Structured-Seed Local Pseudorandom Generators and their Applications

Dung Bui[1], Geoffroy Couteau[2], and Nikolas Melissaris[3]

[1] IRIF, Université Paris Cité, France
bui@irif.fr
[2] CNRS, IRIF, Université Paris Cité, France
couteau@irif.fr
[3] Aarhus University, Denmark
nikolas@cs.au.dk

Abstract. In this note, we introduce structured-seed local pseudorandom generators, a relaxation of local pseudorandom generators. We provide constructions of this primitive under the sparse-LPN assumption, and explore its implications.

1 Introduction

Pseudorandom generators (PRGs) are functions mapping n bits to $m(n) > n$ bits such that no polynomial-time algorithm can distinguish their output on a random input from a random m-bit string. Local pseudorandom generators (local PRGs) are pseudorandom generators where every output bit can be computed from a constant number of input bits (that is, they belong to the complexity class NC^0). The existence of local PRGs was first investigated in the work of Cryan and Miltersen [CM01]. The work of Applebaum, Ishai, and Kushilevitz [AIK04,AIK08] showed that pseudorandom generators in NC^0 with sublinear stretch ($m = n + o(n)$) exist under widely believed standard assumption for the case of PRG with sublinear stretch (such as factorization, or discrete logarithm), and under a specific intractability assumption related to the hardness of decoding "sparsely generated" linear codes for the case of PRG with linear stretch $m = \Theta(n)$.

In recent years, the existence of local pseudorandom generators with *polynomial stretch* ($m = n^{1+\varepsilon}$ for some constant $\varepsilon > 0$) has been shown to enjoy a variety of applications, ranging from secure computation with constant computational overhead [IKOS08], indistinguishability obfuscation [JLS21, JLS22], pseudorandom correlation generators and functions [BCG+17, BCM+24], public key encryption [BKR23] and sublinear secure computation [BCM23], to applications extending beyond the realm of cryptography such as hardness of learning [DV21]. Consequently, the existence of polynomial-stretch local PRGs and the cryptanalysis of existing candidates has been the subject of many works [Gol00, MST03, BQ09, App12, OW14, CEMT14, App15, ABR16, AL16, LV17, CDM+18, AK19, OST19, Méa, YGJL21, Méa22, Üna23b, DMR23, Üna23a]. All existing candidates build upon a design originally suggested in [Gol00] that applies a well-chosen predicate P on constant-size subsets of the bits of the seed, where the subsets are chosen to form the hyperedges of a sufficiently expanding uniform hypergraph.

1.1 Our contribution

In this work, we revisit the applications of local PRGs. Our main observation is that many of the standard applications of local PRGs do not require the full power of local PRGs. In particular, many applications only require the existence of a local pseudorandom mapping from n-bit seeds to m-bit strings, but *do not require the seeds to be sampled uniformly at random.* We formalize this observation by introducing the notion of *structured-seed* local pseudorandom generators, which generalize local PRGs to the setting where the seed should be sampled from a prescribed distribution with support over $\{0,1\}^n$ (instead of being sampled uniformly at random), and provide a sample of applications where structured-seed local PRGs can be used as a drop-in replacement to standard local PRGs. Concretely, we show how to use structured-seed local PRGs in the following applications:

- Indistinguishability obfuscation from well-founded assumptions [JLS21];
- Constant-overhead secure computation [IKOS08];

- Compact homomorphic secret sharing [BCM23];
- Hardness of learning DNFs [DV21].

Beyond introducing structured-seed local PRGs and providing a formal definition, we also introduce constructions of structured-seed local PRGs from well-studied cryptographic assumptions which are not known to imply the existence of standard local PRGs. Concretely, we focus on the *sparse learning parity with noise* assumption, an assumption introduced in the work of Alekhnovich [Ale03] which is equivalent to hardness of decoding random LDPC codes. We provide an extended coverage of various flavors of this assumption and show several constructions of structured-seed local PRGs from these variants. In particular, we obtain:

- A direct construction of structured-seed local PRG from the sparse-LPN assumption with regular noise distribution (where the noise is sampled as a concatenation of random one-hot vector), and
- A construction of structured-seed local PRG with inverse-polynomial security from the sparse-LPN assumption with more common noise distributions. This second construction is more involved and builds upon hashing schemes for balanced allocation.

As a consequence, we show that for the four applications above, the assumption of local PRGs can be replaced by the assumption that sparse-LPN with regular noise is hard (for the application to indistinguishability obfuscation, we require subexponential hardness of the assumption). For the application to hardness of learning, where inverse-polynomial security suffices, we further obtain hardness results from the sparse-LPN assumption without regular noise.

1.2 Concurrent work

In a recent work [RVV24], Ragavan, Vafa, and Vaikuntanathan also introduced the notion of structured-seed local pseudorandom generators and studied its application. Our work is concurrent and independent to theirs, and there is a significant overlap between our results: the core observation (that structured-seed local PRGs can replace local PRGs in some applications) and the definition of structured-seed local PRGs are essentially the same in both works. We outline a few differences:

- The work of [RVV24] focuses on the application of structured-seed local PRGs to indistinguishability obfuscation (iO). While we also consider iO, they provide a much more thorough coverage of this application and achieve stronger results (replacing local PRGs with structured-seed local PRG in the work of [JLS22] rather than in the work of [JLS21], hence avoiding the use of the LWE assumption).
- The other applications we consider are not considered in the work of [RVV24]. While our coverage of these applications is (for now) superficial, we plan to include a significantly more extended coverage of the application to hardness of learning in future versions of this work (since several non-trivial complications arise in this setting).
- Eventually, the work of [RVV24] focused on constructions from sparse-LPN with Bernoulli noise. In contrast, we consider other noise distributions, such as regular noise, and XOR noise (where the noise is sampled as a XOR of unit vectors). Consequently, our constructions of structured-seed local PRGs, while sharing a common intuition, differ significantly from theirs.

The current version of our paper is a work in progress. We are posting this working draft on ePrint due to the concurrent work of [RVV24] being online. In future versions of this work, we plan to include additional results, such as

- Exploring further the implications of structured-seed local PRGs for PAC-learning;
- Providing constructions of structured-seed local PRGs with improved parameters (smaller locality for a given stretch) from the Expand-Accumulate LPN assumption from [BCG+22].

2 Preliminaries

Vector and Matrix. We denote vectors using bold font and matrices using caps. Given a vector $\mathbf{v}$, we write $\mathbf{v}[i]$ to denote its i-th entry, for a given set $S \in [n]$, we denote $\mathbf{v}[S]$ as a set including i-th entries of $\mathbf{v}$ for all $i \in S$. By default, all vectors are column vectors. We call a length-m one-hot vector as a unit vector over $\mathbb{F}_2^n$. We write $[n]$ to denote the set $\{1, \cdots, n\}$ and $\mathsf{unit}_n(i)$ to denote a unit vector of length n having non-zero i-th entry. For any distribution $\mathcal{D}$, we denote $x \leftarrow \mathcal{D}$ is the process of sampling uniformly x over the distribution $\mathcal{D}$.

Distribution. Given an algorithm $\mathcal{A}$ and a pair of distributions $(\mathcal{D}_0, \mathcal{D}_1)$, we write $\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1)$ to denote

$$\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1) = \left| \Pr_{x \leftarrow \mathcal{D}_0}[\mathcal{A}(x) = 0] - \Pr_{x \leftarrow \mathcal{D}_1}[\mathcal{A}(x) = 0] \right|.$$

We say the pair of distributions $(\mathcal{D}_0, \mathcal{D}_1)$ is polynomially indistinguishable and subexponentially indistinguishable if $\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1) \leq \mathsf{negl}(\lambda)$ for all sufficient large $\lambda \in \mathbb{N}$ and $\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0, \mathcal{D}_1) \leq \exp(-\lambda^c)$ for a real number $c > 0$ respectively.

2.1 LPN Assumptions

The LPN assumption over the binary field $\mathbb{F}_2$ states, informally, that no adversary can distinguish $(A, A \cdot \mathbf{x} + \mathbf{e})$ from $(A, \mathbf{b})$, where A is a matrix sampled from some matrix distribution $\mathcal{M}$ over $\mathbb{F}_2^{n \times k}$, $\mathbf{x}$ is sampled uniformly from $\mathbb{F}_2^k$, and $\mathbf{e}$ is a *noise vector* sampled from some noise distribution $\mathcal{E}$ over (typically sparse) $\mathbb{F}_2$-vectors. The vector $\mathbf{b}$ is a uniform vector over $\mathbb{F}_2^n$. More formally, we define below the LPN assumption over $\mathbb{F}_2$ with dimension k and n samples, w.r.t. a matrix distribution $\mathcal{M}$ and a noise distribution $\mathcal{D}$:

Definition 2.1 (Learning parity with noise). *For any integer $k \in \mathbb{N}$, let $n = n(k)$ be a polynomial, and let $\mathcal{M} = \mathcal{M}_{n,k}$ be a distribution over $\mathbb{F}_2^{n \times k}$. Let $\mathcal{E} = \mathcal{E}_n$ be a noise distribution over $\mathbb{F}_2^n$. Then, we say that the $(\mathcal{M}, \mathcal{E})$-LPN problem is (T, ε, δ)-hard if for every probabilistic adversary $\mathcal{A} = (\mathcal{A}_\lambda)_{\lambda \in \mathbb{N}}$ of size at most $T = T(\lambda)$, it holds that for all large enough k,*

$$\Pr_{A \xleftarrow{\$} \mathcal{M}_{n,k}} \left[\mathsf{Adv}_{\mathcal{A}_k}(\mathcal{D}_0^A, \mathcal{D}_1^A) > \varepsilon \right] \leq \delta,$$

where $\mathcal{D}_0^A$ denotes the distribution $\{(A, A \cdot \mathbf{s} + \mathbf{e}) \mid \mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k, \mathbf{e} \xleftarrow{\$} \mathcal{E}\}$ and $\mathcal{D}_1^A$ denotes $\{(A, \mathbf{b}) \mid \mathbf{b} \xleftarrow{\$} \mathbb{F}_2^n\}$.

Types of noise. Denoting w a parameter which governs the average density of nonzero entries in a random noise vector $\mathbf{e}$, common choices of noise distribution are Bernoulli noise (each entry of $\mathbf{e}$ is sampled from a Bernoulli distribution with parameter w/n), exact noise ($\mathbf{e}$ is uniformly random over the set of vectors of Hamming weight w), and regular noise ($\mathbf{e}$ is a concatenation of w random unit vectors). Another natural choice of noise distribution is to set $\mathbf{e}$ to be the XOR of w random one-hot vectors of length n, as the independence of the one-hot vectors is often convenient in security analysis. We will call respectively *Bernoulli noise*, *exact noise*, *regular noise*, and *xor noise* these standard noise distributions, and denote:

- $\mathcal{B}_{n,w}$ the distribution over $\mathbb{F}_2^n$ where each entry is set to 1 with independent probability w/n;
- $\mathcal{S}_{n,w}$ the uniform distribution over the set of vectors of $\mathbb{F}_2^n$ with Hamming weight w;
- $\mathcal{R}_{n,w}$ the distribution obtained by sampling w one-hot vectors over $\mathbb{F}_2^{n/w}$ (assuming for simplicity that w divides n) and outputting their concatenation;
- $\mathcal{X}_{n,w}$ the distribution obtained by sampling w one-hot vectors over $\mathbb{F}_2^n$ and outputting their XOR.

When n is clear from the context, we drop it from the subscript and write $\mathcal{B}_w, \mathcal{S}_w, \mathcal{R}_w, \mathcal{X}_w$ respectively.

When $w \ll \sqrt{n}$ (the "low-noise" setting), the flavors of LPN with noise sampled from $\mathcal{S}_w$ and $\mathcal{X}_w$ are easily shown to be equivalent, since a random sample from $\mathcal{X}_w$ has Hamming weight exactly w with probability at least $1 - w^2/n$. In turn, LPN with noise sampled from $\mathcal{S}_w$ is known to be equivalent to LPN with noise sampled from $\mathcal{B}_w$ [Pie12]. While there are also reductions between LPN with regular noise and other variants, they typically induce a much larger loss in the parameters [LWYY24].

Discussion on the definition. The reader may observe that Definition 2.1 differs slightly from the standard definition of LPN: the standard definition requires that the adversary should have at most advantage ε (for a negligible $\varepsilon = \varepsilon(\lambda)$) in distinguishing $\{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathcal{M}_{n,k}, \mathbf{b} \xleftarrow{\$} \mathcal{D}_0^A\}$ from $\{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathcal{M}_{n,k}, \mathbf{b} \xleftarrow{\$} \mathcal{D}_1^A\}$ (that is, the distinguishing advantage is also quantified over the random choice of A). Definition 2.1 is more fine-grained: it separates the probability δ that the matrix A is a "good matrix" for the LPN problem from the probability ε that the adversary can (given A) distinguish the LPN samples from random. Setting (ε, δ) to be negligible recovers the standard LPN definition. However, this more fine-grained definition style is particularly helpful for the *sparse-LPN* assumption, which we cover in Section 4.

2.2 Useful Lemmas

Lemma 2.2 (Chernoff Bound). *Let $n \in \mathbb{N}$ and $X_1, \cdots, X_n$ be independent random variables taking values in $\{0,1\}$. Let X denote their sum and let $\mu \leftarrow \mathbb{E}[X]$. Then for any $\delta \in (0,1)$,*

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{3}\right) \text{ and } \Pr[X \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu}{2}\right).$$

Lemma 2.3 (Markov inequality). *Let X be a positive random variable with finite expectation μ. Then for any $k > 0$, $\Pr[X \geq k] \leq \mu/k$.*

Lemma 2.4 (Piling-up lemma). *For any $0 < r < 1/2$ and any integer N, given N i.i.d. random variables $X_1, \cdots, X_N$ over $\mathbb{F}_2$ with $\Pr[X_i = 1] = r$, it holds that $\Pr[\bigoplus_{i=1}^N X_i = 0] = 1/2 + (1-2r)^n/2$.*

3 Defining Structured-Seed Local PRGs

In this section, we introduce the notion of structured-seed local pseudorandom generator. Recall that a pseudorandom generator (PRG) is an algorithm $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$, with $n > m$, such that no polynomial-time adversary can distinguish between $\mathsf{PRG}(r)$ (for $r \xleftarrow{\$} \{0,1\}^n$) and a random $z \xleftarrow{\$} \{0,1\}^m$. The uniform string $r \in \{0,1\}^n$ is called the *seed* of the PRG.

Informally, a local pseudorandom generator is a pseudorandom generator where each output bit depends on a constant number of input bits. A *structured-seed* pseudorandom generator relaxes the standard notion of pseudorandom generator to allow for more general distributions of seeds: instead of sampling r uniformly over $\{0,1\}^n$, we sample it as $r \xleftarrow{\$} \mathsf{SampleSeed}$, where

- (small size) the support $\mathsf{Supp}(\mathsf{SampleSeed})$ of $\mathsf{SampleSeed}$ is contained in $\{0,1\}^n$, and
- (efficiency) the running time of the sampler $\mathsf{SampleSeed}$ is much smaller than m.

We provide a formal definition below.

Definition 3.1 (Structured-Seed Local Pseudorandom Generator). *A structured-seed pseudorandom generator with a stretch $m(\cdot)$ is a triple of uniform p.p.t. algorithms $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$, with:*

- $\mathsf{Setup}(1^\lambda)$. *A probabilistic algorithm that on inputs 1^λ and outputs a public parameter pp.*
- $\mathsf{SampleSeed}(\mathsf{pp})$. *A probabilistic algorithm that on inputs pp and outputs a seed value $\mathsf{seed} \in \{0,1\}^n$.*
- $\mathsf{PRG}(\mathsf{pp}, \mathsf{seed})$. *A deterministic algorithm that on inputs seed, public parameter pp and outputs an evaluation value $y \in \{0,1\}^{m(n)}$.*

A structured-seed pseudorandom generator is (T, ϵ, δ)-secure *if for any* non-uniform *p.p.t adversary $\mathcal{A} = (\mathcal{A}_\lambda)_{\lambda \in \mathbb{N}}$ of size at most $T = T(\lambda)$, for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{Adv}_{\mathcal{A}_\lambda}(\mathcal{D}_0, \mathcal{D}_1) > \varepsilon\right] \leq \delta,$$

where $\mathcal{D}_0 = \mathcal{D}_0^{\lambda,n}$ denotes the family of distributions

$$\{(\mathsf{pp}, \mathsf{PRG}(\mathsf{pp}, r)) \mid \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda), r \xleftarrow{\$} \mathsf{SampleSeed}(\mathsf{pp})\}$$

and $\mathcal{D}_1 = \mathcal{D}_1^{\lambda,n}$ denotes the family of distributions

$$\{(\mathsf{pp}, z) \mid \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda), z \xleftarrow{\$} \{0,1\}^{m(n)}\}.$$

Furthermore, a structured-seed PRG is said to be in NC^0, or local, *if PRG is implementable by a uniformly efficiently generatable NC^0 circuit. We say that a structured-seed PRG has* polynomial stretch *if $m(n) = n^{1+\Omega(1)}$. Denoting $\mathsf{T_{ss}} = \mathsf{T_{ss}}(\lambda, n, m)$ the (worst-case) running time of $\mathsf{SampleSeed}$ (implemented by a uniformly efficiently generatable family of boolean circuits), we say that a structured-seed PRG has* strong polynomial stretch *if $m(n) = \mathsf{T_{ss}}^{1+\Omega(1)}$.*

We note that if a structured-seed PRG with input size n and stretch m is local, then there exists a locality parameter l such that for all $i \in [m]$, there exists a subset $S_i \subset [n]$ of size $|S_i| \leq l$ and a predicate P_i such that for all $x \in \mathsf{Supp}(\mathsf{SampleSeed}(\mathsf{pp}))$, defining $y = \mathsf{PRG}(x)$, we have $y_i = P_i(x[S_i])$.

Eventually, we consider a further relaxation of structured-seed local PRGs where we allow Setup to be inefficient:

Definition 3.2 (Non-Uniform Structured-Seed Local PRG). *We say that* $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ *is a* non-uniform *structured-seed local pseudorandom generator if it satisfies the conditions of Definition 3.1, except that* $\mathsf{Setup}(1^\lambda)$ *is not required to run in polynomial time.*

4 The Sparse-LPN Assumption

4.1 The sparse-LPN assumption

In this work, we will mostly focus on a variant introduced in [Ale03] (commonly called the "Alekhnovich assumption" or "sparse-LPN assumption"), where $\mathcal{M}$ is a distribution of *sparse* matrices:

Notation 1 *We denote by* $\mathcal{W}^c = \mathcal{W}^c_{n,k}$ *the distribution over* $\mathbb{F}_2^{n\times k}$ *that samples independently each row* $\mathbf{r}$ *of* A *as* $\mathbf{r}^\intercal \xleftarrow{\$} \mathcal{S}_{c,k}$*; that is,* A *is a uniformly random matrix with row-weight* c *over* $\mathbb{F}_2^{n\times k}$*.*

With these notations, the Aleknovich assumption asserts that for a suitable constant $c \geq 3$, the $(\mathcal{M}^c, \mathcal{S}_w, \mathbb{F}_2)$-$\mathsf{LPN}(k, n)$ assumption holds.

It is not hard to see that δ cannot be negligible for sparse-LPN: with probability at least n^2/k^c, two rows $\mathbf{a}_i, \mathbf{a}_j$ of A will be identical, in which case there is a trivial distinguisher (as $\langle \mathbf{a}_i, \mathbf{x}\rangle + \mathbf{e}[i]$ is very likely to be equal to $\langle \mathbf{a}_j, \mathbf{x}\rangle + \mathbf{e}[j]$ by the sparsity of the noise). In fact, any small set of linearly-dependent rows of A yields a nontrivial distinguisher. Therefore, the standard formulation of sparse-LPN asserts that the $(\mathcal{M}^c, \mathcal{S}_w, \mathbb{F}_2)$-$\mathsf{LPN}(k, n)$ is $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda), \delta)$-hard for a suitable inverse-polynomial δ.

When A does not have a small set of linearly-dependent rows, the best-known attacks on sparse-LPN are the same as the best-known attacks on LPN. This is best explained through the framework of *linear tests*, a framework to heuristically analyze the hardness of variants of LPN which has roots in the works of Naor and Naor [NN90] and Mossel, Shpilka, and Trevisan [MST03], and which was explicitly put forth and stuied in the context of LPN in [ADI+17, BCG+20, CRR21]. The central observation of this framework is that most known attacks against LPN (such as BKW [BKW00, Lyu05], ISD [Pra62], and many more) share a common template, and that to defeat all attacks sharing this template, it suffices (in coding theoretic terms) that the *dual distance* of the code generated by A is large – *i.e.*, that A does not have a small set of linearly-dependent rows.

4.2 Security against linear tests

We briefly overview the linear test framework, and derive from the framework a concrete set of parameters for the sparse-LPN assumption. We stress that the framework is only a heuristic: there are known settings in which a variant of LPN can be broken by an attack that does not fit in the framework (see e.g. the discussions in [CRR21, BCCD23]). Nevertheless, the bounds provided by this framework are in line with the state-of-the-art cryptanalysis on sparse LPN, and provides a convenient heuristic to choose plausible parameters.

Notation 2 *We call* dual distance *of a matrix* M*, and write* $\mathsf{dd}(M)$*, the largest integer* d *such that every subset of* d *rows of* M *is linearly independent.*

Informally, the linear test framework models attacks where the adversary is unbounded and can arbitrarily use the LPN matrix A, but is restricted to computing a linear function of the vector $\mathbf{b}$. Concretely, let $\mathcal{A}$ be a (possibly unbounded) adversary. The attack proceeds in two stages:

1. $\mathcal{A}$ receives the LPN matrix A and outputs a nonzero *test vector* $\mathbf{v}$. Note that $\mathcal{A}$ can run in unbounded time, but sees only the matrix A.

2. In the second stage, the vector $\mathbf{b} \leftarrow A \cdot \mathbf{x} + \mathbf{e}$ is sampled. We say that $\mathcal{A}$ is successful if, with large probability over the random choice of A, the *bias* of the distribution induced by sampling $\mathbf{x}$ and the noise $\mathbf{e}$ and computing $\langle \mathbf{v}, A\mathbf{x} + \mathbf{e} \rangle$ (that is, evaluating the linear function picked by $\mathcal{A}$ on $\mathbf{b}$) is noticeable.

To formally state the definition, we recall the notion of bias of a distribution:

Definition 4.1 (Bias of a Distribution). *Given a distribution $\mathcal{D}$ over $\mathbb{F}_2^n$ and a nonzero vector $\mathbf{u} \in \mathbb{F}_2^n$, the bias of $\mathcal{D}$ with respect to $\mathbf{u}$, denoted $\mathsf{bias}_{\mathbf{u}}(\mathcal{D})$, is equal to*

$$\mathsf{bias}_{\mathbf{u}}(\mathcal{D}) = \left| \mathop{\mathbb{E}}_{\mathbf{x} \leftarrow \mathcal{D}} [\mathbf{u}^\intercal \cdot \mathbf{x}] - \mathop{\mathbb{E}}_{\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^n} [\mathbf{u}^\intercal \cdot \mathbf{x}] \right| = \left| \Pr_{\mathbf{x} \leftarrow \mathcal{D}} [\mathbf{u}^\intercal \cdot \mathbf{x} = 1] - \frac{1}{2} \right|.$$

The bias of $\mathcal{D}$, denoted $\mathsf{bias}(\mathcal{D})$, is the maximum bias of $\mathcal{D}$ with respect to any nonzero vector $\mathbf{u}$.

Definition 4.2 (Security against Linear Test). *For any integer $k \in \mathbb{N}$, let $n = n(k)$ be a polynomial, and let $\mathcal{M} = \mathcal{M}_{n,k}$ be a distribution over $\mathbb{F}_2^{n \times k}$. Let $\mathcal{E} = \mathcal{E}_n$ be a noise distribution over $\mathbb{F}_2^n$. Then, we say that the $(\mathcal{M}, \mathcal{E})$-$\mathsf{LPN}$ problem is* (ε, δ)-secure against linear tests *if if for any (possibly inefficient) adversary $\mathcal{A}$ which, on input a matrix $A \in \mathbb{F}_2^{n \times k}$, outputs a nonzero $\mathbf{v} \in \mathbb{F}_2^n$, it holds that*

$$\Pr[A \xleftarrow{\$} \mathcal{M}, \mathbf{v} \xleftarrow{\$} \mathcal{A}(A) \ : \ \mathsf{bias}_{\mathbf{v}}(\mathcal{D}^A) \geq \varepsilon(\lambda)] \leq \delta(\lambda),$$

where $\mathcal{D}^A$ denotes the distribution induced by sampling $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^m$, $\mathbf{e} \leftarrow \mathcal{E}_n$, and outputting the LPN samples $A \cdot \mathbf{x} + \mathbf{e}$.

The following observation is folklore, and was made explicitly e.g. in [BCG+20]:

Observation 1 *Most existing attacks against LPN, including BKW [BKW00, Lyu05], ISD [Pra62], variants of Gaussian elimination [LF06, EKM17], statistical decoding attacks [Ove06], generalized birthday attacks [Wag02, Kir11], linearization attacks [BM97, Saa07], or attacks based on finding correlations with low-degree polynomials [ABG+14, BR17], can be cast as instances of the linear test framework. Therefore, none of these attacks can provide a polynomial-time distinguisher against any LPN assumption that is provably (ε, δ)-secure against linear tests for negligible functions (ε, δ).*

Given any test vector $\mathbf{v}$, observe that $\langle \mathbf{v}, A\mathbf{x} + \mathbf{e} \rangle = \langle \mathbf{v}, A\mathbf{x} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle$. We recall a simple folklore observation, rooted in [NN90, MST03]: the distribution induced by $A \cdot \mathbf{x}$ is $\mathsf{dd}(A)$-wise independent by definition of $\mathsf{dd}(A)$. Hence, the bias of $\langle \mathbf{v}, A\mathbf{x} + \mathbf{e} \rangle = \langle \mathbf{v}, A\mathbf{x} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle$ is zero if the Hamming weight of $\mathbf{v}$ is less than $\mathsf{dd}(A)$. But if $\mathsf{HW}(\mathbf{v}) \geq \mathsf{dd}(A)$, then $\langle \mathbf{v}, \mathbf{e} \rangle$ has low bias, because $\mathbf{e}$ "hits" a nonzero entry of $\mathbf{v}$ with large probability. Formally:

Lemma 4.3. *Let $\mathcal{M} = \mathcal{M}_{n,k}$ be a distribution over $\mathbb{F}_2^{n \times k}$ and $\mathcal{E}_n$ denote a noise distribution over $\mathbb{F}_2^n$. Then for any $d \in \mathbb{N}$, the the $(\mathcal{M}, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$ and $n = n(\lambda)$ samples is $(\varepsilon_d, \delta_d)$-secure against linear tests, where*

$$\varepsilon_d = \max_{\mathsf{HW}(\mathbf{v}) > d} \mathsf{bias}_{\mathbf{v}}(\mathcal{E}_n), \qquad \text{and} \qquad \delta_d = \Pr_{A \xleftarrow{\$} \mathcal{M}} [\mathsf{dd}(A) < d].$$

The quantity ε_d in Lemma 4.3 depends solely on the noise distribution and can be computed easily for standard types of noise:

Lemma 4.4. *For any integer d and noise distribution $\mathcal{E}_n$, we denote $\varepsilon_d(\mathcal{E}_n) = \max_{\mathsf{HW}(\mathbf{v}) > d} \mathsf{bias}_{\mathbf{v}}(\mathcal{E}_n)$. Then*

- $\varepsilon_d(\mathcal{X}_{n,w}) \leq (1 - 2(d+1)/n)^w/2 \leq \exp(-2(d+1)w/n)/2$
- $\varepsilon_d(\mathcal{R}_{n,w}) \leq (1 - 2(d+1)/n)^w/2 \leq \exp(-2(d+1)w/n)/2$
- $\varepsilon_d(\mathcal{B}_{n,w}) \leq (1 - 2w/n)^{d+1}/2 \leq \exp(-2(d+1)w/n)/2$

The proof of the claim is a straightforward application of the piling-up lemma (Lemma 2.4): for $\mathcal{X}_{n,w}$ or $\mathcal{R}_{n,w}$, the distribution induced by $\langle \mathbf{v}, \mathbf{e} \rangle$ is a xor of w independent Bernoulli samples, each equal to 1 with probability $\mathsf{HW}(\mathbf{v})/n < d/n$. For $\mathcal{B}_{n,w}$, $\langle \mathbf{v}, \mathbf{e} \rangle$ is a xor of $\mathsf{HW}(\mathbf{v}) > d$ Bernoulli sample of rate w/n. The rightmost side of the inequalities follows from the standard inequality $(1 - 1/N)^N \leq \exp(-1)$. We note that a similar bound can be shown (with a slightly more tedious analysis) for $\mathcal{S}_{n,w}$.

4.3 The dual distance of random sparse matrices

We recall below a standard bound on the probability that random sparse matrices have large dual distance.

Theorem 4.5 (Most sparse matrices have large dual distance). *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any large enough $k = k(\lambda)$, there is a constant $\gamma(c)$ such that for any $n \leq k^{(1-\eta)c/2+\eta}$,*

$$\Pr\left[A \xleftarrow{\$} \mathcal{W}^c \ : \ \mathsf{dd}(A) \geq \frac{k^\eta}{\gamma(c)} - 1\right] \geq 1 - \left(\frac{\gamma(c)}{k^\eta}\right)^{c-2}.$$

For example, setting $c = 3$ and $\eta = 1/5$, Theorem 4.5 yields that for $n = k^{1.4}$, a random c-sparse matrix A over $\mathbb{F}_2^{n\times k}$ has dual distance $\mathsf{dd}(A) = \Omega(k^{0.2})$ with probability at least $1 - O(k^{-0.2})$. For completeness, we provide a proof of Theorem 4.5 in Appendix A.

4.4 A parametrized version of the sparse-LPN assumption

Combining Lemma 4.4 (bounding ε_d) and Theorem 4.5 (to bound δ_d) yields a quantified estimate of the security of sparse-LPN against attacks from the linear test framework:

Lemma 4.6. *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{n,w}, \mathcal{R}_{n,w}, \mathcal{B}_{n,w}\}$, the $(\mathcal{S}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, noise $w = w(\lambda)$, and $n = n(k) \leq k^{(1-\eta)c/2+\eta}$ samples is (ε, δ)-secure against linear tests, where*

$$\varepsilon = \frac{1}{2} \cdot \exp\left(-2\frac{w \cdot k^\eta}{\gamma(c) \cdot n}\right), \qquad \delta = \left(\frac{\gamma(c)}{k^\eta}\right)^{c-2},$$

for a constant $\gamma(c) = 2 \cdot (c/2)^{\frac{c/2}{c/2-1}} \cdot e^{\frac{c/2+1}{c/2-1}}$.

For example, $\gamma(3) \approx 1000$, $\gamma(7) \approx 70$, and $\gamma(10) \approx 67$ (note however that no effort has been made to optimize the constant, and the analysis is very loose). We are now ready to state a concrete parametrized version of the sparse-LPN assumption; the assumption basically states that there is no attack on sparse LPN that does significantly better than linear tests:

Assumption 1 *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{n,w}, \mathcal{R}_{n,w}, \mathcal{B}_{n,w}\}$, for every $T = 2^{o(\lambda)}$, the $(\mathcal{S}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = n(k) \leq k^{(1-\eta)c/2+\eta}$, and noise $w = w(\lambda, k) \geq \lambda \cdot k^{(1-\eta)c/2}$ is (T, ε, δ)-secure, with*

$$\varepsilon = \frac{\mathsf{poly}(T)}{2^{\Omega(\lambda)}}, \qquad \delta = \frac{1}{\Omega(k^\eta)}.$$

We note that variants of the concrete assumption above have appeared on multiple occasions in the literature: [ADI+17] makes a very similar assumption (Assumption 6 in [ADI+17]) but for a fixed matrix A, and in the constant rate setting (saying that any circuit of size $T = \exp(\Omega_r(\mathsf{dd}(A)))$ has advantage at most $1/T$ against sparse LPN with noise rate $r = w/n$, where r is treated as a constant and $\Omega_r(\cdot)$ hides the dependency in r). [BCG+23] also makes a very similar assumption (though they again only require the existence of a matrix A with large enough dual distance). Below, we provide two specific parameter settings consistent with the requirements of Assumption 1 that we will use in this work:

Parameter Set 1 (balancing (k, w)) *For every constant $\eta \in (0,1)$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{n,w}, \mathcal{R}_{n,w}, \mathcal{B}_{n,w}\}$, for every $T = 2^{o(\lambda)}$, there is a constant $c(\eta) = 2/(1-\eta)$ such that the $(\mathcal{S}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension k, number of samples $n = k^{1+\eta}$, and noise $w = \lambda \cdot k$ is $(T, 2^{-\Omega(\lambda)}, 1/\Omega(k^\eta))$-secure.*

Parameter Set 2 (minimizing w) *For every constants $\gamma > 0$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{n,w}, \mathcal{R}_{n,w}, \mathcal{B}_{n,w}\}$, for every $T = 2^{o(\lambda)}$ and every $c \geq 2\log\gamma/(\log\gamma - 1)$, the $(\mathcal{S}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension k, number of samples $n = k^{1+\gamma/2}$ samples, and noise $w = \lambda \cdot k^\gamma$ is $(T, 2^{-\Omega(\lambda)}, 1/\Omega(k^{1-\gamma/2}))$-secure.*

4.5 Amplifying advantage

Let $t = t(\lambda, \varepsilon, \delta) \leftarrow \lambda/(\delta\varepsilon^2)$. We prove below that, up to a $\mathsf{poly}(\lambda, 1/\varepsilon, 1/\delta)$ loss in the runtime and number of samples, the (ε, δ)-hardness of sparse-LPN implies its $(\exp(-\Omega(\lambda)), \exp(-\Omega(\lambda)))$-hardness.

Lemma 4.7. *Assume that the $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = k^{(1-\eta)c/2+\eta}$, and noise $w = \lambda \cdot k^{(1-\eta)c/2}$ is $(T', \varepsilon, \delta)$-secure. Then the $(\mathcal{S}^c, \mathcal{X}_{n',w'})$-$\mathsf{LPN}$ problem with dimension k, number of samples $n' = n \cdot t$, and noise $w' = w \cdot t$ is $(T, \exp(-\Omega(\lambda)), \exp(-\Omega(\lambda)))$-secure, with $T' = \mathsf{poly}(T, \lambda, 1/\varepsilon, 1/\delta)$.*

Proof. Let $\mathcal{A}$ be an algorithm running in time T solving the $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension k, number of samples n, and noise w, such that $\Pr_A[\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0^A, \mathcal{D}_1^A) > \varepsilon] > \delta$. Set $t \leftarrow \lambda/(\delta\varepsilon^2)$. We show how to construct from $\mathcal{A}$ an algorithm $\mathcal{B}$ which is given black-box access to $\mathcal{A}$ and $(\mathsf{poly}(\lambda, T, 1/\varepsilon, 1/\delta), 1 - \exp(-\Omega(\lambda)), 1 - \exp(-\Omega(\lambda)))$-solves the $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension k, number of samples $n' = n \cdot t$, an noise $w' = w \cdot t$. We denote $\mathsf{Good}_\mu^{\mathcal{A}}$ the set of all matrices $A \in \mathbb{F}_2^{n \times k}$ such that $\mathsf{Adv}_{\mathcal{A}}(\mathcal{D}_0^A, \mathcal{D}_1^A) > \mu$.

On input a $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ instance $(A, \mathbf{b}) \in \mathbb{F}_2^{n' \times k} \times \mathbb{F}_2^{n'}$, let c denote the index of the distribution from which $(A, \mathbf{b})$ was sampled (*i.e.* $c = 0$ if $\mathbf{b}$ is an LPN sample, and $c = 1$ if $\mathbf{b}$ is uniform). $\mathcal{B}$ proceeds as follows:

- Break $(A, \mathbf{b})$ into $n'/n = t$ smaller instances $(A_i, \mathbf{b}_i)_{i \le t}$ with $(A_i, \mathbf{b}_i) \in \mathbb{F}_2^{k \times n}$.
- Set $S \leftarrow \emptyset$. For each $i \le t$, test whether $A_i \in \mathsf{Good}_\varepsilon^{\mathcal{A}}$ as follows:
  - Repeat $\theta = 32\lambda/\varepsilon^2$ times the following procedure: set $\mathsf{ctr} \leftarrow 0$. Sample a bit $\tilde{\sigma} \xleftarrow{\$} \{0,1\}$. If $\tilde{\sigma} = 0$, sample $(\mathbf{x}, \mathbf{e}) \xleftarrow{\$} \mathbb{F}_2^k \times \mathcal{X}_{n,w}|$ and set $\tilde{\mathbf{b}} \leftarrow A_i\mathbf{x} + \mathbf{e}$. If $\tilde{\sigma} = 1$, set $\tilde{\mathbf{b}} \xleftarrow{\$} \mathbb{F}_2^n$. If $\mathcal{A}(A_i, \tilde{\mathbf{b}}) = \tilde{\sigma}$, set $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$.
  - If $\mathsf{ctr} \ge \theta \cdot (1/2 + \varepsilon/4)$, declare A_i to be "good" and add i to S.
  - If $\lambda/(2\varepsilon^2)$ matrices have been declared "good" (i.e., $|S| = \lambda/(2\varepsilon^2)$), **break**.
- Set $B \leftarrow |S| \cdot (1/2 - \varepsilon/16)$. For each good A_i:
  - Sample $\mathbf{x}_i \xleftarrow{\$} \mathbb{F}_2^k$, $\mathbf{b}_i^1 \xleftarrow{\$} \mathbb{F}_2^n$, and set $\mathbf{b}_i^0 \leftarrow \mathbf{b}_i \oplus A_i\mathbf{x}_i$.
  - Flip a coin $c_i \xleftarrow{\$} \{0,1\}$ and compute $\sigma_i \leftarrow \mathcal{A}(A_i, \mathbf{b}_i^{c_i})$.
  - Output 1 if $\sum_{i \in S} \sigma_i \oplus c_i \le B$.

We prove that $\mathcal{B}$ is successful. We use a sequence of simple claims:

Claim. If $A_i \in \mathsf{Good}_\varepsilon^{\mathcal{A}}$, then $\Pr[\mathcal{B} \text{ declares } A_i \text{ good}] \ge 1 - \exp(-\lambda)$.

Proof. If $A_i \in \mathsf{Good}_\varepsilon^{\mathcal{A}}$, then $\Pr[\mathcal{A}(A_i, \tilde{\mathbf{b}}) = b] \ge 1/2 + \varepsilon/2$ by definition. It follows that $\mathbb{E}[\mathsf{ctr}] \ge (1/2 + \varepsilon/2) \cdot 32\lambda/\varepsilon^2$. By a Chernoff bound 2.2,

$$\Pr[\mathsf{ctr} < (1/2 + \varepsilon/4) \cdot 32\lambda/\varepsilon^2] < \exp\left(-32 \cdot \left(\frac{\varepsilon/2}{1+\varepsilon}\right)^2 \cdot \frac{\lambda(1+\varepsilon)}{4\varepsilon^2}\right) \le \exp(-\lambda).$$

□

Claim. Let $S \subseteq [n'/n]$ be the subset of indices i such that $A_i \in \mathsf{Good}_\varepsilon^{\mathcal{A}}$. Then $\Pr[|S| \le \lambda/(2\varepsilon^2)] \le \exp(-\lambda/8)$.

Proof. The A_i are sampled uniformly and independently from $\mathbb{F}_2^{n \times k}$, and $\Pr[A_i \in \mathsf{Good}_\varepsilon^{\mathcal{A}}] > \delta$ by assumption, hence $\mathbb{E}[|S|] > \delta \cdot n'/n = \lambda/\varepsilon^2$. By a Chernoff bound, it follows that

$$\Pr[|S| \le \lambda/(2\varepsilon^2)] \le \exp\left(-\left(\frac{1}{2}\right)^2 \frac{\lambda}{2\varepsilon^2}\right) < \exp(-\lambda/8).$$

□

Combining these two claims, it follows that $\mathcal{B}$ will declare at least $\lambda/(2\varepsilon^2)$ matrices A_i to be good, except with probability at most $\exp(-\lambda) + \exp(-\lambda/8)$.

Claim. If $A_i \notin \mathsf{Good}_{\varepsilon/4}^{\mathcal{A}}$, then $\Pr[\mathcal{B} \text{ declares } A_i \text{ good}] \le \exp(-4\lambda/15)$.

Proof. If $A_i \notin \mathsf{Good}^{\mathcal{A}}_{\varepsilon/4}$, then $\Pr[\mathcal{A}(A_i, \tilde{\mathbf{b}}) = b] \leq 1/2 + \varepsilon/8$ by definition. It follows that $\mathbb{E}[\mathsf{ctr}] \leq (1/2 + \varepsilon/8) \cdot 32\lambda/\varepsilon^2$. By a Chernoff bound 2.2,

$$\Pr[\mathsf{ctr} \geq (1/2 + \varepsilon/4) \cdot 32\lambda/\varepsilon^2] < \exp\left(-32 \cdot \left(\frac{\varepsilon}{4+\varepsilon}\right)^2 \cdot \frac{\lambda(1/2+\varepsilon/8)}{3\varepsilon^2}\right) \leq \exp(-4\lambda/15).$$

□

Therefore, by a straightforward union bound, with probability at least $1 - \exp(-\lambda) - \exp(-\lambda/8) - \lambda\exp(-4\lambda/15)/2 = 1 - \exp(-\Omega(\lambda))$,

- $\mathcal{B}$ declares $\lambda/(2\varepsilon^2)$ matrices to be good, and
- Every matrix A_i declared good by $\mathcal{B}$ belongs to $\mathsf{Good}^{\mathcal{A}}_{\varepsilon/4}$.

Then, for each A_i, observe that if $(A, \mathbf{b})$ is an LPN sample, $(A_i, \mathbf{b}_i^0 = \mathbf{b}_i \oplus A_i\mathbf{x}_i)$ is a uniformly random LPN sample, while if $\mathbf{b}$ is uniform, the $\mathbf{b}_i^0$ are uniform as well. That is,

- If $\mathbf{b}$ is uniform ($c = 1$), then $\mathbf{b}_i^0$ and $\mathbf{b}_i^1$ are identically distributed for every $i \in S$. Therefore, c_i is perfectly independent of σ_i, and $\sum_{i\in S} \sigma_i \oplus c_i$ is a sum of independent unbiased random coins.
- Else, if $\mathbf{b}$ is an LPN sample ($c = 0$), then distinguishing $(A_i, \mathbf{b}_i^0)$ from $(A_i, \mathbf{b}_i^1)$ is exactly distinguishing between $\mathcal{D}_0^{A_i}$ and $\mathcal{D}_1^{A_i}$. Since each $A_i \in \mathsf{Good}^{\mathcal{A}}_{\varepsilon/4}$ (with overwhelming probability), we have for every $i \in S$,

$$\Pr[\sigma_i \oplus c_i = 1 \mid c_i \xleftarrow{\$} \{0,1\}, \mathcal{A}(A_i, \mathbf{b}_i^{c_i})] \leq 1/2 - \varepsilon/8.$$

Therefore,

$$\begin{aligned}
2 \cdot \Pr[\mathcal{B} \text{ fails}] &= \Pr\left[\sum_{i\in S} \sigma_i \oplus c_i > B \mid c = 0\right] + \Pr\left[\sum_{i\in S} \sigma_i \oplus c_i \leq B \mid c = 1\right] \\
&\leq \exp\left(-\frac{|S| \cdot (1/2 - \varepsilon/8)}{3} \cdot \left(\frac{\varepsilon}{8-2\varepsilon}\right)^2\right) + \exp\left(-\frac{|S| \cdot 1/2}{2} \cdot \left(\frac{\varepsilon}{8}\right)^2\right) \\
&= \exp\left(-\frac{\lambda}{48 \cdot (8-2\varepsilon)}\right) + \exp\left(-\frac{\lambda}{256}\right) \leq \exp(-\Omega(\lambda)),
\end{aligned}$$

where the first inequality follows by applying a Chernoff bound on both terms of the sum. This concludes the proof that $\mathcal{B}$ is successful. □

4.6 Variants: changing the noise or matrix distribution

We focused in the above on the noise distribution $\mathcal{X}$ since it is the most convenient to use in our constructions. However, the same analysis and conjectures apply equivalently to the other standard noise distributions $(\mathcal{B}, \mathcal{S}, \mathcal{R})$ (the analysis in the linear test framework gives identical bounds). Furthermore, since in our regime we typically have $n = k^{1+\eta}$ with $\eta < 1$, it is not too hard to provide tight reductions between $(\mathcal{S}^c, \mathcal{X}_{n,w})$-LPN and $(\mathcal{S}^c, \mathcal{S}_{n,w})$-LPN and/or $(\mathcal{S}^c, \mathcal{B}_{n,w})$-LPN.

More interestingly, one can also consider a different distributions over sparse matrices, in the hope of getting better bounds on the dual distance. And indeed, such a distribution was exhibited in the work of Applebaum and Kachlon [AK19]: Theorem 8.2 in [AK19] states that for every constant $\ell > 1$, $c > 4\ell$, there exists a *negligible-error* polynomial-time algorithm that samples matrices A over $\mathbb{F}_2^{n\times k}$, with $n = k^\ell$ rows of weight c with dual distance $\mathsf{dd}(A) \geq \Omega(k^\eta)$ for some suitable constant $\eta(\ell, c) \leq 1 - 4(\ell-1)/(c-4)$. On the flip side, this sampler achieves only a *slightly* negligible error probability.

4.7 Predicate-conditioned sparse-LPN

We now state a result that will prove useful in our analysis later. Let $\mathcal{P} = \{\mathsf{P} : \mathbb{F}_2^n \to \{0,1\}\}$ denote a family of predicates. For a noise distribution $\mathcal{E}$, let us denote $\mathcal{E}|_{\mathsf{P}}$ the distribution $\{\mathbf{e} : \mathbf{e} \xleftarrow{\$} \mathcal{E} \mid \mathsf{P}(\mathbf{e}) = 1\}$ (that is, $\mathcal{E}|_{\mathsf{P}}$ samples vectors from $\mathcal{E}$ conditioned on $\mathsf{P}(\mathbf{e}) = 1$). Fix a predicate P and consider the following assumption (for suitable parameters (T, ε, δ)):

Definition 4.8 ($\mathcal{P}$-conditioned sparse-LPN). *For any constants $c \geq 3$ and $\eta \in (0,1)$, for every $T = 2^{o(\lambda)}$, we say that the $\mathcal{P}$-conditioned $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = n(k) \leq k^{(1-\eta)c/2+\eta}$, and noise $w = w(\lambda, k) \leq \lambda \cdot k^{(1-\eta)c/2}$ is (T, ε, δ)-hard if for every probabilistic adversary $\mathcal{A} = (\mathcal{A}_\lambda)_{\lambda \in \mathbb{N}}$ of size at most $T = T(\lambda)$, it holds that for all large enough k,*

$$\Pr_{A \xleftarrow{\$} \mathcal{S}^c, \mathsf{P} \xleftarrow{\$} \mathcal{P}} \left[\mathsf{Adv}_{\mathcal{A}_k}(\mathcal{D}_0^{A,\mathsf{P}}, \mathcal{D}_1^{A,\mathsf{P}}) > \varepsilon \right] \leq \delta,$$

where $\mathcal{D}_0^{A,\mathsf{P}}$ denotes the distribution $\{(A, \mathsf{P}, A \cdot \mathbf{s} + \mathbf{e}) \mid \mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k, \mathbf{e} \xleftarrow{\$} \mathcal{X}_{n,w}|_{\mathsf{P}}\}$ and $\mathcal{D}_1^{A,\mathsf{P}}$ denotes $\{(A, \mathsf{P}, \mathbf{b}) \mid \mathbf{b} \xleftarrow{\$} \mathbb{F}_2^n\}$.

Reducing $\mathcal{P}$-conditioned sparse-LPN to sparse-LPN. In this section, we prove a reduction between $\mathcal{P}$-conditioned sparse-LPN and sparse-LPN. The quality of the reduction depends on the quantity $\mathsf{err}(\mathcal{P}) = \max_{\mathbf{e}} \Pr_{\mathsf{P} \xleftarrow{\$} \mathcal{P}}[\mathsf{P}(\mathbf{e}) \neq 1]$.

Lemma 4.9. *Assume that the $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = k^{(1-\eta)c/2+\eta}$, and noise $w = \lambda \cdot k^{(1-\eta)c/2}$ is $(T', \varepsilon, \delta)$-secure. Then the $\mathcal{P}$-conditioned $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = k^{(1-\eta)c/2+\eta}$, and noise $w = \lambda \cdot k^{(1-\eta)c/2}$ is $(O(T), \varepsilon + \mathsf{err}(\mathcal{P}), \delta)$-secure.*

Proof. Let $\mathcal{A}$ be a (T, ε, δ)-adversary against the $\mathcal{P}$-conditioned $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem with dimension $k = k(\lambda)$, number of samples $n = k^{(1-\eta)c/2+\eta}$, and noise $w = \lambda \cdot k^{(1-\eta)c/2}$. We build an adversary $\mathcal{B}$ against the $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem as follows: given as input a $(\mathcal{S}^c, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem $(A, \mathbf{b})$, $\mathcal{B}$ samples $\mathsf{P} \xleftarrow{\$} \mathcal{P}$ and feeds $(A, \mathsf{P}, \mathbf{b})$ to $\mathcal{B}$. Observe that

- If $\mathbf{b}$ is uniformly random, then $(A, \mathsf{P}, \mathbf{b})$ is a valid random $\mathcal{P}$-conditioned instance.
- Else, if $\mathbf{b}$ is an LPN sample $\mathbf{b} = A \cdot \mathbf{x} + \mathbf{e}$, then the distribution of $(A, \mathsf{P}, \mathbf{b})$ *conditioned on* $\mathsf{P}(\mathbf{e}) = 1$ is a valid $\mathcal{P}$-conditioned LPN instance.

For $\sigma = 0, 1$, let us denote

$$p_\sigma(A, \mathsf{P}) = \Pr_{(A, \mathsf{P}, \mathbf{b}) \xleftarrow{\$} \mathcal{D}_\sigma^{A,\mathsf{P}}} [\mathcal{A}(A, \mathsf{P}, \mathbf{b}) = 0].$$

We have $\mathsf{Adv}^{\mathcal{A}}(\mathcal{D}_0^{A,\mathsf{P}}, \mathcal{D}_1^{A,\mathsf{P}}) = |p_0(A, \mathsf{P}^t) - p_1(A, \mathsf{P}^t)|$. Then:

$$\begin{aligned}
&\Pr_A \left[\mathsf{Adv}^{\mathcal{B}}(\mathcal{D}_0^A, \mathcal{D}_1^A) > \varepsilon' \right] \\
&= \Pr_{A,\mathsf{P}} \left[|\Pr[\mathcal{A}(A, \mathsf{P}, \mathbf{b}) = \ \mid \mathbf{b} \text{ random}] - \Pr[\mathcal{B}(A, \mathsf{P}, \mathbf{b}) = \ \mid \mathbf{b} \text{ LPN}]| > \varepsilon' \right] \\
&\leq \max_{x \in [0,1]} \Pr_{A,\mathsf{P}} \left[|p_1(A, \mathsf{P}) - p_0(A, \mathsf{P}) \cdot (1 - \mathsf{err}(\mathcal{P})) - x \cdot \mathsf{err}(\mathcal{P})| > \varepsilon' \right] \text{ by the Bayes rule} \\
&\leq \Pr_{A,\mathsf{P}} [|p_1(A, \mathsf{P}) - p_0(A, \mathsf{P})| - \mathsf{err}(\mathcal{P}) > \varepsilon'] \text{ by triangle inequality} \\
&= \Pr_{A,\mathsf{P}} \left[\mathsf{Adv}^{\mathcal{A}}(\mathcal{D}_0^{A,\mathsf{P}}, \mathcal{D}_1^{A,\mathsf{P}}) > \varepsilon' + \mathsf{err}(\mathcal{P}) \right] \leq \delta,
\end{aligned}$$

and we conclude the proof by setting $\varepsilon' = \varepsilon - \mathsf{err}(\mathcal{P})$.

5 A Structured-Seed Local PRG from Sparse LPN

Our constructions rely on a simple method to compress a length-ℓ unit vector $\mathbf{u}$ into d smaller unit vectors $(\mathbf{u}_1, \cdots, \mathbf{u}_d)$ of length $\ell^{1/d}$ such that each entry of $\mathbf{u}$ can be reconstructed by retrieving a single entry from each $\mathbf{u}_i$.

5.1 Compressing unit vectors

Let $\mathsf{unit}_m(i)$ denote the procedure which, on input $i \in [m]$, outputs a length-m one-hot $\mathbb{F}_2$-vector with a 1 at position i. Conversely, let $\mathsf{nzi}(\mathbf{u})$ denote the procedure which, given as input a length-m one-hot $\mathbb{F}_2$-vector $\mathbf{u}$, returns its non-zero index i. We describe the compression and reconstruction algorithms below:

Algorithms $(\mathsf{Comp}, \mathsf{Rec})$:

$\mathsf{Comp}(\mathbf{u}, d)$: on input a length-$\ell$ unit vector $\mathbf{u}$ and a compression factor d,
1. Set $\ell_d \leftarrow \lceil \ell^{1/d} \rceil$.
2. Compute $i \leftarrow \mathsf{nzi}(\mathbf{u})$ and write i over the ℓ_d-ary basis as $(i_1, \cdots, i_d) \in [\ell_d]^d$.
3. Output $(\mathbf{u}_1, \cdots, \mathbf{u}_d) \leftarrow (\mathsf{unit}_{\ell_d}(i_1), \cdots, \mathsf{unit}_{\ell_d}(i_d))$.

$\mathsf{Rec}(j, (\mathbf{u}_1, \cdots, \mathbf{u}_d))$: on input an index $j \in [\ell]$ and d one-hot $\mathbb{F}_2$ vectors $\mathbf{u}_i \in \mathbb{F}_2^{\ell_d}$, with $\ell_d^d \geq \ell$,
1. Write j over the ℓ_d-ary basis as $(j_1, \cdots, j_d) \in [\ell_d]^d$.
2. Return $b \leftarrow \prod_{i=1}^{d} u_i[j_i]$. // AND operation over $\mathbb{F}_2$.

We note in passing that since $\mathsf{Comp}(\mathbf{u}, d)$ starts by computing $i = \mathsf{nzi}(\mathbf{u})$, it never needs to store or read $\mathbf{u}$ in "expanded form" and can be passed i directly. We will make use of this observation when estimating the running time of our algorithms. From the description above, it is clear that $\mathsf{Rec}(j, (\mathbf{u}_1, \cdots, \mathbf{u}_d))$ always reads exactly d bits from its second input $(\mathbf{u}_1, \cdots, \mathbf{u}_d)$: for every j, $\mathsf{Rec}(j, \cdot)$ is d-local. Furthermore, it is easy to observe that on any input $(j, \mathsf{Comp}(\mathbf{u}, d))$, Rec correctly reconstructs the j-th bit of $\mathbf{u}$:

Claim. For every d and length-ℓ one-hot $\mathbb{F}_2$-vector $\mathbf{u}$, for every $j \leq \ell$, it holds that

$$\mathsf{Rec}(j, \mathsf{Comp}(\mathbf{u}, d)) = \mathbf{u}[j].$$

We say that $(\mathsf{Comp}, \mathsf{Rec})$ is *correct* to denote this property.

5.2 Warm-up: a structured-seed local PRG from regular sparse LPN

Let $(w, k) = (w(\lambda), k(\lambda))$ denote respectively the noise weight and dimension of an LPN instance, and let $n = n(k)$ be the (polynomial) number of samples, chosen such that w divides n. Let $c \geq 3$ and d be two constants. Let $\ell \leftarrow n/w$ and $\ell_d \leftarrow \lceil (n/w)^{1/d} \rceil$. We describe below a structured-seed local PRG whose security reduces to the hardness of the regular sparse-LPN assumption:

- Global parameters: two constants (c, d), the noise weight $w = w(\lambda)$ of, the dimension $k = k(\lambda)$, and the stretch $n = n(k)$. All global parameters are implicitly passed as inputs to all algorithms.
- $\mathsf{Setup}(1^\lambda)$: sample $A \xleftarrow{\$} \mathcal{M}_{k,n}^c$. Let $(\mathbf{a}_1, \cdots, \mathbf{a}_n)$ denote the rows of A (of Hamming weight c). Output $\mathsf{pp} \leftarrow A$.
- $\mathsf{SampleSeed}(\mathsf{pp})$: sample $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k$ and w unit vectors $(\mathbf{e}_1, \cdots, \mathbf{e}_w) \xleftarrow{\$} \mathcal{U}_{n/w} \times \cdots \times \mathcal{U}_{n/w}$. Output $\mathsf{seed} \leftarrow (\mathbf{x}, \mathsf{Comp}(\mathbf{e}_1, d), \cdots, \mathsf{Comp}(\mathbf{e}_w, d))$.
- $\mathsf{PRG}_{\mathsf{pp}}(\mathsf{seed})$: parse seed as $(\mathbf{x}, (\mathbf{u}_{1,1}, \cdots, \mathbf{u}_{1,d}), \cdots, (\mathbf{u}_{w,1}, \cdots, \mathbf{u}_{w,d}))$, where each $\mathbf{u}_{i,j}$ is a unit vector over $\mathbb{F}_2^{\ell_d}$ for every $i \leq w$, $j \leq d$. For $i = 1$ to n, write i as $(\alpha - 1) \cdot (n/w) + \beta$, with $\alpha \in [w]$ and $\beta \in [n/w]$. set $y_i \leftarrow \langle \mathbf{a}_i, \mathbf{x} \rangle + \mathsf{Rec}(\beta, (\mathbf{u}_{\alpha,1}, \cdots, \mathbf{u}_{\alpha,d}))$. Output $(y_1, \cdots, y_n)$.

Theorem 5.1. *Assuming the (T, ε, δ)-hardness of the $(\mathcal{M}^c, \mathcal{R})$-$\mathsf{LPN}$ problem, for any constant $d \geq 3$, $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ is a $(T - O(n), \varepsilon, \delta)$-secure structured-seed local pseudorandom generator with seed length $k + wd \cdot \lceil (n/w)^{1/d} \rceil$, stretch n, and locality $c + d$.*

For example, setting $k = w$ and $n(k) = k^{1.99}$ yields a PRG with seed length $s = O(k^{1+0.99/d})$ and stretch $k^{1.99} = \Omega(s^{1.99/(1+0.99/d)})$; for $d = 10$, this translates to a stretch $\Omega(s^{1.81})$. In general, as n approaches k^2 and d grows, the stretch becomes $\Omega(s^{2-\varepsilon_d})$ for an arbitrarily small constant ε_d.

Proof. Let $\mathbf{e}$ denote the regular vector obtained by concatenating $(\mathbf{e}_1, \cdots, \mathbf{e}_w)$. Note that $\mathbf{e}$ is distributed as a random sample from $\mathcal{R}_n^w$. By correctness of $(\mathsf{Comp}, \mathsf{Rec})$, $\mathsf{Rec}(\beta, (\mathbf{u}_{\alpha,1}, \cdots, \mathbf{u}_{\alpha,d})) = \mathbf{e}_\alpha[\beta] = \mathbf{e}[i]$ (the β-th entry of the α-th block of $\mathbf{e}$ is exactly the i-th entry of $\mathbf{e}$ as $i = (\alpha-1)\cdot(n/w)+\beta$). Therefore, denoting $\mathbf{y} = (y_1, \cdots, y_n)$,

$$\mathbf{y} = (\langle \mathbf{a}_i, \mathbf{x} \rangle + \mathbf{e}[i])_{i \leq n} = A \cdot \mathbf{x} + \mathbf{e}.$$

From there, it follows immediately that breaking the $(T - O(n), \varepsilon, \delta)$-security of $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ translates to breaking the (T, ε, δ)-hardness of the Alekhnovich assumption (the reduction is straightforward; the $O(n)$ term accounts for the cost of sampling the LPN instance and compressing the unit vectors). Eventually, $(c+d)$-locality follows from the fact that each $\mathbf{a}_i$ is c-sparse, hence the mapping $\mathbf{x} \mapsto \langle \mathbf{a}_i, \mathbf{x} \rangle$ is c-local, and the mapping $\mathsf{Rec}(\beta, \cdot)$ is d-local for any $\beta \in [n/w]$. The theorem follows.

5.3 Removing regularity using 2-choice hashing

The construction presented in the previous section requires the sparse-LPN assumption to be secure when the noise has a regular structure. In this section, we explain how to lift this restriction using efficient hashing schemes for allocating elements into bins. We use 2-choice hashing [CRS03, SEK03] for the sake of concreteness, but we note that our construction can be framed generically using the language of batch codes [IKOS04]. Replacing regular noise with random sparse noise in the LPN variant using hashing or batch codes has been done in previous works [BCGI18, SGRR19, BBC+24]. Here, we show how to integrate this approach into our structured-seed local PRG without sacrificing constant locality.

Hash functions are commonly used to distribute items into bins. In its simplest form, N items $(u_1, \cdots, u_N)$ from a universe $\mathbb{U}$ can be placed into $L \cdot N$ bins $(1, \cdots, N)$ (for a suitable constant L) using a hash function $h : \mathbb{U} \to [L \cdot N]$, by placing each item u_i at position $h(i)$; when h is a random function, it is well known that with high probability, the maximum load across all bins will be $O(\log N / \log \log N)$ [RS98]. The question of finding alternative hashing strategies that result in a more balanced load has been an active and fruitful field of research [PSWW18, PRTY19, SGRP19]. Typically, these improved strategies rely on multiple hash functions $(h_1, h_2, \cdots)$, combined with an *allocation scheme* to determine, for each item u, which hash function should be used to allocate u to a bin.

For our application, we will need a hashing scheme that guarantees, with probability $1 - O(1)$ (over the random choice of the hash functions, for an arbitrary set of items to be placed) that each bin will contain a constant number of items. There are multiple options with different degrees of simplicity and parameter tradeoffs. For the sake of concreteness, we focus on one of the simplest possible solutions, called *2-choice hashing* [SEK03].

Definition 5.2 (Allocation). *Let $\mathbb{U}$ be a set, L be a constant, and N be an integer. Let h_0, h_1 be two hash functions from $\mathbb{U}$ to $[L \cdot N]$. For any N-tuple $\mathbf{u} = (u_1, \cdots, u_N) \in \mathbb{U}^N$, we define an* allocation *of $\mathbf{u}$ into the bins $1 \cdots N$ with respect to (h_0, h_1) to be a vector $\mathbf{b} \in \mathbb{F}_2^{L \cdot N}$ indicating which bin each item is mapped to: for any $i \in [N]$, the item u_i is mapped to the bin $h_{\mathbf{b}[i]}(u_i)$. Given (h_0, h_1), a tuple $\mathbf{u}$, and an allocation $\mathbf{b}$, we let $\mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b})$ denote the load of the bin i (*i.e. *the total number of indices j such that $h_{\mathbf{b}[j]}(u_j) = i$).*

We will rely on the following lemma:

Lemma 5.3 (2-choice hashing [SEK03, PRTY19]). *Let $\mathbb{U}$ be a set, L be a constant, and N be an integer. Then, there exists a deterministic algorithm Alloc running in time $O(N \log N)$ which, on input two hash functions h_0, h_1 from $\mathbb{U}$ to $[L \cdot N]$ and an N-tuple $\mathbf{u} \in \mathbb{U}^N$, returns an allocation $\mathbf{b}$, and such that for every $\mathbf{u} \in \mathbb{U}^N$,*

$$\Pr_{h_0, h_1}\left[\mathbf{b} \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u}) \ : \ \max_{i \leq L \cdot N} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) > L + 1\right] \leq O\left(\frac{1}{N^L}\right).$$

For notational convenience, we will define the *quality* of a pair of hash function (h_0, h_1) as the fraction of vectors $\mathbf{u} \in \mathbb{U}^N$ that have a good allocation (*i.e.*, such that $\max_{i \leq L \cdot N} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) \leq L + 1$):

Definition 5.4 (Quality). *We call* quality *of a pair* (h_0, h_1) *of functions from* $\mathbb{U}$ *to* $[L \cdot N]$*, and denote* $\mathsf{Quality}(h_0, h_1)$*, the quantity*

$$\mathsf{Quality}(h_0, h_1) := \Pr_{\mathbf{u} \xleftarrow{\$} \mathbb{U}^N} \left[\mathbf{b} \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u}) \ : \ \max_{i \le L \cdot N} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) > L + 1 \right].$$

Then, Lemma 5.3 says that on average, a random choice of (h_0, h_1) has quality $1 - O(1/N^L)$.

5.4 Sampling the seed

As for the regular construction, we assume that all algorithms implicitly receive as input global parameters $\mathsf{gp} = (w, k, n, c, d, L)$ where $w = w(\lambda)$ is the noise weight parameter, $k = k(\lambda)$ is the LPN dimension, $n = n(k)$ denotes the stretch (or number of samples), and $c, d \ge 3$ and $L \ge 1$ denote three constants.

At a high level, our construction proceeds by using 2-choice hashing to allocate the nonzero entries of the noise vector into unit vectors, such that every entry of the noise vector can be reconstructed by looking at one position in $2(L+1)$ unit vectors, and compressing these unit vectors with the compression algorithm Comp. In more details, assume that the setup $\mathsf{Setup}(1^\lambda)$ produces a matrix $A \xleftarrow{\$} \mathcal{M}^c_{k,n}$ together with two hash functions (h_0, h_1) (we ignore for now the issue of how "good" hash functions (h_0, h_1) are selected). Then, $\mathsf{SampleSeed}(\mathsf{pp})$ proceeds as follows:

Algorithm $\mathsf{SampleSeed}(A, h_0, h_1)$:

1. Sample $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^k$.
2. Sample (the positions of) a noise vector $\mathbf{u} \xleftarrow{\$} [n]^w$.
3. Set $\mathbf{b} \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u})$. If $\max_{i \le Lw} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) > L + 1$, **go to** Item 2.
4. For $i = 1$ to Lw, define $\mathbf{v}_i \in \mathbb{F}_2^n$ as follows:

$$\mathbf{v}_i \leftarrow \bigoplus_{j : h_{\mathbf{b}[j]}(\mathbf{u}[j]) = i} \mathsf{unit}_n(\mathbf{u}[j]).$$

   That is, for every entry j of $\mathbf{u}$ mapped to the bin i via the allocation (computed as $h_{\mathbf{b}[j]}(\mathbf{u}[j])$), sum 1 to the position $\mathbf{u}[j]$ in $\mathbf{v}_i$. Note that by construction, $\mathsf{HW}(\mathbf{v}_i) \le L + 1$.
5. Write $\mathbf{v}_i$ as a sum of $L+1$ (unit or zero) vectors $\mathbf{v}_i = \mathbf{v}_i^0 + \cdots + \mathbf{v}_i^L$ with $\mathsf{HW}(\mathbf{v}_i^\ell) \le 1$ for $\ell \in \{0, \cdots, L\}$.
6. Output $\mathsf{seed} \leftarrow (\mathbf{x}, (\mathsf{Comp}(\mathbf{v}_i^0, d), \cdots, \mathsf{Comp}(\mathbf{v}_i^L, d))_{i \le Lw})$.

Note that $\mathsf{SampleSeed}$ does not have to work with an expanded representation of the vectors $\mathbf{v}_i$: the $\mathbf{v}_i$ can be manipulated in compact form (as the list of their nonzero entries, of size at most 2) throughout the entire execution (including the execution of Comp, since for any $\mathbf{v}$, $\mathsf{Comp}(\mathbf{v}, d)$ only needs the nonzero entry $i = \mathsf{nzi}(\mathbf{v})$, as noted in Section 5.1).

5.5 Expanding the seed

Let us denote $\mathbf{e} = \bigoplus_{j'=1}^{w} \mathsf{unit}_n(\mathbf{u}[j'])$ the noise vector in expanded form. Observe that in the $\mathsf{SampleSeed}$ process above, every vector $\mathsf{unit}_n(\mathbf{u}[j'])$ is XORed to exactly one $\mathbf{v}_i$: the one with index $i = h_{\mathbf{b}[j']}(\mathbf{u}[j'])$. Therefore, for every position $j \in [n]$ of $\mathbf{e}$, two cases can occur:

1. Either there exists j' such that $j = \mathbf{u}[j']$ (*i.e.*, j is a noise position of $\mathbf{e}$). Then the j-th entry of $\mathbf{v}_i$ with $i = h_{\mathbf{b}[j']}(\mathbf{u}[j'])$ is set to 1, and the j-th entry of the alternative option $\mathbf{v}_{i'}$, where $i' = h_{\bar{\mathbf{b}}[j']}(\mathbf{u}[j'])$ (i' is the index of the "other bin", not selected by the allocation) stays at 0.
2. Or there is no such j' (*i.e.*, j is not a noise position), in which case both $\mathbf{v}_{i_0}[j]$ and $\mathbf{v}_{i_1}[j]$ are equal to 0, where $(i_0, i_1) = (h_0(j), h_1(j))$.

In both cases, it holds that $\mathbf{e}[j] = \mathbf{v}_{i_0}[j] \oplus \mathbf{v}_{i_1}[j]$, with $(i_0, i_1) = (h_0(j), h_1(j))$. Hence, to "read" $\mathbf{e}[j]$, it suffices to read the j-th entry in $\mathbf{v}_{i_0}$ and $\mathbf{v}_{i_1}$ for $(i_0, i_1) = (h_0(j), h_1(j))$. This can be done by reading d entries in each of $\mathsf{Comp}(\mathbf{v}_{i_0}^0, d), \cdots, \mathsf{Comp}(\mathbf{v}_{i_0}^L, d), \mathsf{Comp}(\mathbf{v}_{i_1}^0, d), \cdots, \mathsf{Comp}(\mathbf{v}_{i_1}^L, d)$ (hence $2d(L+1)$

entries in total) by the d-locality of $(\mathsf{Comp}, \mathsf{Rec})$. Concretely, given parameters $\mathsf{pp} = (A, h_0, h_1)$ and a seed $\mathsf{seed} = (\mathbf{x}, (\mathsf{Comp}(\mathbf{v}_i^0, d), \cdots, \mathsf{Comp}(\mathbf{v}_i^L, d))_{i \le Lw})$, the j-th entry of the noise vector $\mathbf{e} \in \mathbb{F}_2^n$ (whose nonzero entries are given by $\mathbf{u}$) is reconstructed as follows: compute $(i_0, i_1) \leftarrow (h_0(j), h_1(j))$. Set

$$\mathbf{e}[j] \leftarrow \bigoplus_{\alpha \le L, \beta \in \{0,1\}} \mathsf{Rec}(j, \mathsf{Comp}(\mathbf{v}_{i_\beta}^\alpha, d)) = \bigoplus_{\alpha \le L, \beta \in \{0,1\}} \mathbf{v}_{i_\beta}^\alpha[j] = \mathbf{v}_{i_0}[j] \oplus \mathbf{v}_{i_1}[j].$$

The detailed procedure is represented below.

Algorithm $\mathsf{PRG}_{\mathsf{pp}}(\mathbf{x}, (\mathsf{Comp}(\mathbf{v}_i^0, d), \mathsf{Comp}(\mathbf{v}_i^1, d))_{i \le w})$:

1. Parse pp as (A, h_0, h_1).
2. For $j = 1$ to n, set $(i_0^j, i_1^j) \leftarrow (h_0(j), h_1(j))$.
3. For $j = 1$ to n, set
$$y_j \leftarrow \langle \mathbf{a}_j, \mathbf{x} \rangle + \sum_{\substack{0 \le \alpha \le L \\ \beta \in \{0,1\}}} \mathsf{Rec}\left(j, \mathsf{Comp}\left(\mathbf{v}_{i_\beta^j}^\alpha, d\right)\right).$$
4. Output $(y_1, \cdots, y_n)$.

To complete the construction, it remains to explain how (h_0, h_1) are sampled by Setup. Concretely, the algorithm $\mathsf{SampleSeed}$ requires (h_0, h_1) to be "good" in the following sense: for a random choice of noise positions $\mathbf{u} \xleftarrow{\$} [n]^w$ in step 2, it should hold with sufficiently large probability p that $\max_{i \le Lw} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) \le L + 1$ using the allocation $\mathbf{b} \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u})$, since the expected runtime of $\mathsf{SampleSeed}$ grows as $1/p$. In other words, we want $\mathsf{Quality}(h_0, h_1)$ to be sufficiently large (with overwhelming probability over the choice of (h_0, h_1)).

Lemma 5.3 guarantees that for a *fixed choice of* $\mathbf{u}$, a random choice of (h_0, h_1) has probability close to 1 of yielding a good allocation. Looking ahead, our Setup procedure builds upon Lemma 5.3 to produce with overwhelming probability a pair (h_0, h_1) that yield a good allocation for *a constant fraction* of all $\mathbf{u}$'s (*i.e.*, $\mathsf{Quality}(h_0, h_1)$ is bounded below by a constant). This bounds p by a constant, causing only a constant blowup to the expected runtime of $\mathsf{SampleSeed}$[4].

5.6 Testing the hash functions

We let $\mathcal{F}_{n,Lw}$ denote the set of all functions from $[n]$ to $[Lw]$. Our setup procedure builds upon a Test subroutine to test whether pairs of hash functions (h_0, h_1) are sufficiently good. The procedure is represented below:

Algorithm $\mathsf{Test}(1^\lambda, h_0, h_1)$:

1. Set $\mathsf{good} \leftarrow 0$ and $T \leftarrow 42 \cdot \lambda$. For $j = 1$ to T,
   (a) Sample $\mathbf{u}_j \xleftarrow{\$} [n]^w$.
   (b) Set $\mathbf{b}_j \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u}_j)$.
   (c) If $\max_{i \le Lw} \mathsf{Load}_i(h_0, h_1, \mathbf{u}_j, \mathbf{b}_j) \le L + 1$, set $\mathsf{good} \leftarrow \mathsf{good} + 1$.
2. If $\mathsf{good} \ge 0.3 \cdot T$, output 1. Else, output 0.

In other words, the procedure Test computes an empirical estimate of the quality of (h_0, h_1) on a random test set of $T = 42 \cdot \lambda$ vectors $\mathbf{u}$, and outputs 1 iff the empirical quality is at least 30% (see Lemma 5.5 for a detailed explanation of the choice of number of vectors in the random test set and the empirical quality). Our Setup algorithm will rely on two properties of Test:

[4] We note that one can make the runtime strictly polytime by adding a bound λ on the number of retries, where lambda is some security parameter, and aborting if the bound is reached. This gives a strict polytime algorithm with a λ blowup in runtime and a negligible failure probability $(1-p)^\lambda$.

1. (few false positive) the probability (over the randomness of Test) that $\mathsf{Test}(1^\lambda, h_0, h_1) = 1$ but $\mathsf{Quality}(h_0, h_1) < 0.2$ is negligible;
2. (high success rate) the probability (over the random choice of (h_0, h_1) and the randomness of Test) that $\mathsf{Test}(h_0, h_1) = 1$ is bounded below by a constant.

Both proofs are elementary applications of standard tail bounds; we prove them in Section 5.8.

5.7 Sampling the hash functions

The setup procedure is represented below.

Algorithm $\mathsf{Setup}(1^\lambda, k, n)$:

1. Sample $A \xleftarrow{\$} \mathcal{M}^c_{k,n}$. Let $(\mathbf{a}_1, \cdots, \mathbf{a}_n)$ denote the rows of A (of Hamming weight c).
2. Sample $(h_0, h_1) \xleftarrow{\$} \mathcal{F}_{n,Lw} \times \mathcal{F}_{n,Lw}$.
3. If $\mathsf{Test}(1^\lambda, h_0, h_1) = 0$, **go to** Item 2.
4. Output $\mathsf{pp} \leftarrow (A, h_0, h_1)$.

In Section 5.8, we will prove two lemmas: Lemma 5.6 shows that Test succeeds with probability at least $1/2$ ("Test succeeds often enough") and Lemma 5.5 shows that (h_0, h_1) have good quality $\mathsf{Quality}(h_0, h_1) \geq 0.2$ with overwhelming probability ("Test has few false positives"). We discuss consequences for the running time of Setup and $\mathsf{SampleSeed}$ below.

Since each Test succeeds with probability at least $1/2$ by Lemma 5.6, $\mathsf{Setup}(1^\lambda)$ executed a constant expected number of Test (alternatively, we can let Setup run up to λ tests and abort if none succeeded to get a strict bound on the running time and negligible abortion probability). Furthermore, by Lemma 5.5, the probability that the functions (h_0, h_1) output by $\mathsf{Setup}(1^\lambda)$ have quality $\mathsf{Quality}(h_0, h_1) < 0.2$ is at most $1/2^\lambda$. In turn, this implies that the algorithm $\mathsf{SampleSeed}$ will succeed in sampling $\mathbf{u}$ in step 3 after at most 5 tries in expectation.

5.8 Properties of Test

Test has few false positives. We prove the following:

Lemma 5.5. *Let (h_0, h_1) be two functions from $\mathcal{F}_{n,Lw}$ such that $\mathsf{Quality}(h_0, h_1) < 0.2$. Then,*

$$\Pr\left[\mathsf{Test}(1^\lambda, h_0, h_1) = 1\right] \leq \frac{1}{2^\lambda}.$$

Proof. Let (h_0, h_1) be two functions from $\mathcal{F}_{n,Lw}$ such that $\mathsf{Quality}(h_0, h_1) < 0.2$. Let X_j denote the random variable, for $j = 1$ to T, taking value 1 if $\max_{i \leq Lw} \mathsf{Load}_i(h_0, h_1, \mathbf{u}_j, \mathbf{b}_j) \leq L + 1$ and 0 else. Note that the random variables $X_1, \cdots, X_T$ are independent. Let $X \leftarrow \sum_{j=1}^T X_j$ and $\mu \leftarrow \mathbb{E}[X]$. Note that $\mu = T \cdot \mathbb{E}[X_1] = T \cdot \mathsf{Quality}(h_0, h_1) < 0.2T$. Let us denote $x \leftarrow \mu/T < 0.2$. Then, by the Chernoff bound (Lemma 2.2),

$$\begin{aligned}
\Pr[X \geq 0.3T] &\leq \exp\left(-\left(\frac{0.3T}{\mu} - 1\right)^2 \cdot \frac{\mu}{3}\right) \\
&= \exp\left(-\frac{0.03T^2}{\mu} + 0.2T - \frac{\mu}{3}\right) \\
&= \exp\left(-\left(\frac{0.03}{x} - 0.2 + \frac{x}{3}\right) \cdot T\right).
\end{aligned}$$

Now, writing $f(x) = 0.03/x - 0.2 + x/3$, we have $f'(x) = (1 - (0.3/x)^2)/3 < 0$ since $x < 0.3$. Hence, $f(x)$ is decreasing and bounded above by $f(0.2) = 1/60$, which yields

$$\Pr[X \geq 0.3T] \leq \exp\left(-T/60\right) = 2^{-42 \cdot (\log_2 e/60) \cdot \lambda} \leq 2^{-\lambda},$$

which concludes the proof. □

Test succeeds often enough. We prove the following:

Lemma 5.6. *There exists an integer N such that for all $w, \lambda \geq N$*

$$\Pr_{h_0,h_1 \xleftarrow{\$} \mathcal{F}_{n,Lw}} \left[\mathsf{Test}(1^\lambda, h_0, h_1) = 1\right] \geq \frac{1}{2}.$$

Proof. The proof follows immediately from the following (stronger) claim:

Claim. There exists an integer N such that for all $w \geq N$,

$$\Pr_{h_0,h_1 \xleftarrow{\$} \mathcal{F}_{n,Lw}} [\mathsf{Quality}(h_0, h_1) \geq 0.4] \geq \frac{1}{1.9}.$$

Furthermore, if (h_0, h_1) are two functions from $\mathcal{F}_{n,Lw}$ such that $\mathsf{Quality}(h_0, h_1) \geq 0.4$,

$$\Pr\left[\mathsf{Test}(1^\lambda, h_0, h_1) = 0\right] \leq \frac{1}{2^{9\lambda}}.$$

We prove each part of the stronger claim in turn. The first part of claim is an immediate application of the Markov inequality: by Lemma 5.3 and Markov inequality (Lemma 2.3),

$$\Pr_{h_0,h_1 \xleftarrow{\$} \mathcal{F}_{n,Lw}} \left[\mathsf{Quality}(h_0, h_1) \geq \frac{1 - O(1/w^L)}{1.9}\right] \geq \frac{1}{1.9},$$

and for a large enough w, $(1 - O(1/w^L))/1.9 \geq 0.9/1.9 > 0.4$. For the second claim, the proof is similar to that of (1), with a Chernoff bound in the other direction: let (h_0, h_1) be two functions from $\mathcal{F}_{n,Lw}$ such that $\mathsf{Quality}(h_0, h_1) < 0.2$. Then by the Chernoff bound,

$$\begin{aligned}
\Pr[X < 0.3T] &\leq \exp\left(-\left(1 - \frac{0.3T}{\mu}\right)^2 \cdot \frac{\mu}{2}\right) \\
&= \exp\left(-\frac{0.03T^2}{\mu} + 0.2T - \frac{\mu}{3}\right) \\
&= \exp\left(-\left(\frac{0.03}{x} - 0.2 + \frac{x}{3}\right) \cdot T\right).
\end{aligned}$$

Since $\mathsf{Quality}(h_0, h_1) = \mu/T = x \in [0.4, 1)$, writing $f(x) = 0.03/x - 0.2 + x/3$, we have this time $f'(x) = (1 - (0.3/x)^2)/3 > 0$: $f(x)$ is increasing and bounded above by $f(1) = 49/300$. Then,

$$\Pr[X \geq 0.3T] \leq \exp\left(-49T/300\right) = 2^{-42 \cdot (49 \log_2 e/300) \cdot \lambda} \leq 2^{-9\lambda},$$

which concludes the proof. □

5.9 Efficiency and Security

Let $\mathcal{H}_L = \{\mathsf{P}_{h_0,h_1}\}$ denote the following family of predicates: given two hash functions $h_0, h_1 \in \mathcal{F}_{n,Lw}$ and an input $\mathbf{u} \in [n]^w$, $\mathsf{P}_{h_0,h_1}(\mathbf{u})$ samples $\mathbf{b} \leftarrow \mathsf{Alloc}(h_0, h_1, \mathbf{u})$. It returns 0 if $\max_{i \leq Lw} \mathsf{Load}_i(h_0, h_1, \mathbf{u}, \mathbf{b}) > L+1$, and 1 otherwise. We now summarize the efficiency properties of our construction ($\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG}$) described in the previous subsections.

Theorem 5.7. *Let $L \geq 1$ be a constant. Assume the (T, ε, δ)-hardness of the $\mathcal{H}_L$-conditioned $(\mathcal{M}^c_{n,k}, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem, for some constant $c \geq 3$. Let $d \geq 2$ be a constant. Then there exists a $(T - O(n), \varepsilon, \delta - 1/2^\lambda)$-secure structured-seed local pseudorandom generator $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ with the following characteristics:*

- Parameter size: $|\mathsf{pp}| = n \cdot (c \log k + 2 \log w)$;
- Seed length: $|\mathsf{seed}| = k + w \cdot L(L+1)d \cdot \lceil n^{1/d} \rceil$;
- Stretch: n;
- Locality: $c + 2(L+1)d$.

In addition, the algorithms have the following running time:

- $\mathsf{Setup}(1^\lambda)$ *runs in time* $\lambda \cdot \tilde{O}(n + \lambda w)$,
- $\mathsf{SampleSeed}(\mathsf{pp})$ *runs in time* $O(k + w \cdot (\lambda(\log w + \log n) + n^{1/d}))$.

Before we move on with the security analysis, we briefly overview each of the efficiency properties. We analyze runtime in a RAM model for simplicity, but note that all our algorithms can easily be implemented with similar runtime in a circuit model. The size of pp and seed, and the stretch, can be read immediately from their description and that of Comp. As for locality, computing $\langle \mathbf{a}_j, \mathbf{x} \rangle$ requires reading c bits of $\mathbf{x}$ (since $\mathsf{HW}(\mathbf{a}_j) = c$), and computing $\mathsf{Rec}(j, \mathsf{Comp}(\mathbf{v}^\alpha_{i^j_\beta}, d))$ for $\alpha \leq L, \beta \in \{0,1\}$ requires reading d bits of $\mathsf{Comp}(\mathbf{v}^\alpha_{i_j}, d)$ each time (hence $2(L+1)d$ bits in total).

The running time of Setup is decomposed as follows: sampling A requires sampling $c \cdot n$ elements of $[k]$ in time $O(n \log k)$. Sampling (h_0, h_1) requires sampling $2n$ elements of $[w]$, in time $O(n \log w)$, and running $\mathsf{Test}(1^\lambda, h_0, h_1)$ requires $O(\lambda)$ samples over $[n]^w$ (each in time $w \cdot \log n$), computations of allocation and of the maximum load (in time $O(w \cdot (\log w + \log n))$. Furthermore, with overwhelming probability, Setup terminates after at most λ executions of Test, hence the bound of $\lambda \cdot \tilde{O}(n + \lambda w)$ on the total runtime.

Eventually, the running time of $\mathsf{SampleSeed}$ is decomposed as follows: sampling $\mathbf{x}$ requires tossing k coins. Sampling $\mathbf{u} \xleftarrow{\$} [n]^w$, computing the allocation, and computing the maximum load runs in time $O(w \cdot (\log n + \log w))$. Then, computing the $(\mathbf{v}^0_i, \cdots, \mathbf{v}^L_i)$ (in implicit representation, as $\mathsf{nzi}(\mathbf{v}^0_i), \cdots, \mathsf{nzi}(\mathbf{v}^L_i)$) takes time $O(w \cdot (\log n + \log w))$, and compressing them takes time $O(w \cdot n^{1/d})$. Eventually, with overwhelming probability, $\mathsf{SampleSeed}$ produces $\mathbf{u}$ after at most λ executions of the steps 2 and 3, hence the bound of $O(k + w \cdot (\lambda(\log w + \log n) + n^{1/d}))$ on the total runtime.

Security analysis. We prove security of Theorem 5.9 under the (T, ε, δ)-hardness of the $\mathcal{H}_L$-conditioned $(\mathcal{M}^c_{n,k}, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem. Let $\mathbf{e}$ denote the noise vector, which is sampled randomly from $\mathcal{X}_{n,w}$. We showed in Section 5.5 that $\bigoplus_{\alpha \leq L, \beta \in \{0,1\}} \mathsf{Rec}(j, \mathsf{Comp}(\mathbf{v}^\alpha_{i_\beta}, d)) = \mathbf{e}[j]$ for every $j \leq n$. Therefore, denoting $\mathbf{y} = (y_1, \cdots, y_n)$,

$$\mathbf{y} = (\langle \mathbf{a}_j, \mathbf{x} \rangle + \mathbf{e}[j])_{i \leq n} = A \cdot \mathbf{x} + \mathbf{e}.$$

It follows that distinguishing $\mathbf{y}$ from random is perfectly equivalent to breaking the the $\mathcal{H}_L$-conditioned $(\mathcal{M}^c_{n,k}, \mathcal{X}_{n,w})$-$\mathsf{LPN}$ problem.

Reduction to sparse LPN. Plugging the reduction from predicate-conditioned sparse-LPN to sparse-LPN from Lemma 4.9 yields a structured-seed local PRG under the sparse-LPN assumption. However, this comes at a loss $\mathsf{err}(\mathcal{H})$ in the advantage bound ε. We have

$$\mathsf{err}(\mathcal{H}) = \max_{\mathbf{e}} \Pr_{h_0, h_1} [h_0, h_1 \text{ are bad for } \mathbf{e}] \leq O\left(\frac{1}{w^L}\right).$$

Therefore, using our concrete formulation of the sparse-LPN assumption (Assumption 1), we obtain a (T, ε, δ)-secure $(c + 2(L+1)d)$-local PRG with $\varepsilon = O(1/w^L)$ and $\delta = \Omega(k^\eta)^{c-2}$, for a suitable constant $\eta > 0$.

5.10 Structured-seed local PRGs beyond quadratic stretch

If we plug our Parameter Set 1 in Section 5.9, we obtain a PRG with seed length $|\mathsf{seed}| = O(\lambda n^{1/d}) \cdot k$ and stretch $n = k^{1+\eta}$, where η is a constant arbitrarily close to 1 (using the constant $c(\eta) = 1/(1-\eta)$). As k grows (polynomially), this means that the stretch can be made as large as $|\mathsf{seed}|^{2-\gamma}$ for an arbitrarily small constant $\gamma = \gamma(\eta, d, \log_\lambda k)$. However, the construction cannot achieve a super-quadratic stretch directly.

In general, the stretch of a polynomial-stretch local PRG can always be extended to an arbitrary polynomial stretch (keeping the locality constant) by composing the PRG with itself. However, this does not hold anymore for a structured-seed PRG, since the output distribution of the PRG does not match the required seed distribution. Nevertheless, we show in this section that the stretch of our construction can be extended to an arbitrary polynomial via self-composition. At a high level, we leverage the observation that the seed seed of our construction has two components:

- A random bitstring $\mathbf{x}$ of length k, and

– The items $\mathsf{Comp}(\mathbf{v}_i^j)$, of total length $O(w \cdot n^{1/d})$.

Above, the *structured part* of the seed grows only with w, while the *random part* of the seed grows with k. We leverage this observation by making w as small as we possibly can (while keeping the stretch n to be polynomial, $n \geq k^{1+\gamma/2}$ for some constant $\gamma > 0$), and recursively invoke the structured-seed local PRG to generate the length-k random part of the seed. The relevant set of parameters of the sparse-LPN assumption are given in Parameter Set 2; details follow.

Parameters. For simplicity and concreteness, we use the following parameters throughout:

– $\gamma > 0$ denotes an arbitrarily small constant. Fix $L = 1$.
– n is set to $k^{1+\gamma/2}$ and w to $\lambda \cdot k^\gamma$, according to Parameter Set 2 (where $c = c(\gamma) \geq 1 \log \gamma/(\log \gamma - 1)$.
– We choose a large dimension $k \geq \lambda^{1/\gamma^2}$ and set $d \geq 1/\gamma + 1/2$. Note that this guarantees that $\lambda \leq k^\gamma$ and $n^{1/d} \leq k^\gamma$. Therefore, we have $w \cdot n^{1/d} \leq k^{3\gamma}$.

With the set of parameters above, our structured-seed local PRG has the following characteristics:

– Seed length $|\mathsf{seed}| = k + O(k^{3\gamma})$ (where the $O(\cdot)$ hides a factor proportional to $1/\gamma$ and the size-k part is the random part of the seed),
– Stretch $n = k^{1+\gamma/2}$.

We view the construction as *shrinking* n (pseudorandom) bits into a seed of size (dominated by) $k = n^{1/(1+\gamma/2)}$. Let us denote $\theta(\gamma) = 1/(1+\gamma/2) < 1$ this shrinkage parameter.

Construction. Equipped with the parameters above, we are ready to describe our construction. Let s be the target expansion; that is, we want to map $|\mathsf{seed}|$ bits to $|\mathsf{seed}|^s$ bits. Set γ such that $3\gamma\theta(\gamma) = 3\gamma/(1+\gamma/2) = 1/2s$, and let $t = t(s)$ be a constant such that $\theta^t \leq 1/2s$. For readability, we describe the construction as a sequence of t seed-shrinking step:

– Step 1: fix the target output length n: we aim to generate a pseudorandom vector $\mathbf{x}_0 \in \mathbb{F}_2^n$. Set $k \leftarrow n^\theta$. Define seed_1 to be the seed of a structured-seed local PRG with stretch n, with parameters $(\lambda, k, w, L, d, \gamma, c)$ as defined above. We have

$$\mathsf{seed}_1 = (\mathbf{x}_1, S_1 = (\mathsf{Comp}(\mathbf{v}_i^0, d), \cdots, \mathsf{Comp}(\mathbf{v}_i^L, d))_{i \leq w}),$$

with $|\mathbf{x}_1| = n^\theta$ and $|S_1| = O(n^{3\gamma\theta}) = O(n^{1/2s})$.
– Step 2: in this step, we shrink $\mathbf{x}_1 \in \mathbb{F}_2^{n^\theta}$ using again our structured-seed local PRG. Writing $n_1 = k = n^\theta$, we generate a pseudorandom $\mathbf{x}_1$ using the seed:

$$\mathsf{seed}_2 = (\mathbf{x}_2, S_2),$$

where $|\mathbf{x}_2| = n_1^\theta = n^{\theta^2}$ and $|S_2| = O(n_1^{1/2s}) \leq O(n^{1/2s})$.
– Step i: we maintain the invariant that we generate a pseudorandom vector $\mathbf{x}_{i-1} \in \mathbb{F}_2^{n^{\theta^{i-1}}}$ using a seed $\mathsf{seed}_i = (\mathbf{x}_i, S_i)$ with $|\mathbf{x}_i| = n^{\theta^i}$ and $|S_i| \leq O(n^{1/2s})$.

After t steps of the seed-shrinking step, we end up with a final seed

$$\mathsf{seed} = (\mathbf{x}_t, S_t, S_{t-1}, \cdots, S_1),$$

where $|\mathbf{x}_t| \leq n^{\theta^t} \leq n^{1/2s}$, and $|S_i| \leq O(n^{1/2s})$. The total seed length is therefore $O(n^{1/2s})$ (where the $O(\cdot)$ hides a factor t), which is below $n^{1/s}$ for a large enough n. Security follows immediately from a sequence of t hybrids that "undo" the seed-shrinking steps, replacing everytime $\mathbf{x}_{i-1}$ with a random string given a random seed $(\mathbf{x}_i, S_i)$ for the structured-seed local PRG, under the hardness of the $\mathcal{H}_1$-conditioned sparse-LPN problem. Due to the t hybrids, the reduction looses a factor t in the parameters (ε, δ) of the LPN problem. Plugging in the reduction from predicate-conditioned sparse-LPN to sparse-LPN from Lemma 4.9 yields:

Theorem 5.8. *For every $s > 1$, there exists constants γ such that $3\gamma/(1+\gamma/2) = 1/2s$, $\theta = 1/(1+\gamma/2)$, $c \geq 2\log\gamma/(\log\gamma - 1)$, $d \geq 1/\gamma + 1/2$, and t such that $\theta^t \leq 1/2s$, for all large enough n, if the $(\mathcal{S}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension n^{θ^i} and number of samples $n^{\theta^{i-1}}$ is $(2^{o(\lambda)}, \varepsilon_i = \Omega(1/n^{2\gamma\theta^i}, \delta_i = \Omega(1/n^{\theta^i(1+\gamma/2)})$-secure for $i = 1$ to t, then there exists a $(T, \sum_{i=1}^t \varepsilon_i, \sum_{i=1}^t \delta_i)$-secure structured-seed constant-locality pseudorandom generator $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ with seed size $|\mathsf{seed}| \leq n^{1/s}$.*

6 Applications

In this section, we identify works that make use of a local PRG in their main theorems and explore the possibility of substituting their PRG with our own construction. These works span a variety of applications: indistinguishability obfuscation (iO) [JLS21], constant-overhead secure computation [IKOS08], sublinear secure computation [BCM23], and hardness of learning [DV21]. At a high level, we can substitute local PRGs with structured-seed local PRGs in these works because they do not rely on the seed being uniform, but only require the seed to be *short*, and the sampling of the seed to be *efficient*.

We note that in contrast, in other papers such as [BCG+23], it is not obvious how to use our PRG out of the box, because the output is being used as input to another invocation of the PRG, and in this case our idea breaks down as the seed is not structured anymore.

6.1 Indistinguishability obfuscation

Indistinguishability obfuscation (iO) is a cryptographic primitive that allows to obfuscate the code of a program such that no polynomial-time adversary can distinguish which of two (equal size) functionally equivalent programs has been obfuscated. Code obfuscation has been formalized already in the early 2000s as a cryptographic building block, by Hada [Had00] and Barak et al. [BGI+01], along with a number of early positive [Can97,LPS04,Wee05,HRsV07,HMS07] and negative [BGI+01,GK05,Wee05] results. In a recent sequence of breakthrough results culminating with [JLS21], Jain, Lin, and Sahai have shown how to base indistinguishability obfuscation on the subexponential hardness of four assumptions:

- the LWE assumption,
- the learning parity with noise over a general prime field $\mathbb{F}_p$,
- a boolean local PRG in NC^0,
- the Decision Linear assumption on symmetric bilinear groups of prime order.

At the heart of their construction is a sequence of transformations starting from weak forms of functional encryption which are progressively boosted to full-fledged indistinguishability obfuscation. Above, the local PRG is used for constructing a *structured-seed PRG*. We note that the notion of structured-seed PRGs in [JLS21] differs from the notion of structured-seed local PRG considered in our work. However, it follows by inspection that the construction of structured-seed PRG of [JLS21] goes through identically if the boolean local PRG is replaced by a structured-seed local PRG.

We mention one technicality, though: the local PRG used in [JLS21] needs to have subexponential security. For our construction from regular sparse-LPN, one can reasonably conjecture security against subexponential time adversaries while keeping ε inverse-subexponential: this follows directly from our concrete version of the sparse-LPN assumption (see Assumption 1) by setting T to subexponential in λ. However, the value of δ remains always noticeable due to the non-negligible probability of sampling a matrix with a small dual distance. One can make δ negligible using the alternative matrix distribution introduced in [AK19], but this only makes δ *slightly* negligible, while the construction requires δ to be subexponentially small.

We note that a very similar issue happens with (standard) local PRGs, which require an explicit hypergraph with good expansions property, while random hypergraphs will only satisfy the required property with probability $1-1/\mathsf{poly}$. There are two workarounds to this issue. The first one is identical to the solution that was used in [JLS21]: using an explicit choice of the sparse matrix A and assuming the subexponential hardness of sparse-LPN *with respect to this matrix*; then, Assumption 1 implies that *most* choices of A will yield plausible candidates. This works directly, with the caveat that it only provides a non-uniform construction of iO (which would become uniform if an efficiently sampleable distribution over sparse matrices with subexponentially small probability of having a small dual distance is found in the future).

The second solution is to observe that except with subexponentially small probability, if we sample *multiple* parameters pp for a structured-seed local PRG, at least one of them will be secure against subexponential adversaries. Then, as observed in [JLS22], this implies in turn that the construction of functional encryption will yield multiple candidates functional encryption schemes such that except with subexponentially small probability, one of them is subexponentially secure. Then, one can obtain a full-fledged functional encryption scheme out of these schemes using FE combiners (see Remark 3.1 in [JLS22]). We get the following:

Theorem 6.1 (informal). *Assume sub-exponential security of the following assumptions:*

- *the LWE assumption,*
- *the learning parity with noise over a general prime field $\mathbb{F}_p$,*
- *the sparse-LPN assumption with regular noise,*
- *the Decision Linear assumption on symmetric bilinear groups of prime order,*

there exists a (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits. Further, assuming only polynomial security of the aforementioned assumptions, there exists collusion-resistant public-key functional encryption for all polynomial-size circuits.

We note that the follow-up work of [JLS22] gets rid of the LWE assumption by making a more involved use of the local PRG. It is not immediately obvious how to replace the local PRG by a structured-seed local PRG in their more involved construction, because it requires in particular an affine randomized encoding construction that relies on self-composing the PRG (using its pseudorandom output as a seed), which does not work with structured-seed local PRGs. We had preliminary results in this direction, but in light of the recent concurrent and independent work of [RVV24] that focuses precisely on this application, and which provides a full-fledged solution to overcoming this obstacle (and others), we refrain in this work from pursuing this route further.

6.2 Constant-overhead secure computation

The seminal work of Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS08] showed that assuming polynomial-stretch local pseudorandom generators (and oblivious transfers), any two-party functionality can be securely computed with *constant computational overhead* over the cost of evaluating the functionality in the clear. In this section, we observe that the local PRG in [IKOS08] can be replaced by a structured-seed local PRG. We summarize the result in Theorem 6.2 below.

Theorem 6.2. *Assume the existence of a polynomial-stretch structured-seed local PRG in NC^0, denoted as $G : \{0,1\}^n \to \{0,1\}^m$, and of a standard OT protocol. Given a family of circuit $C = \{C_n\}$ of polynomial size $s(n)$ that defines a two-party computation functionality f, there exists a two-party protocol π_f that realizes f in the semi-honest setting where each party in π_f can be implemented by a circuit of size $O(s(n))$.*

As a direct corollary, we obtain that secure two-party computation with constant computational overhead can be based on (oblivious transfer and) the hardness of regular sparse-LPN, or that of $\mathcal{H}$-conditioned sparse-LPN, diversifying the set of assumptions under which extreme efficiency can be achieved in secure computation.

We sketch the proof of Theorem 6.2 below. The construction of [IKOS08] uses the following sequence of steps:

1. Using the GMW protocol [GMW87], given black-box access to $O(s)$ oblivious transfers, any two-party computation for a circuit of size s can be securely evaluated using $O(s)$ bits of communication and $O(s)$ bit operations. Hence, building constant-overhead secure computation reduces to the problem of constructing $O(s)$ OTs with a constant computational overhead. In particular, denoting $p = O(s)$, [IKOS08] shows how to construct p bit-OTs using a local PRG and $\sqrt{p}$ OT instances (on strings of length $\sqrt{p}$) where each party can be implemented by a circuit of size $O(p)$.
2. Let g be a functionality parametrized with a local PRG G which, on input a seed seed from the receiver and p pairs of bits $(\sigma_0^i, \sigma_1^i)_{i \le p}$ from the sender, outputs $(\sigma^i_{G(\mathsf{seed})_i})_{i \le p}$ to the receiver. Then, given black-box access to g, there is a constant-overhead construction of a secure protocol to generate p (chosen) bit-OTs. The protocol follows from the standard information-theoretic derandomization of OT with random selection bits [Bea92], using $G(\mathsf{seed})$ instead of a random mask to hide the selection bits. Since $G(\mathsf{seed})$ can be computed in linear time and the derandomization involves only $O(p)$ XORs and ANDs, the claim follows.
3. The core component of the reduction is the constant-overhead reduction from securely implementing g to a black-box access to $\sqrt{p}$ OTs on strings of total length $O(p)$. This reduction uses decomposable randomized encodings and builds upon the fact that $g \in \mathsf{NC}^0$.

Above, step 1 and 3 remain perfectly identical if G is replaced by a structured-seed local PRG: in particular, g has the description of the stretching algorithm PRG hardcoded, which is in NC^0, and is oblivious to how the seed seed was sampled. The only difference is in step 2, where the receiver must be instructed to sample $\mathsf{seed} \xleftarrow{\$} \mathsf{SampleSeed}(\mathsf{pp})$ instead of picking a random seed. But since the cost of running $\mathsf{SampleSeed}$ is sublinear in p, this has no effect on the computational complexity of the protocol.

We mention two additional minor technicalities:

- The construction of [IKOS08] is described using a quadratic-stretch local PRG, which is without loss of generality since a local PRG with arbitrary polynomial stretch can be extended to quadratic stretch via self-composition. However, our structured-seed local PRG achieves only *near* quadratic stretch, and structured-seed local PRGs cannot be self-composed. Nevertheless, the assumption of (strict) quadratic stretch in [IKOS08] was made only for notational convenience, and can be generalized in a straightforward way to work with a PRG with a smaller (polynomial) stretch. The reduction then invokes $O(p^{1/2+\varepsilon})$ string-OTs on strings of total length $O(p)$, where ε is such that the local PRG stretches $O(p^{1/2+\varepsilon})$ bits into p bits.
- After completing the reduction, it remains to implement the $O(p^{1/2+\varepsilon})$ string-OTs on strings of total length $O(p)$ with constant computational overhead. This relies again on a constant-overhead PRG: given any pair of strings (α_0, α_1) of length $O(p^{1/2-\varepsilon})$, the sender samples two ℓ-bit seeds $(\mathsf{seed}_0, \mathsf{seed}_1)$ for a local PRG. The sender and the receiver (with bit b) use an ℓ-bit string-OT to let the receiver learn seed_b, using $\mathsf{poly}(\lambda) \cdot \ell$ computation. Then, the sender sends $G(\mathsf{seed}_0) \oplus \alpha_0, G(\mathsf{seed}_1) \oplus \alpha_1$, and the receiver unmasks α_b. The total computation per OT scales as $\mathsf{poly}(\lambda) \cdot \ell + O(|\alpha_0| + |\alpha_1|)$, hence an overall cost of $\mathsf{poly}(\lambda) \cdot \ell \cdot p^{1/2+\varepsilon} + O(p)$. Now, using a structured-seed local PRG with any polynomial stretch yields $\ell = O(p^{(1/2-\varepsilon)\cdot\gamma})$ for some $\gamma < 1$, hence $\mathsf{poly}(\lambda) \cdot \ell \cdot p^{1/2+\varepsilon} = o(p)$ for a large enough p. Here again, an arbitrary structured-seed local PRG with polynomial stretch suffices.

6.3 Sublinear secure computation and compact HSS

Homomorphic secret sharing (HSS) was introduced in the work of [BGI16] as an alternative to fully homomorphic encryption to achieve secure computation with sublinear communication. At a high level, an N-party HSS for a class of functions $\mathcal{F}$ allows to share an input x such that for every function $f \in \mathcal{F}$, each party with input share x_i can locally compute y_i such that $(y_1, \cdots, y_N)$ form additive shares of $y = f(x)$. A *compact* HSS is an HSS where the share size and sharing algorithm runtime are $O(|x|) + \mathsf{poly}(\lambda)$. Combined with a generic MPC protocol with linear communication overhead to securely run the sharing algorithm, a compact HSS scheme immediately gives rise to a secure N-party protocol with essentially optimal communication $O(N \cdot (|x| + |y|)) + \mathsf{poly}(\lambda)$ for every function $f \in \mathcal{F}$.

A standard approach to build compact HSS from HSS is to use a "hybrid encapsulation" trick: to share a long input x, share a short seed seed with HSS among the parties, and reveal $u = x \oplus G(\mathsf{seed})$ to everyone, where G is a PRG. If the PRG runs in linear time, the share size and runtime of sharing are clearly $O(|x|) + \mathsf{poly}(\lambda)$. Then, to get shares of $f(x)$, the parties homomorphically evaluate $g_u(\mathsf{seed}) := f(u \oplus G(\mathsf{seed})) = f(x)$. This approach works as long as $g_u \in \mathcal{F}$. In particular, this means that if the HSS scheme supports a very low function class $\mathcal{F}$, the PRG G needs to belong to a very low complexity class.

The recent work of [BCM23] showed how to achieve sublinear secure computation and compact HSS from assumptions that were not previously known to imply it. In particular, they show:

Theorem 6.3 (Theorem 32 in [BCM23]). *Assuming the superpolynomial hardness of DCR and the existence of PRGs with constant locality, there exists a four-party HSS scheme for the class of loglog-depth circuits with n inputs; the HSS scheme has share size $n \cdot (1 + o(1))$. Furthermore, there exists a protocol with communication complexity $n \cdot (4 + o(1))$ (for large enough n) for securely realizing the four-party functionality that generates HSS shares of the concatenation of the parties' inputs.*

It is immediate from the description of the construction that Theorem 32 in [BCM23] extends directly to the setting where a *structured-seed* constant-locality PRG (with arbitrarily small polynomial stretch) is used instead: in their construction, each party locally samples a short seed seed_i and a generic secure computation protocol is ran on their concatenation $(\mathsf{seed}_1 || \mathsf{seed}_2 || \mathsf{seed}_3 || \mathsf{seed}_4)$. The only

impact of using a structured-seed local PRG is that the parties will locally run $\mathsf{SampleSeed}$ instead of sampling their seed uniformly, which has no influence on the correctness, security, of communication efficiency of the protocol. As a consequence, we immediately obtain the following corollary:

Corollary 6.4. *Assuming the superpolynomial hardness of DCR and the hardness of the regular sparse-LPN assumption (or, alternatively, of the $\mathcal{H}$-conditioned sparse-LPN assumption), there exists a four-party HSS scheme for the class of loglog-depth circuits with n inputs; the HSS scheme has share size $n \cdot (1 + o(1))$. Furthermore, there exists a protocol with communication complexity $n \cdot (4 + o(1))$ (for large enough n) for securely realizing the four-party functionality that generates HSS shares of the concatenation of the parties' inputs.*

Combining this corollary with the compiler of [BCM23] from N-party compact HSS to $(N + 1)$-party secure computation with sublinear communication yields a 5-party protocol with sublinear communication $O(s/\log\log s)$ for all layered circuits of size s under the same assumptions as above.[5]

6.4 Hardness of learning

PAC learning [Val84] is the algorithmic problem of finding a hypothesis that predicts the output of an unknown class of functions with high probability. The hardness of learning focuses on showing a learning algorithm's ability to return a hypothesis. Daniely and Vardi in [DV21] recently proved several hardness of learning results based on the assumption that local PRGs with polynomial stretch and constant distinguishing advantage exist. We show that our structured seed local PRG can be used in place of their PRG, obtaining hardness-of-learning results from the sparse-LPN assumption.

Definition 6.5 (Predicate). *Given a structured-seed ℓ-local PRG $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ with input size k, and stretch n, we let P denote the predicate such that for all $i \in [n]$, there exists a subset $S_i \subset [k]$ of size $|S_i| \leq \ell$ such that for all $x \in \mathsf{Supp}(\mathsf{SampleSeed}(\mathsf{pp}))$, defining $y = \mathsf{PRG}(x)$, we have $y_i = P(x[S_i])$.*

Remark 6.6. In general, an ℓ-local PRG only guarantees that for each output bit y_i, there is a predicate P_i and a size-ℓ subset S_i of the bits of the seed x such that $y_i = P_i(x[S_i])$. However, assuming a single predicate P is without of generality when the PRG has polynomial stretch: since there are at most 2^{2^ℓ} possible predicates P_i on ℓ-bit inputs, and ℓ is a constant, setting P to be the most frequent P_i and keeping only the output bits computed using P yields an ℓ-local PRG with polynomial stretch (reduced by a constant factor at most 2^{2^ℓ}) and a single global predicate P. Furthermore, we note that our constructions from sparse-LPN directly have a single global predicate.

DNFs. We prove that formulas in disjunctive normal form with $\omega(1)$ terms cannot be efficiently PAC-learned assuming the sparse-LPN assumption:

Theorem 6.7 (Theorem 3.1 in [DV21]). *Under the assumptions of Theorem 5.8, for every $q(n) = \omega(1)$ there is no efficient algorithm that PAC-learns DNF formulas with n variables and $q(n)$ terms.*

We note that, contrary with the other applications discussed in this section, we only need to assume a structured-seed local PRG with constant (ε, δ). Therefore, we can rely directly on sparse-LPN rather than regular sparse-LPN via our reduction from Lemma 4.9. However, the result also crucially requires a local PRG with *arbitrary polynomial stretch*. Fortunately, one can achieve an arbitrary polynomial stretch under sparse-LPN via the construction of Theorem 5.8.

At the heart of the proof of Theorem 3.1 in [DV21] is the following clever idea: let ℓ be a constant, and let $P : \{0,1\}^\ell \to \{0,1\}$ be an ℓ-local predicate. Given a size-ℓ subset $S = \{s_1, \cdots, s_\ell\} \subset [n]$, write $\mathbf{v}_i := \mathsf{unit}_n(s_i) \in \{0,1\}^n$ for $i = 1$ to ℓ. Then, consider the following formula ψ:

$$\psi(\mathbf{v}_1, \cdots, \mathbf{v}_\ell) = \bigvee_{\mathbf{x}: P(\mathbf{x})=1} \bigwedge_{i \leq \ell} \bigwedge_{j: \mathsf{seed}_j \neq x_i} \bar{v}_{i,j}.$$

[5] The compiler of [BCM23] requires assuming the hardness of DCR and LPN as it relies on the (DCR+LPN)-based construction of correlated symmetric PIR from [BCM22]. However, their construction can be instantiated with any suitable variant of LPN, including sparse-LPN, hence it requires only assumptions that are redundant with the one we already assume.

We have

$$\begin{aligned}
\psi(\mathbf{v}_1, \cdots, \mathbf{v}_\ell) = 1 &\iff \exists \mathbf{x} \in P^{-1}(1), \forall i \le \ell, \forall \mathsf{seed}_j \neq x_i, \bar{v}_{i,j} = 1 \\
&\iff \exists \mathbf{x} \in P^{-1}(1), \forall i \le \ell, \forall \mathsf{seed}_j \neq x_i, \mathsf{unit}_n(s_i)_j = 0 \\
&\iff \exists \mathbf{x} \in P^{-1}(1), \forall i \le \ell, \forall \mathsf{seed}_j \neq x_i, s_i \neq j \\
&\iff \exists \mathbf{x} \in P^{-1}(1), \forall i \le \ell, \mathsf{seed}_{s_i} = x_i \\
&\iff \exists \mathbf{x} \in P^{-1}(1), \mathsf{seed}_S = \mathbf{x} \\
&\iff P(\mathsf{seed}_S) = 1.
\end{aligned}$$

This means that given a predicate P and a seed $\mathbf{x}$, it is possible to hardcode $(P, \mathbf{x})$ in a DNF ϕ such that for every size-ℓ subset S, there is an encoding $\mathsf{Encode}(S) = (\mathbf{v}_1, \cdots, \mathbf{v}_\ell)$ such that $\psi(\mathsf{Encode}(S)) = P(\mathbf{x}[S])$.

Now, we explain how to adapt the proof of Theorem 3.1 in [DV21] to structured-seed local PRGs. We need the following assumption:

Assumption 2 *For every constant $s > 1$, there exists a constant ℓ such that there exists a $(T, 1/6, 1/6)$-secure structured-seed ℓ-local PRG $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$ with predicate P, mapping k bits to k^s bits, for every $T = \mathsf{poly}(\lambda)$.*

By Theorem 5.8, the assumption above is implied by the sparse-LPN assumption. Then, let $\mathcal{A}$ be a PPT adversary that PAC-learns DNF formulas with k variables and $q = \omega(1)$ terms. Let Q denote the number of queries to the DNF oracle made by $\mathcal{A}$, and set s such that $k^s > 100Q^2$. Define the following distribution $\mathcal{D}$:

- Sample $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
- For any index $i \le k^s$, let S_i denote the size-ℓ subset of the bits of seed used by $\mathsf{PRG}_{\mathsf{pp}}(\mathsf{seed})$ (note that S_i is independent of the particular choice of seed, but might depend on pp).
- Define $\mathcal{D} = \mathcal{D}_{\mathsf{pp}}$ to be the distribution that samples $i \xleftarrow{\$} [k^s]$ and outputs $z = \mathsf{Encode}(S_i)$.

Now, sample $\mathsf{seed} \leftarrow \mathsf{SampleSeed}(\mathsf{pp})$, and let ψ be the DNF (with seed, P hardcoded) encoding the computation of the mapping $\mathsf{PRG}_{\mathsf{pp}}(\mathsf{seed})_i = P(\mathsf{seed}_{S_i}) = \psi(\mathsf{Encode}(S_i))$. Note that ψ is a DNF formula with at most 2^ℓ terms. Given Q samples $(z_i, \psi(z_i))_{i \le Q}$ (the training set), the adversary $\mathcal{A}$ returns a hypothesis h with, with small probability, has a small error on the training set. Note that except with probability at most $1/100$, there are no collisions among the queries. Now, given h, it is straightforward to distinguish the next sample $(z_{Q+1}, \psi(z_{Q+1}))$ from random with high probability.

Other classes. Because any function represented by a DNF formula with $q(n)$ terms can also be represented by a polynomial threshold function over $\{0,1\}^n$ with $q(n)$ monomials, assuming sparse-LPN, the following corollary follows from Theorem 6.7.

Corollary 6.8 (Corollary 3.2 in [DV21]). *For all $q(n) = \omega(1)$ there is no efficient algorithm that learns $q(n)$-sparse polynomial threshold functions over $\{0,1\}^n$.*

One can also consider $\omega(1)$-sparse $GF(2)$ polynomials over $\{0,1\}^n$ which are simply a sum of $\omega(1)$ monomials modulo 2.

Theorem 6.9. *For all $q(n) = \omega(1)$ there is no efficient algorithm that learns $q(n)$-sparse GF(2) polynomials over $\{0,1\}^n$.*

References

ABG+14. A. Akavia, A. Bogdanov, S. Guo, A. Kamath, and A. Rosen. Candidate weak pseudorandom functions in AC^0 *o* MOD_2. In *ITCS 2014*, pages 251–260. ACM, January 2014.

ABR16. B. Applebaum, A. Bogdanov, and A. Rosen. A dichotomy for local small-bias generators. *Journal of Cryptology*, 29(3):577–596, July 2016.

ADI+17. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 223–254. Springer, Heidelberg, August 2017.

AIK04. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

AIK08. B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in nc 0. *Computational Complexity*, 17(1):38–69, 2008.

AK19. B. Applebaum and E. Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In *60th FOCS*, pages 171–179. IEEE Computer Society Press, November 2019.

AL16. B. Applebaum and S. Lovett. Algebraic attacks against random local functions and their countermeasures. In *48th ACM STOC*, pages 1087–1100. ACM Press, June 2016.

Ale03. M. Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

App12. B. Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *44th ACM STOC*, pages 805–816. ACM Press, May 2012.

App15. B. Applebaum. The cryptographic hardness of random local functions – survey. Cryptology ePrint Archive, Report 2015/165, 2015. `https://eprint.iacr.org/2015/165`.

BBC+24. M. Bombar, D. Bui, G. Couteau, A. Couvreur, C. Ducros, and S. Servan-Schreiber. Foleage: F4 ole-based multi-party computation for boolean circuits. *Cryptology ePrint Archive*, 2024.

BCCD23. M. Bombar, G. Couteau, A. Couvreur, and C. Ducros. Correlated pseudorandomness from the hardness of quasi-abelian decoding. LNCS, pages 567–601. Springer, Heidelberg, 2023.

BCG+17. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù. Homomorphic secret sharing: Optimizations and applications. In *ACM CCS 2017*, pages 2105–2122. ACM Press, October / November 2017.

BCG+20. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.

BCG+22. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Correlated pseudorandomness from expand-accumulate codes. LNCS, pages 603–633. Springer, Heidelberg, 2022.

BCG+23. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Oblivious transfer with constant computational overhead. LNCS, pages 271–302. Springer, Heidelberg, 2023.

BCGI18. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BCM22. E. Boyle, G. Couteau, and P. Meyer. Sublinear secure computation from new assumptions. LNCS, pages 121–150. Springer, Heidelberg, 2022.

BCM23. E. Boyle, G. Couteau, and P. Meyer. Sublinear-communication secure multiparty computation does not require FHE. LNCS, pages 159–189. Springer, Heidelberg, 2023.

BCM+24. D. Bui, G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. LNCS, pages 88–118. Springer, Heidelberg, 2024.

Bea92. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO'91*, *LNCS* 576, pages 420–432. Springer, Heidelberg, August 1992.

BGI+01. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, *LNCS* 2139, pages 1–18. Springer, Heidelberg, August 2001.

BGI16. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, *LNCS* 9814, pages 509–539. Springer, Heidelberg, August 2016.

BKR23. A. Bogdanov, P. K. Kothari, and A. Rosen. Public-key encryption, local pseudorandom generators, and the low-degree method. LNCS, pages 268–285. Springer, Heidelberg, 2023.

BKW00. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd ACM STOC*, pages 435–440. ACM Press, May 2000.

BM97. M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *EUROCRYPT'97*, *LNCS* 1233, pages 163–192. Springer, Heidelberg, May 1997.

BQ09. A. Bogdanov and Y. Qiao. On the security of goldreich's one-way function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 392–405. Springer, 2009.

BR17. A. Bogdanov and A. Rosen. Pseudorandom functions: Three decades later. Cryptology ePrint Archive, Report 2017/652, 2017. `https://eprint.iacr.org/2017/652`.

Can97. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO'97*, *LNCS* 1294, pages 455–469. Springer, Heidelberg, August 1997.

CDM+18. G. Couteau, A. Dupin, P. Méaux, M. Rossi, and Y. Rotella. On the concrete security of Goldreich's pseudorandom generator. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 96–124. Springer, Heidelberg, December 2018.

CEMT14. J. Cook, O. Etesami, R. Miller, and L. Trevisan. On the one-way function candidate proposed by goldreich. *ACM Transactions on Computation Theory (TOCT)*, 6(3):14, 2014.

CM01. M. Cryan and P. B. Miltersen. On pseudorandom generators in nc 0. In *International Symposium on Mathematical Foundations of Computer Science*, pages 272–284. Springer, 2001.

CRR21. G. Couteau, P. Rindal, and S. Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. LNCS, pages 502–534. Springer, Heidelberg, 2021.

CRS03. A. Czumaj, C. Riley, and C. Scheideler. Perfectly balanced allocation. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 240–251. Springer, 2003.

DIJL23. Q. Dao, Y. Ishai, A. Jain, and H. Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. LNCS, pages 315–348. Springer, Heidelberg, 2023.

DMR23. A. Dupin, P. Méaux, and M. Rossi. On the algebraic immunity—resiliency trade-off, implications for goldreich's pseudorandom generator. *Designs, Codes and Cryptography*, pages 1–45, 2023.

DV21. A. Daniely and G. Vardi. From local pseudorandom generators to hardness of learning. In *Proceedings of Thirty Fourth Conference on Learning Theory*, *Proceedings of Machine Learning Research* 134, pages 1358–1394. PMLR, 15–19 Aug 2021.

EKM17. A. Esser, R. Kübler, and A. May. LPN decoded. In *CRYPTO 2017, Part II*, *LNCS* 10402, pages 486–514. Springer, Heidelberg, August 2017.

GK05. S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *46th FOCS*, pages 553–562. IEEE Computer Society Press, October 2005.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, *LNCS* 263, pages 171–185. Springer, Heidelberg, August 1987.

Gol00. O. Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. `https://eprint.iacr.org/2000/063`.

Had00. S. Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT 2000*, *LNCS* 1976, pages 443–457. Springer, Heidelberg, December 2000.

HMS07. D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. In *TCC 2007*, *LNCS* 4392, pages 214–232. Springer, Heidelberg, February 2007.

HRsV07. S. Hohenberger, G. N. Rothblum, a. shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In *TCC 2007*, *LNCS* 4392, pages 233–252. Springer, Heidelberg, February 2007.

IKOS04. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *36th ACM STOC*, pages 262–271. ACM Press, June 2004.

IKOS08. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. In *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

JLS21. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. pages 60–73. ACM Press, 2021.

JLS22. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. LNCS, pages 670–699. Springer, Heidelberg, 2022.

Kir11. P. Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `https://eprint.iacr.org/2011/377`.

LF06. É. Levieil and P.-A. Fouque. An improved LPN algorithm. In *SCN 06*, *LNCS* 4116, pages 348–359. Springer, Heidelberg, September 2006.

LPS04. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT 2004*, *LNCS* 3027, pages 20–39. Springer, Heidelberg, May 2004.

LV17. A. Lombardi and V. Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *TCC 2017, Part I*, *LNCS* 10677, pages 119–137. Springer, Heidelberg, November 2017.

LWYY24. H. Liu, X. Wang, K. Yang, and Y. Yu. The hardness of lpn over any integer ring and field for pcg applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 149–179. Springer, 2024.

Lyu05. V. Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. 2005.

Méa. P. Méaux. On the fast algebraic immunity of threshold functions. crypt. commun. 13 (5), 741–762 (2021).

Méa22. P. Méaux. On the algebraic immunity of direct sum constructions. *Discrete Applied Mathematics*, 320:223–234, 2022.

MST03. E. Mossel, A. Shpilka, and L. Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.

NN90. J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *22nd ACM STOC*, pages 213–223. ACM Press, May 1990.

OST19. I. C. Oliveira, R. Santhanam, and R. Tell. Expander-based cryptography meets natural proofs. In *ITCS 2019*, pages 18:1–18:14. LIPIcs, January 2019.

Ove06. R. Overbeck. Statistical decoding revisited. In *ACISP 06*, *LNCS* 4058, pages 283–294. Springer, Heidelberg, July 2006.

OW14. R. ODonnell and D. Witmer. Goldreich's prg: evidence for near-optimal polynomial stretch. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 1–12. IEEE, 2014.
Pie12. K. Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.
Pra62. E. Prange. The use of information sets in decoding cyclic codes. 1962.
PRTY19. B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 401–431. Springer, Heidelberg, August 2019.
PSWW18. B. Pinkas, T. Schneider, C. Weinert, and U. Wieder. Efficient circuit-based PSI via cuckoo hashing. In *EUROCRYPT 2018, Part III*, *LNCS* 10822, pages 125–157. Springer, Heidelberg, April / May 2018.
RS98. M. Raab and A. Steger. "balls into bins" - a simple and tight analysis. In *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '98, page 159–170, Berlin, Heidelberg, 1998. Springer-Verlag.
RVV24. S. Ragavan, N. Vafa, and V. Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and lpn variants. *Cryptology ePrint Archive*, 2024.
Saa07. M.-J. O. Saarinen. Linearization attacks against syndrome based hashes. In *INDOCRYPT 2007*, *LNCS* 4859, pages 1–9. Springer, Heidelberg, December 2007.
SEK03. Sanders, Egner, and Korst. Fast concurrent access to parallel disks. *Algorithmica*, 35:21–55, 2003.
SGRP19. P. Schoppmann, A. Gascón, M. Raykova, and B. Pinkas. Make some ROOM for the zeros: Data sparsity in secure distributed machine learning. In *ACM CCS 2019*, pages 1335–1350. ACM Press, November 2019.
SGRR19. P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova. Distributed vector-OLE: Improved constructions and implementation. In *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.
Üna23a. A. Ünal. New baselines for local pseudorandom number generators by field extensions. *Cryptology ePrint Archive*, 2023.
Üna23b. A. Ünal. Worst-case subexponential attacks on PRGs of constant degree or constant locality. LNCS, pages 25–54. Springer, Heidelberg, 2023.
Val84. L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984.
Wag02. D. Wagner. A generalized birthday problem. In *CRYPTO 2002*, *LNCS* 2442, pages 288–303. Springer, Heidelberg, August 2002.
Wee05. H. Wee. On obfuscating point functions. In *37th ACM STOC*, pages 523–532. ACM Press, May 2005.
YGJL21. J. Yang, Q. Guo, T. Johansson, and M. Lentmaier. Revisiting the concrete security of goldreich's pseudorandom generator. *IEEE Transactions on Information Theory*, 68(2):1329–1354, 2021.

A Proof of Theorem 4.5

We restate the Theorem for convenience:

Theorem A.1. *For any constants $c \geq 3$ and $\eta > 0$, for any large enough $k = k(\lambda)$, there is a constant $\gamma(c)$ such that for any $n \leq k^{(1-\eta)c/2+\eta}$,*

$$\Pr\left[A \xleftarrow{\$} \mathcal{W}^c \ : \ \mathsf{dd}(A) \geq \frac{k^\eta}{\gamma(c)} - 1\right] \geq 1 - \left(\frac{\gamma(c)}{k^\eta}\right)^{c-2}.$$

The proof is a direct adaptation (and slight generalization) of the analysis of [MST03, Section 5.3]. Part of the proof is taken essentially verbatim from [CRR21], and adapted to our parameter setting. Near-identical proofs of essentially the same theorems can be found in other works, e.g. [BCG+23, DIJL23].

Proof. Given a matrix $A \in \mathbb{F}_2^{n\times k}$, we denote by $(\mathbf{a}_1, \cdots, \mathbf{a}_n)$ the rows of A, and by $(\mathbf{a}^1, \cdots, \mathbf{a}^k)$ its columns. For any subset $S \subseteq [n]$ of the rows of A, we call *column-neighbors of S in A*, and write $\mathsf{N}^{\mathsf{c}}_A(S) \subseteq [k]$, the subset of the columns of A which have at least one 1 in a row of S. For any integer d, we say that A has the *d-unique column-neighbor property* if for any set $S \subseteq [n]$ with $|S| \leq d$, there exists a column A^j of A such that $A^j[S]$ contains exactly a single 1.

Lemma A.2 ([MST03]). *If $A \in \mathbb{F}_2^{n\times k}$ has d-unique column-neighbor, then $\mathsf{dd}(A) \geq d - 1$.*

Proof. For any subset S of $[n]$ with $|S| \leq d$, there exists a column of A that has exactly one 1 in a row indexed by S, hence $\bigoplus_{i\in S} \mathbf{a}_i \neq 0^k$.

In the following, we show that a randomly sampled sparse matrix A has d-right unique column-neighbor with high probability, for a certain d. This follows from the fact that A has certain expansion properties. We say that a matrix A is (d,α)-expanding if for every subset $S \subseteq [n]$ with $|S| \leq d$, it holds that $|\mathsf{N}_A(S)| > \alpha \cdot |S|$ (in other words, a matrix A is (d,α)-expanding if it is the adjacency matrix of a (d,α)-expanding bipartite graph).

Lemma A.3 ([MST03]). *Let $A \in \mathsf{Supp}(\mathcal{W}^c)$ be a c-sparse matrix. If A is $(d, c/2)$-expanding, then it has d-unique column-neighbor.*

Proof. Assume that A does not have d-unique column-neighbor. Let $S \subseteq [n]$ be any subset with $|S| \leq d$. Then for every $j \in \mathsf{N}^{\mathsf{c}}_A(S)$, $\mathbf{a}^j[S]$ contains at least two 1's. Since the rows in S contain $c \cdot |S|$ 1's in total, this implies that $|\mathsf{N}^{\mathsf{c}}_A(S)| \leq (c/2) \cdot |S|$.

To prove Theorem 4.5, it remains to show that a random sample from $\mathcal{W}^c$ is sufficiently expanding with high probability.

Lemma A.4. *For any large enough $k = k(\lambda)$, any constants $c \geq 3$ and $\eta > 0$, there is a constant $\gamma(c)$ such that for any $n \leq k^{(1-\eta)c/2+\eta}$,*

$$\Pr\left[A \xleftarrow{\$} \mathcal{W}^c \ : \ A \text{ is } \left(\frac{k^\eta}{\gamma(c)}, c/2\right)\text{-expanding}\right] \geq 1 - \left(\frac{\gamma(c)}{k^\eta}\right)^{c-2}.$$

Proof. For any subset $S \subset [n]$ and any size $c \cdot |S|/2$ subset $T \subset [k]$, the probability over the random choice of $A \xleftarrow{\$} \mathcal{W}^c$ that $\mathsf{N}^{\mathsf{c}}_A(S) \subseteq T$ is at most $(c \cdot |S|/2k)^{c \cdot |S|}$. Since there are $\binom{n}{|S|}$ choices for S and $\binom{k}{c|S|/2}$ choices for T, the probability that A fails to be $(d, c/2)$-expanding is at most

$$\sum_{i=2}^{d} \binom{n}{i} \cdot \binom{k}{c \cdot i/2} \cdot \left(\frac{c \cdot i}{2k}\right)^{c \cdot i},$$

where the sum starts at 2 because any single row j always satisfies $\mathsf{N}^{\mathsf{c}}_A(\{j\}) = c > c/2$. Using the inequality $\binom{a}{b} \leq (ae/b)^b$, we get

$$\begin{aligned}
\sum_{i=2}^{d} \left(\frac{en}{i}\right)^i \cdot \left(\frac{2ek}{ci}\right)^{ci/2} \cdot \left(\frac{ci}{2k}\right)^{ci} &= \sum_{i=2}^{d} \left(\frac{en}{i} \cdot \left(\frac{2ek}{ci}\right)^{c/2} \cdot \left(\frac{ci}{2k}\right)^{c}\right)^i \\
&= \sum_{i=2}^{d} \left(\frac{n}{k} \cdot e^{c/2+1} \cdot (c/2)^{c/2} \cdot \left(\frac{i}{k}\right)^{c/2-1}\right)^i \\
&= \sum_{i=2}^{d} \left(\left(\frac{n}{k}\right)^{\frac{1}{c/2-1}} \cdot \left(\frac{c}{2}\right)^{\frac{c/2}{c/2-1}} \cdot e^{\frac{c/2+1}{c/2-1}} \cdot \frac{i}{k}\right)^{i \cdot (c/2-1)} \\
&\leq \sum_{i=2}^{d} \left(\gamma(c) \cdot \frac{i}{2k^\eta}\right)^{i \cdot (c/2-1)},
\end{aligned}$$

where $\gamma(c)$ denotes the constant $2 \cdot (c/2)^{\frac{c/2}{c/2-1}} \cdot e^{\frac{c/2+1}{c/2-1}}$. Now, setting $d = k^2/(\gamma(c) \cdot n)$, the above is upper bounded by

$$\left(\frac{\gamma(c)}{k^\eta}\right)^{c-2} + \left(\frac{3\gamma(c)}{2k^\eta}\right)^{3(c/2-1)} + \log^2 k \cdot \left(\frac{\gamma(c)^2 \log^2 k}{k^\eta}\right)^{2c-4} + d \cdot \left(\frac{1}{k^{\log k}}\right)^{c/2-1},$$

where the first two terms are the terms for $i = 2, 3$ in the sum, the third term bounds the terms $i = 4, \cdots, \log^2 k$ in the sum, and the last term bounds the sum of the remaining terms. For a large enough k and any integer $c > 2$, this sum is therefore dominated by its first term. The lemma follows.