(x)\right)$. But, according to Lemma 1, the statistical distance between outputs of $\mathcal{O}^{\mathbf{0}}(x)$ and $\mathcal{O}^{\mathbf{0}}_{\text{ideal}}(x)$ is smaller than

$$\Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), \left(\mathbf{a}, e(x)\right)\Big)$$

where $\mathbf{a} \leftarrow \mathbb{F}_2^k$, $\mathbf{r}(x) \leftarrow \mathrm{Ber}(2^x\omega_0)^{\otimes n}$ and $e(x) \leftarrow \mathrm{Ber}(2^x\omega_0 t)$. Furthermore we have the same upper-bound between outputs of $\mathcal{O}^{\mathbf{v}_i}(x)$ and $\mathcal{O}^{\mathbf{v}_i}_{\mathrm{ideal}}(x)$ except that we have to replace $\mathbf{t}$ by $\mathbf{t}+\mathbf{v}_i$ and $e(x) \leftarrow \mathrm{Ber}(2^x\omega_0(t \pm 1))$ as $|\mathbf{v}_i| = 1$. In both cases, the statistical distances are equal up to a factor $(1 + 2^{-\Omega(n)})$.

Therefore, by using the data processing inequality and Equation (3), the procedure will recover the support of $\mathbf{t}$ in the same time and with probability

$$\geqslant 1 - 2^{-\Omega(n)} - N' \max_{x \geqslant 0} \Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), (\mathbf{a}, e(x)) \Big)$$

where N' is the number of queries that our procedure makes to oracles $\mathcal{O}^{\mathbf{0}}(x)$ and $\mathcal{O}^{\mathbf{v}_i}(x)$. According to Lemma 2, the procedure makes $\mathsf{poly}(\alpha)$ queries to its input oracle which is here $\mathcal{A}(\mathcal{O}^{\mathbf{0}}(x))$ and $\mathcal{A}\left(\mathcal{O}^{\mathbf{v}_i}(x)\right)$. But at the same time, $\mathcal{A}$ makes N queries to its input oracles. Therefore $N' = \mathsf{poly}(\alpha)N$ which concludes the proof. □

Remark 4. *This algorithm bares similarities with the* OHCP *framework introduced in* [PRS17] *to prove pseudorandomness of the ring-*LWE *distribution. However, contrary to the lattice-based setting, in the case of codes we do not need to introduce a random walk towards a center. Indeed, in the Hamming metric, the support gathers all the needed information to recover the error. The situation is even simpler in the case of the binary field $\mathbb{F}_2$, for there are only two situations: either the error is 1 or 0. For a bigger finite field $\mathbb{F}_q$, we would have to distinguish between a 0 value or a non-zero error, letting us with $q-1$ choices for the actual error value. However, the information "being in the support or not" is enough to recover the error, even if that means solving a linear system. Note that this remark also applies to the rank metric, which could be a good starting point to design search-to-decision reductions for codes endowed with this metric.*

4. Instantiations

4.1. Plain decoding.

In order to instantiate the above reduction, we need to carefully understand how close our oracle $\mathcal{O}^{\mathbf{z}}(x)$ is to output LPN-like samples, from genuine LPN samples which are produced by $\mathcal{O}^{\mathbf{z}}_{\mathrm{ideal}}(x)$ (see Figure 4). That is to say, we want to understand when the additive term

$$N\mathsf{poly}(\alpha)\, \Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), (\mathbf{a}, e(x)) \Big)$$

in Equation (8) is negligible. Recall that $|\mathbf{t}| = t$, $\mathbf{r}(x) \leftarrow \mathrm{Ber}\left(2^x\omega_0\right)^{\otimes n}$ and $e \leftarrow \mathrm{Ber}\left(2^x\omega_0 t\right)^{\otimes n}$. In other words, we want to understand for which parameters ω_0, x the distribution of $\mathbf{r}(x)$ *smoothes* the dual of the code generated by $\mathbf{G}$. We will consider two situations:

- **Average-case to Average-case reduction:** for cryptographic applications, we need to assess the hardness of our problem *on average*. In this situation, the matrix $\mathbf{G}$ is chosen uniformly at random in $\mathbb{F}_2^{k\times n}$. This yields another search to decision reduction for the plain decoding problem, completely different than that of [FS96]. Furthermore this gives a sense of the best sorts of trade-off between parameters that we can achieve with our reduction. The main ingredient here will be the following lemma proved in Appendix A which is a variation of [DST19, Lemma 3, §C.1], itself a particular case of the famous leftover hash lemma (see [BDK+11]).

  Lemma 4. *Let E, F be finite sets. Let $\mathcal{H} = (h_i)_{i\in I}$ be a finite family of applications from E to F and $T \subseteq E$. Let t be drawn uniformly at random in T and $r \in E$ be a random variable distributed according to some distribution $\mathcal{D}$. Let,*

  $$p \overset{def}{=} \mathbb{P}_{t,r}\left(\langle r, t\rangle = 1\right) \tag{30}$$

  where $\langle\cdot,\cdot\rangle$ is a map from $E \times E \to \{0,1\}$. Let η be the "collision bias" defined by

  $$\mathbb{P}_{h,t,r_0,r_1}\Big(h(r_0) = h(r_1), \langle t, r_0\rangle = \langle t, r_1\rangle\Big) \leqslant \frac{1}{\sharp F}(p^2 + (1-p)^2 + \eta) \tag{31}$$

  where h, t are uniformly drawn in $\mathcal{H}$ and T respectively and r_0, r_1 be independent and distributed according to $\mathcal{D}$.

Let Y be the random variable (u, e) where u is uniform over F and $e \in \{0,1\}$ is a Bernoulli random variable of parameter p and u, e are independent. Let $Y(h,t)$ be the random variable $(h(r), \langle r, t\rangle)$ when r is distributed according to $\mathcal{D}$. We have,

$$\mathbb{E}_{h,t}\left(\Delta(Y(h,t), Y)\right) \leqslant \sqrt{\eta}.$$

In our case, the functions will be defined as $h(r) = \mathbf{r}\mathbf{G}^\top$ where $\mathbf{G}$ ranges over a family of matrices, typically double circulant matrices, or the full space of $k \times n$ matrices; and $\langle \cdot, \cdot \rangle$ will stand for the canonical inner product over $\mathbb{F}_2^n$.

- **Worst-case to Average-case reduction:** on a more theoretical perspective, one can wonder on the worst-case hardness of the decision decoding problem. Such a result has been obtained for lattices, proving for instance that different flavors of LWE are at least as hard as *worst-case* problems on (different classes of) Euclidean lattices. The main ingredient here will be the *smoothing bounds* of [BLVW19, YZ21, DDRT22, DR22]. This is the first time that such a reduction is derived from the OCP framework in the code–based setting.

Average-case to average-case reduction. In this paragraph, we consider the plain decoding problem. First, we prove the following lemma. It will yield the noise allowed in the decision problem of the reduction.

Lemma 5. *Let $\beta, \eta \in (0,1)$, $k \leqslant n \in \mathbb{N}$, $t \in [\![1, n]\!]$ and $\omega_0 \in \mathbb{R}_+$ be such that*

$$\omega_0 \geqslant -\log_2\left(1 - 2\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(\frac{k}{n}\right)\right) \tag{32}$$

with $h^{-1} : [0,1] \to [0, \frac{1}{2}]$ being the inverse of the binary entropy function h. Then, for all $x \geqslant 0$,

$$\mathbb{E}_{\mathbf{G},\mathbf{t}}\left(\Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), (\mathbf{a}, e(x))\Big)\right) = 2^{-\Omega(n)}$$

where $\mathbf{a} \leftarrow \mathbb{F}_2^k$, $\mathbf{r}(x) \leftarrow \mathrm{Ber}(2^x\omega_0)^{\otimes n}$, $e(x) \leftarrow \mathrm{Ber}(2^x\omega_0 t)$, $\mathbf{G} \leftarrow \mathbb{F}_2^{k\times n}$ and $\mathbf{t} \leftarrow \mathcal{S}_t^n$ being the sphere of radius t around $\mathbf{0}$ in $\mathbb{F}_2^n$.

It is a corollary of Lemma 4 and [DR22, Proposition 6.7] recalled below, which shows that the Bernoulli distribution inherits the smoothing properties of the uniform distribution over a Hamming sphere.

Proposition 1 ([DR22, Proposition 6.7])**.** *Let $\mathbf{t} \in \mathbb{F}_2^n$, $\beta > 0$, $\rho \in \mathbb{R}_+$ and $p \overset{def}{=} \frac{1}{2}\left(1 - 2^{-\rho}\right)$. Let $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ be the generator matrix of an $[n,k]$-code. Then,*

$$\Delta\Big(\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{r}, \mathbf{t}\rangle\right), (\mathbf{a}, e)\Big) \leqslant \sum_{r=(1-\beta)np}^{(1+\beta)np} \Delta\Big(\left(\mathbf{r}_r\mathbf{G}^\top, \langle \mathbf{r}_r, \mathbf{t}\rangle\right), (\mathbf{a}, e_r)\Big) + 2^{-\Omega(n)}$$

where $\mathbf{r} \leftarrow \mathrm{Ber}\left(\rho\right)^{\otimes n}$, $\mathbf{a} \leftarrow \mathbb{F}_2^k$, $e \leftarrow \mathrm{Ber}\left(\rho|\mathbf{t}|\right)$, $\mathbf{r}_r \leftarrow \mathcal{S}_r$ and the e_r's are distributed as the $\langle \mathbf{r}_r, \mathbf{t}\rangle$'s.

Remark 5. *Note that Equation (32) is equivalent to*

$$\frac{1}{2}\left(1 - 2^{-\omega_0}\right)(1-\beta) \geqslant (1+\eta)h^{-1}\left(\frac{k}{n}\right). \tag{33}$$

That is to say, require the least index in the sum in Proposition 1 to be above the Gilbert-Varshamov bound. This is a necessary condition for the statistical distances to be negligible.

We are now ready to prove Lemma 5. We will proceed in two steps: first we show that it holds when $\mathbf{r}$ is instead uniformly distributed over the sphere of radius $\frac{n}{2}\left(1 - 2^{-e^x\omega_0}\right)(1-\beta)$; we then apply Proposition 1.

Proof of Lemma 5. To ease the reading, let us drop the dependency in x (the maximum of the statistical distance is reached for $x = 0$; taking $x \geqslant 0$ can only decrease this statistical distance as it increases the noise). Let $r \stackrel{\text{def}}{=} \frac{n}{2}(1 - 2^{-\omega_0})(1 - \beta)$ and $\mathbf{r} \leftarrow \mathcal{S}_r$. Our aim is to show that the result holds for this distribution. To conclude the proof it will just remain to apply Proposition 1. By Lemma 4, it suffices to compute the collision probability (where $\mathbf{r}_0, \mathbf{r}_1 \leftarrow \mathcal{S}_r$, $\mathbf{G} \leftarrow \mathbb{F}_2^{k \times n}$ and $\mathbf{t} \leftarrow \mathcal{S}_t$)

$$\begin{aligned}
&\mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1,\mathbf{G},\mathbf{t}}\Big(\mathbf{r}_0\mathbf{G}^\top = \mathbf{r}_1\mathbf{G}^\top, \langle \mathbf{t}, \mathbf{r}_0\rangle = \langle \mathbf{t}, \mathbf{r}_1\rangle\Big) \\
&= \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1,\mathbf{G},\mathbf{t}}\Big((\mathbf{r}_0 - \mathbf{r}_1)\,\mathbf{G}^\top = \mathbf{0}, \langle \mathbf{t}, \mathbf{r}_0 - \mathbf{r}_1\rangle = 0\Big) \\
&= \sum_{\mathbf{r}\neq\mathbf{0}} \mathbb{P}_{\mathbf{G}}\left(\mathbf{r}\mathbf{G}^\top = \mathbf{0}\right) \mathbb{P}_{\mathbf{t}}\left(\langle \mathbf{t}, \mathbf{r}\rangle = 0\right) \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1}\left(\mathbf{r}_0 - \mathbf{r}_1 = \mathbf{r}\right) + \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1}\left(\mathbf{r}_0 = \mathbf{r}_1\right) \\
&= \frac{1}{2^k} \sum_{\mathbf{r}\neq\mathbf{0}} \mathbb{P}_{\mathbf{t}}\left(\langle \mathbf{t}, \mathbf{r}\rangle = 0\right) \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1}\left(\mathbf{r}_0 - \mathbf{r}_1 = \mathbf{r}\right) + \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1}\left(\mathbf{r}_0 = \mathbf{r}_1\right) \\
&\leqslant \frac{1}{2^k}\left(\mathbb{P}_{\mathbf{t},\mathbf{r}_0,\mathbf{r}_1}\left(\langle \mathbf{t}, \mathbf{r}_0 - \mathbf{r}_1\rangle = 0\right) + 2^k \mathbb{P}_{\mathbf{r}_0,\mathbf{r}_1}\left(\mathbf{r}_0 = \mathbf{r}_1\right)\right) \\
&= \frac{1}{2^k}\left(p^2 + (1-p)^2 + \frac{2^k}{\binom{n}{r}}\right)
\end{aligned}$$

where $p \stackrel{\text{def}}{=} \mathbb{P}_{\mathbf{r},\mathbf{t}}\left(\langle \mathbf{t}, \mathbf{r}\rangle = 1\right)$ and we used in the inequality the law of total probability. By Lemma 4,

$$\mathbb{E}_{\mathbf{G},\mathbf{t}}\left(\Delta\Big(\left(\mathbf{r}_r\mathbf{G}^\top, \langle \mathbf{r}_r, \mathbf{t}\rangle\right), (\mathbf{a}, e_r)\Big)\right) \leqslant \sqrt{\frac{2^k}{\binom{n}{r}}}.$$

Recall that $\binom{n}{r} = 2^{nh(r/n)(1+o(1))}$ where h denotes the binary entropy function. By Equation (33), r verifies $(1+\eta)h^{-1}\left(\frac{k}{n}\right) \leqslant \frac{r}{n} \leqslant 1/2$. Therefore, since h is a strictly increasing function, the above upper-bound is a $2^{-\Omega(n)}$. This yields the claimed result. □

Recall that in Theorem 1, the considered (search) decoding problem is fixed once and for all. However, the above lemma tells us that on average, on the choice of $\mathbf{G}$ and $\mathbf{t}$, the considered statistical distance is negligible. We can actually prove that it holds for *almost* all choices.

Lemma 6. *Let $k \leqslant n \in \mathbb{N}, t \in [\![0, n]\!]$. For a matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ and a vector $\mathbf{t} \in \mathbb{F}_2^n$ of Hamming weight t, denote*

$$X(\mathbf{G}, \mathbf{t}) \stackrel{def}{=} \Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), (\mathbf{a}, e(x))\Big).$$

Let,

$$\gamma \stackrel{def}{=} \mathbb{E}_{\mathbf{G}_u,\mathbf{t}_u}\left(X(\mathbf{G}_u, \mathbf{t}_u)\right)$$

where $\mathbf{G}_u \leftarrow \mathbb{F}_2^{k\times n}$ and $\mathbf{t}_u \leftarrow \mathcal{S}_t$. Then,

$$\frac{\sharp\{(\mathbf{G}, \mathbf{t}) \in \mathbb{F}_2^{k\times n} \times \mathcal{S}_t \mid X(\mathbf{G}, \mathbf{t}) \geqslant \sqrt{\gamma}\}}{2^{kn}\binom{n}{t}} \leqslant \sqrt{\gamma}.$$

Proof. Since $\mathbf{G}_u$ and $\mathbf{t}_u$ are independent and uniformly distributed over their respective domains, this proportion is nothing else than $\mathbb{P}_{\mathbf{G}_u,\mathbf{t}_u}(X(\mathbf{G}_u, \mathbf{t}_u) \geqslant \sqrt{\gamma})$. By Markov inequality, we have

$$\mathbb{P}_{\mathbf{G}_u,\mathbf{t}_u}(X(\mathbf{G}_u, \mathbf{t}_u) \geqslant \sqrt{\gamma}) \leqslant \frac{\mathbb{E}_{\mathbf{G}_u,\mathbf{t}_u}(X(\mathbf{G}_u, \mathbf{t}_u))}{\sqrt{\gamma}} \leqslant \sqrt{\gamma},$$

which concludes the proof. □

We are now ready to instantiate our search-to-decision average-to-average case reduction. However, in the same manner as discussed in [DR22, §6], parameters have to be carefully chosen to ensure that the decision problem is not *too hard* and its search counterpart into which we reduce is not *too easy*.

Notice that the noise of the decision decoding problem of the reduction is distributed as $\mathrm{Ber}\,(\omega_0 t)$ with ω_0 given in Equation (32). If one chooses k, n such that $\frac{k}{n} = \Theta(1)$, one would obtain a noise distributed as $\mathrm{Ber}(\omega_0 t) = \mathrm{Ber}\,(\Theta(t))$. In that case, it seems that we need to choose t as a $O(\log_2(n))$ to reach a noise rate $1/2(1-2^{-\omega_0 t}) = 1/2 - 1/\mathsf{poly}(n)$ in the decision decoding problem. Otherwise, we would reduce the decoding problem into a decision decoding problem with a noise rate *exponentially or sub-exponentially* close to $1/2$; an extremely hard problem which is not very satisfactory. On the other hand, choosing $t = O(\log_2(n))$ is a real disaster for the reduction: decoding a code of length n at distance $O\left(\log_2(n)\right)$ can be done in polynomial time (using for instance Prange algorithm [Pra62]). That is, we would be reducing an *easy* worst-case search decoding problem to an average-case decision decoding problem; which says nothing about the hardness of the decision version. We therefore conclude that the only way to reach an error rate $1/2(1-2^{-\omega_0 t}) = 1/2 - 1/\mathsf{poly}(n)$ is to decrease as much as possible ω_0 given in Equation (32). In particular, we are led to choose $k/n = o(1)$, since in that case $\omega_0 = -\log_2(1-o(1)) = o(1)$. More precisely, for these parameters ω_0 verifies

$$\omega_0 = -\log_2\left(1 - \Theta\left(h^{-1}\left(\frac{k}{n}\right)\right)\right) \approx \frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}$$

where we used the expansion $h^{-1}(\varepsilon) \underset{\varepsilon\to 0}{\approx} \frac{\varepsilon}{\log_2(1/\varepsilon)}$. Therefore, to reach the noise rate $1/2 - 1/\mathsf{poly}(n)$ we need to choose parameters such that

$$\frac{k}{n} = o(1) \quad \text{and} \quad \omega_0 t = \frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}\,t = O\left(\log_2(n)\right). \tag{34}$$

Notice that necessarily in the above choice of parameters, we need t to be sublinear in n, since otherwise k would be too small, allowing an exhaustive search to decode in polynomial time. Fortunately, in that case the reduction is non-trivial. The cost of Prange's algorithm [Pra62] (which is asymptotically the best known decoding algorithm when the decoding distance t is sublinear in the length of the input code, see [CS16]) is given by

$$2^{\Theta\left(t\,\frac{k}{n}\right)} = 2^{\Theta(\log_2(n)\log_2(n/k))} = n^{\Theta(\log_2(n/k))}$$

which is super-polynomial.

In what follows we focus our attention to a noise rate $1/2 - 1/\mathsf{poly}(n)$ in the decision problem, that is to say we propose parameters where the rate k/n of the codes considered in the reduction verifies $k/n = o(1)$.

Theorem 2. *Let $\beta, \eta \in (0,1)$, $C > 0$ and $n, k, t \in \mathbb{N}$ be such that*

$$\frac{k}{n} = o(1) \quad \textit{and} \quad \frac{2}{\ln(2)}\,\frac{1+\eta}{1-\beta}\,\frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}\,t = C\log_2(n). \tag{35}$$

Furthermore, let

$$\omega_0 \;=\; -\log_2\left(1 - 2\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(\frac{k}{n}\right)\right) \quad \textit{i.e.} \quad \frac{1-\beta}{2}\left(1-2^{-\omega_0}\right) \;=\; (1+\eta)h^{-1}\left(\frac{k}{n}\right). \tag{36}$$

Suppose that there exists an algorithm $\mathcal{A}$, with advantage $\varepsilon = \frac{1}{\mathsf{poly}(n)}$, which distinguishes in time T distributions $(\mathbf{A}, \mathbf{sA}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with

$$\mathbf{A} \leftarrow \mathbb{F}_2^{k\times n},\ \mathbf{s} \leftarrow \mathbb{F}_2^{k},\ \mathbf{y} \leftarrow \mathbb{F}_2^{n} \quad \textit{and} \quad \mathbf{e} \leftarrow \mathrm{Ber}\,(\omega_0 t)^{\otimes n}.$$

Then, there exists an algorithm running in time $T\mathsf{poly}(n)$, which takes as inputs $\mathbf{G} \in \mathbb{F}_2^{k\times n}$, $\mathbf{mG}+\mathbf{t}$ where $\mathbf{m} \in \mathbb{F}_2^k$, $\mathbf{t} \in \mathcal{S}_t^n$, and outputs $\mathbf{t}$ (or equivalently $\mathbf{m}$) with probability at least $1 - 2^{-\Omega(n)}$ over a uniform choice of $\mathbf{G}$ and $\mathbf{t}$.

Remark 6. *With the above parameter choice, we have*

$$\omega_0 t = C\log_2(n)(1+o(1))$$

i.e. the error rate in the decision *problem is*

$$\frac{1}{2}(1-2^{-\omega_0 t}) = \frac{1}{2} - \frac{1}{\mathsf{poly}(n)}.$$

Proof. We use the notations of Theorem 1 and Lemma 6. Let $\mathbf{G} \leftarrow \mathbb{F}_2^{k\times n}$ and $\mathbf{t} \leftarrow \mathcal{S}_t$. Notice that, since $k/n = o(1)$, the following computation holds

$$\begin{aligned}\omega_0 t &= -\log_2\left(1-2\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(\frac{k}{n}\right)\right)t\\ &= \frac{2}{\ln(2)}\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(\frac{k}{n}\right)t(1+o(1))\\ &= \frac{2}{\ln(2)}\,\frac{1+\eta}{1-\beta}\,\frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}\,t\,(1+o(1)),\end{aligned}$$

where we used the expansion $h^{-1}(x) = \frac{x}{\log_2(1/x)}(1+o(1))$. Therefore, by Equation (35), we have

$$\omega_0 t = C\log_2(n)(1+o(1)).$$

Let us consider now the algorithm $\mathcal{B}$ given by Theorem 1 which is obtained from an algorithm distinguishing distributions $(\mathbf{A}, \mathbf{sA}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with advantage $\varepsilon = \frac{1}{\mathsf{poly}(n)}$. It will output some $\mathbf{t}'$ in time $T\mathsf{poly}(\alpha)$ and with probability $1-2^{-\Omega(n)}-n\mathsf{poly}(\alpha)X(\mathbf{G},\mathbf{t})$. Notice that we do not have a max here because it is reached when $x=0$: the higher is the noise, the closer our distribution is from the genuine LPN. Since $\alpha = \max\left(\frac{1}{\varepsilon}, n\right) = \mathsf{poly}(n)$, then this probability is $1-2^{-\Omega(n)}$ when $X(\mathbf{G},\mathbf{t}) = 2^{-\Omega(n)}$. But since ω_0 is chosen as in Equation (36), we have $\mathbb{E}_{\mathbf{G},\mathbf{t}}\left(X(\mathbf{G},\mathbf{t})\right) = 2^{-\Omega(n)}$. Therefore, according to Lemma 6, the proportion of $(\mathbf{G},\mathbf{t})$ for which it happens is $1-2^{-\Omega(n)}$ (since ω_0 was chosen such that $\gamma = \mathbb{E}_{\mathbf{G},\mathbf{t}}\left(X(\mathbf{G},\mathbf{t})\right) = 2^{-\Omega(n)}$). Moreover, the success probability of $\mathcal{B}$ is independent from $\mathbf{G}$ and $\mathbf{t}$. Therefore, the probability that $\mathcal{B}(\mathbf{G},\mathbf{t})$ outputs 1 will be greater than $(1-2^{-\Omega(n)})(1-2^{-\Omega(n)}) = 1-2^{-\Omega(n)}$, which concludes the proof. □

Remark 7. *In Theorem 2, we instantiated Theorem 1 with $N = n$ to get a decisional version of the actual decoding problem. However, we are not really limited by the length N of the input code in the decision decoding problem; we have total liberty in the choice of N. Increasing N would only increase the running time of the reduction. In other words, this reduction would also apply in the context of* LPN*, where N is* a priori *unbounded.*

Worst-case to average-case reduction. We will now deal with the worst-case to average-case reduction. Recall that in Theorem 1, we need to set the statistical distance between our produced samples and genuine LPN samples to be negligible. For a worst-case hardness we need it to be negligible for *any* code, *i.e.* for *any* matrix $\mathbf{G}$. To this end we will use smoothing bounds as given in [DR22, Proposition 7.6]. However, this bound is only stated when $\mathbf{G}$ is a generator matrix of an $[n,k]$-code which is balanced (in the same manner than in [BLVW19]).

Definition 3 (Balanced code)**.** *An $[n,k]$-code is δ-balanced if its minimum distance is at least δn and all the codewords have Hamming weight at most $(1-\delta)n$. That is, for all $\mathbf{x} \in \mathcal{C} \setminus \{\mathbf{0}\}$,*

$$\delta n \leqslant |\mathbf{x}| \leqslant (1-\delta)n.$$

In the worst-case to average-case search-to-decision reduction we will restrict "worst" instances to δ-balanced codes. Therefore, we will first need to fix an $[n,k]$-code $\mathcal{C}$ which is δ-balanced. A natural choice for δ is given by the relative *Gilbert-Varshamov* bound $h^{-1}\left(1-\frac{k}{n}\right)$ which appears ubiquitously in the coding-theoretic literature: amongst other contexts, it arises as the (expected) relative minimum distance of a random code of dimension k and length n (see for instance [BF02, §C]). However, for the same reasons as above with random codes, in order to reach a noise rate $1/2 - 1/\mathsf{poly}(n)$ in the decision problem, we will choose parameters k, n so that $k/n = o(1)$. Many other interesting sets of parameters for the reduction can be proposed, for instance choosing $k/n = \Theta(1)$ and $t/n = o(n)$ leading to a noise rate in the decision decoding problem $1/2 - 2^{-o(n)}$.

To reach a negligible statistical distance we will use the following proposition.

Proposition 2 ([DR22, Proposition 7.6]). *Let $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ be the generator matrix of an $[n,k]$-code which is δ-balanced with $1/2 \geqslant \delta \geqslant h^{-1}\left(1-\frac{k}{n}\right) \geqslant C$ for some constant $C>0$. Let $\mathbf{t} \in \mathbb{F}_2^n$ and suppose that $\frac{|\mathbf{t}|}{n} = o(1)$.*

Let $\beta, \eta > 0$ and $\rho \in \mathbb{R}_+$ be such that

$$(1-\beta)\,\frac{1}{2}(1-2^{-\rho}) \geqslant (1+\eta)h^{-1}\left(2\frac{k}{n} + D\frac{|\mathbf{t}|}{n}\right)$$

for some large enough constant D. Then,

$$\Delta\Big(\left(\mathbf{r}\mathbf{G}^\top, \langle \mathbf{r}, \mathbf{t}\rangle\right), (\mathbf{a}, e)\Big) = 2^{-\Omega(n)}$$

where $\mathbf{r} \leftarrow \mathrm{Ber}\,(\rho)^{\otimes n}$, $\mathbf{a} \leftarrow \mathbb{F}_2^k$ and $e \leftarrow \mathrm{Ber}\,(\rho|\mathbf{t}|)$.

This proposition allows to instantiate our reduction in the worst-to-average case in the following theorem.

Theorem 3. *Let $\beta, \eta \in (0,1)$, $C>0$ and $n,k,t \in \mathbb{N}$ be such that*

$$\frac{k}{n} = o(1), \quad \frac{t}{n} = o\left(\frac{k}{n}\right) \quad \text{and} \quad \frac{4}{\ln(2)}\,\frac{1+\eta}{1-\beta}\,\frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}\,t = C\log_2(n). \tag{37}$$

Furthermore, let (for some large enough constant D)

$$\omega_0 = -\log_2\left(1 - 2\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(2\frac{k}{n} + D\frac{t}{n}\right)\right)$$
$$\text{i.e.} \quad \frac{1-\beta}{2}\left(1-2^{-\omega_0}\right) = 2(1+\eta)h^{-1}\left(2\frac{k}{n} + D\frac{t}{n}\right). \tag{38}$$

Suppose that there exists an algorithm $\mathcal{A}$, with advantage $\varepsilon = \frac{1}{\mathsf{poly}(n)}$, which distinguishes in time T distributions $(\mathbf{A}, \mathbf{s}\mathbf{A}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with

$$\mathbf{A} \leftarrow \mathbb{F}_2^{k\times n},\ \mathbf{s} \leftarrow \mathbb{F}_2^k,\ \mathbf{y} \leftarrow \mathbb{F}_2^n \ \text{ and } \ \mathbf{e} \leftarrow \mathrm{Ber}\,(\omega_0 t)^{\otimes n} \qquad \text{where } \ \omega_0 t = C\log_2(n)(1+o(1)).$$

Then, there exists an algorithm running in time $T\mathsf{poly}(n)$, which takes as inputs $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ a (fixed) *generator matrix of a δ-balanced $[n,k]$ code (with $\delta \geqslant h^{-1}\left(1-\frac{k}{n}\right) = \frac{1}{2} - \sqrt{\frac{k}{n}}(1+o(1))$), a noisy codeword $\mathbf{m}\mathbf{G}+\mathbf{t}$ with $\mathbf{t}$ of Hamming weight t, and outputs $\mathbf{t}$ (or equivalently $\mathbf{m}$) with probability at least $1-2^{-\Omega(n)}$ (where the probability is* not *taken over the choice of $\mathbf{m}$, $\mathbf{G}$ and $\mathbf{t}$).*

Proof. We use the notations of Theorem 1 and Proposition 2. Notice that, since $k/n = o(1)$ and $t/n = o(k/n)$, we have the following computation

$$\begin{aligned}\omega_0 t &= -\log_2\left(1 - 2\,\frac{1+\eta}{1-\beta}\,h^{-1}\left(2\frac{k}{n} + D\frac{t}{n}\right)\right) t \\ &= \frac{4}{\ln(2)}\,\frac{1+\eta}{1-\beta}\,\frac{1}{\log_2\left(\frac{n}{k}\right)}\,\frac{k}{n}\,t\,(1+o(1))\end{aligned}$$

where we used the expansion $h^{-1}(x) = \frac{x}{\log_2(1/x)}(1+o(1))$. Therefore, by Equation (37), we have

$$\omega_0 t = C\log_2(n)(1+o(1)),$$

i.e.

$$\frac{1}{2}\left(1-2^{-\omega_0 t}\right) = \frac{1}{2}\left(1 - \frac{1}{n^{C(1+o(1))}}\right).$$

Let,

$$Y(\mathbf{G},\mathbf{t}) \stackrel{\text{def}}{=} \Delta\Big(\left(\mathbf{r}(x)\mathbf{G}^\top, \langle \mathbf{r}(x), \mathbf{t}\rangle\right), (\mathbf{a}, e(x))\Big).$$

Let us consider now the algorithm $\mathcal{B}$ given by Theorem 1 which is obtained from an algorithm distinguishing distributions $(\mathbf{A}, \mathbf{s}\mathbf{A}+\mathbf{e})$ and $(\mathbf{A}, \mathbf{y})$ with advantage $\varepsilon = \frac{1}{\mathsf{poly}(n)}$. It will output some $\mathbf{t}'$ in time $T\mathsf{poly}(\alpha)$ and with probability $1-2^{-\Omega(n)} - n\mathsf{poly}(\alpha)Y(\mathbf{G},\mathbf{t})$. Notice that we do not have

a max here because it is reached when $x = 0$: the higher is the noise, the closer our distribution is from the genuine LPN. Since $\alpha = \max\left(\frac{1}{\varepsilon}, n\right) = \mathsf{poly}(n)$, then this probability is $1 - 2^{-\Omega(n)}$ when $Y(\mathbf{G}, \mathbf{t}) = 2^{-\Omega(n)}$. But since ω_0 is chosen as in Equation (38) we have $Y(\mathbf{G}, \mathbf{t}) = 2^{-\Omega(n)}$. Moreover, the success probability of $\mathcal{B}$ is independent from $\mathbf{G}$ which concludes the proof. □

A set of parameters. One can apply Theorem 3 for instance with the following set of parameters

$$\frac{k}{n} = \frac{1}{n^D} \quad \text{and} \quad \frac{t}{n} = \frac{\log_2(n)^2}{n^{1-D}}$$

with $D < 1/2$. Theorem 3 shows that solving the decision-average decoding problem of codes with length n, dimension n^{1-D} at distance $1/2 - O\left(1/n^{D\ln(2)/4}\right)$ is at least as hard as decoding a fixed δ-balanced code (with $\delta \geqslant h^{-1}\left(1 - \frac{1}{n^D}\right)$) at distance $n^D \log_2(n)^2$. Note that, as noticed in [BLVW19, §1.1] or [YZ21] and even [BF02] (though not under the same terminology), most of the codes are δ-balanced, and no generic decoding algorithm is known to take advantage of this property.

5. Failed attempt: the case of structured codes

In the manner of [PRS17] and [RSW18], it would be very tempting to apply our reduction in the case of structured error correcting codes, such as quasi-cyclic codes. Such codes are used in NIST submissions BIKE and HQC because they offer a very good efficiency while keeping the same security parameter as truly random codes.

Quasi-cyclic codes are codes that have a generator matrix formed out by multiple circulant blocks, *i.e.* of the form

$$\begin{pmatrix} a_0 & a_{n-1} & \dots & a_1 \\ a_1 & a_0 & \dots & a_2 \\ a_2 & a_1 & \dots & a_3 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \dots & a_0 \end{pmatrix}.$$

In order to simplify the discourse, in the sequel we consider the situation of quasi-cyclic codes formed out by a *single* row of circulant blocks. This generalizes easily to multiple rows, which corresponds to what is sometimes called *module*-LPN in the literature (see for instance [BCG+20]).

Very conveniently, quasi-cyclic codes benefit from a so-called *polynomial* representation. Indeed, each vector of length n can be represented as an element of $\mathbb{F}_2[X]/(X^n - 1)$; such that the matrix-vector product is nothing but the usual product of polynomials.

Consider a quasi-cyclic code generated by a matrix $\mathbf{G}$ of rate R, *i.e.* with $1/R$ circulant blocks. A noisy codeword $\mathbf{y} = \mathbf{mG} + \mathbf{t}$ where $\mathbf{t}$ is a *regular* error of weight t (*i.e.* a concatenation of $1/R$ words of Hamming weight t, which is the usual noise considered with quasi-cyclic codes) yields $1/R$ noisy polynomials of the form $\mathbf{ma} + \mathbf{t}' \in \mathbb{F}_2[X]/(X^n - 1)$, using the polynomial representation.

Hence, we could change the inner product $\langle \cdot, \cdot \rangle$ in Theorem 1 by the following inner product (with value in $\mathbb{F}_2[X]/(X^n - 1)$): if $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_{1/R}) \in (\mathbb{F}_2[X]/(X^n - 1))^{1/R}$, define

$$\langle \mathbf{x}, \mathbf{y} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^{1/R} \mathbf{x}_i \mathbf{y}_i.$$

With this inner product in hand, given $\mathbf{y} = (\mathbf{ma}_1 + \mathbf{t}'_1, \dots, \mathbf{ma}_{1/R} + \mathbf{t}'_{1/R}) \in (\mathbb{F}_2[X]/(X^n - 1))^{1/R}$, we can compute $\langle \mathbf{y}, \mathbf{r} \rangle$ where $\mathbf{r} = (\mathbf{r}_1, \dots, \mathbf{r}_{1/R})$ and each $\mathbf{r}_i$ are distributed according to $\mathrm{Ber}(\omega_0)$, meaning in this context that all the n coefficients of $\mathbf{r}_i$ are distributed according to $\mathrm{Ber}(\omega_0)$. Then,

$$\langle \mathbf{y}, \mathbf{r} \rangle = \mathbf{m} \left(\sum_{i=1}^{1/R} \mathbf{a}_i \mathbf{r}_i \right) + \underbrace{\sum_{i=1}^{1/R} \mathbf{t}'_i \mathbf{r}_j}_{\mathsf{LPN}\text{ noise}} \tag{39}$$

We can then follow the same strategy than previously to prove a structured analogue of Theorem 1. However we have to show that sampling elements as in (39) is close to sample from the ring-LPN distribution instead of the plain LPN. Lemma 4 can be adapted to this case and collisions can be easily computed; therefore we are able to compute the noise from which the Bernoulli distribution smoothes the distribution, which would actually be roughly the same as in the unstructured case (this is actually a consequence of the fact that random quasi-cyclic codes of use in cryptography have on average a minimum distance reaching the Gilbert-Varshamov bound [GZ06], so as genuine random codes). However there is a strong caveat when one wants to estimate the noise in a sample given in Equation (39).

For the sake of simplicity, let us consider $R = 1$. Let $\mathbf{t} = \sum_{i=0}^{n-1} t_i X^i \in \mathbb{F}_2[X]/(X^n - 1)$ with Hamming weight t, namely with t non-zero coefficients. Now, sample $\mathbf{r} = \sum_{i=0}^{n-1} r_i X^i \in \mathbb{F}_2[X]/(X^n - 1)$ where $(r_0, \dots, r_{n-1}) \leftarrow \mathrm{Ber}(\omega_0)^{\otimes n}$. The noise in the built LPN-samples is given by the inner product between $\mathbf{t}$ and $\mathbf{r}$, namely

$$\mathbf{tr} = \sum_{k=0}^{n-1} \sum_{i+j \equiv k \bmod n} t_i r_j X^k. \tag{40}$$

Notice that each coefficient of $\mathbf{tr}$ is exactly the sum of t independent $\mathrm{Ber}(\omega_0)$ random variables, therefore is a Bernoulli random variable of parameter $t\omega_0$. It may seem at first glance that we obtain the same analysis than in the plain case, starting from a noisy codeword $\mathbf{y} = \mathbf{ma} + \mathbf{t}$ we build LPN-like samples with Bernoulli noise given by $\mathrm{Ber}(\omega_0 t)$. There is a strong caveat here though: the coefficients of the product in Equation (40) are *not* independent, even though this inner product would have the good Hamming weight on average. Therefore our new noise does not follow the right distribution.

It turns out that this distribution is very difficult to analyze, and this fact was already emphasized in the HQC submission to the NIST [AAB+21b] when studying the Decoding Failure Rate (DFR) of the scheme. In particular, the authors replaced this weird distribution by an actual Bernoulli distribution and made experimental results to support their modelization. Such a modelization is not enough from a theoretical standpoint, and we cannot use it to build reductions. In other words, in order to apply our reduction, we lack a *random self reducibility* for structured codes, such as quasi-cyclic codes as the direct approach given in Equation (39) does not seem to work directly.

In the world of Euclidean lattices, this caveat is avoided since the error distribution is taken through the Mikowski embedding. The noise would then affect each coordinate independently. The reduction from [RSW18, Section 4] benefits from the fact that the Vandermonde matrix, which maps the so-called coefficient embedding onto the Mikowski embedding, does not distort the noise too much. In the case of codes, such a Fourier-based approach takes an exagerated toll on the noise distribution.

6. Conclusion

We gave the first reduction from the worst-case search decoding problem to the average-case decision decoding problem by following the OCP framework introduced in [PRS17]. This reduction paradigm applied to lattices also permitted to obtain many new reductions for structured variants. Therefore it is tantalizing to try to apply such an approach in order to get worst-case to average-case and search-to-decision reductions for structured codes such as quasi-cyclic codes which are used for instance in BIKE and HQC, two of the three code-based proposals remaining in the fourth round of NIST post-quantum competition. However, as mentioned in Section 5, such an extension to structured codes is far from being straightforward and represents a highly interesting challenge.

References

[AAB+21a] Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo

Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE. Round 3 Submission to the NIST Post-Quantum Cryptography Call, v. 4.2, September 2021.

[AAB+21b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. HQC. Round 3 Submission to the NIST Post-Quantum Cryptography Call, June 2021. https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf.

[AAB+22a] Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE. Round 4 Submission to the NIST Post-Quantum Cryptography Call, v. 5.1, October 2022.

[AAB+22b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, Gilles Zémor, and Jurjen Bos. HQC. Round 4 Submission to the NIST Post-Quantum Cryptography Call, October 2022. https://pqc-hqc.org/.

[Ale03] Alekhnovich, Michael. More on Average Case vs Approximation Complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[BCCD23] Maxime Bombar, Geoffroy Couteau, Alain Couvreur, and Clément Ducros. Correlated Pseudorandomness from the Hardness of Quasi-Abelian Decoding. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43nd International Cryptology Conference*. Springer, August 2023.

[BCD22] Maxime Bombar, Alain Couvreur, and Thomas Debris-Alazard. On Codes and Learning With Errors over Function Fields. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd International Cryptology Conference*, volume 13508 of *LNCS*. Springer, August 2022.

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO*, pages 387–416, Cham, 2020. Springer International Publishing.

[BDK+11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 1–20, 2011.

[BF02] Alexander Barg and G. David Forney. Random codes: Minimum distances and error exponents. *IEEE Trans. Inf. Theory*, 48(9):2568–2573, 2002.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.

[BJRW20] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. Towards classical hardness of module-lwe: The linear rank case. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020*. Springer, December 2020.

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *LNCS*, pages 619–635. Springer, 2019.

[BM17] Leif Both and Alexander May. Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security. In *WCC Workshop on Coding and Cryptography*, September 2017.

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

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT*, pages 423–447. Springer, April 2023.

[CDMT22] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In *Advances in Cryptology - ASIACRYPT 2022*, LNCS. Springer, 2022.

[CS16] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016*, LNCS, pages 144–161, Fukuoka, Japan, February 2016.

[DDRT22] Thomas Debris-Alazard, Léo Ducas, Nicolas Resch, and Jean-Pierre Tillich. Smoothing codes and lattices: Systematic study and new bounds. *CoRR*, abs/2205.10552, 2022.

[DR22] Thomas Debris-Alazard and Nicolas Resch. Worst and average case hardness of decoding via smoothing bounds. preprint, December 2022. eprint.

[DRT21] Thomas Debris-Alazard, Maxime Remaud, and Jean-Pierre Tillich. Quantum reduction of finding short code vectors to the decoding problem. preprint, November 2021. arXiv:2106.02747.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51, Kobe, Japan, December 2019. Springer.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

[FS96] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *LNCS*, pages 245–255. Springer, 1996.

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.

[GZ06] Philippe Gaborit and Gilles Zémor. Asymptotic improvement of the Gilbert-Varshamov bound for linear codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2006*, pages 287–291, Seattle, USA, June 2006.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT2010*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75:565–599, 2015.

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on sidh. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT*, pages 423–447. Springer, April 2023.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[PMS21] Alice Pellet-Mary and Damien Stehlé. On the hardness of the NTRU problem. In *Asiacrypt 2021 - 27th Annual International Conference on the Theory and Applications of Cryptology and Information Security*, Advances in Cryptology – ASIACRYPT 2021. Lecture Notes in Computer Science, vol 13090., Singapore, Singapore, December 2021.

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

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 461–473, 2017.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[Rob23] Damien Robert. Breaking sidh in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT*, pages 423–447. Springer, April 2023.

[RSW18] Miruna Rosca, Damien Stehlé, and Alexandre Wallet. On the ring-LWE and polynomial-LWE problems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 146–173. Springer, 2018.

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

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47, 2011.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *LNCS*, pages 106–113. Springer, 1988.

[YZ21] Yu Yu and Jiang Zhang. Smoothing out binary linear codes and worst-case sub-exponential hardness for LPN. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *LNCS*, pages 473–501. Springer, 2021.

Appendix A. Proof of Proposition 4

Lemma 4. *Let E, F be finite sets. Let $\mathcal{H} = (h_i)_{i\in I}$ be a finite family of applications from E to F and $T \subseteq E$. Let t be drawn uniformly at random in T and $r \in E$ be a random variable distributed according to some distribution $\mathcal{D}$. Let,*

$$p \stackrel{def}{=} \mathbb{P}_{t,r}\left(\langle r, t\rangle = 1\right) \tag{30}$$

where $\langle\cdot,\cdot\rangle$ is a map from $E \times E \to \{0,1\}$. Let η be the "collision bias" defined by

$$\mathbb{P}_{h,t,r_0,r_1}\Big(h(r_0) = h(r_1), \langle t, r_0\rangle = \langle t, r_1\rangle\Big) \leqslant \frac{1}{\sharp F}(p^2 + (1-p)^2 + \eta) \tag{31}$$

where h, t are uniformly drawn in $\mathcal{H}$ and T respectively and r_0, r_1 be independent and distributed according to $\mathcal{D}$.

Let Y be the random variable (u, e) where u is uniform over F and $e \in \{0,1\}$ is a Bernoulli random variable of parameter p and u, e are independent. Let $Y(h,t)$ be the random variable $(h(r), \langle r, t\rangle)$ when r is distributed according to $\mathcal{D}$. We have,

$$\mathbb{E}_{h,t}\left(\Delta(Y(h,t), Y)\right) \leqslant \sqrt{\eta}.$$

Proof. By definition of the statistical distance we have

$$\begin{aligned}
\mathbb{E}_{h,t}\left(\Delta(Y(h,t), Y)\right) &= \sum_{\substack{h\in\mathcal{H}\\ t\in T}} \frac{1}{\sharp\mathcal{H}\ \sharp T}\ \Delta\left(\left(h(r), \langle r, t\rangle\right), (u, e)\right) \\
&= \frac{1}{2}\sum_{\substack{h\in\mathcal{H}\\ t\in T}} \frac{1}{\sharp\mathcal{H}\ \sharp T} \sum_{\substack{f\in F\\ b\in\{0,1\}}} \left|\mathbb{P}_r(h(r) = f, \langle r, t\rangle = b) - \frac{\mathbb{P}(e=b)}{\sharp F}\right| \\
&= \frac{1}{2}\sum_{\substack{h,t\\ f,b}} \left|\mathbb{P}_{h_0,t_0,r}\left(h_0 = h, t_0 = t, h_0(r) = f, \langle r, t_0\rangle = b\right) - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\right| \\
&= \frac{1}{2}\sum_{\substack{h,t\\ f,b}} \left|q_{h,t,f,b} - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\right| \qquad (41)
\end{aligned}$$

where $q_{h,t,f,b} \stackrel{\text{def}}{=} \mathbb{P}_{h_0,t_0,r}\left(h_0 = h, t_0 = t, h_0(r) = f, \langle r, t_0\rangle = b\right)$ with (h_0, t_0) being uniformly chosen at random in $\mathcal{H} \times T$ and r be distributed according to $\mathcal{D}$. Using the Cauchy-Schwarz inequality, we obtain

$$\sum_{\substack{h,t\\ f,b}} \left|q_{h,t,f,b} - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\right| \leqslant \sqrt{\sum_{\substack{h,t\\ f,b}} \left(q_{h,t,f,b} - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\right)^2}\sqrt{2\sharp\mathcal{H}\ \sharp T\ \sharp F}. \tag{42}$$

Let us observe now that

$$\begin{aligned}
\sum_{\substack{h,t\\ f,b}} \left(q_{h,t,f,b} - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\right)^2 &= \sum_{\substack{h,t\\ f,b}} \left(q_{h,t,f,b}^2 - 2\mathbb{P}(e=b)\frac{q_{h,t,f,b}}{\sharp\mathcal{H}\ \sharp T\ \sharp F} + \frac{\mathbb{P}(e=b)^2}{\sharp\mathcal{H}^2\ \sharp T^2\ \sharp F^2}\right) \\
&= \sum_{\substack{h,t\\ f,b}} q_{h,t,f,b}^2 - \frac{1}{\sharp\mathcal{H}\ \sharp T\ \sharp F}\ \sum_b \mathbb{P}(e=b)\left(2\sum_{\substack{h,t\\ f}} q_{h,t,f,b} - \mathbb{P}(e=b)\right)
\end{aligned} \tag{43}$$

Let us observe now that

$$\begin{aligned}\sum_{\substack{h,t\\f}} q_{h,t,f,b} &= \sum_{\substack{h,t\\f}} \mathbb{P}_{h_0,t_0,r}\left(h_0 = h, t_0 = t, h_0(r) = f, \langle r, t_0\rangle = b\right)\\ &= \mathbb{P}_{t_0,r}\left(\langle r, t_0\rangle = b\right)\\ &= \mathbb{P}(e = b)\end{aligned}$$

where in the last line we used Equation (30) and the assumption on e. Plugging this in Equation (43) we obtain

$$\begin{aligned}\sum_{\substack{h,t\\f,b}} \left(q_{h,t,f,b} - \frac{\mathbb{P}(e=b)}{\sharp\mathcal{H}\,\sharp T\,\sharp F}\right)^2 &= \sum_{\substack{h,t\\f,b}} q_{h,t,f,b}^2 - \frac{\mathbb{P}(e=0)^2 + \mathbb{P}(e=1)^2}{\sharp\mathcal{H}\,\sharp T\,\sharp F}\\ &= \sum_{\substack{h,t\\f,b}} q_{h,t,f,b}^2 - \frac{p^2 + (1-p)^2}{\sharp\mathcal{H}\,\sharp T\,\sharp F} \qquad (44)\end{aligned}$$

Consider now for $i \in \{0,1\}$ independent random variables h_i, t_i and r_i that are drawn uniformly at random in $\mathcal{H}$, T and according to $\mathcal{D}$ respectively. We continue this computation by noticing now that

$$\begin{aligned}\sum_{h,t,f,b} q_{h,t,f,b}^2 &= \sum_{h,f} \mathbb{P}_{h_0,t_0,r_0}\left(h_0 = h, t_0 = t, h_0(r) = f, \langle r_0, t_0\rangle = b\right)\\ &\qquad \mathbb{P}_{h_1,t_1,r_1}\left(h_1 = h, t_1 = t, , h_1(r_1) = f, \langle r_1, t_1\rangle = b\right)\\ &= \mathbb{P}_{h_0,h_1,t_0,t_1,r_0,r_1}\left(h_0 = h_1, t_0 = t_1, h_0(r_0) = h_1(r_1), \langle t_0, r_0\rangle = \langle t_1, r_1\rangle\right)\\ &= \frac{\mathbb{P}_{h_0,t_0,r_0,r_1}\left(h_0(r_0) = h_0(r_1), \langle t_0, r_0\rangle = \langle t_0, r_1\rangle\right)}{\sharp\mathcal{H}\,\sharp T}\\ &\leqslant \frac{p^2 + (1-p)^2 + \eta}{\sharp\mathcal{H}\,\sharp T\,\sharp F}. \qquad (45)\end{aligned}$$

where in the last line we used the definition of η given in Equation (31). By substituting for $\sum_{h,f} q_{h,f}^2$ the expression obtained in (45) into (44) and then back into (42) we finally obtain

$$\begin{aligned}\sum_{\substack{h,t\\f,b}} \left|q_{h,t,f,b} - \frac{\mathbb{P}(e=0)^2 + \mathbb{P}(e=1)^2}{\sharp\mathcal{H}\,\sharp T\,\sharp F}\right| &\leqslant \sqrt{\frac{p^2 + (1-p)^2 + \eta}{\sharp\mathcal{H}\,\sharp T\,\sharp F} - \frac{p^2 + (1-p)^2}{\sharp\mathcal{H}\,\sharp T\,\sharp F}}\ \sqrt{2\sharp\mathcal{H}\,\sharp T\,\sharp F}\\ &= \sqrt{\frac{\eta}{\sharp\mathcal{H}\,\sharp T\,\sharp F}}\ \sqrt{2\sharp\mathcal{H}\,\sharp T\,\sharp F}\\ &= \sqrt{2\eta}.\end{aligned}$$

which concludes the proof. □