Proving modern code-based dual attacks

Charles Meyer-Hilfiger

Univ Rennes, Inria, CNRS, IRISA
charles.meyer-hilfiger@inria.fr

Abstract. In code-based cryptography, dual attacks to solve the decoding problem have recently been improved. They are now competitive and beat information set decoders for a significant regime. These recent dual attacks, starting from Carrier et al. (Asiacrypt 2022), work by reducing decoding to an LPN problem where the secret and the noise involve parts of the error vector coming from the decoding problem. However, their analysis relies on some heuristics. While in the original Asiacrypt 2022 work, an LPN modeling was used to carry out the analysis, Meyer-Hilfiger and Tillich (TCC 2023) showed that this assumption could not be used. As a result, this TCC paper analyzed this attack with a new technique based on Fourier theory and on modeling the weight enumerator of a random linear code as a Poisson variable. The analysis the newest and most efficient dual attack, doubleRLPN, introduced by Carrier et al. (Eurocrypt 2024) also relies on this technique and on this model.
Our main contribution is to devise a variant of doubleRLPN that we can fully prove without using any model. We show that our variant has the same performance, up to polynomial factors, as doubleRLPN. The final algorithm and its analysis are also simpler. Our technique involves flipping the coordinates of the noisy codeword and observing the fine changes in the amount of noise of the related LPN problem to reconstruct the entire error. The analysis is based on the second-order behavior of the bias of the noise which was already used in the original analysis.
Secondly, the performance of our algorithm, as was the case for doubleRLPN, heavily depends on having access to a good code along with an efficient decoder. We instantiate this code by choosing a Cartesian product of a constant (instead of sublinear in the original proposal) number of random linear codes. We use a decoder based on blockwise error enumeration which was already used by Guo et al. (Asiacrypt 2014). We show that our approach is optimal up to polynomial (instead of superpolynomial) factors.

Note: preliminary version not ready for diffusion.

1 Introduction

The security of code-based schemes relies on the hardness of the decoding problem. We focus here on the binary variant of this problem.

Definition 1.1 (Binary Decoding problem). *Let $\mathcal{C}$ be a binary linear code of dimension k and length n,* i.e. *a linear subspace of dimension k of $\mathbb{F}_2^n$. Given $\mathcal{C}$ and a noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e}$ has Hamming weight $|\mathbf{e}| = t$, the goal is to find an error $\mathbf{e}'$ of weight t such that $\mathbf{y} - \mathbf{e} \in \mathcal{C}$.*

For the right choice of parameters n, k, t the algorithms solving this problem are exponential in t. In particular, when the rate of the code $R \overset{\text{def}}{=} k/n$ is constant and the relative decoding weight $\tau = t/n$ is a well-chosen constant, the runtime of all algorithms is of the order $2^{\alpha(R,\tau)n}$ where the constant $\alpha(R, \tau)$ depends on the algorithm.

The two main families of algorithms solving this problem are, on the one hand, Information set Decoders (ISD) and, on the other hand, Dual attacks. The ISDs are essentially improvements of Prange's algorithm from 1962 [Pra62]. They are the most widely studied decoders and have benefited from many improvements over the years : [Ste88, Dum89, MMT11, BJMM12, MO15, BM18] to cite a few. Dual attacks, on the contrary, can be seen as improvements of Al-Jabri's statistical decoding algorithm from 2001 [Jab01]. However, this decoder was forgotten for a long time as it was shown to be uncompetitive to attack McEliece cryptosystem [Ove06] and asymptotically uncompetitive [DT17] against Prange decoder, the simplest of the ISDs. Recently, however, new dual attacks have been developed [CDMT22, MT23, CDMT24] which are now competitive and even outperform the ISD's for some significant regimes. More precisely, the best dual attack [CDMT24] asymptotically outperforms the best ISD [BM18] in terms of time complexity when decoding codes of constant rate $R \overset{\text{def}}{=} k/n$ that are smaller than 0.42 and when the relative error weight of the decoding problem is the relative Gilbert-Varshamov distance, namely when $t/n = h_2(1 - R)$ where $h_2(x) = -x \log_2(x) - (1 - x) \log_2(1 - x)$ is the binary entropy function. This is the distance where the problem is the hardest and represents the distance where we expect a unique non-planted solution to the decoding problem.

1.1 Dual attacks.

The main ingredient for dual attacks is the dual code, defined as $\mathcal{C}^\perp \overset{\text{def}}{=} \{\mathbf{h} \in \mathbb{F}_2^n : \langle \mathbf{h}, \mathbf{c} \rangle = 0 \forall \mathbf{c} \in \mathcal{C} \}$ where $\langle \mathbf{h}, \mathbf{c} \rangle = \sum_{i=0}^{n} h_i c_i \in \mathbb{F}_2$ and where $h_i \in \mathbb{F}_2^n$ is the i'th coordinate of $\mathbf{h}$. To decode, dual attacks leverage the fact that, for a dual vector $\mathbf{h} \in \mathcal{C}^\perp$, the inner product between $\mathbf{h}$ and the noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$,

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle$$

is more biased toward 0 as the Hamming weight of $\mathbf{e}$ and $\mathbf{h}$, namely $|\mathbf{e}|$ and $|\mathbf{h}|$, are low.

1.2 Modern dual attacks.

The most recent dual attacks [CDMT22, MT23, CDMT24] essentially reduce decoding to an LPN problem which is then solved with standard solvers.

LPN problem. In essence, an LPN problem is a problem where given access to an oracle which upon each call returns $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{F}_2^s \times \mathbb{F}_2$ where $\mathbf{a}$ is uniformly random in $\mathbb{F}_2^s$ and the noise e is taken as a Bernoulli of a fixed parameter $\frac{1-\varepsilon}{2}$ and $\mathbf{s}$ is a fixed secret and the goal is to recover the secret $\mathbf{s}$ with as many calls to the oracle as wanted.

Reducing decoding to LPN. The reduction presented in [CDMT22] works as follows : by splitting the support $[\![1,\ n]\!]$ in two complementary parts $\mathscr{P}$ and $\mathscr{N}$ and by computing a dual vector of low weight on the part $\mathscr{N}$, we get the following LPN sample

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{a}, \mathbf{s} \rangle + e \quad \text{ where } \quad \begin{cases} \mathbf{s} & = \mathbf{e}_{\mathscr{P}} \\ \mathbf{a} & = \mathbf{h}_{\mathscr{P}} \\ e & = \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \end{cases}$$

where $\mathbf{e}_{\mathscr{P}}$ is the secret.

Strategy to recover the secret. The algorithm RLPN presented in [CDMT22] essentially computes many such dual vectors, each yielding an LPN sample, and tries to recover the secret $\mathbf{e}_{\mathscr{P}}$ with an LPN solver. Very roughly the LPN solver returns the $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ such that $\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle$ is the most biased toward 0. Denoting by $\mathscr{H}$ the set of computed dual vectors, this was done in RLPN by computing *exhaustively* for all $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ a score function encoding this bias

$$F_{\mathscr{H},\mathbf{y}}(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{h} \in \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle}.$$

The algorithm then considers the $\mathbf{x}$'s such that the score is big enough. One can show that when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ this score is expected to be big while when $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ this score is roughly expected to be 0.

Estimating the bias of the noise. A key quantity underlying the hardness of recovering the secret $\mathbf{e}_{\mathscr{P}}$ is the bias of the noise of the LPN sample where the bias of a Bernoulli variable X is defined as

$$\text{bias}(X) = \mathbb{P}(X = 0) - \mathbb{P}(X = 1).$$

Say, for the sake of the discussion, that $\mathbf{h}$ is taken uniformly at random in $\{\ \mathbf{h} \in \mathcal{C}^{\perp}\ :\ |\mathbf{h}_{\mathscr{N}}| = w\}$. To estimate the bias of the noise, $\text{bias}(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle)$, it was previously assumed in [Jab01, Ove06, DT17] and rigorously showed by [CDMT22] that it could reasonably be approximated by the bias when forgetting the code structure, namely by the bias of $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle$ where $\mathbf{h}'_{\mathscr{N}}$ is taken uniformly at random in the Hamming sphere of weight w (and not in its intersection with the dual code). This is very convenient as the later bias can be expressed as a closed-form function depending only on the weight and length of the vectors. For that purpose let us define

$$\delta_w^{(|\mathscr{N}|)}\left(|\mathbf{e}_{\mathscr{N}}|\right) \overset{\text{def}}{=} \text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \rangle\right). \tag{1}$$

which is a function of $|\mathscr{N}|$, the length of the vectors, and w, the weight of $\mathbf{h}'_{\mathscr{N}}$, and $|\mathbf{e}_{\mathscr{N}}|$, the weight of $\mathbf{e}_{\mathscr{N}}$ (suppose without loss of generality that this last weight is known and assumed for example to be of typical weight). This function has a simple closed form expression involving Krawtchouk polynomials. When the context is clear we forget the dependencies and denote it more simply by

$$\delta \overset{\text{def}}{=} \delta_w^{(|\mathscr{N}|)}\left(|\mathbf{e}_{\mathscr{N}}|\right)$$

.

High level rationale of the analysis and the ideal LPN model. The key question behind the analysis of dual attacks is the number of LPN samples required in order the make the right decision above, *i.e.* to be able to recover the secret of our LPN sample with good probability. Generally, given N LPN samples coming from a *standard* LPN oracle, *i.e.* where each sample $(\langle \mathbf{a}, \mathbf{s} \rangle + e)$ are drawn *independently* and where $\mathbf{a}$ is drawn uniformly in $\mathbb{F}_2^{|\mathscr{P}|}$ and e is drawn as a Bernoulli variable of bias $\varepsilon = \text{bias}\,(e)$, where the bias is defined as

$$\text{bias}\,(e) \overset{\text{def}}{=} \mathbb{P}\,(e = 0) - \mathbb{P}\,(e = 1)$$

then, Shannon's first theorem states essentially that it is sufficient that $N > 1/\varepsilon^2$ to be able to recover the secret $\mathbf{s}$ with good probability. This secret can be recovered by maximizing the associated score function. Of course the samples we get are not distributed as standard LPN samples but to make the analysis tractable [CDMT22] made an *LPN modelling* that the distributions of the LPN samples obtained in the algorithm are that of standard LPN samples.

The LPN modelling along with the estimation of the bias yielded the simple condition that if $N > 1/\delta^2$ then the RLPN algorithm should be able to recover $\mathbf{e}_{\mathscr{P}}$ by maximizing the score function, *i.e.* by outputting $\mathbf{x}$ such that $\mathbf{x} = \arg\max_{\mathbf{x}} F_{\mathscr{H},\mathbf{y}}\,(\mathbf{x})$. Some experimental discrepancy were noticed but they were conjectured to be not problematic asymptotically.

Technical difficulties : the rise of the Poisson model. However, as it was shown shortly after by [MT23] this LPN modelling really could not be used. This comes from the fact that the $\mathbf{h}_{\mathscr{N}}$ intervening in the noise shared a lot of intersection of their support and that $\mathbf{h}_{\mathscr{P}}$ and $\mathbf{h}_{\mathscr{N}}$ are linearly related. As a result, there exists some $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ such that their associated score function is bigger than that associated to the secret $\mathbf{e}_{\mathscr{P}}$, even if the condition that $N > 1/\delta^2$ is met. This critically means that $\mathbf{e}_{\mathscr{P}}$ cannot be recovered by maximizing the score function as it was believed before.

To contravene this [MT23] proposed a corrected variant of RLPN. The idea was, instead of returning the vector maximizing the score function, to rather consider a rather small set of potential candidates for $\mathbf{e}_{\mathscr{P}}$. A vector $\mathbf{x}$ would be considered as candidate for $\mathbf{e}_{\mathscr{P}}$ if its associated score, $F_{\mathscr{H},\mathbf{y}}(\mathbf{x})$ was big enough, superior to a well-chosen threshold. The algorithm would then test each of those candidates $\mathbf{x}$ for $\mathbf{e}_{\mathscr{P}}$ by solving a smaller decoding problem. This decoding problem would return the whole error $\mathbf{e}$ when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ and fail else. In particular, and importantly, testing a candidate is exponentially costly, thus a key part of the analysis was now to precisely estimate their number. This all boils down to understanding the tail distribution of the score function : given some $\mathbf{x}$ what is the probability that $F_{\mathscr{H},\mathbf{y}}(\mathbf{x})$ is superior to the threshold that was chosen? Interestingly, the second-order behavior of this score function is known from [CDMT22]. In essence, it is shown there that if $N \overset{\text{def}}{=} |\mathscr{H}|$, the number of distinct dual vector drawn, is such that $N > n/\delta^2$ then with probability $1 - o(1)$,

$$F_{\mathscr{H},\mathbf{y}}(\mathbf{e}_{\mathscr{P}}) = N\delta + o(N\delta). \tag{2}$$

We call this a second-order bound because it is derived by computing the expected value and variance of the score function and applying the Bienaymé-Tchebychev inequality to conclude. In fact, a similar bound could be derived for any fixed $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ to show that under the same condition and probability we have

$$F_{\mathscr{H},\mathbf{y}}(\mathbf{x}) = 0 + o(N\delta).$$

Basically, both these bounds allows to show that, as long as $N > n/\delta^2$, $\mathbf{e}_{\mathscr{P}}$ can be distinguished with probability $1 - o(1)$ from $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$. Problematically these bounds are completely insufficient for the purpose of [MT23]. Indeed, the optimal parameters of this algorithm are such that $|\mathscr{P}| = \Omega(n)$ and as a result the space in which $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ leaves is exponentially big, thus one would need exponential bounds, *i.e.* bounds that holds with probability $1 - 2^{-\Omega(n)}$ in order to say anything non-trivial about the size of the set of candidates. Sadly, [MT23] had to rely on some assumptions to obtain such exponential bound. Their technique relied on using the Poisson summation formula on the score function along with the model that when $\mathcal{C} + \mathbf{z}$ is a random coset of a random linear code of length n and dimension k then the number of codewords of weight i can be modeled as a Poisson variable of right expected value, $\binom{n}{i}/2^{n-k}$. This was verified experimentally.

Under such model, and equipped with this bound, they were able to show that the number of false candidate was polynomial and that consequently the cost of checking the candidates was completely dominated by the other costs of the algorithm (say by the part where we compute the dual vectors). All in all this showed that even though the LPN model was not valid, the overall additional cost incurred by this structure was of polynomial nature. Later, [CDMT24] improved this RLPN algorithm by using an additional reduction from sparse LPN to plain LPN and its analysis relied on a similar estimation of a number of candidates. This estimation relied on a similar Poisson model and it was showed that for the parameters of interest in the article this number was exponential. However, again, it was sufficiently low so that the cost of checking these candidates never dominated the complexity of the other steps of the algorithm.

1.3 Contributions

- Our main contribution is to devise a variant of double-RLPN that we can fully prove without using any model whatsoever up to rate $R \leqslant 0.5$. We show that our variant has the same performance, up to polynomial factors, as the original double-RLPN algorithm. In practice our result holds for higher rates but then depends on the parameters of the algorithm but in any cases, $R \leqslant 0.5$ already encompasses the rate regime where the best dual attacks is currently known to beat the best ISD's.
- Secondly, as we will recall later, the performance of our algorithm, as it was the case for the doubleRLPN, heavily depends on having access to a good code along with an efficient decoder. We instantiate this code by choosing cartesian product of a constant (instead of sublinear in the original proposal of [CDMT24]) number of random linear codes. We use a syndrome decoder based on blockwise error enumeration that was already used in [GJL14]. We show overall that our algorithm, when using this code, looses only a polynomial factor (instead of superpolynomial) compared to the ideal case where we would suppose that we have access to a random linear code that we could decode efficiently.

1.4 Our technique

We give the idea of our provable variant. Our goal here is to make a variant whose proof relies only on the tractable second-order bounds on the score function given previously. We focus here on giving the idea of a fully provable variant of RLPN only, the fully provable variant of double-RLPN will follow a similar rationale.

On a very high level our method is as follows : instead of computing one LPN problem and an associated score function, we compute $|\mathscr{N}| + 1$ LPN problems and associated score function. This will allow us to make for each $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ a guess $g\left(\mathbf{x}\right)$ for the value of $\mathbf{e}_{\mathscr{N}}$, in *polynomial* time. The two crucial point will be that i) a guess can be tested in *polynomial* and ii) we can show that on the secret $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ the guess is valid, namely $g\left(\mathbf{e}_{\mathscr{P}}\right) = \mathbf{e}_{\mathscr{N}}$ with good probability.

Base observation. Our base observation is that we can change the amount of noise of the LPN samples by flipping the coordinates of the received noisy codeword $\mathbf{y}$ that intervene in the noise of the LPN samples. More precisely, by computing

$$\mathbf{y}^{(i)} \quad \text{such that} \quad \left(\mathbf{y}^{(i)}\right)_{\mathscr{P}} = \mathbf{y}_{\mathscr{P}} \quad \text{and} \quad \left(\mathbf{y}^{(i)}\right)_{\mathscr{N}} = \mathbf{y}_{\mathscr{N}} + \xi_i$$

one can readily see that for a dual vector $\mathbf{h} \in \mathcal{C}$ of low weight on $\mathscr{N}$ we have the "flipped" LPN sample

$$\left\langle \mathbf{y}^{(i)}, \mathbf{h} \right\rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{N}} + \xi_i, \mathbf{h}_{\mathscr{N}} \rangle$$

which is more or less noisy depending on the weight of $\mathbf{e}_{\mathscr{N}} + \xi_i$. Said differently, if $\left(\mathbf{e}_{\mathscr{N}}\right)_i = 1$ then the weight of $\mathbf{e}_{\mathscr{N}} + \xi_i$ increases and thus the "flipped" score function evaluated on the secret $\mathbf{e}_{\mathscr{P}}$, namely $F_{\mathscr{H}, \mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\right) = \sum_{\mathbf{h} \in \mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}} + \xi_i, \mathbf{h}_{\mathscr{N}} \rangle}$, is expected to increase.

Observation 1.1.

If $\quad \left(\mathbf{e}_{\mathscr{N}}\right)_i = 1 \quad$ *we expect that* $\quad F_{\mathscr{H}, \mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\right) > F_{\mathscr{H}, \mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\right).$

The algorithm. On a high level our idea is as follows : instead of computing one score function $F_{\mathscr{H}, \mathbf{y}}$, we will also compute the flipped scores $F_{\mathscr{H}, \mathbf{y}^{(i)}}$ for $i \in [\![1, |\mathscr{N}|]\!]$. Computing these additional scores allows us to make for each $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ a guess $g(\mathbf{x})$ for the value of $\mathbf{e}_{\mathscr{N}}$, in *polynomial* time. This guess is given as follows:

$$g(\mathbf{x})_i \leftarrow 1 \quad \text{If} \qquad F_{\mathscr{H}, \mathbf{y}^{(i)}}(\mathbf{x}) > F_{\mathscr{H}, \mathbf{y}}(\mathbf{x}).$$

Following Observation 1.1 it is readily seen that when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ we expect that the guess is right, namely that $g(\mathbf{x}) = \mathbf{e}_{\mathscr{N}}$. Now, the key remark we use is that in any cases, we can test a guess in *polynomial* time by checking that $\mathbf{x}$ concatenated with the associated guess $g(\mathbf{x})$ is a solution to the original decoding problem (by making sure it is of weight t and that, when we remove $\mathbf{y}$ the result is in $\mathcal{C}$). As such our algorithm simply go through all $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$, make a guess and check this guess.

Proving our algorithm. Proving our algorithm only relies on proving that when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ then the guess is good with high probability. It is clear here that the second order concentration bounds given in Eq. (2) are sufficient to prove this. Indeed, for each position i we compare only two distributions, the one related to $\mathbf{e}_{\mathscr{N}_i}$ being 0 and the one for $\mathbf{e}_{\mathscr{N}_i}$ being 1. Of course, we must apply these bounds for each $i \in [\![1, n]\!]$, but this only incurs a polynomial loss compared to stronger bounds. We make no claim whatsoever regarding the other cases corresponding to $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ as they will be naturally discarded by our checking phase.

Our variant has the same performance as the original algorithm. Last, we argue that in fact our algorithm have the same performance, up to polynomial factors, than the original algorithm. One could argue that we have somehow lost in performance since the distribution we are trying to distinguish now are closer than the original distributions in double-RLPN. This is essentially not the case. In RLPN the distributions being distinguished had either expected value $\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\right)\right) = N\delta$ or expected value $\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{x}\right)\right) = 0$ when $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$. In our provable variant the distribution that we compare distributions that have expected values respectively

$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\right)\right) = N\delta \quad \text{with} \quad \delta \overset{\text{def}}{=} \delta_w^{(|\mathscr{N}|)}\left(u\right)$$
$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\right)\right) = N\delta' \quad \text{with} \quad \delta' \overset{\text{def}}{=} \delta_w^{(|\mathscr{N}|)}\left(u + (-1)^{(\mathbf{e}_{\mathscr{N}})_i}\right)$$

and where we recall that $\delta^{()}\left(\right)$ was defined in Eq. (1). The key point is that we can show that in our regime of interest (*i.e.* in the non-oscillatory regime of Krawtchouk polynomials) then $\delta' - \delta$ is polynomially relatable to δ, namely we can show that

$$\delta' - \delta \geqslant \frac{1}{\text{poly}\left(n\right)}\delta.$$

This shows that our algorithm solves the decoding problem under a mild polynomial strengthening of the original condition, namely it requires that $N > \text{poly}\left(n\right)/\delta^2$.

2 Acknowledgements

The author would like to warmly thank Jean-Pierre Tillich for his proofreading and comments on the counterpart of this paper that I wrote as a part of my PhD manuscript. The author is funded by pepr PQ-TLS. One part of this work was done in the Cosmiq team, Inria Paris and the other part in the Capsule team, Irisa, Rennes.

3 Notation and Preliminaries

3.1 Notation

Set, vector and matrix notation. The set $[a, b]$ is closed set of reals between a and b. $\llbracket a, b \rrbracket$ indicates the closed integer interval between a and b. $\mathbb{F}_2$ is the binary field. $|E|$ is the cardinality of a finite set E. Vectors are indicated by lowercase bold letters $\mathbf{x}$ and matrices by uppercase bold letters $\mathbf{A}$. For a vector $\mathbf{x} = (x_i)_{1 \leqslant i \leqslant n}$ and $\mathscr{I} \subset \llbracket 1, n \rrbracket$, $\mathbf{x}_{\mathscr{I}}$ is given by $\mathbf{x}_{\mathscr{I}} = (x_i)_{i \in \mathscr{I}}$ and $\mathbf{x}^{\mathsf{T}}$ is the transpose of $\mathbf{x}$ and $|\mathbf{x}|$ stands for the Hamming weight of $\mathbf{x}$. Given two vectors $\mathbf{x} \in \mathbb{F}_2^n$ and $\mathbf{y} \in \mathbb{F}_2^n$, their canonical inner product in $\mathbb{F}_2$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$. We denote by $(\mathbf{x} \mid\mid \mathbf{y}) \in \mathbb{F}_2^{2n}$ the concatenation of $\mathbf{x}$ and $\mathbf{y}$. For a matrix $\mathbf{A} \in \mathbb{F}_2^{k \times n}$ and $\mathscr{I} \subset \llbracket 1, n \rrbracket$, $\mathbf{A}_{\mathscr{I}}$ is the matrix $\mathbf{A}$ where we kept only the columns whose indices are in $\mathscr{I}$. $\text{rank}\left(\mathbf{A}\right)$ is the rank of $\mathbf{A}$. $\mathbf{I}_n$ is the identity matrix with n rows and columns. $\mathbf{0}_{k \times n} \in \mathbb{F}_2^{k \times n}$ is the all zero matrix. $\mathbf{0}_n \in \mathbb{F}_2^n$ is the null vector. $\mathcal{S}_w^n \overset{\text{def}}{=} \{\mathbf{x} \in \mathbb{F}_2^n \ : |\mathbf{x}| = w\}$ is the Hamming sphere of weight w of $\mathbb{F}_2^n$. $\mathbf{1}_A$ is the indicator function of the set A, namely $\mathbf{1}_A\left(x\right) = 1$ if $x \in A$ otherwise 0.

Probability. We denote respectively by $\mathbb{E}(X)$ and $\mathbf{Var}(X)$ the expected value and variance of a random variable X. We will use Bienaymé–Chebyshev inequality given as follows.

Proposition 3.1 (Bienaymé–Chebyshev inequality). *For any random variable X and any $\alpha > 0$ we have*

$$\mathbb{P}\left(|X - \mathbb{E}(X)| > \alpha\right) \leqslant \frac{\mathbf{Var}(X)}{\alpha^2}.$$

We will also use the union bound.

Proposition 3.2 (Union bound). *Given N event $(E_i)_{i \in [\![1,\, N]\!]}$ we have that $\mathbb{P}\left(\bigcup_{i=1}^{N} E_i\right) \leqslant \sum_{i=1}^{n} \mathbb{P}(E_i)$.*

When $\mathcal{D}$ is a probability distribution we write that $\boldsymbol{X} \sim \mathcal{D}$ to specify that $\boldsymbol{X}$ is distributed according to $\mathcal{D}$. If A is a set, we denote by $\mathcal{U}(A)$ the uniform distribution over A. We denote by $\mathrm{Ber}(p)$ is the Bernouilli distribution of parameter p. By definition if $X \sim \mathrm{Ber}(p)$ then X take value in $\mathbb{F}_2$ and $\mathbb{P}(X = 1) = p$. We define the bias of a binary random variable X as

$$\mathrm{bias}(X) \overset{\text{def}}{=} \mathbb{P}(X = 0) - \mathbb{P}(X = 1).$$

In particular if $X \sim \mathrm{Ber}\left(\frac{1-\varepsilon}{2}\right)$ then $\mathrm{bias}(X) = \varepsilon$.

Lemma 3.3 (Pilling up lemma). *If $X_1 \sim \mathrm{Ber}\left(\frac{1-\varepsilon_1}{2}\right)$ and $X_2 \sim \mathrm{Ber}\left(\frac{1-\varepsilon_2}{2}\right)$ then* $\mathrm{bias}(X_1 + X_2) = \varepsilon_1 \varepsilon_2$.

Fourier Transform. Let $f : \mathbb{F}_2^n \to \mathbb{R}$ be a function. We define its Fourier transform $\widehat{f} : \mathbb{F}_2^n \to \mathbb{R}$ as

$$\widehat{f}(\mathbf{x}) = \sum_{\mathbf{a} \in \mathbb{F}_2^n} f(\mathbf{a})(-1)^{\langle \mathbf{x}, \mathbf{a} \rangle} \quad \forall \mathbf{x} \in \mathbb{F}_2^n.$$

We call any algorithm computing this Fourier transform in time $\mathcal{O}(n2^n)$ a Fast Fourier transform.

Landau and asymptotic notation. For real valued functions defined over $\mathbb{R}$ or $\mathbb{N}$ we define $o()$, $\mathcal{O}()$, $\Omega()$, $\Theta()$, in the usual way. We write that $f = \omega(g)$ when f dominates g asymptotically that is when $\lim\limits_{x \to \infty} \frac{|f(x)|}{g(x)} = \infty$. We use the less common notation $\widetilde{\mathcal{O}}()$, where $f = \widetilde{\mathcal{O}}(g)$ means that $f(x) = \mathcal{O}\left(x^k g(x)\right)$ for some constant k.

3.2 Linear codes and decoding problem

Definition 3.4 (Binary linear code). *A binary linear code $\mathcal{C}$ of length n and dimension k is a linear subspace of $\mathbb{F}_2^n$ of dimension k. We say that $\mathcal{C}$ is an $[n,k]$ linear code.*

We call $R = \frac{k}{n}$ the rate of the code. We denote by $\dim(\mathcal{C})$ the dimension of $\mathcal{C}$ as a linear space. We say that $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ is a generator matrix of $\mathcal{C}$ if $\mathcal{C} = \{\mathbf{mG} \; : \; \mathbf{m} \in \mathbb{F}_2^k\}$ and that $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ is a parity-check matrix of $\mathcal{C}$ if $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \; : \; \mathbf{Hc}^\intercal = \mathbf{0}\}$.

Definition 3.5 (Dual code). *For any binary linear code $\mathcal{C}$ of length we denote the dual code by $\mathcal{C}^\perp \overset{def}{=} \{\mathbf{h} \in \mathbb{F}_2^n : \langle \mathbf{c}, \mathbf{h}\rangle = 0, \; \forall \mathbf{c} \in \mathcal{C}\}$.*

We have that if $\mathcal{C}$ is an $[n,k]$-linear code then $\mathcal{C}^\perp$ is an $[n, n-k]$-linear code and that if $\mathbf{G}$ is a generator matrix of $\mathcal{C}$ then $\mathbf{G}$ is a parity-check matrix of $\mathcal{C}^\perp$. We will denote by $\mathcal{C}_\mathcal{I}$ the punctured code obtained from $\mathcal{C}$ by keeping only the positions in $\mathcal{I}$, *i.e.* :

$$\mathcal{C}_\mathcal{I} = \{\mathbf{c}_\mathcal{I} : \mathbf{c} \in \mathcal{C}\}.$$

Two of the most standard distributions on linear codes are given as follows.

Definition 3.6 (Distribution on linear codes). *We denote by $\mathcal{U}_\mathbf{G}(n, \; k)$ the distribution on linear codes obtained by taking a generator matrix of the code uniformly at random in $\mathbb{F}_2^{k\times n}$. We denote by $\mathcal{U}_\mathbf{H}(n, \; k)$ the distribution on linear codes obtained by taking a parity-check matrix of the code uniformly at random in $\mathbb{F}_2^{(n-k)\times n}$.*

In this work we will focus on designing an algorithm solving the following average binary decoding problem.

Definition 3.7 (Binary Decoding problem $\mathrm{DP}_\mathbf{G}(n,k,t)$). *Given $(\mathbf{G}, \mathbf{y})$ where $\mathbf{G}$ is a matrix taken uniformly at random in $\mathbb{F}_2^{k\times n}$ and $\mathbf{y} = \mathbf{mG} + \mathbf{e}$ where $\mathbf{m}$ is taken uniformly at random in $\mathbb{F}_2^k$ and $\mathbf{e}$ is taken uniformly at random among vectors of $\mathbb{F}_2^n$ of Hamming weight t, the goal is to output an error vector $\mathbf{e}'$ of Hamming weight t such that $\mathbf{y}$.*

The so called Gilbert-Varshamov distance is an importance quantity which represents essentially the decoding distance where we expect a unique non-planted solution to the decoding problem.

Definition 3.8 (Gilbert-Varshamov distance). *Let $n, k \in \mathbb{N}$ be such that $k \leqslant n$. We define the Gilbert-Varshamov distance $d_{\mathrm{GV}}(n, \; k)$ as the largest integer such that*

$$\sum_{i=0}^{d_{\mathrm{GV}}(n,\; k)} \binom{n}{i} \leqslant 2^{n-k}.$$

The above could be proved using some slight variant of the following standard lemma.

Lemma 3.9 (Probability of belonging to a code). *Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ and let $\mathbf{c} \in \mathbb{F}_2^n \setminus \{\ \mathbf{0}\}$ be a fixed vector. We have that*

$$\mathbb{P}(\mathbf{c} \in \mathcal{C}) = \frac{1}{2^{n-k}}.$$

If furthermore $\mathbf{d} \in \mathbb{F}_2^n \setminus \{\ \mathbf{0}\}$ is a fixed vector such that $\mathbf{d} \neq \mathbf{c}$ we have that

$$\mathbb{P}(\mathbf{c} \in \mathcal{C}, \mathbf{d} \in \mathcal{C}) \leqslant \left(\frac{1}{2^{n-k}}\right)^2.$$

3.3 Krawtchouk polynomial

We recall here some properties about Krawtchouk polynomial that will be useful in the article.

Definition 3.10. *(Krawtchouk polynomial) We define the Krawtchouk polynomial $K_w^{(n)}$ of degree w and of order n as $K_w^{(n)}(X) \overset{def}{=} \sum_{j=0}^{w} (-1)^j \binom{X}{j}\binom{n-X}{w-j}$.*

The following fact is well known, it gives an alternate expression of the Krawtchouk polynomial (see for instance [vL99, Lemma 5.3.1]) :

Lemma 3.11. *For any $\mathbf{a} \in \mathbb{F}_2^n$,*

$$K_w^{(n)}(|\mathbf{x}|) = \widehat{\mathbf{1}_{\mathcal{S}_t^n}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n : |\mathbf{y} = w} (-1)^{\langle \mathbf{x}, \mathbf{y} \rangle}. \tag{3}$$

Equipped with this lemma it is easy to see that Krawtchouk polynomials are related to the bias of the inner product of two random binary vectors.

Lemma 3.12 (Bias of an inner product). *Let $\mathbf{e} \in \mathcal{S}_t^n$ be a fixed vector and let $\mathbf{h}$ be taken uniformly at random in $\mathcal{S}_w^n$. We have that*

$$\text{bias}(\langle \mathbf{e}, \mathbf{h} \rangle) = \delta_w^{(n)}(t)$$

where

$$\delta_w^{(n)}(t) \overset{def}{=} K_w^{(n)}(t) / \binom{n}{w}$$

3.4 Asymptotic expansion

To simplify our proofs we will often use the following standard asymptotic expansion of the binomial coefficient.

Proposition 3.13. *There exists a positive poly-bounded function f such that for any $n, t \in \mathbb{N}$ such that $t \leqslant n$ we have*

$$\frac{1}{f(n)} 2^{h(t/n)n} \leqslant \binom{n}{t} \leqslant f(n) 2^{h(t/n)n}$$

where $h(x) = -x \log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function. Furthermore, the binary entropy function is differentiable in $]0,\ 1[$.

We will also use this standard result about the asymptotic expansion of Krawtchouk polynomials.

Proposition 3.14 (About the asymptotic expansion of Krawtchouk polynomials). *Let us define* $A \overset{def}{=} \{ (\omega,\ \tau) \in [0,\ 1]^2 \ :\ \tau < 1/2 - \sqrt{\omega(1-\omega)} \}$. *There exists a positive poly-bounded function* f *and a bivariate function* $\kappa(\omega,\ \tau)$ *that is differentiable on* A *and that is such that for any* w, t, n *such that* $t < \mathrm{Root}\left(K_w^{(n)}\right)$ *we have*

$$\frac{1}{f(n)} 2^{\kappa(w/n,\ t/n)n} < K_w^{(n)}(t) < f(n) 2^{\kappa(w/n,\ t/n)n}.$$

where the delimitation of the root region is defined as

$$\mathrm{Root}\left(K_w^{(n)}\right) \overset{def}{=} n/2 - \sqrt{w(n-w)} \tag{4}$$

Proof. This is a direct corollary of [KS21, Section 2.2, point 6.] together with [KS21, Section 2.1.2].

4 Essential on the reduction of double-RLPN

Our provable variant of double-RLPN reuses a major part of the original double-RLPN algorithm [CDMT24]. Recall that our goal here is to solve the following decoding problem, namely, given an $[n, k]$-linear code $\mathcal{C}$ and a noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$, the goal is to recover $\mathbf{e}$.

We recall here the 4 steps of the dual attack double-RLPN to solve this problem.

1. Choose at random $\mathscr{P}$ and $\mathscr{N}$ two complementary subsets of $[\![1,\ n]\!]$
2. Compute a list of LPN samples where the secret is related to $\mathbf{e}_{\mathscr{P}}$.
3. Compute a score function associated to the LPN samples that encodes how likely a vector is the secret of the LPN samples.
4. Somehow recover $\mathbf{e}_{\mathscr{P}}$ using the values of the score function then recover the rest of the error $\mathbf{e}_{\mathscr{N}}$.

Those steps are iterated as certain number N_{iter} of times until a bet on the weight of the error on the part $\mathscr{P}$ and $\mathscr{N}$ is verified, say $|\mathbf{e}_{\mathscr{N}}| = u$. The rationale of step 4. is that it succeed in recovering $\mathbf{e}$ when this bet is verified, else it fails.

We discuss in detail of step 2. and step 3. in Section 4.1 and Section 4.2 respectively. These steps represent the core of double-RLPN and will be used by our provable variant. We discuss quickly of step 4. as well as the original analysis of double-RLPN to recall the key quantities that will also appear in our analysis.

4.1 Computing a list of LPN samples

In this section we suppose that two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ of $[\![1,\ n]\!]$ such that

$$|\mathscr{P}| = s, \qquad \text{and} \quad |\mathscr{N}| = n - s.$$

are given and explain how double-RLPN computes some LPN samples whose secret is related to $\mathbf{e}_{\mathscr{P}}$.

The reduction of [CDMT24]. The reduction starts from the base remark from [CDMT22] that a dual vectors $\mathbf{h}$ of small weight on $\mathscr{N}$, directly yield an LPN sample

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle .$$

Then, [CDMT24] further noticed that the dimension of the LPN samples could be reduced using a technique from [GJL14], at the cost of a mild increase of noise by using the sparsity of the secret $\mathbf{e}_{\mathscr{P}}$. The reduction works by considering an $[s, k_{\text{aux}}]$-linear auxiliary code $\mathcal{C}_{\text{aux}}$ and decoding $\mathbf{h}_{\mathscr{P}}$ onto $\mathcal{C}_{\text{aux}}$, *i.e.* , by finding $\mathbf{c}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$ and an error $\mathbf{e}_{\text{aux}}$ of low weight t_{aux} such that

$$\mathbf{h}_{\mathscr{P}} = \mathbf{c}_{\text{aux}} + \mathbf{e}_{\text{aux}} \qquad\qquad |\mathbf{e}_{\text{aux}}| = t_{\text{aux}}.$$

Now considering $\mathbf{G}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}} \times s}$ a generator matrix of $\mathcal{C}_{\text{aux}}$, there exists a unique $\mathbf{m}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}}}$ such that $\mathbf{c}_{\text{aux}} = \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}}$. This allows to get the following LPN sample $(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ where

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{a}, \mathbf{s} \rangle + e \quad \text{ where } \quad \begin{cases} \mathbf{s} & = \mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal} \in \mathbb{F}_2^{k_{\text{aux}}} \\ \mathbf{a} & = \mathbf{m}_{\text{aux}} \\ e & = \langle \mathbf{e}_{\text{aux}}, \mathbf{e}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle . \end{cases} \tag{5}$$

Definition 4.1 (List of LPN samples from a list of decoded dual vectors.). *Given a set of decoded dual vectors* $\mathscr{H} \subset \{ (\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathcal{C}^{\perp} \times \mathcal{S}_{t_{\text{aux}}}^{s} : \mathbf{h}_{\mathscr{P}} + \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}} \}$ *and* $\mathbf{G}_{\text{aux}}$ *a generator matrix of an* $[s, k_{\text{aux}}]$*-linear code* $\mathcal{C}_{\text{aux}}$ *and a word* $\mathbf{y} \in \mathbb{F}_2^n$ *we define the list of LPN samples as*

$$\mathcal{L}\left(\mathscr{H},\ \mathbf{y},\ \mathbf{G}_{\text{aux}},\ \mathscr{P}\right) \overset{def}{=} \left(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle\right)_{(\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{H}}$$

where for each $(\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{H}$*, the associated* $\mathbf{m}_{\text{aux}}$ *is the unique vector of* $\mathbb{F}_2^{k_{\text{aux}}}$ *such that* $\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} + \mathbf{e}_{\text{aux}} = \mathbf{h}_{\mathscr{P}}$*.*

Note that $\mathcal{L}\left(\mathscr{H},\ \mathbf{G}_{\text{aux}},\ \mathbf{y}\right)$ can be computed with standard linear algebra in time and memory $\text{poly}\left(n\right)|\mathscr{H}|$. The rest of Section 4.1 is dedicated to recalling precisely how the set of decoded dual vectors is computed.

About the computation of low weight dual vectors We recall here the standard fact that computing a vector $\mathbf{h} \in \mathcal{C}^{\perp}$ of small weight on $\mathscr{N}$ reduces to computing a vector of weight w of $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ and lifting it uniquely onto $\mathcal{C}^{\perp}$.

Definition 4.2 (Information set and lifting). *A subset* $\mathscr{I}$ *of* $[\![1,\ n]\!]$ *is an information set of a linear code* $\mathcal{D}$ *if for any* $\mathbf{x} \in \mathcal{D}_{\mathscr{I}}$ *there exists a unique codeword* $\mathbf{d} \in \mathcal{D}$ *such that* $\mathbf{d}_{\mathscr{I}} = \mathbf{x}$*. We define*

$$\text{Lift}\left(\mathcal{D},\ \mathscr{I},\ \mathbf{x}\right) \overset{def}{=} \mathbf{d} \textit{ where } \mathbf{d} \textit{ is the unique } \mathbf{d} \in \mathcal{D} \textit{ such that } \mathbf{d}_{\mathscr{I}} = \mathbf{x}.$$

We call it the lift of $\mathbf{x}$ *into* $\mathcal{D}$*.*

This is possible only if $\mathscr{N}$ if an information set of $\mathcal{C}^{\perp}$, which is the case with high probability since the condition that $|\mathscr{N}| \overset{\text{def}}{=} n - s > n - k$ are naturally verified by the double-RLPN parameters. Checking if $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$ can be done in polynomial time by checking that $\text{rank}\left(\mathbf{G}_{\mathscr{P}}\right) = |\mathscr{P}|$ where $\mathbf{G}$ is a generator matrix of $\mathcal{C}$. Lifting $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ can be done in polynomial time with a gaussian elimination to find a generator matrix $\mathbf{G}'$ of $\mathcal{C}$ of the form $\mathbf{G}' = \begin{bmatrix} \mathbf{I}_s & \mathbf{R} \\ \mathbf{0}_{s\times s} & \mathbf{A} \end{bmatrix}$ and by defining $\mathbf{h}_{\mathscr{P}} \leftarrow \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}$.

With these reductions presented We describe next two algorithms: the first creates couples $(\mathbf{h}, \mathbf{e}_{\text{aux}})$ of decoded dual vectors and second transforms these couples into the samples $(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle)$.

The procedure computing the decoded dual vectors. The procedure computing the LPN samples starts by calling a procedure Decoded-dual-vectors$(\mathcal{C}, \mathscr{P}, \mathscr{N}, \mathcal{F})$ that we describe here. $\mathcal{F}$ is the family of $[s, k_{\text{aux}}]$-linear code that will be used for the reduction. This procedure outputs a set $\mathscr{H}$ composed of couples $(\mathbf{h}, \mathbf{e}_{\text{aux}})$ as described previously and a generator matrix $\mathbf{G}_{\text{aux}}$ of the code that was used for the reduction.

The procedure starts by calling a procedure Compute-Small-Codeword$(\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}, w)$ returning a subset of $\{\ \mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} : |\mathbf{h}_{\mathscr{N}}| = w\}$. Then each of these vectors is lifted into a vector of $\mathcal{C}^{\perp}$ to form a subset $\mathscr{W}$ of $\{\mathbf{h} \in \mathscr{H} : |\mathbf{h}_{\mathscr{N}}| = w\ \}$. Then the $[s, k_{\text{aux}}]$-linear code $\mathcal{C}_{\text{aux}}$ is drawn uniformly at random from $\mathcal{F}$ this family comes equipped with a list decoder Decode-Auxilary$(\mathcal{C}_{\text{aux}}, \mathbf{a}, t_{\text{aux}})$ that returns a subset of $\{\ \mathbf{e}_{\text{aux}} \in \mathcal{S}^{s}_{t_{\text{aux}}} : \mathbf{a} + \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}\}$. For each $\mathbf{h}$, we call Decode-Auxilary$(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}}, t_{\text{aux}})$ and happens the couple $(\mathbf{h}, \mathbf{e}_{\text{aux}})$ to the set $\mathscr{H}$ of decoded dual vectors.

Algorithm 1

```
Name: Decoded-dual-vector-double-RLPN($\mathcal{C}, \mathscr{P}, \mathscr{N}, \mathcal{F}$)
 1: Continue if $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$
 2: $\mathscr{W}_{\mathscr{N}} \leftarrow$ Compute-Small-Codeword$(\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}, w)$ ▷ Returns a subset of $\{\ \mathbf{h} \in \mathcal{C}^{\perp} : |\mathbf{h}_{\mathscr{N}}| = w\}$
 3: $\mathscr{W} \leftarrow \{\ \mathbf{h} \in \mathcal{C}^{\perp} : \mathbf{h} = \text{Lift}\,(\mathcal{D}, \mathscr{I}, \mathbf{x})\}$
 4: $\mathcal{C}_{\text{aux}}, \mathbf{G}_{\text{aux}} \xleftarrow{\$} \mathcal{F}$
 5: $\mathscr{H} \leftarrow \emptyset$
 6: for $\mathbf{h} \in \mathscr{W}$ do
 7:     $\mathcal{E} \leftarrow$ Decode-Auxilary$(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}})$ ▷ Returns a set of error of small weight $\mathbf{e}_{\text{aux}} \in \mathcal{S}^{s}_{t_{\text{aux}}}$ s.t $\mathbf{h}_{\mathscr{P}} - \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$
 8:     for $\mathbf{e}_{\text{aux}} \in \mathcal{E}$ do
 9:         $\mathscr{H}$.append$((\mathbf{h}, \mathbf{e}_{\text{aux}}))$          ▷ The set of decoded dual vectors
10: return $(\mathscr{H}, \mathbf{G}_{\text{aux}})$
```

4.2 Computing the score function

The main quantity intervening in modern dual attacks is the score function of the related LPN sample. It basically encodes how likely a vector $\mathbf{z}$ is the secret $\mathbf{s}$ of the LPN problem.

Definition 4.3 (Score function). *Let $\mathcal{L}$ be a list of LPN samples of the form $(\mathbf{a}, b) \in \mathbb{F}_2^{k_{\text{aux}}} \times \mathbb{F}_2$. For all $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ we define the score function as*

$$F_{\mathcal{L}}(\mathbf{z}) \overset{def}{=} \sum_{(\mathbf{a},b)\in\mathcal{L}} (-1)^{b-\langle \mathbf{a},\mathbf{z}\rangle}.$$

Intuitively this score is expected to be big when $\mathbf{z}$ is the secret of the LPN problem. In particular, in double-RLPN, the score function evaluated on the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ (see Eq. (5)) gives the following.

Lemma 4.4. *We have that*

$$F_{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) \overset{def}{=} \sum_{(\mathbf{h},\mathbf{e}_{\text{aux}})\in\mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{P}},\mathbf{e}_{\text{aux}}\rangle+\langle \mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}$$

where $\mathcal{L} \overset{def}{=} \mathcal{L}(\mathscr{H}, \mathbf{G}_{\text{aux}}, \mathbf{y})$ is defined in Definition 4.1.

The procedure double-RLPN Computes this score function using a Fast Fourier Transform based approach which was already used in RLPN and more generally was introduced by [LF06] to seed-up LPN solvers. We refer to the previous article for more details. The procedure is described in Score-function($\mathcal{L}$) and is detailed in Algorithm 2.

Algorithm 2

```
Name: SCORE-FUNCTION($\mathcal{L}$)
1: for $\mathbf{a} \in \mathbb{F}_2^{k_{\text{aux}}}$ do
2:     $f_{\mathcal{L}}(\mathbf{r}) \leftarrow \sum_{(\mathbf{a},)\in\mathcal{L}\,:\,\mathbf{a}=\mathbf{r}} (-1)^b$
3: $F_{\mathcal{L}} \leftarrow \mathrm{FFT}(f_{\mathcal{L}})$ ▷ FFT$(f)$ is a Fast Fourier transform algorithm that outputs the Fourier Transform $\widehat{f}$ of $f$.
4: return $F_{\mathcal{L}}$
```

Proposition 4.5. *Given a list of LPN samples $\mathcal{L}$, Algorithm 2 returns $F_{\mathcal{L}}$ in time and memory respectively*

$$\textbf{\textit{Time}} = \mathrm{poly}(n)\left(|\mathcal{L}| + 2^{k_{\text{aux}}}\right), \qquad \textbf{\textit{Memory}} = \mathrm{poly}(n)\left(|\mathcal{L}| + 2^{k_{\text{aux}}}\right)$$

4.3 Step 4. Recovering e

We do not describe in detail this step because it will not be useful in our provable variant ou double-RLPN. We give the basic outline just for the sake of comparison. At this stage of the double-RLPN algorithm, the set of decoded dual vectors $\mathscr{H}$ is computed as well as the associated score function $F_{\mathscr{H},\mathbf{y}}$. Recovering $\mathbf{e}$ was done by first considering a set of candidates for the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal} \in \mathbb{F}_2^{k_{\text{aux}}}$ of the obtained LPN problem by filtering out vectors whose score were low, namely by considering the set of candidates $\{ \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}} : F_{\mathscr{H},\mathbf{y}}(\mathbf{z}) > T\}$ where T is a well-chosen threshold. Each candidate is then tested by solving some smaller decoding problems which returns $\mathbf{e}$ if the considered candidate is such that $\mathbf{z} = \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$. Testing a candidate is exponentially costly.

4.4 Key quantity of the analysis of double-RLPN

We recall here the basic quantities intervening in the analysis of double-RLPN and that of our provable variant. This section allows the reader to interpret the quantities that will appear in the main theorem of this paper and that we state in the next section.

The analysis of double-RLPN was done when $\mathcal{C}$ is chosen by taking its generator matrix uniformly at random in $\mathbb{F}_2^{k\times n}$ and by supposing that the set $\mathscr{W}$ of computed dual vectors is the whole set $\{ \mathbf{h} \in \mathcal{C}^{\perp} : |\mathbf{h}_{\mathscr{N}}| = w\}$. So that the quantities are well-defined we condition all the probabilities by the event that $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$. To simplify the analysis the code $\mathcal{C}_{\text{aux}}$ was taken as a uniformly random $[s, k_{\text{aux}}]$-linear codes equipped with a genie-aided list decoder, namely by supposing that the $\textsc{Decode}(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}}, t_{\text{aux}})$ returns the set $\{ \mathbf{e}_{\text{aux}} \in \mathcal{S}_{t_{\text{aux}}}^{s} : \mathbf{e}_{\text{aux}} - \mathbf{h}_{\mathscr{P}} \in \mathcal{C}_{\text{aux}}\}$. All in all this means that the set of decoded dual vectors $\mathscr{H}$ is comprised of all possible couples

$$\mathscr{H} = \{ (\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathcal{C}^{\perp} \times \mathcal{S}_{t_{\text{aux}}}^{s} : |\mathbf{h}_{\mathscr{N}}| = w \text{ and } \mathbf{h}_{\mathscr{P}} + \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}\}.$$

There are two relevant quantities to the analysis : the average number of LPN samples and the bias of the noise of the samples.

Number of available LPN samples. First, one could show that the expected number of samples is

$$\begin{aligned}\mathbb{E}(|\mathcal{L}|) &= \mathbb{E}(|\mathscr{H}|)\\ &= \frac{\binom{n-s}{w}}{2^{k-s}}\frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}}\\ &= \frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}.\end{aligned}$$

The reader interested in a rigorous proof can see this statement as a direct corollary of the later stated Proposition 8.1. More roughly, this equality comes from the fact that i) by construction, the expected number of dual vectors of

$\mathcal{C}$ weight w on $\mathscr{N}$ is given by $\mathbb{E}\left(\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} \bigcap \mathcal{S}_w^{n-s}\right) = \binom{n-s}{w}/2^{k-s}$ and ii) the expected number of error returned by the auxiliary list decoder for each $\mathbf{h}_{\mathscr{P}}$ is $\mathbb{E}\left((\mathcal{C}_{\text{aux}} + \mathbf{h}_{\mathscr{P}}) \bigcap \mathcal{S}_{t_{\text{aux}}}^{s}\right) = \binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}}$.

Bias of the LPN samples. The second key quantity is the bias of the noise $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle$ of the LPN samples. This quantity obviously depends on the weight of the vectors. We recall that by construction

$$|\mathbf{h}_{\mathscr{N}}| = w, \qquad |\mathbf{e}_{\text{aux}}| = t_{\text{aux}}$$

and that a bet is made on the weight of the error on each subpart, namely that

$$|\mathbf{e}_{\mathscr{N}}| = u, \qquad |\mathbf{e}_{\mathscr{P}}| = t - u.$$

Forgetting about the fact that all these vectors come from a decoding problem we can easily estimate the bias of this noise.

Lemma 4.6. *Bias of the noise by forgetting about the code structure. Let $\mathscr{P}$ and $\mathscr{N}$ two complementary subset of $[\![1,\, n]\!]$ of size s and $n-s$ respectively. Let $\mathbf{e} \in \mathcal{S}_t^n$ such that $|\mathbf{e}_{\mathscr{P}}| = u$ and $|\mathbf{e}_{\mathscr{N}}| = t-u$. Then*

$$\text{bias}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle\right) = \delta_w^{(n-s)}(u)\, \delta_{t_{\text{aux}}}^{(s)}(t-u)$$

where $\mathbf{h}_{\mathscr{N}}$ is taken uniformly at random in $\mathcal{S}_w^{n-s}$ and $\mathbf{e}_{\text{aux}}$ is taken uniformly at random in $\mathcal{S}_{t_{\text{aux}}}^{s}$.

Proof. This is direct consequence of the Pilling-Up lemma and Lemma 3.12.

Of course, one cannot completely forget this code structure but ultimately [CDMT24, Proposition 2] showed with a second-order technique that under some conditions, the quantity $\delta_w^{(n-s)}(u)\, \delta_{t_{\text{aux}}}^{(s)}(t-u)$ was with good probability a good approximation of the bias of the noise of the LPN samples.

Rationale behind the attack. Under the flawed [MT23, CDMT24] LPN modelling one would expect to require that the number N of LPN samples is superior to n/ε^2 in order to be able to recover the secret of an LPN problem with a bias of noise ε. In our case this would roughly be that

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} > \frac{n}{\left(\delta_w^{(n-s)}(u)\, \delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}$$

to recover the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$. Of course because this modelling is flawed [CDMT24] had to use a model [CDMT24, Model 1] in order to carry out the analysis. This model was used to bound the size of the number of candidates in Step 4., see Section 4.3.

5 Results and main theorem

In this section we state the main result of this paper. We state a theorem giving the performance of our provable variant of double-RLPN, we will describe in detail this provable variant in the next Section 6. For simplicity, we state, as it was the case for double-RLPN, our theorem in the case where we suppose that we have access to a procedure that computes all the vectors of $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ of weight w. To make the quantities appearing in our theorem more intelligible, the reader can refer to the previous Section 4.4. We mark with a "*" the new constraints that did not appear in the original paper [CDMT24, Proposition 9].

Theorem 5.1. *There exists a positive poly-bounded function f such that for any $k, t, s, k_{\text{aux}}, t_{\text{aux}}, w, u \in \mathbb{N}$ implicit functions of a parameter $n \in \mathbb{N}$ and any procedure* Compute-small-codewords *that are such that*

1. *(Computing all the whole set of dual vectors $\{\mathbf{h} \in \mathcal{C}^{\perp} \,:\, |\mathbf{h}_{\mathscr{N}}| = w\}$)*
$$\mathbb{P}\left(\text{Compute-small-codewords}(\mathcal{D}) \neq \mathcal{D} \bigcap \mathcal{S}_w^{n-s}\right) \in o(1)\,,$$
where $\mathcal{D}$ is taken by choosing its parity-check matrix uniformly at random in $\mathbb{F}_2^{(k-s)\times(n-s)}$.
2. *(Main constraint that we have enough dual vectors)*
$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{f(n^{s/k_{\text{aux}}})}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right),$$
3. *(Decoding the auxiliary code below Gilbert-Varshamov)* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$,
4. **(Linear scaling of the parameters) $s/k_{\text{aux}} \in \mathcal{O}(1)$,*
5. **($\mathscr{N}$ is an information set of the code $\mathcal{C}$) $n-s-k \in \omega(1)$,*
6. **(Small technical constraints)* $\text{Root}\left(K_w^{(n-s)}\right) - u \in \Omega(n-s)$ *and* $\text{Root}\left(K_{t_{\text{aux}}}^{(s)}\right) - (t-u) \in \Omega(s)$ *and $k-s \in \omega(1)$ and $n-k \in \omega(1)$*

then there exists an algorithm solving $\text{DP}_{\mathbf{G}}(n,k,t)$ with probability $1-o(1)$ in time and memory

$$\textit{\textbf{Time}} = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{s}{t-u}\binom{n-s}{u}}\left(T_{\text{eq}} + 2^{k_{\text{aux}}}\right)\right), \qquad \textit{\textbf{Memory}} = \widetilde{\mathcal{O}}\left(\left(M_{\text{eq}} + 2^{k_{\text{aux}}}\right)\right)$$

where T_{eq} and M_{eq} are respectively time and memory complexity of one call to the procedure Compute-small-codewords$(\mathcal{D})$ *and where we recall that $\delta_w^{(n)}(t)$ is defined in Lemma 3.12 and* $\text{Root}\left(K_w^{(n)}\right)$ *is defined in Eq. (4). In particular the said algorithm is the one defined later in Definition 7.4 by taking $b \overset{def}{=} \lceil s/k_{\text{aux}} \rceil$ along with a goood choice for N_{iter}.*

Next we make a comparison with [CDMT24, Proposition 9] giving the performance of double-RLPN and the theorem we have just stated.

- The proposition giving the performance of the algorithm in [CDMT24, Proposition 9] requires the use of the Poisson model [CDMT24, Model 1] whereas our theorem do not require any model.
- [CDMT24, Proposition 9] the complexity is in $2^{k_{\text{aux}}} + T_{\text{eq}} 2^{o(n)} + T_{\text{check}}$ where T_{check} is an additional term for checking false candidates and the $2^{o(n)}$ is the cost of one call to the auxiliary decoder. But recall that in practice [CDMT24] verified that T_{check} never dominates the complexity.
- Our provable variant will require that $\mathscr{N}$ is an information set of $\mathcal{C}$. So that this happens with good probability we had to add the constraint on the parameters that $n - s - k \in \omega(1)$ where we recall that $n - s = |\mathscr{N}|$. However, because $s < k$ we have that this is unconditionally true for codes of rate $R \stackrel{\text{def}}{=} k/n$ smaller than 0.5. We believe we could remove this constraint but at the cost of a slightly more complex algorithm.
- The constraint that $s/k_{\text{aux}} \in \mathcal{O}(1)$ comes from the choice we made for the auxiliary code $\mathcal{C}_{\text{aux}}$: it is the condition required for i) the code to be optimal and ii) that we can decode it in polynomial time. In practice for the parameters regime we are interested in this article, namely $k = \Theta(n)$ and $t = \Theta(n)$, the optimal values of the parameters (*i.e.* matching the main constraint and minimizing the complexity) always verify this condition. In particular this is the case for all the given parameters of [CDMT24]. We believe that this condition could be removed by using Polar codes as suggested in [Car20, CDMT24], but then the proof would not be as simple.
- In [CDMT24, Proposition 9] the main constraint needed for the proof under the model was that

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{n^8}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right). \tag{6}$$

  Here we have some other poly-bounded function $f(n^b)$ instead of n^8. The point is that, regardless of the polynomial, it is easy to create from a set of parameter verifying constraint given by Eq. (6) a new set of parameters verifying this new constraint by increasing w only slightly by a $\mathcal{O}(\log_2(n))$ term. This could rigorously be shown by computing the derivative in w of the asymptotic expansion of $\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2$.
- We added some small additional constraints (Point 6. in the theorem) to make our statement rigorous. In practice those are always verified for non-degenerate parameters. And, in particular, they are verified for all the asymptotic parameter datasets given in [CDMT24].

In particular, any complexity claims about double-RLPN made using [CDMT24, Proposition 9] under the Poisson model [CDMT24, Model 1] are provably achieved (by an algorithm that we describe next), up to polynomial factors, when the rate of the code $R \stackrel{\text{def}}{=} k/n$ is smaller than 0.5.

6 Fully provable double-RLPN

Let us now describe our provable variant of double-RLPN. Our algorithm retakes the main loop of double-RLPN by choosing at random two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ and the main ingredient we reuse are the procedures describes in **??** The procedure DOUBLE-RLPN-SCOREFUNCTION which given a vector $\mathbf{y}$ computes the double-RLPN score function associated to the LPN problem generated by $\mathbf{y}$ (and some given subpart $\mathscr{P}$ and $\mathscr{N}$ of the support).

With these ingredients we build a GUESSING-$\mathbf{e}_{\mathscr{N}}(\mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$ that will for each vector in the secret space of the underlying LPN problem make a guess for the value of $\mathbf{e}_{\mathscr{N}}$ by calling the previous procedure multiple times. We then build a checking procedure which will check each guess for $\mathbf{e}_{\mathscr{N}}$ and will try to reconstruct the whole error vector $\mathbf{e}$. Importantly each guess is tested in polynomial time.

Algorithm 3 Provable-double-RLPN algorithm

```
Name: PROVABLE-DOUBLE-RLPN($\mathcal{C}, \mathbf{y}, t$)
Input: $\mathcal{C}$ a linear code of length $n$, $\mathbf{y} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_2^n$ with $\mathbf{c} \in \mathcal{C}$ and $|\mathbf{e}| = t$
Parameter: $s, w, u$ and $N_{\text{iter}}$
1: while $i = 1 \dots N_{\text{iter}}$ do
2:     $\mathscr{P} \overset{\$}{\leftarrow} \{\mathscr{P} \subset [\![1, n]\!] \; : \; |\mathscr{P}| = s\}$        ▷ Hope that $|\mathbf{e}_{\mathscr{N}}| = u$
3:     $\mathscr{N} \leftarrow [\![1, n]\!] \setminus \mathscr{P}$
4:     $\mathcal{G} \leftarrow$GUESSING-$\mathbf{e}_{\mathscr{N}}(\mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$
5:     $\mathbf{e} \leftarrow$ CHECKING-$\mathbf{e}_{\mathscr{N}}(\mathcal{G}, \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N}, t)$
6:     if $\mathbf{e} \neq \perp$ then
7:         return $\mathbf{e}$
```

6.1 Guessing phase

Recalling that $\mathbb{F}_2^{k_{\text{aux}}}$ the space in which lives the secret of the underlying LPN problem $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}$, our goal here is for each $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$, to make a guess on the value of $\mathbf{e}_{\mathscr{N}}$, we make this guess bit by bit. We will exploit the fact that when $\mathbf{z}$ is secret of our LPN samples, namely $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ we have that

Lemma 6.1.

$$F_{\mathscr{H}, \mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \sum_{(\mathbf{h}, \, \mathbf{e}_{\text{aux}}) \in \mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle}.$$

Observe that flipping the positions of the received word $\mathbf{y}$ is exactly flipping the positions of the error vector $\mathbf{e}$. Thus, we have an impact on the score function by flipping the i'th bit of $\mathbf{y}$ in $\mathscr{N}$.

Definition 6.2. *For any $i \in [\![1, n]\!]$ we denote by $\mathbf{y}^{(i)} \in \mathbb{F}_2^n$ the vector defined as $\left(\mathbf{y}^{(i)}\right)_{\mathscr{P}} \overset{def}{=} \mathbf{y}_{\mathscr{P}}$ and $\left(\mathbf{y}^{(i)}\right)_{\mathscr{N}} \overset{def}{=} \mathbf{y}_{\mathscr{N}} + \xi_i$ where $\xi_i \in \mathbb{F}_2^{n-s}$ is the vector which is zero everywhere except on its i'th coordinate.*

Clearly we have that

$$F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \sum_{(\mathbf{h},\ \mathbf{e}_{\text{aux}})\in\mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}}+\xi_i,\mathbf{h}_{\mathscr{N}}\rangle + \langle \mathbf{e}_{\mathscr{P}},\mathbf{e}_{\text{aux}}\rangle}.$$

It is readily seen that this last quantity is expected to be bigger than original $F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$ if we flipped an erroneous position, namely if $\left(\mathbf{e}_{\mathscr{N}}\right)_i = 1$. This is our decision rationale for our guess on $\mathbf{e}_{\mathscr{N}}$.

Algorithm We compute for each $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ and associated guess for $\mathbf{e}_{\mathscr{N}}$, call it $G(\mathbf{z}) \in \mathbb{F}_2^{n-s}$ by successively flipping the bits of $\mathbf{y}$ as described above. More precisely the i'th bit of $G(\mathbf{z})$ is determined by computing a reference score function $F_{\mathscr{H},\mathbf{y}}$ and computing the flipped score function $F_{\mathscr{H},\mathbf{y}^{(i)}}$ and comparing them.

Definition 6.3. *For each* $\mathbf{x} \in \mathbb{F}_2^{k_{\text{aux}}}$ *we denote by* $G(\mathbf{z})$ *the vector of* $\mathbb{F}_2^{n-s}$ *whose* i*'th coordinate is equal to*

$$G(\mathbf{z})_i \overset{def}{=} \begin{cases} 1 & \textit{if} \quad F_{\mathscr{H},\mathbf{y}^{(i)}}(\mathbf{z}) > F_{\mathscr{H},\mathbf{y}}(\mathbf{z}) \\ 0 & \textit{else.} \end{cases}.$$

We call $G(\mathbf{z})$ *the guess for* $\mathbf{e}_{\mathscr{N}}$ *related to* $\mathbf{z}$*.*

We then store these guesses in a set $\mathcal{G} = \{(\mathbf{z},\ G(\mathbf{z}))\ :\ \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\}$ and outputs it.

Algorithm 4

```
Name: GUESSING-$\mathbf{e}_{\mathscr{N}}(\mathcal{C},\mathbf{y},\mathscr{P},\mathscr{N})$
Input: $\mathcal{C},\mathbf{y},\mathscr{P},\mathscr{N}$
1: $\mathscr{H} \leftarrow$ DECODED-DUAL-VECTOR-DOUBLE-RLPN$(\mathcal{C},\mathscr{P},\mathscr{N})$
2: $F_{\mathscr{H},\mathbf{y}} \leftarrow$ SCORE-FUNCTION$(\mathbf{y},\mathscr{H})$      ▷ Reference value of the score function
3: while $i = 1\dots n-s$ do
4:     $\mathbf{y}^{(i)} \leftarrow \mathbf{y} + \xi_{\mathscr{N}_i}$      ▷ This flips the i'th bit of $\mathbf{e}_{\mathscr{N}}$
5:     $F^{\mathcal{L}}_{\mathscr{H},\mathbf{y}^{(i)}} \leftarrow$ SCORE-FUNCTION$(\mathbf{y}^{(i)},\mathscr{H})$      ▷ Comparative score function
6: Compute $G(\mathbf{z})$ (Definition 6.3) for all $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$
7: return $\{(\mathbf{z},\ G(\mathbf{z}))\ :\ \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\}$
```

6.2 Checking phase

Here we want to check each guess for $\mathbf{e}_{\mathscr{N}}$. It is important that each guess is checked in polynomial time as we range over the whole secret space $\mathbb{F}_2^{k_{\text{aux}}}$, and we want this step to no dominate in front of the FFT say. Note that in the case of double-RLPN the secret of the LPN samples is not $\mathbf{e}_{\mathscr{P}}$ but rather some linear combination of $\mathbf{e}_{\mathscr{P}}$, say $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ where $\mathbf{G}_{\text{aux}}$ is the generator matrix of the code used in the reduction from sparse $-$ LPN to plain $-$ LPN. Consequently, given $\mathbf{z}$

a candidate for $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ and an associated guess for $\mathbf{e}_{\mathscr{N}}$ we cannot easily verify if this couple is indeed the solution to our decoding problem as we could have done if we had access to $\mathbf{e}_{\mathscr{P}}$. Notably, trying to recover $\mathbf{e}_{\mathscr{P}}$ from $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ would be exponentially harmful in here as we observe that in practice our double-RLPN optimal parameters are such that this compression of $\mathbf{e}_{\mathscr{P}}$ is extremely lossy, namely the Hamming sphere in which $\mathbf{e}_{\mathscr{P}}$ lives is exponentially larger than this arrival space. We believe there are workarounds for this issue but make rather use the following stategy to keep the algorithm simple.

The procedure. Note that if $\mathscr{N}$ is an information set of the code $\mathcal{C}$ then we can easily check a guess for $\mathbf{e}_{\mathscr{N}}$. Indeed, a guess $\mathbf{z}$ for $\mathbf{e}_{\mathscr{N}}$ can be verified in polynomial time by simply computing the unique codeword $\mathbf{c}$ of $\mathcal{C}$ which is such that $\mathbf{c}_{\mathscr{N}} = \mathbf{y}_{\mathscr{N}} - \mathbf{z}$ and then check that $\mathbf{y} - \mathbf{c}$ is of right Hamming weight t (the weight of the error). Importantly, denoting by $s \overset{\text{def}}{=} |\mathscr{P}|$, the setting where $\mathscr{N}$ is an information set with good probability, is when

$$n - s \geqslant k$$

which are trivially verified by all our parameters when the rate R of the code $\mathcal{C}$ is smaller than 0.5 as $s \leqslant k$. This is more than enough to account for the interesting parameters for which we beat the ISD's. The checking algorithm is described in Algorithm 5.

Algorithm 5

```
Name: CHECKING-$\mathbf{e}_{\mathscr{N}}$ $(\mathcal{G}, \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$
Input: $\mathcal{G}, \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N}$
Parameter: $t$
1: while $\mathbf{e}_{\mathscr{N}}^{(\mathbf{z})} \in \mathcal{G}$ do
2:     $\mathbf{c}_{\mathscr{N}} \leftarrow \mathbf{y}_{\mathscr{N}} - \mathbf{e}_{\mathscr{N}}^{(\mathbf{z})}$
3:     $\mathbf{c} \leftarrow \text{Lift}(\mathcal{C},\ \mathscr{N},\ \mathbf{c}_{\mathscr{N}})$
4:     if $|\mathbf{y} - \mathbf{c}| = t$ then
5:         $\mathbf{e} \leftarrow \mathbf{y} - \mathbf{c}$
6:         return $\mathbf{e}$
```

It is readily seen that we have that If the guess for $\mathbf{e}_{\mathscr{N}}$ related to the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ is good, namely if $G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \mathbf{e}_{\mathscr{N}}$ and if $\mathscr{N}$ contains an information set of $\mathcal{C}$ then CHECKING-$\mathbf{e}_{\mathscr{N}}$ outputs $\mathbf{e}$.

7 Instantiation with a juxtaposition codes

Goal. Now, let us instantiate our algorithm with a specific family of $[s, k_{\text{aux}}]$-linear code $\mathcal{F}$ that are used in the reduction from sparse LPN to plain LPN : the so called auxiliary code $\mathcal{C}_{\text{aux}}$ is drawn uniformly at random in $\mathcal{F}$ in Algorithm 1.

The codes and its decoder must be as efficient as possible : given $\mathbf{h}_{\mathscr{P}}$ one wants that the decoder returns a codeword $\mathbf{c}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$ at small distance t_{aux} possible while having a small decoding time the decoding distance, *i.e.* the weight of $\mathbf{h}_{\mathscr{P}} - \mathbf{c}_{\text{aux}}$, crucially intervene in the noise of the generated LPN samples. Information theory tells us that the best we could hope the decoding distance t_{aux} equals to the Gilbert-Varshamov distance $d_{\text{GV}}(s, k_{\text{aux}})$ for a proportion $1 - o(1)$ of the word of the space $\mathbb{F}_2^s$.

Codes/Decoder used in [CDMT24]. Forgetting about the question of actually having a decoder running in polynomial time, this could be achieved with a random code. In fact, in double-RLPN the analysis was carried out when $\mathcal{C}_{\text{aux}}$ is taken uniformly at random among $[s, k_{\text{aux}}]$-linear codes and it was supposed to be equipped with an ideal genie-aided decoder that finds all the codeword at distance t_{aux} and where each codeword found in time $\text{poly}(n)$. In [CDMT24,] it was also proposed to use a Cartesian product (what we call juxtaposition codes) of $b = s/\log_2(s)$ smaller random linear codes $\mathcal{C}_{\text{aux}} = \mathcal{C}^{(1)} \times \mathcal{C}^{(2)} \times \cdots \times \mathcal{C}^{(b)}$, each of length $s/\log_2(s)$ and dimension $k_{\text{aux}}/\log_2(s)$.

Definition 7.1 (Juxtaposition code). *We define the set of juxtaposition codes with b blocks, and of length n and dimension k, namely $\mathfrak{C}^{\text{juxt}}[b,\ n,\ k]$, as the set of linear codes $\mathcal{C}$ such that there exists b linear codes, $(\mathcal{C}^{(i)})_{i \in [\![1,\ b]\!]}$ that are such that for every $i \in [\![1,\ b]\!]$, $\mathcal{C}^{(i)}$ is an $[n^{(i)}, k^{(i)}]$-linear code and such that*

$$\mathcal{C} = \mathcal{C}^{(1)} \times \mathcal{C}^{(2)} \times \cdots \times \mathcal{C}^{(b)}$$

where we denote implicitly (in b), for each integer $v \in \mathbb{N}$ its i'th part as:

$$v^{(i)} \overset{def}{=} \begin{cases} \lfloor v/b \rfloor + 1 & \text{if } i \leqslant (v \mod b) \\ \lfloor v/b \rfloor & \text{else} \end{cases} \tag{7}$$

Definition 7.2. *When the context is clear we will implicitly denote the i'th part of a vector $\mathbf{x} \in \mathbb{F}_2^n$ relative to the support given by n by $\mathbf{x}^{(i)} \overset{def}{=} \mathbf{x}_{\mathscr{I}}$ where $\mathscr{I} \overset{def}{=} [\![\sum_{j=1}^{i-1} n^{(j)},\ \sum_{j=1}^{i} n^{(j)}]\!]$. In the same manner, given $\mathcal{C} \in \mathfrak{C}^{\text{juxt}}[b,\ n,\ k]$ we denote by $\mathcal{C}^{(i)}$ its i'th constituent code.*

Decoding $\mathbf{h}_{\mathscr{P}}$ is done by enumerating independently the codewords of each code and decode independently on each part. It was shown overall that this incurs only a loss of order $2^{o(n)}$ in double-RLPN compared to the ideal case where a random code with a genie-aided decoder is used.

The decoder we use here. Here we use the base observation that even if $\mathcal{C}_{\text{aux}}$ is a random $[s, k_{\text{aux}}]$-linear code, it can be decoded at distance t_{aux} in polynomial amortized time as long as number total number of calls, to the decoders is superior to the number of admissible errors of good weight $\mathbf{e}_{\text{aux}} \in \mathcal{S}_{t_{\text{aux}}}^s$, namely $\binom{t_{\text{aux}}}{s}$. In this case one can simply create a syndrome table $\{\ (\mathbf{H}_{\text{aux}}\mathbf{e}_{\text{aux}},\ \mathbf{e}_{\text{aux}})\ :$

$\mathbf{e}_{\text{aux}} \in \mathcal{S}^{s}_{t_{\text{aux}}}\}$ once and for all and, when one wants to decode $\mathbf{a} \in \mathbb{F}_2^s$ onto $\mathcal{C}_{\text{aux}}$ at distance t_{aux}, the decoders look up the table and returns the $\mathbf{e}_{\text{aux}}$ that are such that $\mathbf{H}_{\text{aux}}\mathbf{a} = \mathbf{H}_{\text{aux}}\mathbf{e}_{\text{aux}}$. This idea can be extended by considering a juxtaposition code with b block and decoding each code independently at distance t_{aux}/b with this error enumeration technique, this decoder was already used in [GJL14]. Basically the decoder returns the following set of errors.

Definition 7.3 (Set of admissible errors). *Let $b, n, k, t \in \mathbb{N}$. Let $\mathcal{C} \in \mathfrak{C}^{\text{juxt}}[b,\ n,\ k]$ be a juxtaposition code of length n and dimension k. Let $\mathbf{y} \in \mathbb{F}_2^n$, we define the set of admissible errors as*

$$Dec^{\text{juxt}}(\mathcal{C}, \mathbf{y}, t) \overset{def}{=} \{\mathbf{e} \in \mathcal{S}^n_t\ :\ \left|\mathbf{e}^{(i)}\right| = t^{(i)} \text{ and } \mathbf{y}^{(i)} - \mathbf{e}^{(i)} \in \mathcal{C}^{(i)},\ \forall i \in [\![1,\ b]\!]\}.$$

We have in particular that

$$Dec^{\text{juxt}}(\mathcal{C}, \mathbf{y}, t) \subset \{\mathbf{e} \in \mathcal{S}^n_t\ :\ \mathbf{e} + \mathbf{y} \in \mathcal{C}\}.$$

This is particularly useful when one have to decode an exponential number of vectors, say $2^{\lambda s}$ with $\lambda > 0$. The point being that there exists a constant b such that the cost of computing the syndrome table is $\text{poly}(n)\binom{s/b}{t_{\text{aux}}/b} = \text{poly}(n)\sqrt[b]{\binom{s}{t_{\text{aux}}}}$ is smaller than $2^{\lambda s}$. The fact that b is constant allows on to argue that

$$\mathbb{E}\left(\left|\text{Dec}^{\text{juxt}}(\mathcal{C}, \mathbf{y}, t)\right|\right) = \tilde{\Omega}\left(\frac{\binom{t_{\text{aux}}}{s}}{2^{s-k_{\text{aux}}}}\right).$$

With a variance argument we could show that this allows to conclude that juxtaposition can decode returning this set of admissible errors is essentially as good as a random linear codes with a genie aided-decoder.

All in all here is our provable-double-RLPN algorithm instantiated with this family of code and this decoder.

Definition 7.4 (Provable-DoubleRLPN with juxtaposition codes). *We define an instantiation of Algorithm 3 where we have an additionnal parameter $b \in \mathbb{N}$ and where:*

- *The family $\mathcal{F} \subset \mathfrak{C}[s, k_{\text{aux}}]$ of auxiliary codes is defined as*

$$\mathcal{F} = \mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]$$

- *After having drawn the code $\mathcal{C}_{\text{aux}}$ from $\mathcal{F}$, compute and store for each $i \in [\![1,\ b]\!]$ an hash table $T^{(i)}$ indexed by syndromes, namely $T^{(i)}[\mathbf{s}] = \{\mathbf{e}^{(i)}_{\text{aux}} \in \mathcal{S}^{s^{(i)}}_{t^{(i)}_{\text{aux}}}\ :\ \mathbf{H}_{\text{aux}}{}^{(i)}\mathbf{e}^{(i)}_{\text{aux}} = \mathbf{s}^{(i)}\ \}$ where $\mathbf{H}_{\text{aux}}{}^{(i)}$ is a parity-check matrix of $\mathcal{C}^{(i)}_{\text{aux}}$.*
- *A call to* $\textsc{Decode}(\mathcal{C}_{\text{aux}}, \mathbf{a},\ t_{\text{aux}})$ *returns $Dec^{\text{juxt}}(\mathcal{C}, \mathbf{a}, t_{\text{aux}})$ by looking up the previous syndrome tables.*

Importantly, our regime of interests here will be when $R, t = \Theta(n)$, this corresponds to regime in which the results of [CDMT24] where given. In this case, the optimal parameters of double-RLPN are such that $s, k_{\text{aux}}, t_{\text{aux}} = \Theta(s)$ and, in particular, the complexity of an iteration of double-RLPN is exponential in n and at least equal to the complexity of the FFT given by $k_{\text{aux}}\, 2^{k_{\text{aux}}}$. Basically this means that there exists a constant b such that $\sqrt[b]{\binom{s}{t_{\text{aux}}}} < 2^{k_{\text{aux}}}$, making the syndrome table creation step negligible compared to the cost of the FFT.

We directly have the following complexity result.

Proposition 7.5 (Complexity of fully provable double-RLPN with juxtaposition codes). *Let n and let $k, t, s, k_{\text{aux}}, t_{\text{aux}}, N_{\text{iter}}, b$ be implicit functions of n. Suppose that $b = \lceil s/k_{\text{aux}} \rceil$ and that $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$. Then, given an instance of $\mathrm{DP}_{\mathbf{G}}(n,k,t)$, the expected time and memory complexity of Definition 7.4 is given by*

$$\textbf{\textit{Time}} = \widetilde{\mathcal{O}}\left(N_{\text{iter}}\left(T_{\text{eq}} + 2^{k_{\text{aux}}}\right)\right), \qquad \textbf{\textit{Memory}} = \widetilde{\mathcal{O}}\left(\left(M_{\text{eq}} + 2^{k_{\text{aux}}}\right)\right)$$

where T_{eq} and M_{eq} are respectively the expected time and memory complexity of one call to Compute-Small-Codewords.

8 Analysis

In this section we prove the main Theorem 5.1 by analyzing our provable variant of double-RLPN instantiated with juxtaposition code. Forgetting about the other technical details that we will deal with later in the rigorous proof, in essence, proving our main theorem, mainly relies on proving that, we have with probability $1 - o(1)$ that when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$, the guess on $\mathbf{e}_{\mathscr{N}}$ is $\mathbf{e}_{\mathscr{N}}$. By construction of the guess, it is sufficient to prove that with probability $1 - o(1/n)$ we have that

$$F_{\mathscr{H},\mathbf{y}}(\mathbf{e}_{\mathscr{P}}) > F_{\mathscr{H},\mathbf{y}^{(i)}}(\mathbf{e}_{\mathscr{P}}) \qquad \text{if } (\mathbf{e}_{\mathscr{N}})_i = 0, \tag{8}$$

$$F_{\mathscr{H},\mathbf{y}}(\mathbf{e}_{\mathscr{P}}) < F_{\mathscr{H},\mathbf{y}^{(i)}}(\mathbf{e}_{\mathscr{P}}) \qquad \text{if } (\mathbf{e}_{\mathscr{N}})_i = 1. \tag{9}$$

The result can then be concluded with a union bound on $i \in [\![1,\, n-s]\!]$. Our proof mainly relies on the two following propositions. The first is a second-order concentration bound on the score function.

Proposition 8.1. *There exists a positive poly-bounded function f_0 such that for any $t, k, s, k_{\text{aux}}, t_{\text{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $\frac{\prod_{j=1}^{b}\binom{s^{(j)}}{t^{(j)}_{\text{aux}}}}{2^{s-k_{\text{aux}}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$ then*

$$\mathbb{P}\left(\left|F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - \mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)\right| \geqslant f_0(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right)$$

and

$$\mathbb{E}\left(F_{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\delta_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{j=1}^{b}\delta_{t^{(j)}_{\text{aux}}}^{(s^{(j)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right),$$

where N is the expected number of LPN samples, namely $N \overset{def}{=} \mathbb{E}\left(|\mathscr{H}|\right)$ and is given by

$$N = N_{\text{eq}} N_{\text{aux}}, \qquad N_{\text{eq}} \overset{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}}, \qquad N_{\text{aux}} \overset{def}{=} \frac{\prod_{j=1}^{b} \binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s-k_{\text{aux}}}}$$

and where the distributions considered in the probabilities are as follows

- *$\mathscr{N}$ and $\mathscr{P}$ are two fixed complementary subsets of $[\![1,\, n]\!]$ of size s and $n-s$ respectively.*
- *$\mathcal{C}$ is chosen by taking its generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ uniformly at random in $\{\ \mathbf{G} \in \mathbb{F}_2^{k\times n}\ :\ \text{rank}\left(\mathbf{G}_{\mathscr{P}}\right) = s\}$ and $\mathcal{C}_{\text{aux}}$ is chosen by taking its generator matrix $\mathbf{G}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}} \times s}$ uniformly at random among matrices of rank k_{aux}.*
- *$\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \sim \mathcal{U}\left(\mathcal{C}\right)$ and $\mathbf{e} \in \mathcal{S}_t^n$ is a fixed vector.*
- *The set $\mathscr{H}$ is*

$$\mathscr{H} \overset{def}{=} \{(\mathbf{m}_{\text{aux}}, \mathbf{h}) \in \mathbb{F}_2^{k_{\text{aux}}} \times \mathcal{C}^{\perp} : |\mathbf{h}_{\mathscr{N}}| = w,\ \forall i \in [\![1, b]\!], (\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} + \mathbf{h}_{\mathscr{P}})^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\}.$$

- *Where $\mathcal{L} \overset{def}{=} \mathcal{L}\left(\mathscr{H}, \mathbf{y},\ \mathbf{G}_{\text{aux}}, \mathscr{P}\right)$ is defined in Definition 4.1 and where we recall that $F_{listL}\left(\mathbf{z}\right)$ is defined in Definition 4.3.*

This proposition is proved in Section 10.1.

Remark 8.2. This proposition is only a slight variant of [CDMT24, Proposition 2].

The main part of our proof relies on the following lemma that state that the expected value of $F_{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$ and $F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$ are essentially as far as the expected value of $F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$ and 0.

Lemma 8.3. *There exists a positive poly-bounded function f_1 such that for any $s, w, u \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that $\text{Root}\left(K_w^{(n-s)}\right) - u \in \Omega\left(n-s\right)$ we have that*

$$\delta_w^{(n-s)}\left(u-1\right) - \delta_w^{(n-s)}\left(u\right) \geqslant \frac{1}{f_1(n)} \delta_w^{(n-s)}\left(u\right),$$
$$\delta_w^{(n-s)}\left(u\right) - \delta_w^{(n-s)}\left(u+1\right) \geqslant \frac{1}{f_1(n)} \delta_w^{(n-s)}\left(u\right).$$

This lemma is proved in Section 10.2.

8.1 Rest of the proof.

Recall that our algorithm is iterated a number N_{iter} of times. In RLPN this was done to ensure that there existed at least one iteration such that $|\mathbf{e}_{\mathscr{N}}| = u$ where u is a parameter. Clearly, from Proposition 8.1 one can see that, because we used juxtaposition codes with b block, the distribution of the score function evaluated on the secret, $F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$, finely depends on the weight of $\mathbf{e}$ on each subpart, namely on $|\mathbf{e}_{\mathscr{N}}|$ and $\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|$ for each $i \in [\![1,\, b]\!]$. We will make the additionnal, finer bet that $\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|$ is of typical weight roughly $(t-u)/b$. More precisely we will make the bet that

$$|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}\,.$$

First, because b is constant, essentially this bet is verified with the same probability, up to polynomial factors, as the simpler bet that $|\mathbf{e}_{\mathscr{N}}| = w$. Namely, we have that

Lemma 8.4. *There exists a positive poly-bounded function g such that for any t, s, b implicit functions of n we have that*

$$\mathbb{P}\left(|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}\right) \geqslant \frac{1}{g(n^b)} \frac{\binom{n-s}{u}\binom{s}{t-u}}{\binom{n}{t}}$$

where $\mathscr{N}$ and $\mathscr{P}$ are any fixed complementary subsets of $[\![1,\, n]\!]$ and $\mathbf{e}$ is taken uniformly at random in $\mathcal{S}_t^n$.

The proof is straightforward using Proposition 3.13. Moreover, using again the fact b is constant one can show easily that all the quantities appearing in Proposition 8.1, like N and the product of the biases, polynomially relates to the ideal case $b = 1$. More precisely we have the following.

Proposition 8.5 (Relating the quantities to the case $b = 1$). *There exists a positive poly-bounded function f_2 such that for any $k, t, s, u, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$\text{Root}\left(K_w^{(n-s)}\right) - u \in \Omega\,(n-s) \qquad \textit{and} \qquad \forall i \in [\![1, b]\!] \text{Root}\left(K_{t_{\text{aux}}}^{(s)}\right) - (t-u) \in \Omega\,(s)$$

then we have that

$$\prod_{j=1}^{b} \delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left(\left|(t-u)^{(i)}\right|\right) \in \Omega\left(\frac{1}{f_2(n^b)} \delta_{t_{\text{aux}}}^{(s)}\,(t-u)\right),$$

$$\frac{\prod_{j=1}^{b} \binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s-k_{\text{aux}}}} \in \Omega\left(\frac{1}{f_2(n^b)} \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}}\right).$$

The proof is straightforward using Proposition 3.13 and Proposition 3.14.

Proof (Proof of the main Theorem 5.1). In Theorem 5.1 we choose the positive poly-bounded function f appearing as $f(n) = (f_0(n) f_1(n))^2 f_2(n)^3$ where f_0, f_1 and f_2 are defined in Proposition 8.1, Lemma 8.3 and Proposition 8.5 respectively. Let us now consider parameters $k, t, s, w, u, s, k_{\text{aux}}, t_{\text{aux}}, b$ satifying the conditions of the theorem. Now, about the choice for N_{iter}, by using Lemma 8.4

we can show that there exists N_{iter} that is such that i) $N_{\text{iter}} = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{s}{t-u}\binom{n-s}{u}}\right)$ and ii) there exists with probability $1-o(1)$ an iteration of the algorithm that is such that the bet on the error is valid, namely such that

$$|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}. \tag{10}$$

Let us consider such a value for N_{iter} and suppose we are in one of the iteration where the bet is valid. It is easy to convince oneself that the conditions in the theorem are sufficient to ensure that the conditions to apply Proposition 8.1 are met with probability $1-o(1)$. We recall that those conditions to apply Proposition 8.1 are that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and that $\mathscr{H}$ is equal to the full set of decoded dual vectors. We suppose that this iteration is such that those conditions are met. Let $i \in [\![1,\, n-s]\!]$ and let us now prove that

$$\mathbb{P}\left(G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)_i = \left(\mathbf{e}_{\mathscr{N}}\right)_i\right) = 1 - o(1/n). \tag{11}$$

Note that proving it would directly prove our theorem. Indeed we then we can easily prove that

$$\mathbb{P}\left(G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \mathbf{e}_{\mathscr{N}}\right) = 1 - o(1)$$

by using the union bound. In turn, we can show that $\mathscr{N}$ is an information set of the code $\mathcal{C}$ with probability $1-o(1)$ by using the condition that $n-s-k \in \omega(1)$ in the theorem. This means that with probability $1-o(1)$ the procedure Testing-$\mathbf{e}_{\mathscr{N}}$ returns the error $\mathbf{e}$.

Let us now prove Eq. (11). Suppose, without loss of generality that $\left(\mathbf{e}_{\mathscr{N}}\right)_i = 1$, the proof in the other case is similar. Recall that by construction $G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)_i = 1$ if and only if $F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) < F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$. Thus to prove our result we only have to prove that

$$\mathbb{P}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) < F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = 1 - o(1).$$

We prove it using Proposition 8.1. Indeed we can write that

$$\mathbb{P}\left(\left|F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - E(0)\right| \geqslant f_1(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right),$$
$$\mathbb{P}\left(\left|F_{\mathscr{H},\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - E(1)\right| \geqslant f_1(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right)$$

where

$$E(x) \overset{\text{def}}{=} N\delta_{w}^{(n-s)}(u+x)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right), \qquad N \overset{\text{def}}{=} \frac{\binom{n-s}{w}\prod_{j=1}^{b}\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{k-k_{\text{aux}}}}.$$

This allows saying that we make the good guess for each coordinate with probability $1-o(1/n)$ as long as

$$E(-1) - E(0) > f_0(n^b)\sqrt{N} \tag{12}$$

Using successively Lemma 8.3 and Proposition 8.5 we get

$$\begin{aligned} E(-1) - E(0) &\geqslant \frac{1}{f_1(n^b)} N \delta_w^{(n-s)}(u) \prod_{j=1}^{b} \delta_{t_{\mathrm{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right) \\ &\geqslant \frac{1}{f_1(n) f_2(n)} N \delta_w^{(n-s)}(u)\, \delta_{t_{\mathrm{aux}}}^{(s)}(t-u) \end{aligned}$$

Plugging this into Eq. (12), this means that we make the right decision with probability $1 - o(1)$ if

$$\frac{1}{f_1(n^b) f_2(n^b)} N \delta_w^{(n-s)}(u)\, \delta_{t_{\mathrm{aux}}}^{(s)}(t-u) \geqslant f_0(n^b)\sqrt{N}$$

This is equivalent to asking that

$$N \geqslant \frac{\left(f_0(n^b) f_1(n^b) f_2(n^b)\right)^2}{\left(\delta_w^{(n-s)}(u)\, \delta_{t_{\mathrm{aux}}}^{(s)}(t-u)\right)^2}.$$

But, from Proposition 8.5 we have that

$$N \geqslant \frac{1}{f_2(n^b)} \frac{\binom{s}{t_{\mathrm{aux}}}}{2^{s-k_{\mathrm{aux}}}}.$$

Thus replacing N in the previous equation we get that our condition to make the right decision with probability $1 - o(1)$ is

$$\frac{\binom{n-s}{w}\binom{s}{t_{\mathrm{aux}}}}{2^{s-k_{\mathrm{aux}}}} \geqslant \frac{\left(f_0(n^b) f_1(n^b)\right)^2 f_2(n^b)^3}{\left(\delta_w^{(n-s)}(u)\, \delta_{t_{\mathrm{aux}}}^{(s)}(t-u)\right)^2}.$$

which is exactly the condition of the theorem.

9 Conclusion

In this work we have presented a variant of the most recent code-based dual attack, double-RLPN, but that we can fully prove without using any heuristics up to rate $R \leqslant 0.5$ and that has the same performances, up to polynomial factors, as the original algorithm. We believe that this proof strategy could also be adapted to lattice-based dual attacks.

References

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.

BM18. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2018*, volume 10786 of *LNCS*, pages 25–46, Fort Lauderdale, FL, USA, April 2018. Springer.

Car20. Kevin Carrier. *Recherche de presque-collisions pour le décodage et la reconnaissance de codes correcteurs.* Theses, Sorbonne Université, June 2020.

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.

CDMT24. Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Reduction from sparse LPN to LPN, dual attack 3.0. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *LNCS*, pages 286–315. Springer, 2024. Artifact available at `https://artifacts.iacr.org/eurocrypt/2024/a10/`.

DT17. Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. preprint, January 2017. arXiv:1701.07416.

Dum89. Il'ya Dumer. Two decoding algorithms for linear codes. *Probl. Inf. Transm.*, 25(1):17–23, 1989.

GJL14. Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. In *Advances in Cryptology - ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 1–20. Springer, 2014.

Jab01. Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *Cryptography and coding. Proceedings of the 8^{th} IMA International Conference*, volume 2260 of *LNCS*, pages 1–8, Cirencester, UK, December 2001. Springer.

KS21. Naomi Kirshner and Alex Samorodnitsky. A moment ratio bound for polynomials and some extremal properties of krawchouk polynomials and hamming spheres. *IEEE Trans. Inform. Theory*, 67(6):3509–3541, 2021.

LF06. Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In *Proceedings of the 5th international conference on Security and Cryptography for Networks*, volume 4116 of *LNCS*, pages 348–359. Springer, 2006.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

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

MT23. Charles Meyer-Hilfiger and Jean-Pierre Tillich. Rigorous foundations for dual attacks in coding theory. In *Theory of Cryptography Conference, TCC 2023*, volume 14372 of *LNCS*, pages 3–32. Springer Verlag, December 2023.

Ove06. Raphael Overbeck. Statistical decoding revisited. In Reihaneh Safavi-Naini Lynn Batten, editor, *Information security and privacy : 11^{th} Australasian conference, ACISP 2006*, volume 4058 of *LNCS*, pages 283–294. Springer, 2006.

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

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

vL99. Jacobus Hendricus van Lint. *Introduction to coding theory.* Graduate texts in mathematics. Springer, 3rd edition edition, 1999.

10 Appendices

10.1 Proof of the second-order concentration bound

The goal of this section is to prove Proposition 8.1 that we recall here.

Proposition 8.1. *There exists a positive poly-bounded function f_0 such that for any $t, k, s, k_{\text{aux}}, t_{\text{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $\frac{\prod_{j=1}^{b}\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s-k_{\text{aux}}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$ then*

$$\mathbb{P}\left(\left|F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - \mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)\right| \geqslant f_0(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right)$$

and

$$\mathbb{E}\left(F_{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\delta_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right),$$

where N is the expected number of LPN samples, namely $N \overset{def}{=} \mathbb{E}\left(|\mathscr{H}|\right)$ and is given by

$$N = N_{\text{eq}}N_{\text{aux}}, \qquad N_{\text{eq}} \overset{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}}, \qquad N_{\text{aux}} \overset{def}{=} \frac{\prod_{j=1}^{b}\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s-k_{\text{aux}}}}$$

and where the distributions considered in the probabilities are as follows

- *$\mathscr{N}$ and $\mathscr{P}$ are two fixed complementary subsets of $[\![1, n]\!]$ of size s and $n-s$ respectively.*
- *$\mathcal{C}$ is chosen by taking its generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ uniformly at random in $\{\, \mathbf{G} \in \mathbb{F}_2^{k\times n} : \text{rank}\left(\mathbf{G}_{\mathscr{P}}\right) = s\}$ and $\mathcal{C}_{\text{aux}}$ is chosen by taking its generator matrix $\mathbf{G}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}}\times s}$ uniformly at random among matrices of rank k_{aux}.*
- *$\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \sim \mathcal{U}\left(\mathcal{C}\right)$ and $\mathbf{e} \in \mathcal{S}_t^n$ is a fixed vector.*
- *The set $\mathscr{H}$ is*

$$\mathscr{H} \overset{def}{=} \{(\mathbf{m}_{\text{aux}}, \mathbf{h}) \in \mathbb{F}_2^{k_{\text{aux}}}\times\mathcal{C}^{\perp} : |\mathbf{h}_{\mathscr{N}}| = w,\ \forall i \in [\![1, b]\!], (\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} + \mathbf{h}_{\mathscr{P}})^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\}.$$

- *Where $\mathcal{L} \overset{def}{=} \mathcal{L}\left(\mathscr{H}, \mathbf{y},\ \mathbf{G}_{\text{aux}}, \mathscr{P}\right)$ is defined in Definition 4.1 and where we recall that $F_{listL}\left(\mathbf{z}\right)$ is defined in Definition 4.3.*

The main technical lemma that we will use here is the following giving the first two moments of the score function.

Lemma 10.1. *(Main technical lemma.) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ are implicit functions of $n \in \mathbb{N}$ we have that*

$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\, \delta_{w}^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right) \prod_{j=1}^{b} \delta_{t_{\text{aux}}^{(j)}}^{\left(s^{(j)}\right)}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right)$$

$$\mathbf{Var}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = \mathcal{O}\left(n^{b+1}\, N\, \max\left(1,\ N_{\text{aux}}\right)\right)$$

Remark 10.2. We believe that the term n^{b+1} appearing in the variance is much more reasonable and is rather some $\mathcal{O}\left(2^{b}\right)$ in the constant rate regime.

Proof (Proof of Proposition 8.1). This is directly obtained by using Byenemé-Chebyshev inequality along with the moments given in Lemma 10.1.

Proving the second-order moments. Let us now prove Lemma 10.1. We recall for convenience two lemmas we will be useful to prove the proposition. We recall here the distribution appearing in the previous expression and which will

Lemma 10.3 (Distribution of some related quantities). *For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ we have that*

$$\mathbf{R} \sim \mathcal{U}\left(\mathbb{F}_{2}^{s\times(n-s)}\right), \tag{13}$$

$$\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} \sim \mathcal{U}_{\mathbf{H}}\left(n-s,\ n-k\right), \tag{14}$$

$$\mathbf{R} \text{ and } \mathcal{C}^{\mathscr{N}} \text{ are independent.} \tag{15}$$

where $\mathbf{R} \overset{def}{=} \text{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right)$ and where the other quantities are defined in **??**.

Proof. The second property is trivial and the first and third can be shown by observing the way $\mathbf{R}$ is constructed in **??** along with the fact that by assumption in Proposition 8.1 $\mathbf{G}_{\mathscr{N}}$ is distributed uniformly at random in $\mathbb{F}_{2}^{k\times(n-s)}$.

This allows us to get the following lemma

Lemma 10.4. *We have that*

$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\, \delta_{w}^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right) \prod_{j=1}^{b} \delta_{t_{\text{aux}}^{(j)}}^{\left(s^{(j)}\right)}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right),$$

$$\mathbf{Var}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant N\left[1 + \sum_{\mathbf{c}\in\{0,1\}^{b}\,:\,\mathbf{c}\neq\mathbf{0}} \prod_{i=1}^{b}\left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i}\right]$$

Proof. Let us first compute the expected value. Recall that we have that

$$|\mathbf{e}_{\mathscr{N}}| = u,\ \left|\mathbf{e}_{\mathscr{P}}^{(1)}\right| = (t-u)^{(1)},\ \cdots,\ \left|\mathbf{e}_{\mathscr{P}}^{(b)}\right| = (t-u)^{(b)} \tag{16}$$

We will show that

$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\ \delta_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right).$$

Rewriting the score function we have that

$$F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\sum_{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}_{t_{\text{aux}}^{(1)}}^{s^{(1)}}}\cdots\sum_{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}_{t_{\text{aux}}^{(b)}}^{s^{(b)}}}(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\left[\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}} \tag{17}$$

By linearity of the expected value we have that

$$\mathbb{E}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\sum_{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}_{t_{\text{aux}}^{(1)}}^{s^{(1)}}}\cdots\sum_{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}_{t_{\text{aux}}^{(b)}}^{s^{(b)}}} \left[(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]\left[\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\right] \tag{18}$$

Now from the independence of the indicator variable we get that

$$\begin{aligned}\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right) &= \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\right)\prod_{i=1}^{b}\mathbb{E}\left(\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\\ &= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right).\end{aligned} \tag{19}$$

From Lemma 10.3 we have that $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\sim\mathcal{U}_{\mathbf{H}}\left(n-s,\ n-k\right)$ thus using Lemma 3.9 we get

$$\mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right) = \frac{1}{2^{k-s}}.$$

From Lemma 10.3 we have that for each $i\in[\![1,b]\!]$, $\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\sim\mathcal{U}\left(\mathbb{F}_2^{s^{(i)}}\right)$, thus as $\mathcal{C}_{\text{aux}}^{(i)}$ is an $[s^{(i)},k_{\text{aux}}^{(i)}]-$linear code we get that

$$\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) = \frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}. \tag{20}$$

Plugging these last equation into Eq. (19) yield that

$$\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)=\frac{1}{2^{k-s}}\prod_{i=1}^{b}\frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}$$
$$=\frac{1}{2^{k-k_{\text{aux}}}} \tag{21}$$

where in the last equation we used the fact that by definition of the i'th part of a vector in Eq. (7) we have that

$$\sum_{i=1}^{b}s^{(i)}=s,\qquad\sum_{i=1}^{b}k_{\text{aux}}^{(i)}=k_{\text{aux}}.$$

Finally, plugging this last equality back into Eq. (18) gives that

$$\begin{aligned}\mathbb{E}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)&=\frac{1}{2^{k-k_{\text{aux}}}}\sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\sum_{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}^{s^{(1)}}_{t_{\text{aux}}^{(1)}}}\cdots\sum_{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}^{s^{(b)}}_{t_{\text{aux}}^{(b)}}}(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\\&=\frac{1}{2^{k-k_{\text{aux}}}}K_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{i=1}^{b}K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right)\\&=\frac{\binom{n-s}{w}\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{k-k_{\text{aux}}}}\frac{K_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{i=1}^{b}K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right)}{\binom{n-s}{w}\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}\\&=N\delta_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{i=1}^{b}\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right)\end{aligned}$$

where in the last lines we used the definition of δ in Lemma 3.12.

Let us now compute the variance of $F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$. Starting again from Equation (17) and denoting by $A\in\{-1,1\}$ the value:

$$A\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}}^{(1)},\ \ldots,\ \mathbf{e}_{\text{aux}}^{(b)}\right)\overset{\text{def}}{=}(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\left[\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]$$

we have that

$$F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)=\sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\sum_{\left(\mathbf{e}_{\text{aux}}^{(i)}\in\mathcal{S}^{s^{(i)}}_{t_{\text{aux}}^{(i)}}\right)_{i\in[\![1,\,b]\!]}}A\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}}^{(1)},\ \ldots,\ \mathbf{e}_{\text{aux}}^{(b)}\right)\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\perp})_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}.$$

Now we use the fact that we can upper bound the variance of $\sum_i A_i\mathbf{X}_i$ where $\mathbf{X}_i$ are some random variables the $A_i\in\{-1,1\}$ are some fixed coefficient by upper bounding the covariance as

$$\begin{aligned}Cov\left(A_i\,X_i,A_j\,X_j\right)&=A_iA_jCov\left(X_i,X_j\right)\\&\leqslant\left|Cov\left(X_i,X_j\right)\right|.\end{aligned}$$

This observation allows us to write that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant V + C \tag{22}$$

where

$$V \overset{\text{def}}{=} \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \mathbf{Var}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}} \in (\mathcal{C}^{\perp})_{\mathscr{N}}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) \tag{23}$$

and

$$C \overset{\text{def}}{=} \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\mathbf{g}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \sum_{\left(\mathbf{z}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \mathbf{1}_{(\mathbf{h}_{\mathscr{N}},\, \mathbf{e}_{\text{aux}}) \neq (\mathbf{g}_{\mathscr{N}},\, \mathbf{z}_{\text{aux}})} \left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right| \tag{24}$$

where

$$\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) \overset{\text{def}}{=} \mathbf{Cov}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}} \in (\mathcal{C}^{\perp})_{\mathscr{N}}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}},\ \mathbf{1}_{\mathbf{g}_{\mathscr{N}} \in (\mathcal{C}^{\perp})_{\mathscr{N}}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right). \tag{25}$$

Let us first compute the term V. As V is the variance of a Bernoulli distribution, we can upper bound it by the expected value of this Bernoulli:

$$\begin{aligned}\mathbf{Var}\left(\mathbf{1}_{(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}, \mathbf{h}_{\mathscr{N}}) \in \mathscr{W}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) &\leqslant \mathbb{E}\left(\mathbf{1}_{(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}, \mathbf{h}_{\mathscr{N}}) \in \mathscr{W}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) \\ &= \frac{1}{2^{k-k_{\text{aux}}}} \qquad (Eq.\ (21))\end{aligned}$$

And thus, plugging this last equation in Eq. (23) we get

$$\begin{aligned}V &\leqslant \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \frac{1}{2^{k-k_{\text{aux}}}} \\ &= \frac{\binom{n-s}{w} \prod_{i=1}^{b} \binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{k-k_{\text{aux}}}} \\ &= N.\end{aligned}$$

Let us now compute the covariance terms C by first rewriting $\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)$. We have

$$\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)$$
$$-\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}},\ \prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}},\ \prod_{i=1}^{b}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)$$

Thus

$$\begin{aligned}
&\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \\
&= \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}}\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2} \\
&= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2}
\end{aligned} \tag{26}$$

where in the last line we used the independence of the variables.

1. Case $\mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}$. Suppose here that $\mathbf{h}_{\mathscr{N}}\neq\mathbf{h}_{\mathscr{N}}$ Let us first compute $\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)$. First, as $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\sim\mathcal{U}_{\mathbf{H}}\left(n-s,\ n-k\right)$ we get from Lemma 3.9 that

$$\mathbb{P}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right) = \left(\frac{1}{2^{k-s}}\right)^{2}$$

Moreover, regardless of the values of $\mathbf{e}_{\text{aux}}^{(i)}$ and $\mathbf{z}_{\text{aux}}^{(i)}$ we have that $\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}$ and $\mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}$ are independent and uniformly distributed which yields that

$$\text{if}\quad \mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}\quad\text{then}\quad \mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) = \left(\frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}\right)^{2}.$$

Plugging this last equality back into $\mathbf{C}$ gives that

$$\textbf{If}\quad \mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}\quad\textbf{then}\quad \mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = 0.$$

Using this fact in Eq. (24) gives that

$$C = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_{w}^{n-s}}\ \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)}\in\mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i\in[\![1,\ b]\!]}}\ \sum_{\left(\mathbf{z}_{\text{aux}}^{(i)}\in\mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i\in[\![1,\ b]\!]}} \mathbf{1}_{\mathbf{e}_{\text{aux}}\neq\mathbf{z}_{\text{aux}}}\left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right| \tag{27}$$

2. Case $\mathbf{h}_{\mathscr{N}}=\mathbf{g}_{\mathscr{N}}$. Let us now compute $\left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right|$. Recall that from Eq. (26) we have that

$$\begin{aligned}
&\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \\
&= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+\left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+\left(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2}.
\end{aligned}$$

But as we have that

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right) \prod_{i=1}^{b} \mathbb{P}\left(\mathbf{e}_{\mathrm{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\mathrm{aux}}^{(i)},\ \mathbf{z}_{\mathrm{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\mathrm{aux}}^{(i)}\right) \\
&\leqslant \frac{1}{2^{k-s}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{1+\mathbf{1}_{\mathbf{e}_{\mathrm{aux}}^{(i)} \neq \mathbf{z}_{\mathrm{aux}}^{(i)}}} \\
&= \frac{1}{2^{k-k_{\mathrm{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\mathrm{aux}}^{(i)} \neq \mathbf{z}_{\mathrm{aux}}^{(i)}}}
\end{aligned}$$

This yield that

$$\left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\mathrm{aux}},\ \mathbf{z}_{\mathrm{aux}}\right)\right| = \mathcal{O}\left(\frac{1}{2^{k-k_{\mathrm{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\mathrm{aux}}^{(i)} \neq \mathbf{z}_{\mathrm{aux}}^{(i)}}}\right).$$

Finally,

$$\begin{aligned}
C &= \mathcal{O}\left(\sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \ \sum_{\left(\mathbf{e}_{\mathrm{aux}}^{(i)} \in \mathcal{S}^{s^{(i)}}_{t_{\mathrm{aux}}^{(i)}}\right)_{i \in [\![1,\ b]\!]}} \ \sum_{\left(\mathbf{z}_{\mathrm{aux}}^{(i)} \in \mathcal{S}^{s^{(i)}}_{t_{\mathrm{aux}}^{(i)}}\right)_{i \in [\![1,\ b]\!]}} \mathbf{1}_{\mathbf{e}_{\mathrm{aux}} \neq \mathbf{z}_{\mathrm{aux}}} \frac{1}{2^{k-k_{\mathrm{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\mathrm{aux}}^{(i)} \neq \mathbf{z}_{\mathrm{aux}}^{(i)}}}\right) \\
&= \mathcal{O}\left(\frac{\binom{n-s}{w} \prod_{i=1}^{b} \binom{s^{(i)}}{t_{\mathrm{aux}}^{(i)}}}{2^{k-k_{\mathrm{aux}}}} \sum_{\mathbf{c} \in \{0,1\}^b \,:\, \mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\mathrm{aux}}^{(i)}}}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{c_i}\right) \\
&= \mathcal{O}\left(N \sum_{\mathbf{c} \in \{0,1\}^b \,:\, \mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\mathrm{aux}}^{(i)}}}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{c_i}\right).
\end{aligned}$$

Plugging this last upper bound for C along with the expression of V in Eq. (23) in Eq. (22) allows writing

$$\mathbf{Var}\left(F_{\mathscr{H},\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right)\right) \leqslant N\left(1 + \mathcal{O}\left(\sum_{\mathbf{c} \in \{0,1\}^b \,:\, \mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\mathrm{aux}}^{(i)}}}{2^{s^{(i)}-k_{\mathrm{aux}}^{(i)}}}\right)^{c_i}\right)\right)$$

We can now prove our main lemma.

Proof (Proof of Lemma 10.1). The expected values are already given by Lemma 10.4, we only have to show that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right)\right) = \mathcal{O}\big(N\, n^{b+1}\ \max\left(1,\ N_{\mathrm{aux}}\right)\big).$$

Recall that from Lemma 10.4, we have that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right)\right) \leqslant N\left[1 + \sum_{\mathbf{c} \in \{0,1\}^b \,:\, \mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\mathrm{aux}}^{(i)}}}{2^{s^{(i)}-t_{\mathrm{aux}}^{(i)}}}\right)^{c_i}\right]. \qquad (28)$$

The result will come from the fact that, as for any i, j we have that

$$\begin{aligned}s^{(i)} &= s^{(j)} \pm 1,\\ k_{\text{aux}}^{(i)} &= k_{\text{aux}}^{(j)} \pm 1,\\ t_{\text{aux}}^{(i)} &= t_{\text{aux}}^{(i)} \pm 1,\end{aligned}$$

we can write that

$$\begin{aligned}\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}} &= \mathcal{O}\left(\frac{s^{(i)}+1}{t_{\text{aux}}^{(i)}+1}\frac{\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s^{(j)}-k_{\text{aux}}^{(j)}}}\right) && (29)\\ &= \mathcal{O}(n)\,\frac{\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s^{(j)}-k_{\text{aux}}^{(j)}}} && (30)\end{aligned}$$

The previous equation yields that, regardless of $\mathbf{c} \in \{0,1\}^b$ in Eq. (28) we have that

$$\begin{aligned}\prod_{i=1}^{b}\left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i} &= \mathcal{O}\left(n^b\ \max\left(1,\ \frac{\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s-k_{\text{aux}}}}\right)\right)\\ &= \mathcal{O}\big(n^b\ \max\left(1,\ N_{\text{aux}}\right)\big)\,.\end{aligned}$$

Plugging this last equation in Eq. (28) we get:

$$\begin{aligned}\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &\leqslant N\left[1+\sum_{\mathbf{c}\in\{0,1\}^b\,:\,\mathbf{c}\neq\mathbf{0}}\mathcal{O}\big(n^b\ \max\left(1,\ N_{\text{aux}}\right)\big)\right]\\ &\leqslant N\left[1+\ \mathcal{O}\big(n^{b+1}\ \max\left(1,\ N_{\text{aux}}\right)\big)\right]\\ &= \mathcal{O}\big(N\,n^{b+1}\ \max\left(1,\ N_{\text{aux}}\right)\big)\,.\end{aligned}$$

Which is our desired result.

10.2 Proof of the lemma comparing the biases

In this section we prove Lemma 8.3 and which we recall here.

Lemma 8.3. *There exists a positive poly-bounded function f_1 such that for any $s, w, u \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that* $\text{Root}\left(K_w^{(n-s)}\right) - u \in \Omega\left(n-s\right)$ *we have that*

$$\begin{aligned}\delta_w^{(n-s)}\left(u-1\right) - \delta_w^{(n-s)}\left(u\right) &\geqslant \frac{1}{f_1(n)}\delta_w^{(n-s)}\left(u\right),\\ \delta_w^{(n-s)}\left(u\right) - \delta_w^{(n-s)}\left(u+1\right) &\geqslant \frac{1}{f_1(n)}\delta_w^{(n-s)}\left(u\right).\end{aligned}$$

To prove it we use the following recurrence relations.

Lemma 10.5 (Difference of bias). *Let $n, w, t \in \mathbb{N}$, we have that*

$$\delta_w^{(n)}(t-1) - \delta_w^{(n)}(t) = 2\frac{w}{n}\delta_{w-1}^{(n-1)}(t-1).$$

Proof. Recall that by definition in Lemma 3.12

$$\delta_w^{(n)}(t) = \frac{K_w^{(n)}(t)}{\binom{n}{w}}.$$

First let us show the following recurrence relations

$$K_w^{(n)}(t-1) = K_w^{(n-1)}(t-1) + K_{w-1}^{(n-1)}(t-1), \tag{31}$$

$$K_w^{(n)}(t) = K_w^{(n-1)}(t-1) - K_{w-1}^{(n-1)}(t-1). \tag{32}$$

By Lemma 3.11 for any $\mathbf{x} \in \mathcal{S}_v^n$ we have

$$\begin{aligned} K_w^{(n)}(v) &= K_w^{(n)}(\mathbf{x}) \\ &= \sum_{\mathbf{h}\in\mathcal{S}_w^n} (-1)^{\langle \mathbf{h},\mathbf{x}\rangle}. \end{aligned}$$

By taking any $\mathbf{x}' \in \mathcal{S}_{t-1}^{n-1}$ and constructing $\mathbf{x} = (0 \mid\mid \mathbf{x}') \in \mathcal{S}_{t-1}^n$ we get, by decomposing the previous sum on the values of $\mathbf{h}$ on the first position, that

$$K_w^{(n)}(t-1) = K_w^{(n-1)}(t-1) + K_{w-1}^{(n-1)}(t-1).$$

Thus, Eq. (31) is showed. In the same manner, taking this time $\mathbf{x} = (1||\mathbf{x}') \in \mathcal{S}_t^n$ we get

$$K_w^{(n)}(t) = K_w^{(n-1)}(t-1) - K_{w-1}^{(n-1)}(t-1).$$

Thus, Eq. (32) is showed. Now, using the fact that

$$\binom{n}{w} = \frac{n}{w}\binom{n-1}{w-1}$$

and these recurrence relations we get

$$\begin{aligned} \delta_w^{(n)}(t-1) - \delta_w^{(n)}(t-1) &= \frac{2\,K_{w-1}^{(n-1)}(t-1)}{\binom{n}{w}} \\ &= 2\,\frac{w}{n}\frac{K_{w-1}^{(n-1)}(t-1)}{\binom{n-1}{w-1}}. \end{aligned}$$

The proof of Lemma 8.3 is now straightforward.

Proof (Proof of Lemma 8.3). Using the previous recurrence relations we have that

$$\delta_w^{(n-s)}(u-1) - \delta_w^{(n-s)}(u) = \delta_{w-1}^{(n-s-1)}(u-1).$$

Now, using Proposition 3.14, there exists a bivariate function κ such that

$$\delta_{w-1}^{(n-s-1)}(u-1) = \tilde{\Theta}\left(2^{\kappa((w-1)/(n-s-1),(u-1)/(n-s-1))n}\right).$$

By assumption in the lemma, we have that $u < \mathrm{Root}\left(K_w^{(n-s)}\right)$, this allows to show that the couple $((w-1)/(n-s-1),(u-1)/(n-s-1))$ belongs to $A \overset{\text{def}}{=} \{(\omega,\tau) \in [0,\ 1]^2 \ : \ \tau < 1/2 - \sqrt{\omega(1-\omega)}\ \}$. Using the fact, from Proposition 3.14 that κ is differentiable on A along with Taylor's theorem allows to conclude that $\delta_{w-1}^{(n-s-1)}(u-1) = \tilde{\Omega}\left(\delta_w^{(n-s)}(u)\right)$.