Towards Worst-case Hardness for Low-Noise LPN

Divesh Aggarwal* Rishav Gupta† Hai Hoang Nguyen‡ Kel Zin Tan§
Prashant Nalini Vasudevan¶

National University of Singapore

September 8, 2026

Abstract

The hardness of the Learning Parity with Noise (LPN) problem is a foundational assumption in cryptography, forming the basis of constructions ranging from symmetric-key primitives to public-key encryption and beyond. A central open question is whether the average-case hardness of LPN can be based on worst-case complexity assumptions, as has been achieved for the analogous Learning With Errors (LWE) problem.

Existing worst-case-to-average-case reductions for LPN [BLVW19, YZ21] rely on statistical smoothing of linear codes, which inherently limits the resulting average-case hardness to noise rates as large as $1/2 - 1/\mathrm{poly}(n)$, which is insufficient for public-key applications.

We explore a new approach towards obtaining such reductions: rather than requiring that random sparse combinations of the rows of the generator matrix of a code be *statistically* close to uniform, we only require that they be *computationally indistinguishable* from uniform. This leads to a clean win-win structure: we show that any efficient LPN solver can be transformed into a pair of efficient algorithms (S, D) such that for *every* matrix A of appropriate dimensions over $\mathbb{F}_2$, either S decodes the code generated by A from random noise, or D distinguishes random noisy codewords of the dual of this code from uniform.

By instantiating this reduction with appropriate parameters, we obtain the average-case hardness of LPN with inverse-polynomial noise rate $n^{-\alpha}$ for any constant $\alpha < 1$, assuming the worst-case simultaneous hardness of decoding a code from random noise and distinguishing random noisy codewords of its dual from uniform. In particular, setting $\alpha = 1/2$, our reduction yields LPN hardness in the parameter regime required for Alekhnovich's construction of public-key encryption [Ale03], a regime that was previously inaccessible via worst-case reductions.

* Email: divesh@comp.nus.edu.sg
† Email: rishavg@u.nus.edu
‡ Email: hai.h.nguyen@nus.edu.sg
§ Email: kelzin@u.nus.edu
¶ Email: prashvas@nus.edu.sg

1 Introduction

Learning Parity with Noise (LPN) [BFKL94] is essentially the problem of decoding random noisy codewords of a random linear code. An instance consists of a uniformly random matrix $A \leftarrow \mathbb{F}_2^{m \times n}$ and a vector $A \cdot s + e \in \mathbb{F}_2^m$, where $s \leftarrow \mathbb{F}_2^n$ is a uniformly random vector and $e \leftarrow Ber(\eta)^m$ is a vector of Bernoulli variables, where $\eta \in (0, 1/2)$ is called the *noise rate*. In the Search LPN problem, the task is to recover s given $(A, As + e)$. In the Decision LPN problem, the task is to distinguish this distribution from (A, b), where b is an independent uniformly random vector in $\mathbb{F}_2^m$. These variants are known to be equivalent [AIK07, MM11].

The LPN problem has been of significant consequence to cryptography since its first consideration by Blum et al. [BFKL94]. It is one of the few known sources of secure Public-Key Encryption (PKE) [Ale03, YZ16], and has been used to construct various advanced cryptographic primitives (see, for instance, the survey by Pietrzak [Pie12]). It may be interpreted as the problem of solving a system of noisy linear equations, and for large enough η, worst-case versions of this problem are known to be NP-hard [AK11]. The best algorithms we have for LPN run in time $2^{O(\eta \cdot n)}$ for any non-trivial m [Pra62, Ste89, Dum91, MMT11, BJMM12, MO15, BM17, BM18, DEEK24], or in time $2^{O(n/\log n)}$ if $m > 2^{\Omega(n/\log n)}$ [BKW03], or $2^{O(n/\log\log n)}$ if $m > n^{1+\Omega(1)}$ [Lyu05].

In many ways, LPN is similar to its analogue in the Euclidean metric, the Learning With Errors (LWE) problem [Reg09]. One notable difference, however, is that the (average-case) hardness of the LWE problem is supported by the worst-case hardness of lattice problems [Reg09, Pei09], leading to a construction of PKE and other primitives from the worst-case hardness of these problems. With LPN, the worst-case to average-case reductions we have are much more limited.

Known Reductions for LPN. The only known method for reducing from the plausible worst-case hardness of any problem to LPN is that of Brakerski et al. [BLVW19]. Their reduction is from a worst-case version of the LPN problem itself, called the Nearest Codeword Problem (NCP). Here, given a matrix $A \in \mathbb{F}_2^{m \times n}$ and vector $As + e \in \mathbb{F}_2^m$ for some e guaranteed to have Hamming weight at most $\eta \cdot m$, the problem is to recover s. They show that the worst-case hardness of NCP (with some minor restrictions) with noise rate $\eta = O(\log^2 n/n)$ implies the average-case hardness of LPN with noise rate $(1/2 - 1/\mathrm{poly}(n))$.

With additional careful analysis, Yu and Zhang [YZ21] extend this reduction to sub-exponential-time algorithms. For example, they show that if NCP with noise rate $\eta = O(n^{-1/2})$ and $m = 2^{\Omega(n^{1/2})}$ is worst-case hard for algorithms running in time $2^{O(n^{1/2})}$, then LPN with constant noise-rate and similar values of m is also hard for such algorithms. They also point out that the hardness of LWE with certain noise parameters implies hardness of an analogue of LPN over large fields with noise rate close to 1.

Apart from worst-case-to-average-case reductions, hardness amplification results are also known for LPN [AGZ26]. Roughly speaking, these results show that an algorithm solving LPN with secret length kn and noise rate $k\eta$ with success probability ε can be used to solve LPN with secret length n and noise rate η with success probability at least $1 - \delta$, where $k = \Theta\left(\frac{1}{\delta} \log \frac{1}{\varepsilon}\right)$.

Limits of Known Reductions. The aforementioned results are already quite remarkable as they imply the possibility of symmetric-key cryptography based on worst-case hardness assumptions. Unfortunately, they fall well short of providing the hardness of LPN required for public-key applications, which is hardness at noise rate $O(n^{-1/2})$ [Ale03] against polynomial-time algorithms,

or at constant noise rate against $2^{\omega(n^{1/2})}$-time algorithms [YZ16].

Further, the worst-case hardness assumptions in their hypotheses are quite strong. Recall that [BLVW19] requires hardness of NCP with a noise rate of $O(\log^2 n/n)$. This is only slightly larger than $O(\log n/n)$, at which rate NCP can be solved in polynomial time [Pra62]. Further, if arbitrary pre-processing of the matrix A is allowed, even NCP with noise rate $O(\log^2 n/n)$ can be solved in polynomial time [BCLV26].

The source of these limitations is as follows. The core of their reduction is, given an arbitrary matrix A, to consider the vector $r^T A$, where $r \in \mathbb{F}_2^m$ is a random vector with Hamming weight w. Clearly, if $w > n$ and A is sufficiently non-degenerate, then $r^T A$ will be close to being uniformly random. They show that this is true even if $w = O(n/\log n)$ and m is a large enough polynomial in n. Then, given an NCP instance $(A, As + e)$, they generate the rows of their LPN instance as $(r^T A, r^T(As + e))$. The first part $r^T A$ is uniform as required, and the second part is $(r^T A)s + r^T e$, with $r^T e$ playing the part of the noise. This phenomenon of the distribution of $r^T A$ being close to uniform for an appropriate distribution of r is referred to as *smoothing*,[1] and has been studied in detail for this and other distributions of r, and in various metrics [BLVW19, YZ21, DDRT23, DR25, PB25].

This method of generating the instance causes the noise rate to increase. If e had noise rate η, then $r^T e$ will have noise rate $\alpha \approx 1/2 - e^{-2\eta w}$. This α has to be non-negligibly removed from $1/2$ for the problem to be meaningful. So if w is set to $\Omega(n/\log n)$, then only noise rates η between $\omega(\log n/n)$ and $O(\log^2 n/n)$ are useful, and the resulting α will never be smaller than $1/2 - o(1)$.

One way to try to deal with this issue is to set w to be something smaller. However, it is easy to check that if $w = o(n/\log n)$, then for any m that is polynomial in n, a random vector of Hamming weight w does not have enough entropy for the distribution of $r^T A \in \mathbb{F}_2^n$ to be close to uniform. This remains true for other natural distributions such as Bernoulli vectors with expected Hamming weight w.

Another possibility is that instead of using the same distribution for every matrix A, one could try to use a distribution of r specifically tailored to the given A such that $r^T A$ will be uniform. However, without bounds on the Hamming weight of r, it is unclear that $r^T e$ will remain unbiased for all possible noise vectors e. In fact, it was recently shown by Pathegama and Barg [PB25] that, at least if $m = O(n)$ and the noise rate is a constant, there are matrices $A \in \mathbb{F}_2^{m \times n}$ for which there is no such distribution of r that will satisfy both conditions: $r^T A$ is sufficiently close to uniform, and $r^T e$ has noise rate $1/2 - 1/\mathrm{poly}(n)$. Though not explicitly stated, their result also applies when e is a random Bernoulli vector. This still leaves open the possibility of this approach working if m is much larger, or if we start with hardness of NCP with a smaller noise rate, but pursuing it will require significantly new techniques.

1.1 Our Results

Our proposal to improve on existing reductions is somewhat different, and is to weaken the notion of smoothing used – instead of asking that $r^T A$ be statistically close to uniform, we only ask that it be *computationally indistinguishable* from uniform. Interestingly, this again relates to decoding a linear code – if we view A^T as the parity-check matrix of its dual code, then $A^T r$ may be regarded as the syndrome of r. If B generates the dual code of A, then multiplying a noisy dual codeword $Bs + r$ by

[1] Strictly speaking, "smoothing" actually refers to a dual phenomenon that this is known to be equivalent to [PB25].

A^T results in the codeword part being cancelled out, giving us: $A^T(Bs+r) = (A^TB)s+A^Tr = A^Tr$. So recovering r given r^TA is equivalent to decoding the dual code of A with r as the noise vector, and distinguishing r^TA from uniform corresponds to solving a decision version of the decoding problem for the dual code of A.

This now leads to a win-win situation. If r^TA is computationally indistinguishable from uniform, then the existing smoothing-based reduction approach can be used to generate *pseudorandom* LPN samples of the form $(r^TA, r^T(As + e))$, which can then be given to an LPN solver. Since this distribution is pseudorandom, the LPN solver will continue to work and find the secret s. On the other hand, if r^TA is *not* pseudorandom, then the distinguisher that breaks pseudorandomness is essentially solving a decision version of the decoding problem for the dual code of A.

Following this observation, we show that the hardness of LPN can be derived from the worst-case simultaneous hardness of decoding both a code and its dual from random noise. An informal version of this reduction is described below. We refer to each row of $(A, As + e)$ as a *sample* of LPN, and each bit of the secret s as a *variable*.

Theorem 1.1 (Informal, see Theorem 3.1 and Corollary 4.1)**.** *Suppose there is an efficient algorithm that has non-negligible advantage in solving LPN with n variables, t samples, and noise rate $n^{-\alpha}$ for some constant $\alpha < 1$. Then, for any constant $\beta > 0$ such that $\alpha+\beta < 1$, and $m = O(t^2n^{2\beta})$, there are efficient algorithms S and D such that, for any $A \in \mathbb{F}_2^{m\times n}$, either S can decode the code generated by A from random noise of rate $n^{-(\alpha+\beta)}$, or D can distinguish noisy codewords of the dual of this code from random for noise rate n^β/m.*

To the best of our knowledge, the complexity of decoding a generic linear code and decoding its dual are not known to be negatively correlated. So we expect the complexity of the task specified in the theorem to be the minimum of the worst-case complexity of an algorithm that can decode any code generated by an $m \times n$ matrix from random noise of rate $n^{-(\alpha+\beta)}$ and that of an algorithm that can decode any code generated by an $m \times (m - n)$ matrix from random noise of rate n^β/m. As long as α and β are constants such that $\alpha + \beta < 1$, the best known algorithms for both of these tasks run in sub-exponential time.

Public-Key Cryptography. The most remarkable consequence of our reduction is that, making reasonable assumptions about the hardness of decoding worst-case codes and their duals from random noise, we can get the hardness of LPN required for constructions of Public-Key Encryption using the Alekhnovich paradigm [Ale03]. In particular, this requires hardness for LPN with $t = 2n$ samples and noise rate $n^{-1/2}$, which can be obtained by setting the parameters in the theorem to, for instance, $\alpha = 0.5$, $\beta = 0.4$, and $m = O(n^{2.8})$.

Challenges. While the observations stated earlier that lead to our reduction are relatively straightforward, realising them fully turns out to be quite challenging. The primary obstacle here is that computational randomness is much harder to obtain and use than actual randomness. One detail that we have been glossing over in our discussion so far is that for the smoothing reduction approach to work, it is not sufficient to show that r^TA is close to uniform – one also needs to show that the joint distribution of (r^TA, r^Te) is close to $(U_n, Ber(\alpha))$, where U_n is the uniform distribution over n bits and α is the noise rate of LPN one is reducing to. In the statistical setting of [BLVW19, YZ21], the Fourier analysis they use can easily show this using nearly the same argument that shows that r^TA is close to uniform.

In our computational setting, however, we do not have powerful tools like Fourier analysis that can readily incorporate this additional biased bit. So even if we are promised that $r^T A$ is pseudo-random, and we also know that $r^T e$ is a Bernoulli variable with the appropriate parameter α, it is non-trivial to argue that $(r^T A, r^T e)$ is computationally indistinguishable from $(U_n, Ber(\alpha))$. This is because the leakage $r^T e$ could lead to $r^T A$ becoming slightly distinguishable from uniform, and this could add up over many such samples to render the resulting distribution easily distinguishable from an LPN instance.

It is due to this issue that our reduction only produces an S that decodes from random noise, whereas the earlier smoothing reductions could produce algorithms that can decode from worst-case noise. We show that for most choices of the set of vectors r chosen while constructing the LPN instance, even after fixing the r's, the randomness in e ensures that the set of bits $r^T e$ are independent Bernoulli variables. This also requires us to use a different error model for the dual code – the r's are chosen to be vectors of some fixed Hamming weight, rather than as Bernoulli vectors (see Theorem 3.1).

Nevertheless, we believe that our reduction should work even with r being a Bernoulli vector and e being any worst-case error vector of bounded Hamming weight. Proving that it works seems quite challenging, however, and will likely require the development of new analytical tools for computational entropy.

1.2 Future Directions

Our work opens up several natural directions for further investigation.

More standard worst-case foundations. The worst-case assumption that we need, the simultaneous hardness of decoding a code and distinguishing syndromes of its dual from uniform, is non-standard and, to our knowledge, has not been independently studied. An important next step is to determine whether our approach can be instantiated from more widely studied worst-case assumptions, such as the hardness of the Nearest Codeword Problem (NCP) alone, or classical problems like the Minimum Distance Problem (MDP). More broadly, one would like to understand whether the joint hardness of a code and its dual is inherently needed to improve the smoothing approach to reductions, or whether there is some way around it.

Worst-case noise. Our reduction produces a decoder S that works against *random* Bernoulli noise, whereas the smoothing-based reductions of Brakerski et al. [BLVW19] and Yu–Zhang [YZ21] yield decoders for *worst-case* bounded-weight noise. Extending our approach to worst-case noise would significantly strengthen the result, but appears to require new tools for reasoning about computational entropy in the presence of adversarially chosen error vectors. We believe this is a tractable and important challenge.

1.3 Paper Outline

In Section 2, we recall the notation and technical tools used throughout the paper. In Section 3, we present our main reduction. In Section 4, we instantiate our reduction with suitable parameters to derive useful hardness regimes for LPN.

2 Preliminaries

Notation We use x_i to denote the i-th bit of the binary string (vector) x and $A[i]$ as the i-th row of the matrix A. Denote $D_1 \equiv D_2$ if the distributions D_1 and D_2 are the same. The notation $x \leftarrow D$ means that x is sampled from the distribution D; when D is a set (we abuse notation), it means x is sampled uniformly from the set. Let $\mathbb{F}_p$ denote the order-p prime field. $\mathrm{Ber}(\eta)$ is the Bernoulli distribution with parameter η. Write $[n] \coloneqq \{1, 2, \ldots, n\}$. The notation $\binom{[n]}{s}$ denotes the set of all vectors in $\mathbb{F}_2^n$ with Hamming weight s. Next, $\mathrm{wt}(v)$ denotes the Hamming weight of $v \in \mathbb{F}_2^n$, and $\mathrm{Supp}(v) \coloneqq \{i \in [n] \mid v_i \neq 0\}$ is the support of the vector.

Fact 2.1. (Piling Up Lemma). *Let $X_1, X_2, \ldots, X_k$ be independent random variables where $X_i \leftarrow \mathsf{Ber}(\eta)$, then the random variable $S = X_1 + X_2 + \cdots + X_k$, where addition is over $\mathbb{F}_2$, follows the distribution* $\mathsf{Ber}\left(\dfrac{1-(1-2\eta)^k}{2}\right)$.

Proof. See Section A. □

We now define the set of vectors with fixed Hamming weight.

Definition 2.2 (Fixed weight vector)**.** *Denote $\mathcal{W}(n, w)$ as the set of vector $v \in \mathbb{F}_2^n$ with $\mathrm{wt}(v) = w$.*

Next, we formally define the Learning Parity with Noise problem (LPN).

Definition 2.3 (Learning Parity with Noise (LPN))**.** *Let $n, m \in \mathbb{Z}$, and $\eta \in [0, 1]$. For sample count m, secret dimension n, and noise parameter η, algorithms S_{LPN} and D_{LPN} are said to have advantage $\varepsilon \in [0, 1]$ in solving the* Search LPN *and* Decision LPN *problems, respectively, if they satisfy the following conditions:*

- Search LPN: S_{LPN} *recovers the secret s with probability at least ε:*

$$\Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ s \leftarrow \mathbb{F}_2^n \\ e \leftarrow \mathrm{Ber}(\eta)^m}} [\mathsf{S}_{LPN}(A, As + e) = s] \geq \varepsilon.$$

- Decision LPN: D_{LPN} *distinguishes LPN samples from uniform with advantage at least ε:*

$$\left| \Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ s \leftarrow \mathbb{F}_2^n \\ e \leftarrow \mathrm{Ber}(\eta)^m}} [\mathsf{D}_{LPN}(A, As + e) = 1] - \Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ b \leftarrow \mathbb{F}_2^m}} [\mathsf{D}_{LPN}(A, b) = 1] \right| \geq \varepsilon.$$

We now define the Fixed Weight Syndrome Decoding problem SD, which can be viewed as a slight variation of the dual form of LPN.

Definition 2.4 (Fixed Weight Random Syndrome Decoding (SD))**.** *Let $n, m \in \mathbb{Z}$, and $\theta \in [0, 1]$. For code length m, code dimension n, and noise parameter θ, algorithms S_{SD} and D_{SD} are said to have advantage $\varepsilon \in [0, 1]$ in solving the* Search SD *and* Decision SD *problems, respectively, if they satisfy the following conditions:*

- Search SD: S_{SD} *decodes a syndrome with probability at least* ε*:*

$$\Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ r \leftarrow \mathcal{W}(m, \theta \cdot m)}} [\mathsf{S}_{SD}(A, r^T A) = r] \geq \varepsilon.$$

- Decision SD: D_{SD} *distinguishes a syndrome from uniform with advantage at least* ε*:*

$$\left| \Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ r \leftarrow \mathcal{W}(m, \theta \cdot m)}} [\mathsf{D}_{SD}(A, r^T A) = 1] - \Pr_{\substack{A \leftarrow \mathbb{F}_2^{m \times n} \\ c \leftarrow \mathbb{F}_2^n}} [\mathsf{D}_{SD}(A, c^T) = 1] \right| \geq \varepsilon.$$

Remark 2.5. Our definition of Syndrome Decoding Problem deviates from the more natural dual to LPN in that the distribution of r is a randomly sampled fixed weight vector rather than a Bernoulli vector.

3 Average-Case Reduction

Our main contribution is the following theorem, which shows that for every worst-case code generated by a matrix A, an LPN solver yields one of two consequences: either it can be used to solve a random decoding instance associated with A, or it can be used to distinguish fixed-weight random syndrome decoding instance with A^T as the parity-check matrix from a random string.

Theorem 3.1. *Consider* $n, m, t \in \mathbb{Z}$ *and* $\eta, \theta \in (0, 1)$ *such that* $t^2\theta^2 m < 1$*. Suppose there exists an algorithm for Search LPN with sample count* t*, secret dimension* n*, and noise parameter* $\tau = (1 - (1 - 2\eta)^{\theta \cdot m})/2$*, which runs in time* T *and has advantage* ε*. Then there exist algorithms* D *and* S *such that for every matrix* $A \in \mathbb{F}_2^{m \times n}$*, at least one of the following is true:*

- S *runs in time* $\text{poly}\left(n, m, T, (1/\varepsilon)^{\log_m(m/\varepsilon)}\right)$ *and decodes codewords of* A *from random noise such that,*

$$\Pr_{\substack{s \leftarrow \mathbb{F}_2^n \\ e \leftarrow Ber(\eta)^m}} [\mathsf{S}(A, As + e) = s' \text{ and } \text{wt}(As' - (As + e)) \leq 2\eta m] \geq \varepsilon/2 - o(\varepsilon).$$

- D *runs in time* $\text{poly}(n, m) + T$ *and distinguishes a sparse combination of rows of* A *from a uniform string such that,*

$$\Pr_{c \leftarrow \mathbb{F}_2^n} [\mathsf{D}(A, c^T) = 1] - \Pr_{r \leftarrow \mathcal{W}(m, \theta \cdot m)} [\mathsf{D}(A, r^T A) = 1] \geq \varepsilon/2t.$$

The rest of this section constitutes the proof of Theorem 3.1. Fix any set of parameters n, m, t, η, θ, and τ that satisfy the conditions in the theorem statement. Suppose there is an algorithm S_{LPN} for Search LPN that has running time T and advantage ε as specified in the theorem. The main idea in the reduction is to check how good S_{LPN} is at decoding a code whose generator matrix is derived by randomly combining rows of a given matrix $A \in \mathbb{F}_2^{m \times n}$, and based on this, using it to either decode the code generated by A, or the code whose parity check matrix is A.

For any $A \in \mathbb{F}_2^{m \times n}$, let $Adv(A)$ be the advantage of $\mathsf{S}_{\mathrm{LPN}}$ at decoding a noisy codeword of the code generated by the matrix $R \cdot A \in \mathbb{F}_2^{t \times n}$, where R is a $t \times m$ matrix and each row of R is an independently sampled random vector of Hamming weight $\theta \cdot m$, written as $R \leftarrow \mathcal{W}(m, \theta \cdot m)^t$:

$$Adv(A) = \Pr_{\substack{s \leftarrow \mathbb{F}_2^n \\ e \leftarrow \mathrm{Ber}(\tau)^t \\ R \leftarrow \mathcal{W}(m, \theta \cdot m)^t}} [\mathsf{S}_{\mathrm{LPN}}(RA, RAs + e) = s] \tag{1}$$

We can then split into two cases. If $Adv(A)$ is sufficiently large (i.e. $Adv(A) \geq \varepsilon/2$), then $\mathsf{S}_{\mathrm{LPN}}$ can be used to construct an algorithm that decodes a noisy codeword $As + e$ of A by running $\mathsf{S}_{\mathrm{LPN}}$ on $(RA, R(As + e))$. On the other hand, if $Adv(A)$ is sufficiently smaller than ε (i.e. $Adv(A) < \varepsilon/2$), then this means RA does not look random to $\mathsf{S}_{\mathrm{LPN}}$, and we can use a hybrid argument to distinguish the syndrome $r^T A$ from uniform when $r \leftarrow \mathcal{W}(m, \theta \cdot m)$.

We next state the lemmas capturing the properties of the two resulting algorithms. Note that these lemmas are stated under the assumption that the algorithm $\mathsf{S}_{\mathrm{LPN}}$ as described above exists. The algorithms themselves and the proofs of these lemmas are presented in Sections 3.1 and 3.2. Together, these lemmas directly imply Theorem 3.1.

Lemma 3.2 (Large Advantage)**.** *There exists an algorithm* S *that, for any* $A \in \mathbb{F}_2^{m \times n}$ *such that* $Adv(A) \geq \varepsilon/2$*, decodes the code generated by* A *from random Bernoulli noise of rate* η*, with success probability at least* $\varepsilon/2 - o(\varepsilon)$*. More precisely,*

$$\Pr_{\substack{s \leftarrow \mathbb{F}_2^n \\ e \leftarrow Ber(\eta)^m}} [\mathsf{S}(A, As + e) = s' \text{ and } \mathrm{wt}(As' - (As + e)) \leq 2\eta m] \geq \varepsilon/2 - o(\varepsilon).$$

Further, S *runs in time* $\mathrm{poly}\left(n, m, T, (1/\varepsilon)^{\log_m(m/\varepsilon)}\right)$.

Lemma 3.3 (Small Advantage)**.** *There exists an algorithm* D *that, for any* $A \in \mathbb{F}_2^{m \times n}$ *such that* $Adv(A) < \varepsilon/2$*, distinguishes between random syndromes of the code with parity check matrix* A *and the uniform distribution with advantage at least* $\varepsilon/2t$*. More precisely,*

$$\Pr_{c \leftarrow \mathbb{F}_2^n}[\mathsf{D}(A, c^T) = 1] - \Pr_{r \leftarrow \mathcal{W}(m, \theta \cdot m)}[\mathsf{D}(A, r^T A) = 1] \geq \varepsilon/2t.$$

Further, D *runs in time* $T + \mathrm{poly}(n, m)$.

3.1 Large Advantage

In this section, we prove Lemma 3.2, about the ability of an algorithm S to decode the code generated by A when $Adv(A) \geq \varepsilon/2$.

Lemma 3.2 (Large Advantage)**.** *There exists an algorithm* S *that, for any* $A \in \mathbb{F}_2^{m \times n}$ *such that* $Adv(A) \geq \varepsilon/2$*, decodes the code generated by* A *from random Bernoulli noise of rate* η*, with success probability at least* $\varepsilon/2 - o(\varepsilon)$*. More precisely,*

$$\Pr_{\substack{s \leftarrow \mathbb{F}_2^n \\ e \leftarrow Ber(\eta)^m}} [\mathsf{S}(A, As + e) = s' \text{ and } \mathrm{wt}(As' - (As + e)) \leq 2\eta m] \geq \varepsilon/2 - o(\varepsilon).$$

Further, S *runs in time* $\mathrm{poly}\left(n, m, T, (1/\varepsilon)^{\log_m(m/\varepsilon)}\right)$.

Algorithm 1 Algorithm $\mathsf{S}(A \in \mathbb{F}_2^{m \times n}, b \in \mathbb{F}_2^m)$

1: Sample matrix $R \leftarrow \mathcal{W}(m, \theta \cdot m)^t$. ▷ Smoothing matrix
2: Set threshold $= \max(8 \log_m(m/\varepsilon) \log(1/\varepsilon), 12)$. ▷ Threshold
3: Set $C = \{j \in [t] \mid \exists i < j : \mathrm{Supp}(R[i]) \cap \mathrm{Supp}(R[j]) \neq \emptyset\}$. ▷ Collision Set
4: Convert C into a list with arbitrary ordering.
5: **if** $|C| \geq$ threshold **then**
6: **return** $\perp$. ▷ Abort
7: **for** each $v \in \mathbb{F}_2^{|C|}$ **do**
8: Set $b' \leftarrow Rb$.
9: **for** each j in C **do** ▷ Iterate through collided rows
10: Replace $b'_j \leftarrow (v_j + \mathsf{Ber}(\tau))$ ▷ Recall that $\tau = (1 - (1-2\eta)^{\theta \cdot m})/2$
11: Query $s' \leftarrow \mathsf{S}_{\mathrm{LPN}}(RA, b')$. ▷ Query oracle
12: **if** $\mathrm{wt}(b - As') \leq 2\eta m$ **then** ▷ Verify answer
13: **return** s'.
14: **return** $\perp$

The algorithm S that proves the lemma is described below, followed by the proof of the lemma. Here, $\mathsf{S}_{\mathrm{LPN}}$ is the assumed search algorithm for LPN that has running time T and advantange ε.

Proof of Lemma 3.2. Fix an A such that $Adv(A) \geq \varepsilon/2$. Recall that this means that $\mathsf{S}_{\mathrm{LPN}}$ has $\varepsilon/2$ probability of successfully recovering s when given a sample from the distribution $(RA, RAs + \mathsf{Ber}(\tau)^t)$, with $\tau = (1 - (1-2\eta)^{\theta \cdot m})/2$. To show the correctness of the algorithm, it suffices to prove that in at least one iteration, the input (RA, b') provided to $\mathsf{S}_{\mathrm{LPN}}$ in line 11 has the same distribution as $(RA, RAs + \mathsf{Ber}(\tau)^t)$, and also that the algorithm does not prematurely terminate at line 6.

In the following claim, we show that the algorithm does not abort in line 6 with high probability.

Claim 3.4. *The probability that* $|C| \geq \text{threshold} = \max(8 \log_m(\frac{m}{\varepsilon}) \log(\frac{1}{\varepsilon}), 12)$ *is at most* $o(\varepsilon)$. *Formally,*

$$\Pr\left[|C| \geq \max(8 \log_m(m/\varepsilon) \log(1/\varepsilon), 12)\right] \leq o(\varepsilon).$$

Proof. Using union bound, we first upperbound the probability that a column in $R \leftarrow \mathcal{W}(m, \theta \cdot m)^t$ has at least k bits that are 1. Let $P_i \in \mathbb{F}_2^t$ be the i-th column vector of R. For any $i \in [m]$,

$$\Pr\left[\mathrm{wt}(P_i) \geq k\right] \leq \binom{t}{k} (\frac{\theta m}{m})^k \leq (\frac{et\theta}{k})^k. \tag{2}$$

Since $t^2\theta^2 m < 1$, which implies $t\theta < 1/\sqrt{m}$. We can conveniently set $k = 4 \log_m(m/\varepsilon)$ and do a union bound to say that with high probability $1 - o(\varepsilon)$, no columns will have more than k non-zero bits. Formally,

$$\Pr\left[\exists i \in [m], \mathrm{wt}(P_i) \geq 4 \log_m(m/\varepsilon)\right] \leq m(\frac{e}{\sqrt{m}})^{4 \log_m(m/\varepsilon)} = o(\varepsilon). \tag{3}$$

Setting $k = O(\log_m(m/\varepsilon))$ is essentially having k being a constant when $\varepsilon = 1/\text{poly}(n)$. Next, we show that with high probability $1-o(\varepsilon)$, there are at most $h = 2\log(1/\varepsilon)$ columns that have weight at least 2. This is done again by using union bound over all possible h subsets of columns,

$$\Pr\left[\exists S \in \binom{[m]}{h}, \forall j \in S,\, \text{wt}(P_j) \geq 2\right] \leq \binom{m}{h} \Pr\left[\forall j \in [h],\, \text{wt}(P_j) \geq 2\right]. \tag{4}$$

Since each row is a fixed weight vector, if a column happens to have a weight ≥ 2, conditioned on this event, the probability of the next column having weight ≥ 2 is smaller. This negative correlation property implies,

$$\Pr\left[\forall j \in [h],\, \text{wt}(P_j) \geq 2\right] \leq \Pr\left[\text{wt}(P_1) \geq 2\right]^h \leq (\frac{t^2\theta^2}{2})^h. \tag{5}$$

Therefore, combining Eq. (4) and Eq. (5), gives us an upperbound

$$\Pr[\exists \text{ at least } h \text{ columns with weight} \geq 2] \leq \binom{m}{h} (\frac{t^2\theta^2}{2})^h \leq (\frac{emt^2\theta^2}{2h})^h. \tag{6}$$

Since $mt^2\theta^2 < 1$. If $\varepsilon \leq 1/2$, we can plug in $h = 2\log(1/\varepsilon)$ to get

$$(\frac{emt^2\theta^2}{2h})^h \leq (\frac{e}{4})^{2\log(1/\varepsilon)} = o(\varepsilon)$$

Otherwise, if $\varepsilon > 1/2$, we can plug in $h = 3$ to get

$$(\frac{emt^2\theta^2}{2h})^h \leq (\frac{e}{6})^3 = o(\varepsilon)$$

Therefore, if $h = \max(2\log(1/\varepsilon), 3)$, the probability in Eq. (6) is upper bounded by $o(\varepsilon)$. In summary, probability bounds we have on the two bad events are,

- The probability that there exists a column in R with more than k non zero entries where, $k = 4\log_m(m/\varepsilon)$ is at most $o(\varepsilon)$ (Eq. (3)).
- The probability that there exists $h = \max(2\log(1/\varepsilon), 3)$ columns in R with weight ≥ 2 is at most $o(\varepsilon)$ (Eq. (6)).

By union bound, with probability $1 - o(\varepsilon)$, neither of the bad events happens. In that scenario, the maximum size of $|C|$ is upper-bounded as follows,

$$|C| < k \cdot h = \max(8\log_m(m/\varepsilon)\log(1/\varepsilon), 12\log_m(m/\varepsilon)).$$

Because the second value in the maximum expression happens only when $\varepsilon > 1/2$, that also means $\log_m(m/\varepsilon) \leq 1$. Therefore, the statement can be further simplified

$$|C| < k \cdot h = \max(8\log_m(m/\varepsilon)\log(1/\varepsilon), 12)$$

Hence, the probability of the abort in line 6, is at most $o(\varepsilon)$. □

Now, assume that abort does not occur in line 6, we will show that in that case at least one iteration, the input (RA, b') provided to $\mathsf{S}_{\mathrm{LPN}}$ in line 11 has the same distribution as $(RA, RAs + \mathsf{Ber}(\tau)^t)$.

By definition of $C = \{j \in [t] \mid \exists i < j : \mathrm{Supp}(R[i]) \cap \mathrm{Supp}(R[j]) \neq \emptyset\}$, we get that for all distinct $p, q \in [t] \setminus C$,

$$\mathrm{Supp}(R[p]) \cap \mathrm{Supp}(R[q]) = \emptyset.$$

We can therefore split the smoothing matrix R into two parts: 'touched' submatrix $R_t \in \mathbb{F}_2^{|C| \times m}$, consisting of the rows indexed by the elements of C and 'untouched' submatrix $R_{ut} \in \mathbb{F}_2^{(t-|C|) \times m}$, consisting of the rows not indexed by the elements of C. Formally, viewing both C and $[t] \setminus C$ as lists with arbitrary but fixed orderings, we define

$$R_t[i] := R[C[i]] \quad \text{for every } i \in [|C|],$$

and

$$R_{ut}[i] := R[([t] \setminus C)[i]] \quad \text{for every } i \in [t - |C|].$$

To simplify the argument, ignore the ordering of the rows in R_{ut} and R_t as we can always combine them back to R and maintain the original ordering.

For the rows represented by R_{ut}, because the rows' supports are disjoint, each bit of $R_{ut} \cdot e$, where $e \leftarrow \mathsf{Ber}(\eta)^m$, is independently and identically distributed. The distribution of $(R_{ut}e)_i$ is exactly the sum of $\theta \cdot m$ independent and identically distributed $\mathsf{Ber}(\eta)$ random variables. By Fact 2.1, this is $(R_{ut}e)_i \equiv \mathsf{Ber}(\tau)$ where $\tau = (1 - (1 - 2\eta)^{\theta \cdot m})/2$ and implies that $(R_{ut}, R_{ut}e) \equiv (R_{ut}, \mathsf{Ber}(\tau)^{t-|C|})$. Then

$$(R_{ut}A, R_{ut}b) = (R_{ut}A, R_{ut}As + R_{ut}e) \equiv (R_{ut}A, R_{ut}As + \mathsf{Ber}(\tau)^{t-|C|}).$$

For the rows represented by R_t, since we are enumerating all possible $\mathbb{F}_2^{|C|}$ assignments. There will be one assignment that is exactly the value of $v = R_tAs$, and in this case the replacement procedure in line 10 sets the corresponding part of b' to exactly $R_tAs + \mathsf{Ber}(\tau)^{|C|}$.

Finally, we combine R_t and R_{ut} back to R respecting the original ordering. Assuming that we are in the iteration of getting a correct assignment, $v = R_tAs$, the bits in $R_tb = R_tAs + R_te$ are replaced with $R_tAs + \mathsf{Ber}(\tau)^{|C|}$. This does not leak any information about $R_{ut}e$, and the query to $\mathsf{S}_{\mathrm{LPN}}$ is exactly the distribution $(RA, RAs + \mathsf{Ber}(\tau)^t)$. As $\mathsf{S}_{\mathrm{LPN}}$ has $\varepsilon/2$ probability of success on this distribution and the probability of abort in line 6 is at most $o(\varepsilon)$, with probability at least $\varepsilon/2 - o(\varepsilon)$, this execution $\mathsf{S}_{\mathrm{LPN}}(RA, b')$ will return $s' = s$. For the verification of the secret s in line 12, having the correct candidate secret $s' = s$, the probability that error vector $b - As'$ has Hamming weight greater than $2\eta m$ is exponentially small, and so the verification will fail with negligible probability. So the algorithm S succeeds with probability at least $\varepsilon/2 - o(\varepsilon)$.

The remaining possibility is that the algorithm $\mathsf{S}_{\mathrm{LPN}}(RA, b')$ returns a candidate s' that is not equal to s in some other iteration (i.e $v \neq R_tAs$). This is still fine because s' must have the property that $\mathrm{wt}(b - As') \leq 2\eta m$ to pass the verification. Therefore, S recovers a secret candidate s' from noisy codeword samples of A with advantage at least $\varepsilon/2 - o(\varepsilon)$.

The runtime of the algorithm is determined by the size of the matrix $A \in \mathbb{F}_2^{m \times n}$ and the number of queries called to $\mathsf{S}_{\mathrm{LPN}}$. Since $|C|$ is upper bounded by $\max(8 \log_m(m/\varepsilon) \log(1/\varepsilon), 12)$, the number of iterations is $O(2^{\log_m(m/\varepsilon) \log(1/\varepsilon)}) = O((1/\varepsilon)^{\log_m(m/\varepsilon)})$ .Therefore, the runtime of S is $\mathrm{poly}\left(n, m, T, (1/\varepsilon)^{\log_m(m/\varepsilon)}\right)$ □

3.2 Small Advantage

In this section, we prove Lemma 3.3, about the ability of an algorithm D to distinguish random syndromes from uniform when $Adv(A) < \varepsilon/2$.

Lemma 3.3 (Small Advantage)**.** *There exists an algorithm D that, for any $A \in \mathbb{F}_2^{m\times n}$ such that $Adv(A) < \varepsilon/2$, distinguishes between random syndromes of the code with parity check matrix A and the uniform distribution with advantage at least $\varepsilon/2t$. More precisely,*

$$\Pr_{c\leftarrow\mathbb{F}_2^n}[\mathsf{D}(A, c^T) = 1] - \Pr_{r\leftarrow\mathcal{W}(m,\theta\cdot m)}[\mathsf{D}(A, r^T A) = 1] \geq \varepsilon/2t.$$

Further, D runs in time $T + \mathrm{poly}(n, m)$.

The algorithm D that proves the lemma is described below, followed by the proof of the lemma. Here, $\mathsf{S}_{\mathrm{LPN}}$ is the assumed search algorithm for LPN that has running time T and advantage ε.

Algorithm 2 Algorithm $\mathsf{D}(A \in \mathbb{F}_2^{m\times n}, c^T \in \mathbb{F}_2^{1\times n})$

1: Sample $i \leftarrow [0, t-1]$. ▷ Select which hybrid to use
2: Sample matrices $R \leftarrow \mathcal{W}(m, \theta\cdot m)^i$, $B \leftarrow \mathbb{F}_2^{(t-1-i)\times n}$
3: Set $H \leftarrow \begin{pmatrix} RA \\ c^T \\ B \end{pmatrix} \in \mathbb{F}_2^{t\times n}$ ▷ Create a sample from the hybrid distribution
4: Sample $s \leftarrow \mathbb{F}_2^n$ and $e \leftarrow \mathsf{Ber}(\tau)^t$ ▷ Recall that $\tau = (1-(1-2\eta)^{\theta\cdot m})/2$
5: **if** $\mathsf{S}_{\mathrm{LPN}}(H, Hs + e)$ returns s **then** ▷ Check whether $\mathsf{S}_{\mathrm{LPN}}$ succeeds
6: **return** 1
7: **else return** 0

Proof of Lemma 3.3. Fix an A such that $Adv(A) < \varepsilon/2$. We begin the proof by defining a sequence of hybrid distributions. For any $i \in [0, t]$, the hybrid distribution $\mathcal{H}_i$ over $\mathbb{F}_2^{t\times n}$ is defined by the following sampling process:

1. Sample a matrix $R \leftarrow \mathcal{W}(m, \theta\cdot m)^i$.
2. Sample a matrix $B \leftarrow \mathbb{F}_2^{(t-i)\times n}$
3. Output the row-concatenated matrix $\begin{pmatrix} RA \\ B \end{pmatrix} \in \mathbb{F}_2^{t\times n}$.

We now use the fact that, by our assumption in this case, the $\mathsf{S}_{\mathrm{LPN}}$ solver succeeds with low probability on the hybrid corresponding to $i = t$, whereas it succeeds with high probability on the hybrid corresponding to $i = 0$. To be concrete, first fix any $i \in [0, t-1]$, sample $R \leftarrow \mathcal{W}(m, \theta\cdot m)^i$, $B \leftarrow \mathbb{F}_2^{(t-i-1)\times n}$, we can make the following conclusion on the distribution $\mathcal{H}$ of $H = \begin{pmatrix} RA \\ c^T \\ B \end{pmatrix}$.

- If $c \leftarrow \mathbb{F}_2^n$, $\mathcal{H}$ is exactly $\mathcal{H}_i$.

- If $c = r^T A$ where $r \leftarrow \mathcal{W}(m, \theta \cdot m)$, $\mathcal{H}$ is exactly $\mathcal{H}_{i+1}$.

With s and e sampled as $s \leftarrow \mathbb{F}_2^n$ and $e \leftarrow \mathsf{Ber}(\tau)^t$, define Adv_0, Adv_1 as

$$Adv_0 = \Pr_{\substack{i \leftarrow [0,t-1] \\ H \leftarrow \mathcal{H}_i}}[\mathsf{S}_{\mathrm{LPN}}(H, Hs + e) = s], \quad Adv_1 = \Pr_{\substack{i \leftarrow [0,t-1] \\ H \leftarrow \mathcal{H}_{i+1}}}[\mathsf{S}_{\mathrm{LPN}}(H, Hs + e) = s].$$

Then, the behavior of the algorithm D we defined can be described exactly with Adv_0, Adv_1

$$Adv_0 = \Pr_{c \leftarrow \mathbb{F}_2^n}[\mathsf{D}(A, c^T) = 1], \quad Adv_1 = \Pr_{r \leftarrow \mathcal{W}(m, \theta \cdot m)}[\mathsf{D}(A, r^T A) = 1].$$

By a standard hybrid argument, Adv_0 and Adv_1 have a gap of $Adv_0 - Adv_1 \geq \varepsilon/2t$. This is stated in the following claim, whose proof is in Section A.

Claim 3.5. (Hybrid Argument). $Adv_0 - Adv_1 \geq \varepsilon/2t$.

Therefore, algorithm D can distinguish random syndromes of A from uniform with advantage at least $\varepsilon/2t$. As D only calls $\mathsf{S}_{\mathrm{LPN}}$ once, its running time is $\mathrm{poly}(n, m) + T$. □

4 Corollaries

In this section, we discuss some implications of Theorem 3.1. We show how the theorem gives meaningful hardness for LPN with parameters that are within the statistical-computational gap. In particular, we will show that Theorem 3.1 gives partial worst-case hardness for some parameters for LPN that can be used to construct public-key cryptography. The next corollary describes the contrapositive statement of Theorem 3.1 with parameters simplified and rearranged.

Corollary 4.1 (Hardness for LPN)**.** *Consider any polynomial $t(n) \in \mathbb{Z}$ with $t(n) > n$, constants $\alpha, \beta \in (0.01, 0.99)$ with $\alpha + \beta \in (0.01, 0.99)$, and $m(n) = \lceil 2t^2 n^{2\beta} \rceil$. Suppose that for every pair of PPT algorithms (S, D), there exists a negligible function $negl$ such that, for all large enough n, there exists an $\mathbb{F}_2$-matrix A of dimension $m \times n$ for which both of these conditions are true:*

- $\Pr[\mathsf{S}(A, As + e) = s' \text{ and } \mathrm{wt}(As' - (As + e)) \leq 2m/n^{\alpha+\beta}] \leq negl(n)$, *where $s \leftarrow \mathbb{F}_2^n$, $e \leftarrow Ber(n^{-(\alpha+\beta)})^m$.*
- $\left|\Pr[\mathsf{D}(A, c^T) = 1] - \Pr[\mathsf{D}(A, r^T A) = 1]\right| \leq negl(n)$, *where $c \leftarrow \mathbb{F}_2^n$ and $r \leftarrow \mathcal{W}(m, n^\beta)$.*

Then, for the Search LPN problem with sample count $t(n)$, secret dimension n, and noise parameter $n^{-\alpha}$, no PPT algorithm has non-negligible advantage.

Proof. This corollary can be proven using the contrapositive statement of Theorem 3.1 with an appropriate setting of variables as follows:

$$\eta = \frac{1}{n^{\alpha+\beta}} \quad \theta = \frac{n^\beta}{m}$$

The resulting noise rate of LPN from the theorem can be bounded as follows using the Bernoulli inequality,

$$\tau = \frac{1}{2}\left(1 - \left(1 - \frac{2}{n^{\alpha+\beta}}\right)^{n^\beta}\right) \leq n^{-\alpha}.$$

The noise rate in the LPN instance can later be artificially increased to be equal to $n^{-\alpha}$ as required. Next, we can also check that the condition of $t^2\theta^2 m < 1$ required by Theorem 3.1 (that the probability of intersection between two rows is bounded) is also satisfied ,

$$\frac{(t)^2(n^\beta)^2}{m} \leq \frac{(t^2)n^{2\beta}}{2t^2n^{2\beta}} = \frac{1}{2} < 1.$$

So if there is a polynomial-time algorithm for Search LPN that has non-negligible advantage $\varepsilon(n)$ on infinitely many values of n, then by Theorem 3.1, it contradicts the assumption in the statement of the corollary. Note that the runtime of S resulting from the theorem is poly $\left(n, m, T, (1/\varepsilon)^{\log_m(m/\varepsilon)}\right)$, and when $\varepsilon = 1/\text{poly}(n)$, we have that $(1/\varepsilon)^{\log_m(m/\varepsilon)} = \text{poly}(n)$, so the runtime is still polynomial in n. That concludes the proof of the corollary. □

Non-Triviality of Hardness. For the reductions in Theorem 3.1 and Corollary 4.1 to be non-trivial and useful for cryptography, they have to show that under a reasonable worst-case hardness assumption, we can infer the hardness of LPN with parameters for which it is possible to solve statistically.

For the Search LPN problem to be statistically feasible to solve, the parameters should be in a regime where the instance $(A, As + e)$ contains enough information about the secret s. Using the fact that random linear codes achieve Shannon capacity for the binary symmetric channel [GRS25], an LPN instance with secret size n, noise rate η, and sample size $t \gg n/(1 - h(\eta))$, where h is the binary entropy function, satisfies this condition. For $\eta = n^{-\alpha}$ with $\alpha \in (0.01, 0.99)$, this condition is satisfied if $t > n + \omega(n^{1-\alpha} \log n)$.

To the best of our knowledge, to design a pair of algorithms (S, D) such that S can decode the code generated by a matrix A from random noise while D can distinguish random syndromes generated by it from uniform is to pick the optimal algorithm for each task separately, just keep the more efficient one as S or D, and let the other algorithm simply not do anything. If $\alpha, \beta \in (0.01, 0.99)$, the best such algorithms run in time $\exp(O(n^{1-(\alpha+\beta)}))$ for S and $\exp(O(n^\beta))$ for D [Pra62]. Thus, if $(\alpha + \beta) \in (0.01, 0.99)$, the best algorithms for the worst-case problem are still sub-exponential time.

Parameter settings. It remains to show that we can set our parameters so that the above conditions are simultaneously satisfied. Some of these are as follow. For each setting of parameters below, we make the corresponding assumption as stated in the hypothesis of Corollary 4.1, and the conclusion is the result of applying the corollary with these parameters.

1. Setting $\alpha = 0.1$, $\beta = 0.1$, and $m = \lceil 8n^{2.2} \rceil$ implies the hardness of LPN with $t = 2n$ samples and noise rate $n^{-0.1}$.
2. Setting $\alpha = 0.5$, $\beta = 0.45$, and $m = \lceil 8n^{2.9} \rceil$ implies the hardness of LPN with $t = 2n$ samples and noise rate $n^{-0.5}$.
3. Setting $\alpha = 0.8$, $\beta = 0.1$, and $m = \lceil 2n^{6.2} \rceil$ implies the hardness of LPN with $t = n^3$ samples and noise rate $n^{-0.8}$.

Note that the last two parameter settings above, together with known search-to-decision reductions for LPN [Reg09, AIK07, MM11], are sufficient for use in Alekhnovich's construction of a

Public-Key Encryption (PKE) scheme from the hardness of LPN [Ale03]. There are other viable settings of the parameter β for which this would be true as well. The following is one specific corollary of the many that are correspondingly possible.

Corollary 4.2. *If the hypothesis of Corollary 4.1 is satisfied for the parameter setting $t = 2n$, $\alpha = 0.5$, $\beta = 0.45$ and $m = \lceil 8n^{2.9} \rceil$, then there is a secure construction of PKE.*

5 Acknowledgements

We would like to thank Yuval Ishai for pointing us to relevant references on barriers to the statistical smoothing approach.

This work was supported by the National Research Foundation, Singapore, under awards no. NRF-NRFF14-2022-0010 and NRF-NRFI09-0005.

References

[AGZ26] Divesh Aggarwal, Rishav Gupta, and Li Zeyong. Hardness amplification for (sparse) lpn, 2026. URL: https://arxiv.org/abs/2605.10056, arXiv:2605.10056.

[AIK07] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 92–110, Santa Barbara, CA, USA, August 19–23, 2007. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-74143-5_6.

[AK11] Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 474–485, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. doi:10.1109/SFCS.2003.1238204.

[BCLV26] Andrej Bogdanov, Rohit Chatterjee, Yunqi Li, and Prashant Nalini Vasudevan. Decoding balanced linear codes with preprocessing. In Shubhangi Saraf, editor, *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Bocconi University, Milan, Italy, January 27-30, 2026*, LIPIcs, pages 23:1–23:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. URL: https://doi.org/10.4230/LIPIcs.ITCS.2026.23, doi:10.4230/LIPICS.ITCS.2026.23.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48329-2_24.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_31.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 619–635, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-17659-4_21.

[BM17] Leif Both and Alexander May. Optimizing BJMM with nearest neighbors: Full decoding in $2^{2n/21}$ and McEliece security. In *Proceedings of the Tenth International Workshop on Coding and Cryptography (WCC 2017)*, page 214, September 2017. URL: https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/bjmm+.pdf.

[BM18] Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 25–46, Fort Lauderdale, Florida, United States, April 9–11, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-79063-3_2.

[DDRT23] Thomas Debris-Alazard, Léo Ducas, Nicolas Resch, and Jean-Pierre Tillich. Smoothing codes and lattices: Systematic study and new bounds. *IEEE Trans. Inf. Theory*, 69(9):6006–6027, 2023. doi:10.1109/TIT.2023.3276921.

[DEEK24] Léo Ducas, Andre Esser, Simona Etinski, and Elena Kirshanova. Asymptotics and improvements of sieving for codes. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VII*, volume 14657 of *Lecture Notes in Computer Science*, pages 151–180, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58754-2_6.

[DR25] Thomas Debris-Alazard and Nicolas Resch. Worst and average case hardness of decoding via smoothing bounds. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025: 28th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 15675 of *Lecture Notes in Computer Science*, pages 363–392, Røros, Norway, May 12–15, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91823-0_12.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In G. Kabatianskii, editor, *Proceedings of the 5th Joint Soviet-Swedish International Workshop on Information Theory*, pages 50–52, Moscow, 1991. Nauka.

[GRS25] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. 2025. Version of August 26, 2025. URL: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005. doi:10.1007/11538462_32.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484, Santa Barbara, CA, USA, August 14–18, 2011. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-22792-9_26.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124, Seoul, South Korea, December 4–8, 2011. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-25385-0_6.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_9.

[PB25] Madhura Pathegama and Alexander Barg. Limitations of the decoding-to-LPN reduction via code smoothing. *Designs, Codes and Cryptography*, 93(7):2761–2778, 2025. doi:10.1007/s10623-025-01617-9.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536461.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2012. doi:10.1007/978-3-642-27660-6_9.

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962. doi:10.1109/TIT.1962.1057777.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[Ste89] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1989. doi:10.1007/BFb0019850.

[YZ16] Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 214–243, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53018-4_9.

[YZ21] Yu Yu and Jiang Zhang. Smoothing out binary linear codes and worst-case sub-exponential hardness for LPN. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 473–501, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84252-9_16.

A Supplementary Proofs

Fact 2.1. (Piling Up Lemma). *Let $X_1, X_2, \ldots, X_k$ be independent random variables where $X_i \leftarrow \mathsf{Ber}(\eta)$, then the random variable $S = X_1 + X_2 + \cdots + X_k$, where addition is over $\mathbb{F}_2$, follows the distribution* $\mathsf{Ber}\left(\dfrac{1-(1-2\eta)^k}{2}\right)$.

Proof. Let $Y_i := (-1)^{X_i} \in \{\pm 1\}$. Then $\mathbf{E}[Y_i] = (1-\eta)\cdot 1 + \eta \cdot (-1) = 1 - 2\eta$. Notice that,

$$(-1)^S = Y_1 \cdot Y_2 \cdots Y_k \ .$$

Since $Y_1, \ldots, Y_k$ are independently distributed, we have that

$$\mathbf{E}\left[(-1)^S\right] \ = \ \mathbf{E}\left[\prod_{i=1}^{k}(-1)^{X_i}\right] \ = \ \prod_{i=1}^{k}\mathbf{E}\left[(-1)^{X_i}\right] \ = \ (1-2\eta)^k.$$

From $\mathbf{E}\left[(-1)^S\right] = \Pr[S=0] - \Pr[S=1] = 1 - 2\Pr[S=1]$. Hence

$$\Pr[S=1] \ = \ \frac{1-(1-2\eta)^k}{2}.$$

□

Claim 3.5. (Hybrid Argument). *$Adv_0 - Adv_1 \geq \varepsilon/2t$.*

Proof. Suppose $s \leftarrow \mathbb{F}_2^n, e \leftarrow \mathrm{Ber}(\tau)^t$ throughout the proof. Recall that,

$$Adv = \Pr_{R \leftarrow \mathcal{W}(m,\theta\cdot m)^t}[\mathsf{S}_{\mathrm{LPN}}(RA, RAs+e) = s].$$

From $\mathsf{S}_{\mathrm{LPN}}$ having advantage ε and $Adv < \varepsilon/2$,

$$\Pr_{B \leftarrow \mathbb{F}_2^{t\times n}}[\mathsf{S}_{\mathrm{LPN}}(B, Bs+e) = s] - \Pr_{R \leftarrow \mathcal{W}(m,\theta\cdot m)^t}[\mathsf{S}_{\mathrm{LPN}}(RA, RAs+e) = s] \geq \varepsilon/2. \tag{7}$$

Note that $\mathcal{H}_t$ is exactly the distribution RA, while $\mathcal{H}_0$ is the uniform distribution. Telescoping,

$$\begin{aligned}
&\Pr_{B\leftarrow\mathbb{F}_2^{t\times n}}[\mathsf{S}_{\mathrm{LPN}}(B, Bs+e)=s] - \Pr_{R\leftarrow\mathcal{W}(m,\theta m)^t}[\mathsf{S}_{\mathrm{LPN}}(RA, RAs+e)=s] \\
&= \sum_{i=0}^{t-1}\left(\begin{array}{c}\Pr_{H_i\leftarrow\mathcal{H}_i}[\mathsf{S}_{\mathrm{LPN}}(H_i, H_is+e)=s] \\ -\Pr_{H_{i+1}\leftarrow\mathcal{H}_{i+1}}[\mathsf{S}_{\mathrm{LPN}}(H_{i+1}, H_{i+1}s+e)=s]\end{array}\right).
\end{aligned}$$

There are two components in the equation, look at each component individually, by the law of total probability,

$$\begin{aligned}
&\sum_{i\in[0,t-1]}\Pr_{H_i\leftarrow\mathcal{H}_i}[\mathsf{S}_{\mathrm{LPN}}(H_i, H_is+e)=s] \\
&= t\sum_{i\in[0,t-1]}\frac{1}{t}\cdot\Pr_{H_i\leftarrow\mathcal{H}_i}[\mathsf{S}_{\mathrm{LPN}}(H_i, H_is+e)=s] \\
&= t\sum_{i\in[0,t-1]}\Pr_{X\leftarrow[0,t-1]}[X=i]\Pr_{H_X\leftarrow\mathcal{H}_X}[\mathsf{S}_{\mathrm{LPN}}(H_X, H_Xs+e)=s \mid X=i] \\
&= t\Pr_{\substack{i\leftarrow[0,t-1]\\ H_i\leftarrow\mathcal{H}_i}}[\mathsf{S}_{\mathrm{LPN}}(H_i, H_is+e)=s] = t\cdot Adv_0.
\end{aligned}$$

By a similar argument,

$$\sum_{i\in[0,t-1]}\Pr_{H_{i+1}\leftarrow\mathcal{H}_{i+1}}[\mathsf{S}_{\mathrm{LPN}}(H_{i+1}, H_{i+1}s+e)=s] = t\cdot Adv_1.$$

Substitute into Eq. (7), and that concludes the claim

$$t\cdot Adv_0 - t\cdot Adv_1 \geq \varepsilon/2 \implies Adv_0 - Adv_1 \geq \varepsilon/2t.$$

□