Structured-Seed Local Pseudorandom Generators and their Applications

Benny Applebaum[1], Dung Bui[2⋆], Geoffroy Couteau[3], and Nikolas Melissaris[4]

[1] Tel-Aviv University, Israel
benny.applebaum@gmail.com
[2] LIP6, Université Sorbonne, France
bui@irif.fr
[3] CNRS, IRIF, Université Paris Cité, France
couteau@irif.fr
[4] CNRS, IRIF, Université Paris Cité, France
nikolas@irif.fr

Abstract. We introduce *structured-seed local pseudorandom generators* (SSL-PRGs), pseudorandom generators whose seed is drawn from an efficiently sampleable, structured distribution rather than uniformly. This seemingly modest relaxation turns out to capture many known applications of local PRGs, yet it can be realized from a broader family of hardness assumptions. Our main technical contribution is a generic template for constructing SSL-PRGs that combines the following two ingredients:

i. noisy-NC^0 PRGs, computable by constant-depth circuits fed with sparse noise, with
ii. new local compression schemes for sparse vectors derived from combinatorial batch codes.

Instantiating the template under the *sparse Learning-Parity-with-Noise (LPN)* assumption yields the first SSL-PRGs with polynomial stretch and constant locality from a subquadratic-sample search hardness assumption; a mild strengthening of sparse-LPN gives strong SSL-PRGs of arbitrary polynomial stretch. We further show that for all standard noise distributions, noisy-local PRGs cannot be emulated by ordinary local PRGs, thereby separating the two notions.

Plugging SSL-PRGs into existing frameworks, we revisit the canonical applications of local PRGs and demonstrate that SSL-PRGs suffice for:

i. indistinguishability obfuscation,
ii. constant-overhead secure computation,
iii. compact homomorphic secret sharing, and
iv. deriving hardness results for PAC-learning DNFs from sparse-LPN.

Our work thus broadens the landscape of low-depth pseudorandomness and anchors several primitives to a common, well-motivated assumption.

1 Introduction

We study the existence of cryptographic primitives that minimize parallel time-complexity, continuing a fruitful line of work initiated by [AIK04]. We are particularly interested in *pseudorandom generators* with *constant* parallel time-complexity (local PRGs), an influential primitive that has found important applications to learning theory [ABW10, AR16a, DV21], hardness of approximation [Ale03, AIK08, App12], secure computation [IKOS08, ADI$^+$17, BCG$^+$17, BCM23, BCM$^+$24], indistinguishability obfuscation [JLS21, JLS22], and more [ABW10, BKR23].

Our main contribution is threefold: first, we put forward the notion of *structured-seed local PRG*, a relaxation of local PRGs, and we study its relation to local PRGs. Second, we provide a general template for constructing structure-seed local PRGs using a natural notion of *noisy*-NC^0 circuits, and we show how to instantiate the template from sparse variants of the learning parity with noise (LPN) assumption. Third, we revisit multiple applications of local PRGs and show that, in many cases, structured-seed local PRGs can be used as a drop-in replacement. As a consequence, we obtain a variety of results such as hardness of PAC learning DNFs, constant-overhead secure computation, and indistinguishability obfuscation, from the sparse LPN assumption (and other standard cryptographic assumptions).

⋆ Work done while author was at IRIF, Université Paris Cité.

1.1 Local pseudorandom generators

A pseudorandom generator (PRG) is a function $G : \{0,1\}^n \to \{0,1\}^m$ with $m = m(n) > n$ such that no polynomial-time algorithms can distinguish the distribution $G(\mathcal{U}_n)$ from $\mathcal{U}_m$ (where $\mathcal{U}_t$ denotes the uniform distribution over t-bit strings). An NC^0 pseudorandom generators (aka local PRG) is a PRG that each of its output bits can be computed based on a constant number of its input bits. The existence of local PRGs was first investigated in the work of Cryan and Miltersen [CM01] and has been the subject of a lot of attention ever since, due in large part to their many applications.

The status of local PRGs varies significantly depending on its *stretch* that is measured by the number of bits $m = m(n)$ that the generator outputs given an n-bit random seed. Local PRGs with sublinear stretch ($m = n + o(n)$) have been shown to exist by Applebaum, Ishai, and Kushilevitz [AIK04] under the widely believed assumption that PRGs exist in the class NC^1 — an assumption implied, in particular, by most standard cryptographic assumptions. When $m = n + \Theta(n)$ (the linear-stretch regime), the same authors showed in [AIK08] that their existence is implied by a specific intractability assumption related to the hardness of decoding "sparsely generated" linear codes that was originally suggested by Alekhnovich [Ale03].

Eventually, the polynomial-stretch regime, where $m = n^c$ for some $c > 1$, is both the most challenging settingand the setting required in most applications. In this regime, all existing constructions build upon a heuristic design originally put forth by Goldreich [Gol00] where a predicate P is evaluated on constant-size subsets of the bits of the seed, chosen to form the hyperedges of a sufficiently expanding uniform hypergraph. Existing constructions either assume that such a function is one-way and show how to transform it into a PRG while preserving locality [App12, AK19], or assume that the direct construction is a PRG.[5] The latter assumption and its variants have been extensively studied and analyzed in the past two decades [MST03, BQ09, OW14, CEMT14, ABR16, AL16, LV17, Zic17, CDM+18, AK19, OST19, Méa, YGJL21, Méa22, Üna23, DMR23]. (See [App16] for a survey.)

1.2 Learning parity with noise

The basic problem. For integers n (length of secret) and m (number of samples), the input to the *Learning Parity with Noise* problem consists of a public matrix $A \in \mathbb{F}_2^{m \times n}$ (typically sampled uniformly at random) and an m-bit vector

$$b = Ax + e,$$

where the secret vector $x \in \mathbb{F}_2^n$ is sampled uniformly at random and $e \in \mathbb{F}_2^m$ is a noise vector that is sampled from some probability distribution over sparse vectors whose Hamming weight is a small fraction of m. Given the pair (A, b) the goal in the search-LPN problem is to recover x. In the decisional variant of the problem (decisional-LPN) the goal is to distinguish (A, b) from (A, r) where r is sampled uniformly from $\mathbb{F}_2^m$. The corresponding hardness assumptions assert that the problems are infeasible for polynomial-time adversaries.

Typical noise distributions include: (i) Bernoulli noise, where every coordinate of e is flipped independently, (ii) exact-weight noise, where e is chosen uniformly at random from all vectors with some fixed Hamming weight, (iii) regular noise, where e is split into equal-sized blocks and in each block a single 1 is placed at a uniformly random position (all other coordinates are 0), and (iv) xor-of-unit, where e is XOR of w unit vectors over $\mathbb{F}_2^m$. Since the seminal work of Blum, Kalai, and Wasserman [BKW00], the best-known algorithms (the BKW family and Information-Set Decoding variants) still run in time about $2^{\tilde{\Theta}(n)}$ for a constant noise rate and $m = \mathsf{poly}(n)$.

[5] The latter assumption seems to require strictly-stronger conditions on the predicate and on the expansion properties of the hypergraph compared to the corresponding one-wayness assumption.

Sparse-LPN. A variant that plays a central role in our work is *sparse-LPN* where the matrix A is chosen at random so that each row of its rows has weight exactly $c \geq 3$. Alekhnovich [Ale03] conjectured that decision-LPN remains hard even for this highly sparse A as long as the matrix is expanding. This property is both common (holds with high probability for random sparse matrices) and also implies large dual distance (i.e., there are no short linear relations among the rows). The latter property guarantees pseudorandomness against many concrete attacks (see, e.g., [ABW10]). Additionally, it is known [ABW10, App12] that the decisional version of the problem reduces to its search version.

Sparse-LPN is tightly connected to several important problems in computer science. For example, breaking sparse-LPN amounts to distinguishing whether instances of the c-XOR problem are close or far from satisfiable, connecting the study of sparse-LPN to that of random CSPs [AIK08, App12] and to the field of hardness of approximation [Fei02]. Sparse-LPN is also strongly connected to learning theory: treating rows of A as examples and b_i as labels turns LPN into learning half-spaces with noise. Feldman et al. [FGKP06] show how efficiently learning noisy (non-sparse) parities implies breaking several other problems including learning half-spaces with random or adversarial classification noise.

1.3 Our contributions

The starting point of our work is the observation that in most of the applications listed above, full-fledged local PRGs are not required: instead, the minimal primitive required is an NC^0-mapping $g : \{0,1\}^n \to \{0,1\}^m$ such that $g(\mathcal{R})$ is indistinguishable from uniform over $\{0,1\}^m$, where $\mathcal{R}$ is a distribution samplable in time $o(m)$. That is, in these applications, *the seed is not required to be uniformly sampled.* Informally, this captures all applications where the use of the local PRG is "single-shot", such as in applications to indistinguishability obfuscation [JLS21, JLS22] and secure computation [IKOS08, ADI+17, BCG+17, BCM23, BCM+24], but excludes constructions where the output of the local PRG would be used as its own seed (such as in the low-complexity distributed point function from [CK24])[6]. Building upon this observation, we provide four main contributions:

1. We formally introduce the notion of structured-seed local PRGs (SSL-PRGs), and revisit several applications of local PRGs, showing how the underlying requirement of local PRGs can be relaxed to our new notion.
2. We exhibit a natural and general strategy to derive SSL-PRGs from a new notion of *noisy-local PRG*. Our approach combines a noisy-local PRG with a *local compression scheme*: lossless compression schemes for subset of strings, equipped with a local decompression procedure. Building upon the negligible-error combinatorial batch code construction of [AK19], we show a simple local compression scheme for the set of all sparse vectors. This scheme could be of independent interest.
3. We put forth a general study of noisy-local PRGs, showing a number of results. First, we separate the power of noisy-local PRGs from that of local PRGs for a wide class of "non-degenerate" noise distributions (that encompasses all standard noise distributions used in the LPN and coding theory literature), showing that all polynomial-stretch local PRGs fail to sample non-degenerate noise distribution within small statistical distance. Second, we revisit previous results on local PRGs and extend them to noisy-local PRGs, such as weak-to-strong hardness amplification[7], stretch expansion, and (in a special case) search-to-decision reductions.

[6] Other applications, such as PAC learning lower bounds [AR16a, DV21] leverage more specifically the specific structure of concrete local PRG candidates based on random local functions. Looking ahead, we observe that our new candidates also enjoy similar applications.

[7] A *weak* PRG only guarantees an inverse-polynomial bound on the distinguishing advantage, while a strong PRG guarantees a negligible advantage in distinguishing a PRG output from random.

4. Eventually, we provide several constructions of noisy-local PRGs from several sparse variants of the learning parity with noise assumption, achieving different tradeoffs between simplicity, locality, and conservativeness of the underlying assumption.

We summarize our main result in an informal theorem below:

Theorem 1.1 (Informal). *Assume the search hardness of the c-sparse LPN assumption for any of the {Bernoulli, exact, regular, xor-of-units} noise distributions, with dimension n, number of samples $m = n^{1+\eta}$ for some constant $\eta = 1-2/c \in (0,1)$, and noise rate $r = n^{\alpha} \cdot n^{-\eta}$ for some constant $\alpha < 1$. Then there exists a strong structured-seed local PRG that stretches n' bits into $m' = (n')^{1+\theta}$ bits, for some constant θ arbitrarily close to η. Furthermore, setting instead the LPN parameters (m, r) to $(n^{1+\gamma/2}, n^{\alpha+\gamma/2-1})$ for a constant $\gamma = \gamma(c)$ with $2\log\gamma/(\log\gamma - 1) \leq c$, there exists a weak structured-seed local PRG with arbitrary polynomial stretch.*

For concreteness, we sketch a few corollaries of our main contributions. We refer the reader to Section 8 for discussions and definitions of the notions involved. All of the below-mentioned results were previously known assuming the existence of local PRGs (and other standard assumptions), which can be based on the one-wayness of random local functions [App12].

Corollary 1.2. *Under the assumption of Theorem 1.1, and further assuming the existence of an oblivious transfer 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 $O(s(n))$.*

Corollary 1.3. *Under the assumption of Theorem 1.1, and further assuming the Learning-With-Errors assumption, the learning parity with noise over an exponentially-large prime-order field $\mathbb{F}_p$, and the Decision Linear assumption on symmetric bilinear groups of order p, there exists collusion-resistant public-key functional encryption for all polynomial-size circuits. Furthermore, under the subexponential hardness of the above assumptions but replacing the assumption of Theorem 1.1 with the subexponential hardness of* decisional *sparse-LPN with* regular *noise, there exists a (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits.*

Corollary 1.4. *Under the assumption of Theorem 1.1, and further assuming the superpolynomial hardness of the Decisional Composite Residuosity assumption, there exists a four-party homomorphic secret sharing scheme for the class of loglog-depth circuits with n inputs with share size $n \cdot (1 + o(1))$.*

Corollary 1.5. *Under the assumption of Theorem 1.1, for every $q(n) = \omega(1)$ there is no efficient algorithm that PAC-learns DNF formulas with n variables and $q(n)$ terms.*

1.4 Concurrent work

Concurrently to our work, the core observation that we build upon was also made in a recent paper [RVV24a]. The full version of their work [RVV24b] appeared online shortly before a preliminary version of the current work. While the observation was made concurrently in both works, the first online occurrence of this observation should therefore be credited to the authors of [RVV24a]. Below, we expand on the differences between the current work and the work of [RVV24a].

The work of [RVV24a] focuses on indistinguishability obfuscation (iO). Their main observation is that the polynomial-stretch local PRG used in a recent construction of iO [JLS22] can be replaced by a structured-seed local PRG. Then, as their main technical contribution, they provide a direct construction of structured-seed local PRGs from the decision-hardness of the sparse-LPN assumption with Bernoulli noise.

In contrast, in the current work, we do not explore the application to iO to the same extent as [RVV24a], confining ourselves to observing that structured-seed local PRGs can be used as a drop-in replacement in [JLS21], without revisiting the (more involved) construction of [JLS22] that gets rid of the learning-with-error assumption but makes more indirect use of the local PRG (replacing their local PRG with an SSL-PRG requires diving more carefully in their security analysis, and was the focus of [RVV24a]). Compared with [RVV24a], the main advantage of our work is three-fold: (1) we initiate a general study of structured-seed local PRGs and noisy-local PRGs, establishing a number of useful results; (2) as a result of our study, we construct SSL-PRGs from a broader class of assumptions, including in particular constructions from the *search* hardness of sparse LPN ([RVV24a] relied on the hardness of the decisional problem) for all standard noise distributions (the construction of [RVV24a] requires sparse LPN with Bernoulli noise); (3) we explore a broader range of applications, outlined in the previous section.

Subsequent work. After a preliminary version of our work appeared online, a subsequent work [BBTV24] has used our result on the hardness of PAC-learning DNFs to prove that under the subexponential hardness of the standard LPN assumption, there is no efficient algorithm that PAC-learns n-variable DNF formulas with $\exp((\log\log n)^{\omega(1)})$ terms.

2 Technical Overview

We start with a simple warm-up example to motivate structured-seed local PRGs, using a minimal amount of niche cryptographic jargon. Recall that a public-key encryption scheme is a triple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ where KeyGen generates a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret key, $\mathsf{Enc}(\mathsf{pk}, M)$ encrypts a message M with the public key, and $\mathsf{Dec}(\mathsf{sk}, C)$ decrypts a ciphertext C with the secret key. Security states, informally, that $\mathsf{Enc}(\mathsf{pk}, M)$ leaks no information on M. Constructions of public-key encryption schemes are well-known and form the cornerstone of modern cryptography. Yet, standard constructions such as ElGamal or RSA require expensive operations (exponentiations over a large group): using them directly to encrypt a large file M would be prohibitively slow. When encrypting a large file, the common (and universally deployed) practice is to use *hybrid encryption*: fix a pseudorandom generator $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{|M|}$ that stretches a short n-bit seed into an $|M|$-bit pseudorandom string. To encrypt M, sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^n$ and set

$$\mathsf{Enc}^*(\mathsf{pk}, M) = (\mathsf{Enc}(\mathsf{pk}, \mathsf{seed}), \mathsf{PRG}(\mathsf{seed}) \oplus M),$$

where $\oplus$ denotes the bitwise-XOR. It is a simple exercise to show that the security of the hybrid scheme reduces to that of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ together with the pseudorandomness of PRG. Moreover, when $|M| \gg n$, the cost of computing $\mathsf{Enc}(\mathsf{pk}, \mathsf{seed})$ using an expensive public-key scheme vanishes asymptotically, and the overall cost of encryption is dominated by the computation of $\mathsf{PRG}(\mathsf{seed})$ (plus the XOR). Decryption of a pair (C_0, C_1) proceeds similarly: it first recovers $\mathsf{seed} \leftarrow \mathsf{Dec}(\mathsf{sk}, C_0)$ and computes $M \leftarrow \mathsf{PRG}(\mathsf{seed}) \oplus C_1$; here again, the cost is asymptotically dominated by the cost of PRG. Using a *local* pseudorandom generator pushes efficiency to an extreme: encrypting or decrypting a large message M of length m requires only $O(m)$ Boolean operations. Furthermore, encrypting or decrypting *any subset* S of the bits of M requires $O(|S|)$ Boolean operations, a highly desirable feature.

In the above example, it is clear that restricting seed to being uniform serves no purpose: the same security and efficiency features would be obtained even if seed was sampled using a different sampling procedure $\mathsf{SampleSeed}$ supported over $\{0,1\}^n$, as long as the cost of running $\mathsf{SampleSeed}$ vanishes as $|M|$ grows. In turn, relaxing the condition on how seed is sampled can plausibly extend our options for instantiating the local PRG: intuitively, rather than enforcing a fully-local mapping, we allow any (possibly expensive and non-local) *preprocessing step* (the

sampling of seed from the random coins of SampleSeed) followed by a fast local expansion step, as long as the preprocessing step is not too costly (the cost vanishes with the output size) and outputs n bits. Formally, we define structured-seed local PRGs (SSL-PRGs) as follows: an SSL-PRG is a 3-tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{SampleSeed}, \mathsf{PRG})$, where PRG can be implemented by an NC^0 circuit. On input 1^n, Setup samples a set of public parameters pp. Given pp, $\mathsf{SampleSeed}$ runs in time much smaller than m (the output size) and outputs a seed seed. Given $(\mathsf{pp}, \mathsf{seed})$, PRG outputs a string $y \in \{0,1\}^m$. An SSL-PRG is (ε, δ)-secure if with probability at least $1-\delta$ over the choice of pp, no polytime adversary $\mathcal{A}$ can distinguish y from random with advantage better than ε.

2.1 Instantiating structured-seed local PRGs via sparse LPN

Recall that the c-sparse LPN assumption with dimension n, m samples, and noise distribution $\mathcal{E} \subset \mathbb{F}_2^m$, states that pairs $(A, Ax+e)$ are indistinguishable from (A, b) where A is a random c-sparse matrix in $\mathbb{F}_2^{m \times n}$ (*i.e.* each row of A is a random weight-c vector), $x \xleftarrow{\$} \mathbb{F}_2^n$, $e \xleftarrow{\$} \mathcal{E}$ is a random noise, and $b \xleftarrow{\$} \mathbb{F}_2^m$. Since its introduction in [Ale03] and due to its appealing efficiency features, many flavors of sparse LPN have been used to design cryptographic primitives, including variants of pseudorandom generators such as local PRGs [AIK08] and pseudorandom correlation generators [BCGI18,BCG+22], but also more advanced primitives such as multiparty homomorphic secret sharing [DIJL23] and somewhat homomorphic encryption [CGHTKV25].

At first sight, sparse LPN does *almost* imply a natural construction of a local PRG: the mapping $x \mapsto A \cdot x$ is c-local. Therefore, if the noise distribution $\mathcal{N}$ is such that sampling $e \leftarrow \mathcal{E}(r)$ can be done via a d-local circuit running on a random tape r of size $|r| < m - n$, we immediately get a $(c+d)$-local PRG $\mathsf{PRG} : (x, r) \mapsto A \cdot x + \mathcal{E}(r)$. Alas, we typically do not know of a local procedure to sample from the standard LPN noise distributions using little randomness. In fact, for some distributions such as uniform fixed-Hamming-weight vectors (the most common flavor in the coding theory literature), it is known that local circuits strongly fail to sample them within small statistical distance even when using a large amount of randomness [Vio10,KOW24]. For other distributions, such as Bernoulli noise with parameter $p = 2^{-k}$, local samplers are known when k is constant [AIK08] (set each output bit to $\prod_{i=1}^{k} r_i$ where the r_i are fresh random bits), but sampling an m-bit vector this way requires $m \cdot k \gg m - n$ bits of randomness; whether Bernoulli distributions can be locally sampled using much less randomness is a long-standing open problem [Vio10][8]. Besides, nothing is known when k is subconstant (even using a large amount of randomness). To circumvent these limitations, we ask:

Is it possible to efficiently preprocess the randomness r into a string r' of size $|r'| \ll m - n$ such that $e = \mathcal{E}(r)$ can be computed from r' via a local mapping?

In this work, we show that the above is indeed feasible, for every distribution $\mathcal{E}$ such that a sample $e \xleftarrow{\$} \mathcal{E}$ is sufficiently sparse with high probability (which captures all standard noise distributions for LPN). As a warm-up, consider the case of sparse LPN with *regular noise*: in this setting, the noise distribution $\mathcal{E}$ is parametrized with a weight parameter w (we assume that w divides m) and a sample e from $\mathcal{E}$ is a concatenation of w random one-hot vectors of length m/w (that is, e is divided into w equal-length "blocks", and each block receives a random 1). The random tape r is a $w \cdot \log(m/w)$-bit string (we assume that m/w is a power of 2 for simplicity) that is parsed into w strings $r_i \in \{0,1\}^{\log(m/w)}$, where r_i encodes the position of the 1 in the i-th block. Fix a constant d, that will provide a tradeoff between the locality and the size of r'. We preprocess r' from r as follows:

[8] The work of [AIK08] introduced an improved strategy that "recycles" the leftover entropy via local hashing. However, due to k being a constant, this approach is necessarily restricted to providing a *linear stretch* local PRG, which does not suffice for our target applications.

- Let k denote the smallest integer such that $k^d \geq m/w$. For each r_i (viewed as an integer in $[0, m/w)$), let $(r_{i,1}, \cdots, r_{i,d}) \in [k]^d$ denote the d coordinates of r_i in the d-dimensional hypercube.
- For each $r_{i,j}$, let $r'_{i,j} \in \{0,1\}^k$ denote the one-hot-vector with a 1 at position $r_{i,j}$. Output $r' = (r'_{i,j})_{i \leq w, j \leq d}$.

The procedure is efficient, and outputs r' of size $w \cdot d \cdot k \approx w \cdot d \cdot (m/w)^{1/d}$, that can be arbitrarily (polynomially) smaller than m. Furthermore, given r', reconstructing e can be done via a d-local mapping: to reconstruct the j-th entry of the i-th block of e, let $(j_1, \cdots, j_d)$ denote the d coordinates of j over the d-dimensional hypercube, and output $e_{i,j} \leftarrow \prod_{\ell=1}^{d} r'_{i,\ell}[j_\ell]$ (where $r'_{i,\ell}[j_\ell]$ denotes the j_ℓ-th bit of $r'_{i,\ell}$). Observe that $e_{i,j} = 1$ iff $j = r_i$, iff $j_\ell = r_{i,\ell}$ for $\ell = 1$ to d, iff $\prod_{\ell=1}^{d} r'_{i,\ell}[j_\ell] = 1$.

Downsides of the previous method. The preprocessing method outlined above can alternatively be seen as a compression scheme for regular vectors that admits a local decompression procedure; we call *local compression scheme* (LCS) such a procedure. Combining the LCS for regular vector with the sparse LPN assumption with regular noise immediately yields a structured-seed local PRG: let Setup sample a random c-sparse matrix A, and $\mathsf{SampleSeed}$ sample $x \xleftarrow{\$} \mathbb{F}_2^n$, $r \xleftarrow{\$} \{0,1\}^{w \log(m/w)}$, and output (x, r') where r' is computed from r via the previous LCS. Eventually, $\mathsf{PRG}(A, (x, r'))$ first locally decompresses r' into e and outputs $A \cdot x + e$. However, this first attempt suffers from two downsides:

- The above procedure is tailored to the regular noise distribution, and it is not immediately clear at first how to extend the ideas to other noise distributions.
- The above SSL-PRG is not fully secure, because the sparse LPN assumption can only guarantee an inverse-polynomial advantage on the distinguishing advantage of a polytime adversary. In line with the standard cryptographic terminology, we say that the SSL-PRG is a *weak* SSL-PRG. This is easily seen by observing that, e.g., the first two rows $a_0^\intercal, a_1^\intercal$ of A are equal with inverse-polynomial probability at least $1/m^c$. Since e is sparse, it follows that $a_0^\intercal \cdot x + e_0$ and $a_1^\intercal \cdot x + e_1$ are very likely to be equal (whenever $e_0 = e_1 = 0$, which happens with probability $\approx 1 - 2w/m$). This yields a simple distinguisher that, given b or $A \cdot x + e$, checks whether pairs of entries corresponding to equal rows of A (if they exist) are equal.

To address each of the issues separately, we first consider a modular approach that abstracts out the previous strategy. At a high level, we isolate two components:

1. A *noisy-local PRG*, a new notion that we introduce. Informally, a noisy-local PRG is a PRG that is computed by a *noisy* NC^0-circuit, which is an NC^0-circuit equipped with the ability to sample from a noise distribution $\mathcal{N}$ (we call $\mathcal{N}$-NC^0 the corresponding class). The sparse LPN assumption immediately implies a *weak* noisy-local PRG, that computes the (local) mapping $x \mapsto A \cdot x$, samples $e \xleftarrow{\$} \mathcal{N}$, and outputs $Ax + e$.
2. A *local compression scheme* for $\mathcal{N}$, that compresses samples from $\mathcal{N}$ to $o(m)$-size strings in time $\ll m$ and admits a local decompression procedure.

In a similar fashion as before, combining a noisy-local PRG in $\mathcal{N}$-NC^0 with an LCS for $\mathcal{N}$ immediately yields an SSL-PRG. We study each of these components and prove a number of results:

Noisy-local PRGs. We show (i) **weak-to-strong amplification**: weak noisy-local PRGs with subquadratic stretch (i.e., $m = o(n^2)$) imply *strong* noisy-local PRGs with subquadratic stretch (where the distinguishing advantage of any polytime adversary is negligible), and (ii) **stretch amplification**: weak noisy-local PRGs with subquadratic stretch imply weak noisy-local PRGs with *arbitrary polynomial stretch*. We note that we leave open the intriguing question of boosting weak noisy-local PRGs with subquadratic stretch all the way to strong noisy-local PRGs with arbitrary stretch: our two reductions are in a sense incompatible.

Local compression schemes. Second, defining noise distributions to be any distribution that samples sparse vectors with high probability (this captures all standard noise distributions), we introduce a negligible-failure LCS that works for all noise distributions – in fact, our LCS polynomially compresses *any* sparse vector.

A summary of these reductions, and their implications to our applications, is provided on Figure 1. Taken together, these results imply that we can start from any flavor of sparse LPN (with inverse-polynomial bound on the adversarial advantage) with any standard noise distribution $\mathcal{N}$, use it to build a weak noisy-local PRG, generically boost it to a (subquadratic stretch) strong noisy-local PRG (or to a long-stretch weak local PRGs, depending on the target application), and finally convert the strong noisy-local PRG into a full-fledged SSL-PRG via our generic LCS for sparse vectors. In the following sections, we briefly overview these reductions.

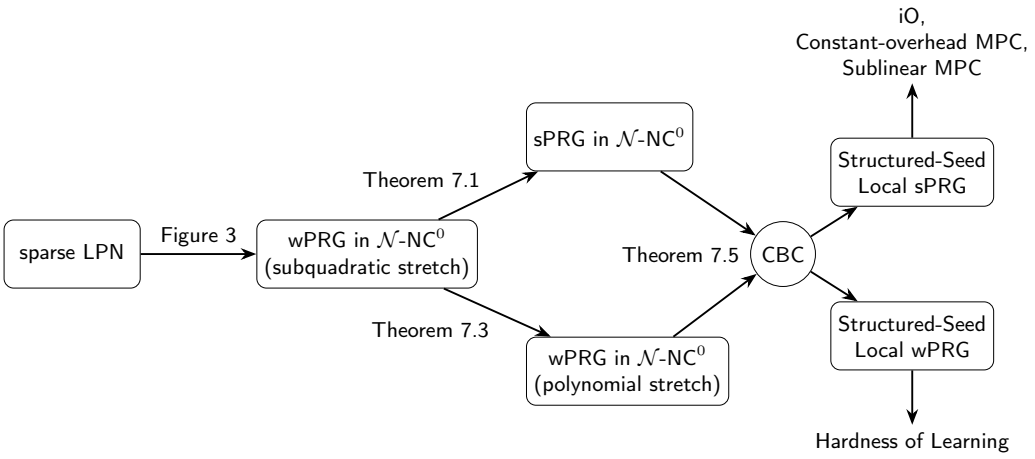

Fig. 1. Summary of the implications shown in this work.

2.2 Noisy-local PRGs

We now overview our results on noisy-local PRGs.

Weak-to-strong amplification. A generic weak-to-strong amplification for local PRGs was described in the work of [AK19, Theorem 2.12]; our proof mostly follows the original analysis and adapts it to the setting of noisy-local PRGs, which introduces some minor technicalities. A first technicality is that the proof of Theorem 2.12 in [AK19] converts an (arbitrary stretch) local PRG into a subquadratic stretch local PRG. This should be clear from Figure 2: the strong PRG takes as input n PRG seeds x_i (each of length n bits) and m extractor seeds s_i (each of length n^α for some $\alpha < 1$) to produce $m \cdot n$ bits. This is not an issue in [AK19] as a local PRG with any polynomial stretch immediately implies a local PRG with an arbitrary polynomial stretch (by recursively evaluating the PRG on its output for a large enough constant number of iterations). However, in the setting of noisy-local PRG, both the reduction outlined on Figure 2 and the recursive self-composition *blow-up the noise*. Looking ahead, when combining both reductions, the amount of noise becomes so large that, even after (optimal) compression, the resulting SSL-PRG always has subquadratic stretch. A last minor technicality is that the proof of [AK19] requires using the fact that a PRG is in particular an unpredictable generator (see Definition 3.6) via a result of Yao. In our setting, in order to apply this result directly, we have to first convert the noisy-local circuit into a regular (non-local) circuit using by composing it with the sampler for the noise distribution, and applying Yao's lemma to the resulting circuit. Here, a simple

but important observation is that the resulting circuit might *not* be a PRG (as it might not be expanding its input), but it nevertheless samples from a pseudorandom distribution, which suffices to apply Yao's result.

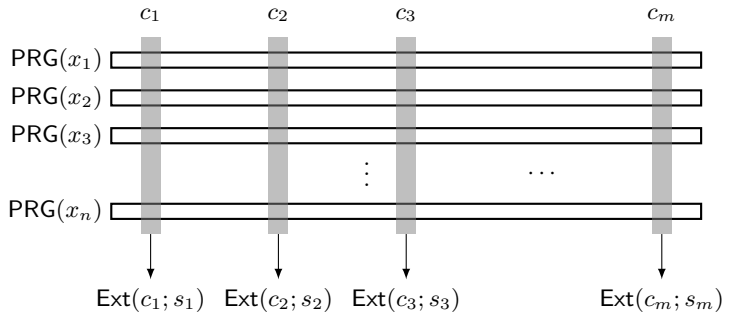

Fig. 2. Representation of the weak-to-strong amplication for local PRGs from [AK19]. Each x_i is an n bit seed stretched into an m-bit string. The c_i denotes the columns (the i-th bit of each string) and Ext denotes a local extraction procedure. Each invocation of Ext takes as auxiliary input an independent seed s_i of size n^α, for some constant $\alpha < 1$, and extracts n bits from the n-bit string c_i. The final PRG stretches $n^2 + m \cdot n^\alpha$ bits to $m \cdot n$ bits.

Stretch amplification. We also outline a simple stretch amplification procedure that recursively invokes the PRG on itself. The reduction itself is straightforward, but the precise statement of the resulting lemma requires carefully tracking the evolution of the noise parameters through the recursive calls.

On the power of noisy-local PRGs. Eventually, we study the sampling power of noisy-local PRGs. Indeed, noisy-local circuits are a relaxation of local circuits whose aim is to enable a broader class of instantiations, but this relaxation can only be useful if it non-trivially extends the sampling power of local circuits. Concretely, we ask the following: consider a noise distribution $\mathcal{N}$ over $\{0,1\}^s$ that samples vectors of weight at most w. Is there a circuit $\mathcal{C}^* : \{0,1\}^n \to \{0,1\}^m$ such that for all inputs $x \in \{0,1\}^n$ and every NC^0-circuit $\mathcal{C} : \{0,1\}^{s^\alpha} \to \{0,1\}^m$, it holds that $\mathcal{C}$ fails to sample the distribution induced by $\mathcal{C}^*$, *i.e.,* $\mathcal{C}^*(x)$ and $\mathcal{C}(\mathcal{U}_{s^\alpha})$ are far in statistical distance? Here, $\alpha < 1$ is some constant, and the reader can think of s as $s = m$, as it is the case for the sparse-LPN based constructions of noisy-local PRGs (though we consider the setting in its full generality). If the answer is negative, then every noisy-local PRG stretching n bits to m bits could be converted into a standard local PRG stretching $n + m^\alpha$ bits to m bits.

We prove that the answer is yes for a large class of *non-degenerate* noise distributions. Informally, we say that a distribution $\mathcal{N}$ over vectors of $\{0,1\}^s$ of weight at most w is non-degenerate if

- it has high Shannon entropy: $H(\mathcal{N}) \geq w$;
- its nonzero outputs are reasonably well-spread. Formally, we consider both a notion of *weak* non-degeneracy, where we only require that the marginal distribution of each output bit has some inverse-polynomial probability of being 1, and a notion of *strong* non-degeneracy, where we instead require that the marginal distribution of each sufficiently large subset of the bits (e.g., subsets of size at least $s/2$) retains a sufficiently large amount of entropy (e.g., at least $w/3$ bits).

Both conditions are mild (they hold for all standard noise distributions for LPN variants) and allow ruling uninteresting degenerate cases, such as distributions outputting always the same fixed strings, or distributions outputting w uniform random bits concatenated with zeroes – such degenerate distributions, while sparse, are easy to sample via local circuits. Given a non-degenerate distribution, the proof proceeds in two cases:

1. Either w is close to s. In this case, we let $\mathcal{C}^*$ be the $\mathcal{N}$-NC^0 circuit that simply outputs its noisy sample r. $H(\mathcal{N}) \geq w$ is much larger than $H(\mathcal{C}(\mathcal{U}_{s^\alpha})) \leq s^\alpha$ for every circuit $\mathcal{C}$. Using the Fannes-Audenaert inequality from quantum mechanics, the large entropy gap implies a large statistical distance between the distribution.
2. Or w is significantly smaller than s (say, $w \leq s/f$ for some slowly growing f), in which case there must be at least $s/2$ output bits that are 1 with probability at most $1/(2f)$. Let $\mathcal{C}^*$ denote the noisy-local circuit that simply outputs these bits. Fix any d-local circuit $\mathcal{C}$: the marginal distribution of any of its output bits (on a random input) is either the always-0 distribution, or a Bernoulli distribution with parameter larger than $1/2^d$, which is a constant. The latter case implies a large statistical gap with the marginal distribution on any output bit of $\mathcal{C}^*$ (since their marginal distribution is Bernoulli with parameter at most $1/(2f)$). In the former case, we get again a statistical gap using either the weak or the strong non-degeneracy of $\mathcal{N}$, where the gap is much larger if we assume strong non-degeneracy.

It is instructive to briefly compare our separation with that of Viola [Vio10]. Both results are incomparable: on the one hand, Viola's result considers a much larger class of local circuits, that are allowed to take *slightly more* than $H(\mathcal{N})$ random bits as input, while we restrict the circuits to taking at most s^α bits for a constant α (this is without loss of generality for our application, since we focus on polynomial-stretch local PRGs). On the other hand, Viola's result is restricted to a single specific noise distribution (the uniform distribution over vector of Hamming weight exactly w), while we cover a large class of non-degenerate noise distributions.

2.3 Local compression schemes

We now turn our attention to local compression schemes for sparse vectors. At the heart of our construction is a negligible-failure *combinatorial batch-code* (CBC) from [AK19]. At a high level, the CBC of [AK19] provides a way to distribute the bits of any string $x \in \{0,1\}^s$ across $T = w^{1.01}$ *buckets* such that any w-tuple of coordinates of x can be recovered by reading at most a single entry from each bucket. Furthermore, the CBC is t-uniform for a constant t: each bit of x is assigned to at most t buckets. Equivalently, the CBC defines a bipartite graph with n nodes on top, T nodes on the bottom, where the top nodes have degree at most t and with the property that for every w-subset of selected top nodes, there exists a choice of edges such that all selected nodes are connected to distinct buckets. In addition, *finding* these edges amounts to solving a maximum matching in the graph, which can be done in almost-linear time $w^{1+o(1)}$.

Given a t-uniform CBC, we construct a LCS for w-sparse vectors as follows: given a w-sparse vector $x \in \{0,1\}^s$, find in time $w^{1+o(1)}$ an assignment of each index i of a nonzero entry of x to distinct buckets. Define the content of the bucket that i gets assigned to to be the length-s one-hot vector with a 1 at position i. To all unassigned buckets, assign the all-zero vector 0^s. Then, compress the unit vector in each bucket using the local compression procedure from Section 2.1: each bucket gets compressed to $d \cdot s^{1/d}$ bits, where d is a constant. Output the string r' of length $ds^{1/d}w^{1.01}$ concatenating the compressed buckets.

For local decompression, observe that by the t-uniformity of the scheme, for each input bit at position i, there cannot be more than t buckets that i can possibly have been assigned to. If i corresponds to a 1 in x, then exactly one of these buckets will have a 1 at position i (before compression), else all of these buckets will have a 0 at position i. Therefore, decompression simply decompresses the i-th bit in each of the t buckets, using total locality $d \cdot t$, and XOR the outputs.

3 Preliminaries

Notation and Distributions

Notation. We denote $[n]$ as the set $\{1, \cdots, n\}$, and λ as the security parameter. For any distribution $\mathcal{D}$, we denote $x \xleftarrow{\$} \mathcal{D}$ is the process of sampling uniformly x over the distribution $\mathcal{D}$.

We write vectors using bold font and matrices using caps. Given a vector $\mathbf{v}$, the Hamming weight of $\mathbf{v}$ is $\mathsf{HW}(\mathbf{v})$, 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 $\mathbb{F}_2^m$, $\mathsf{u}_n(i)$ to denote a unit vector of length n having non-zero i-th entry.

Distributions. The uniform distribution $\mathcal{U}_n$ over the set $\{0,1\}^n$ assigns equal probability to each binary string of length n. We denote $\mathcal{A} = (\mathcal{A}_n)_{n\in\mathbb{N}}$ as the family of non-uniform adversaries having the circuit size of at most $T = T(n)$, $\mathcal{A}$ is said to be efficient, subexponential and exponential depending on whether the circuit size of T is polynomial, subexponential, or exponential in n, respectively.
Given an adversary $\mathcal{A}$ and a pair of distribution ensembles $Y = \{Y_n\}$ and $Z = \{Z_n\}$, we write $\mathsf{Adv}_{\mathcal{A}}(Y, Z)$ to denote

$$\mathsf{Adv}_{\mathcal{A}}(Y, Z) = \left|\Pr[\mathcal{A}(1^n, Y_n) = 1] - \Pr[\mathcal{A}(1^n, Z) = 1]\right|.$$

We say that the pair of distributions (Y, Z) is polynomially indistinguishable and subexponentially indistinguishable if $\mathsf{Adv}_{\mathcal{A}}(Y, Z) \leq \mathsf{negl}(n)$, and $\mathsf{Adv}_{\mathcal{A}}(Y, Z) \leq \exp(-n^c)$ for a real number $c > 0$ respectively.
The statistical distance (also known as total variation distance) between two distributions Y and Z over a finite domain $\mathcal{X}$ is defined as:

$$\Delta(Y, Z) = \frac{1}{2}\sum_{x\in\mathcal{X}} |Y(x) - Z(x)|$$

3.1 LPN Assumptions

Let $\mathbb{F}_2$ be the binary field. The LPN assumption states, informally, that given a generating matrix $A \in \mathbb{F}_2^{m\times n}$ it is computationally intractable to distinguish between a random vector $A \cdot \mathbf{x}$ in the image of A that is perturbed with some noise vector $\mathbf{e}$, from a uniform vector $\mathbf{b}$ of length n. More formally, let λ be a security parameter, we parameterize the assumption with n and m that are functions of λ, and with an ensemble of matrix distributions $\mathcal{M}_{m,n}$ and noise distributions $\mathcal{D}$. As usual, indistinguishability is captured by two parameters T (complexity of the adversary) and ε (distinguishing gap). It will be convenient to introduce another parameter δ that captures the event of sampling a "bad" matrix for which the assumption does not hold.

Definition 3.1 (Learning parity with noise (decisional version)). *Let $m(\cdot) : \mathbb{N} \to \mathbb{N}$ be a polynomially-bounded function. For any integer $n \in \mathbb{N}$, let $m = m(n)$, let $\mathcal{M}_{m,n}$ be a matrix distribution over $\mathbb{F}_2^{m\times n}$, and let $\mathcal{E}_m$ be a "noise" distribution over $\mathbb{F}_2^m$. Then, we say that the $(\mathcal{M}, \mathcal{E})$-$\mathsf{LPN}$ problem is (T, ε, δ)-hard if for every probabilistic non-uniform adversary $\mathcal{A} = (\mathcal{A}_n)_{n\in\mathbb{N}}$ of size at most $T = T(n)$, it holds that for all large enough n,*

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

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

In this work, we will also consider the search version of Definition 3.1:

Definition 3.2 (Learning parity with noise (search version)). *Fix the same notations as Definition 3.1. We say that the* search $(\mathcal{M}, \mathcal{E})$-LPN *problem is* (T, ε, δ)*-hard if for every probabilistic non-uniform adversary* $\mathcal{A} = (\mathcal{A}_n)_{n\in\mathbb{N}}$ *of size at most* $T = T(n)$*, it holds that for all large enough* n*,*

$$\Pr_{A\xleftarrow{\$}\mathcal{M}_{m(n),n}}\left[\Pr_{x\xleftarrow{\$}\{0,1\}^n}[\mathcal{A}_n(A, Ax+e) = x] > \varepsilon(n)\right] \leq \delta(n).$$

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/m), exact noise ($\mathbf{e}$ is uniformly random over the set of vectors of Hamming weight w), and "regular" noise ($\mathbf{e}$ is composed of m/w blocks where each block is sampled as a 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:

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

When m 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 $w \ll \sqrt{m}$ (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 $1 - w^2/m$. In turn, LPN with noise sampled from $\mathcal{S}_w$ is known to be equivalent to LPN with noise sampled from $\mathcal{B}_w$ [AIK08,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 3.1 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 $\{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathcal{M}_{m,n}, \mathbf{b} \xleftarrow{\$} \mathcal{D}_0^A\}$ from $\{(A, \mathbf{b}) \mid A \xleftarrow{\$} \mathcal{M}_{m,n}, \mathbf{b} \xleftarrow{\$} \mathcal{D}_1^A\}$ (that is, the distinguishing advantage is also quantified over the random choice of A). Definition 3.1 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 5.

3.2 Probability lemmas

Lemma 3.3 (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 3.4 (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 3.5 (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 $\Pr[\bigoplus_{i=1}^{N} X_i = 0] = 1/2 + (1-2r)^n/2$.*

3.3 Extractors and unpredictability

Collections of functions. It will be convenient to work with an extension of cryptographic primitives to the setting of *collections of functions* where each function is indexed by some public identifier that is sampled by some efficient sampling algorithm and is always given to the adversary. Formally, such a collection is given by a PPT index generation algorithm $K(1^n)$ that samples indices, and an efficiently computable universal evaluation function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ that takes an index σ and an input x and outputs the evaluation of the σ-th function on x. By default, ensembles are length-regular and there exists polynomially-bounded function $(s(n), k(n), m(n))$ such that $K(1^n)$ samples an identifier $\sigma \in \{0,1\}^{s(n)}$ and $F(\sigma, \cdot)$ is a mapping from $k(n)$-bits to $m(n)$ bits. By default, the input length $k(n)$ will be taken to be n and all functions will be efficiently computatble. We slightly abuse notation and denote such a collection by $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$.

Definition 3.6 (Unpredictable Generator (UG)). Let $m = m(n) > n$ be a length parameter. A collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ with index-sampler K is ϵ-pseudorandom generator (PRG) if the ensemble $(K(1^n); F_{K(1^n)}(U_n))$ is ϵ-indistinguishable from the ensemble $(K(1^n), U_m)$, where U_n is the uniform distribution over $\{0,1\}^n$. When ϵ is negligible, we refer to F as a pseudorandom generator.

The collection of functions F is an ε-unpredictable generator if for every efficient adversary $\mathcal{A}$ and efficiently samplable family of indices $\{i_n\}$, where $i_n \in [m(n)]$, we have that

$$\Pr_{\sigma \leftarrow K(1^n), x \leftarrow \{0,1\}^n}[\mathcal{A}(\sigma, F_\sigma(x)_{[1 \ldots i_n - 1]}) = F_\sigma(x)_{i_n} < \varepsilon(n)$$

Lemma 3.7 ([Yao82]). *A $(1/2 + \varepsilon)$-unpredictable generator of output length $m(n)$ is an $(m \cdot \varepsilon)$-PRG.*

We note that while the condition $m > n$ is typically required in applications of Yao's lemma (to obtain a non-trivial PRG), it is not required *per se* to prove Lemma 3.7.

Definition 3.8 (Randomness Extractors). Let X be a random variable, distributed over $\{0,1\}^N$. Define the minentropy of X by $H_\infty(X) := \log \frac{1}{max_x \Pr[X=x]}$. Let $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^s \to \{0,1\}^m$ be a function. We say that Ext is a (ℓ, μ)-extractor if for every distribution X with minentropy at least ℓ it holds that $\Delta(\mathsf{Ext}(X; U_s); U_m) \leq \mu$, that is, the statistical distance between the random variable $\mathsf{Ext}(X; U_s)$ and U_m is at most μ.

The first entry of Ext is called the source and the second entry is called the seed. An extractor Ext is *local* if each output bit can be computed from a constant number of bits of the source and the seed. While explicit constructions of constant-locality extractors are not known, the work of [AK19] exhibited an efficient sampling procedure that outputs a constant-locality extractor with overwhelming probability:

Lemma 3.9 (Local Extractors [AK19]). *Let c, d be constants such that $d > 4c^2 + 8c + 1$. Let $\delta = \delta(n) < 1$. For a prime power n there exists a $\mathsf{poly}(n)$-time negligible-error construction of a local function (represented as a circuit) $\mathsf{Ext} : \{0,1\}^{n^c} \times \{0,1\}^{3n} \to \{0,1\}^{n^c}$ which is a (k, μ)-extractor, for $k = (1-\delta)n^c$ and $\mu = \exp(-\Omega(n^\varepsilon)) \cdot 2^{\delta n^c/2 - 1/2}$ for any $\varepsilon < \frac{r-2c}{(r-2)(2c+1)}$, for $r = (d - (2c+1)^2)/2$. Moreover, each bit of the output depends on exactly one bit of the input and d bits of the seed.*

We denote by $\mathsf{SampleExt}(1^n, c, d)$ the procedure promised by Lemma 3.9.

4 Defining Noisy-Local PRGs and Structured-Seed Local PRGs

4.1 Noisy local circuits

The class NC^0 is the class of functions computable by uniformly generatable boolean circuits with constant depth and bounded fan-in (equivalently, boolean circuits where every output bit depends on a constant number of input bits). Given an integer ℓ, we denote by NC^0_ℓ the class of Boolean circuits where every output bit depends on at most ℓ input bits.

Noise distributions. In this work, we say that a distribution is a *noise distribution* if it is concentrated over low-weight vectors. Formally, we say that a family of distributions $\mathcal{N} = \{\mathcal{N}_n\}_{n\in\mathbb{N}}$ is a $(w(n), s(n))$-noise distributions if $\mathsf{Supp}(\mathcal{N}_n) \subseteq \{0,1\}^{s(n)}$ and $\Pr_{x\xleftarrow{\$}\mathcal{N}_n}[\mathsf{HW}(x) > w(n)] \leq \mathsf{negl}(n)$. In this work, we will typically be interested in the inverse-polynomial noise regime, where $w(n) \leq s(n)^\alpha$ for some constant $\alpha < 1$.

The noisy-NC^0 model. We introduce below the notion of $\mathcal{N}$-NC^0 circuits, which are randomized NC^0 circuits where the random tape is filled with a sample from a noise distribution $\mathcal{N}$. More formally, we define the model of $\mathcal{N}$-NC^0 circuits as follows:

Definition 4.1. *Let $\mathcal{N} = \{\mathcal{N}_n\}_{n\in\mathbb{N}}$ be a noise distribution over $\{0,1\}^{s(n)}$. An $\mathcal{N}$-NC^0 circuit $C : \{0,1\}^n \to \{0,1\}^m$ is a bounded fan-in constant-depth circuit with two types of input gates: n standard input gates and at most $s(n)$* noisy *input gates. on input $x \in \{0,1\}^n$, a random sample $r \xleftarrow{\$} \mathcal{N}_n$ is picked from $\mathcal{N}$ and placed on the noisy input gates (the i-th noisy gate receives r_i). We denote $C(x;r)$ the result of evaluating the noisy circuit C on input x with randomness r.*

The above definition is very general. We specialize it to the common case where the noise distribution is a Bernoulli distribution with parameter p:

Definition 4.2. *Let $p \in (0,1)$ be a real. A p-NC^0 circuit $C : \{0,1\}^n \to \{0,1\}^m$ is a bounded fan-in constant-depth circuit with two types of input gates: n standard input gates and* noisy *input gates. on input $x \in \{0,1\}^n$, each noisy input gate receives a random sample from $\mathcal{B}_p(\mathbb{F}_2)$.*

4.2 Noisy local PRGs

A cryptographic pseudorandom generator (PRG) is an algorithm $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$, with $m > n$, 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. A pseudorandom generator is *local* if each output bit depends on a constant number of input bits; equivalently, PRG can be computed by an NC^0 circuit.

In the formal definition below and given a noise distribution $\mathcal{N}$, we expand the notion of local pseudorandom generator to also capture *$\mathcal{N}$-local PRGs*, where PRG is a randomized algorithm computable by an $\mathcal{N}$-NC^0 circuit. Formally,

Definition 4.3 (Noisy Local Pseudorandom Generator). *Let $m(\cdot)$ be a polynomial, and let $\mathcal{N} = \{\mathcal{N}_n\}_{n\in\mathbb{N}}$ be a family of efficiently samplable distributions over $\{0,1\}^n$. A noisy pseudorandom generator with stretch $m(\cdot)$ is a pair of uniform p.p.t. algorithms $(\mathsf{Setup}, \mathsf{PRG})$, with:*

- $\mathsf{Setup}(1^n)$. *A probabilistic algorithm that on input 1^n outputs public parameters pp.*
- $\mathsf{PRG}(\mathsf{pp}, \mathsf{seed}; r)$. *A probabilistic algorithm that on inputs a seed $\mathsf{seed} \in \{0,1\}^n$, public parameters pp, and running on random tape $r \in \mathsf{Supp}(\mathcal{N})$, outputs a string $y \in \{0,1\}^{m(n)}$.*

A noisy local PRG must be local *and* secure*:*

Locality. *A noisy PRG is* $\mathcal{N}$-local *if* $\mathsf{PRG}(\mathsf{pp}, \cdot)$ *is implementable by an* $\mathcal{N}$-NC^0 *circuit for every* pp *in the support of* $\mathsf{Setup}(1^n)$*; we typically think of* Setup *as an efficient algorithm that samples the noisy local circuit computing* $x \mapsto \mathsf{PRG}(\mathsf{pp}, x)$*.*

Security. *A noisy local PRG is* (T, ϵ, δ)-secure *if for any* non-uniform *p.p.t adversary* $\mathcal{A} = (\mathcal{A}_n)_{n\in\mathbb{N}}$ *of size at most* $T = T(n)$*, for all* $n \in \mathbb{N}$*,* $\Pr[\mathsf{Adv}_{\mathcal{A}_n}(\mathcal{D}_0, \mathcal{D}_1) > \varepsilon] \leq \delta$*, where the probability is over the choice of* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^n)$ *and for* $b \in \{0,1\}$*,* $\mathcal{D}_b = \mathcal{D}_b^n(\mathsf{pp})$ *denotes the family of distributions*

$$\{(\mathsf{pp}, z_b) \mid x \xleftarrow{\$} \{0,1\}^n, r \xleftarrow{\$} \mathcal{N}, z_0 \xleftarrow{\$} \{0,1\}^{m(n)}, z_1 \leftarrow \mathsf{PRG}(\mathsf{pp}, x; r)\}.$$

We say that a noisy pseudorandom generator is (ε, δ)*-secure if it is* (T, ε, δ)*-secure for every* $T(n) = \mathsf{poly}(n)$*, and that it is* γ*-secure if it is* $(\varepsilon, \gamma - \varepsilon)$*-secure for some* $\varepsilon \leq \gamma$*.*

If $(\mathsf{Setup}, \mathsf{PRG})$ is a γ-secure noisy PRG with $\gamma(n) = \mathsf{negl}(n)$, we say that $(\mathsf{Setup}, \mathsf{PRG})$ is a *strong noisy PRG*; otherwise, we call it a *weak noisy PRG*. If $(\mathsf{Setup}, \mathsf{PRG})$ is (ε, δ)-secure with $\varepsilon = \mathsf{negl}(n)$ and $\delta = 1/\mathsf{poly}(n)$, we say that $(\mathsf{Setup}, \mathsf{PRG})$ is an *almost-strong* noisy PRG. We say that a noisy local PRG has *polynomial stretch* if $m(n) = n^{1+\Omega(1)}$. We recover the standard definition of local PRGs by considering noisy NC^0 circuits with no noisy input gates.

4.3 Separating noisy-local circuits from local circuits

In this section, we study the power of noisy-local circuits. In essence, we ask: do noisy-local circuits bring additional power over (standard) local circuits for computing pseudorandom generators? Slightly more formally, we consider local circuits that stretch s bits of outputs from s^α bits of inputs (for some constant $\alpha < 1$), and we ask whether every noisy-local circuit can be emulated within small statistical distance by such a circuit. We prove that this is not the case:

Claim (Informal). For every non-degenerate noise distribution $\mathcal{N}$, there exists a circuit in $\mathcal{N}$-NC^0 whose output distribution cannot be emulated within small statistical distance by any NC^0 circuit $\mathcal{C} : \{0,1\}^{s^\alpha} \to \{0,1\}^s$ that receives a uniform seed as input.

Non-degenerate noise distributions. Before we proceed with the formal statement and proof, let us discuss the notion of a *non-degenerate* noise distribution. Intuitively, there exists (w, s)-noise distributions that can clearly be emulated by local circuits: consider for example the distribution $\mathcal{N}$ that always outputs the string $1^w 0^{s-w}$. It is clear that an $\mathcal{N}$-NC^0 circuit has no more sampling power than a standard local circuit. More generally, if $\mathcal{N}$ has very low Shannon entropy, it can plausibly be sampled efficiently by a local circuit in many cases (though not in *all* cases).

Hence, as a first non-triviality requirement, we restrict our attention to *high-entropy* noise distributions. Because a (w, s)-noise distribution cannot hope to have Shannon entropy much higher than $\log_2 \binom{s}{w} \approx w \log s$, we define (somewhat arbitrarily) a high-entropy (w, s)-noise distribution $\mathcal{N}$ to be such that for every large enough n, it holds that $H(\mathcal{N}_n) \geq w(n)$.

Is high entropy a sufficient condition for proving our target claim? Unfortunately not: consider the distribution $\mathcal{N}$ that samples $r \xleftarrow{\$} \{0,1\}^w$ and outputs $r0^{s-w}$. $\mathcal{N}$ has Shannon entropy exactly w and satisfies the sparsity condition. Yet, if $w \leq s^\alpha$, an NC^0 circuit $\mathcal{C} : \{0,1\}^{s^\alpha} \to \{0,1\}^s$ receiving a random seed as input can clearly emulate sampling from $\mathcal{N}$! Intuitively, this stems from the fact that the nonzero entries are too concentrated over a small subset of output bits. As a second nontriviality requirement, we therefore enforce that the 1's are reasonably "spread out" by requiring that there exists a polynomial $p = p(n)$ such that for every $i \leq s$, it holds that $\Pr_{r \xleftarrow{\$} \mathcal{N}}[r_i = 1] \geq 1/p(n)$. Eventually, we also consider a stronger notion of "well-spread" distribution that requires, roughly, that the *entropy* is well-spread over the outputs (rather than just the marginal probabilities of being equal to 1), by requiring that any restriction of $\mathcal{N}$ to half of its bits has entropy at least $w(n)/3$ (here again, the exact bound is somewhat arbitrary).

Noisy-local circuits versus local circuits. We now proceed with a formal statement of the non-degeneracy condition.

Definition 4.4. *Let $p : \mathbb{N} \to \mathbb{N}$ be a polynomial. We say that a $(w(n), s(n))$-noise distribution $\mathcal{N}$ is* p-non-degenerate *if $\mathcal{N}$ is high entropy, and for every $i \leq s(n)$, it holds that $\Pr_{r \xleftarrow{\$} \mathcal{N}}[r_i = 1] \geq 1/p(n)$. We say that $\mathcal{N}$ is* strongly non-degenerate *if it is high entropy and the restriction of $\mathcal{N}$ to any subset of at least half of its bits has Shannon entropy at least $w(n)/3$.*

We note that these are mild requirements satisfied by most natural noise distribution, and in particular, by all noise distributions considered in this work and in the LPN literature. We now show that non-degeneracy of the noise distribution suffices to separate noisy-local circuits from local circuits.

Lemma 4.5. *Let $p : \mathbb{N} \to \mathbb{N}$ be a polynomial, $s : \mathbb{N} \to \mathbb{N}$ be any strictly increasing function of n, $12 < w(n) \leq s(n)$, and $\mathcal{N}$ be a $p(n)$-non-degenerate $(w(n), s(n))$-noise distribution. There exists an $\mathcal{N}$-NC^0 circuit $\mathcal{C}^* : \{0,1\}^n \to \{0,1\}^m$ such that for any constant $\alpha < 1$, any large enough $n \in \mathbb{N}$, any input $x \in \{0,1\}^n$, and any NC^0 circuit $\mathcal{C} : \{0,1\}^{s(n)^\alpha} \to \{0,1\}^m$, it holds that*

$$\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha})) \geq \min\left(\frac{1}{p(n)}, \frac{1}{\log^\star n}\right).$$

Furthermore, if $\mathcal{N}$ is strongly non-degenerate, we have the stronger inequality

$$\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha})) \geq \min\left(\frac{w(n)}{3s(n)}, \frac{1}{\log^\star n}\right).$$

Our proof will use a special case of the Fannes-Audenaert inequality [Aud07] in quantum mechanics. The original lemma provides a general bound on the difference between the von Neumann entropies of two density matrices as a function of their trace distance. When specializing the density matrices to diagonal matrices, the von Neumann entropy coincides with the Shannon entropy of the probability distribution defined by the diagonal, and the trace distance coincides with the statistical distance. We state the special case below:

Lemma 4.6 (Fannes-Audenaert [Aud07]). *Let P, Q denote two probability distributions over the support $\{0,1\}^s$. For any $\varepsilon \in (0,1)$, if $|H(P) - H(Q)| > 1 + \varepsilon \cdot s$, then $\Delta(P, Q) > \varepsilon$.*

Proof. Fix $\alpha < 1$. Let f denote an arbitrarily slowly growing function with $f(n) < s(n)/(2(s(n)^\alpha + 1)$ (e.g. $f(n) = \log^\star(n)$). For every large enough $n \in \mathbb{N}$, we consider two cases.

Case 1: $w(n) \geq s(n)/f(n)$. In this case, by assumption, $H(\mathcal{N}_n) \geq s(n)/f(n)$. Set $m = s$ and let $\mathcal{C}^* : \{0,1\}^n \to \{0,1\}^{s(n)}$ denote the noisy-local circuit that ignores its input and outputs its noisy input $r \xleftarrow{\$} \mathcal{N}_n$. For any circuit $\mathcal{C}$, $H(\mathcal{C}(\mathcal{U}_{s(n)^\alpha}))) \leq s(n)^\alpha$, hence for any $x \in \{0,1\}^n$,

$$H(\mathcal{C}^*(x)) - H(\mathcal{C}(\mathcal{U}_{s(n)^\alpha}))) \geq s(n)/f(n) - s(n)^\alpha > 1 + \varepsilon \cdot s(n),$$

where $\varepsilon(n) = 1/(2f(n))$, and the inequality holds for a sufficiently large n. Then, by the Fannes-Audenaert inequality (Lemma 4.6), it holds that $\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha}))) > 1/(2f(n))$.

Case 2: $w(n) < s(n)/f(n)$. In this case, $\Pr[\mathsf{HW}(r) \leq s(n)/f(n)] \geq 1 - \mathsf{negl}(n)$ over the choice of $r \xleftarrow{\$} \mathcal{N}_n$. By a pigeonhole principle, this implies that there exists a subset $S \subset [s(n)]$ of size $|S| > s(n)/2$ such that for all $i \in S$,

$$\Pr_{r \xleftarrow{\$} \mathcal{N}_n}[r_i = 1] \leq 2/f(n) + \mathsf{negl}(n).$$

Fix S, and let $\mathcal{C}^* : \{0,1\}^n \to \{0,1\}^{|S|}$ denote the circuit that, on input x and noisy input r, ignores x and outputs $(r_i)_{i\in S}$. Now, fix any circuit $\mathcal{C} : \{0,1\}^{s(n)^\alpha} \to \{0,1\}^{|S|}$ in NC^0, and let d denote the (constant) locality of $\mathcal{C}$. For every $i \le |S|$, let $\mathcal{C}_i : \{0,1\}^d \to \{0,1\}$ denote $\mathcal{C}$ restricted to its i-th output and the (at most) d inputs that influence the i-th output. Let $\mathbf{c}_i \in \{0,1\}^{2^d}$ denote the outputs of $\mathcal{C}_i$ on all possible inputs $j \in [2^d]$ (identifying $[2^d]$ with $\{0,1\}^d$). Observe that $q_i := \Pr_{j\xleftarrow{\$}\{0,1\}^d}[\mathcal{C}_i(j) = 1] = \mathsf{HW}(\mathbf{c}_i)/2^d$. Hence, two cases can occur:

- Either there exists $i \le |S|$ such that $\mathsf{HW}(\mathbf{c}_i) \neq 0$. Then $q_i \ge 1/2^d = \Omega(1)$, hence we have $\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha})) \ge \Omega(1)$ for every x.
- Or $\mathsf{HW}(\mathbf{c}_i) = 0$ for all $i \le |S|$, implying that $\mathcal{C}$ is the all-zero circuit. In this case, we get $\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha})) \ge 1/p(n)$ by the p-non-degeneracy of $\mathcal{N}$.

The last item above provides a very weak bound. However, in general, we cannot hope for better, as a p-non-degenerate distribution $\mathcal{N}$ could have a subset S of $s(n)/2$ outputs that are all simultanously equal to 1 with probability $1/p(n)$ (provides that $p(n)$ is small enough) without violating any of the conditions. However, if $\mathcal{N}$ is instead *strongly* non-degenerate, we can replace the last item with

- $\mathsf{HW}(\mathbf{c}_i) = 0$ for all $i \le |S|$, implying that $\mathcal{C}$ is the all-zero circuit. In this case, denoting $\mathcal{C}^*(x)|_S$ the restriction of $\mathcal{C}^*(x)$ to S, $H(\mathcal{C}^*(x)|_S) - H(\mathcal{C}(\mathcal{U}_{s(n)^\alpha})) = H(\mathcal{C}^*(x)|_S) \ge w(n)/3$ by the strong non-degeneracy of $\mathcal{N}$. Therefore, by the Fannes-Audenaert inequality, using $\varepsilon = w(n)/(4s(n))$ (hence $1+\varepsilon\cdot s(n) < w(n)/3$ since $w(n) > 12$), we get $\Delta(\mathcal{C}^*(x), \mathcal{C}(\mathcal{U}_{s(n)^\alpha})) \ge w(n)/(3s(n))$,

which concludes the proof of Lemma 4.5

Discussion. We briefly comment on a few aspects of our result. Viola [Vio10] has shown that no local circuit can sample from a distribution statistically close to uniform over the set of w-sparse strings. Our result is weaker in that Viola's result applies also to circuits that can use *more* than $H(\mathcal{N})$ bits of randomness, but stronger in that it applies to a much larger class of distributions $\mathcal{N}$ (and in particular to Bernoulli noise; proving the equivalent of Viola's result for Bernoulli noise is a long standing open problem).

4.4 Structured-seed local PRGs

In this section, we introduce the notion of structured-seed local pseudorandom generator. 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 4.7 (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:*

- $\mathsf{Setup}(1^n)$. *A probabilistic algorithm that on inputs 1^n 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* $\mathcal{A} = (\mathcal{A}_n)_{n\in\mathbb{N}}$ *of size at most* $T = T(n)$*, for all* $n \in \mathbb{N}$*,*

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

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

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

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

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

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* $\mathsf{T}_{\mathsf{ss}} = \mathsf{T}_{\mathsf{ss}}(\lambda, n, m)$ *the (worst-case) running time of* SampleSeed *(implemented by a uniformly efficiently generatable family of boolean circuits), we say that a structured-seed PRG has* strong polynomial stretch *if* $m(n) = \mathsf{T}_{\mathsf{ss}}^{1+\Omega(1)}$*.*

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 4.8 (Non-Uniform Structured-Seed Local PRG). *We say that a triple* (Setup, SampleSeed, PRG) *is a* non-uniform *structured-seed local pseudorandom generator if it satisfies the conditions of Definition 4.7, except that* $\mathsf{Setup}(1^n)$ *is not required to run in polynomial time.*

5 Noisy-NC^0 Weak PRGs from the Sparse-LPN Assumption

5.1 The sparse-LPN assumption

We refer the reader to Section 3 for preliminaries on the LPN assumption and standard noise distributions. 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* $\mathcal{M}^c = \mathcal{M}^c_{m,n}$ *the distribution over* $\mathbb{F}_2^{m\times n}$ *that samples independently each row* $\mathbf{r}$ *of* A *as* $\mathbf{r}^\intercal \xleftarrow{\$} \mathcal{S}_{c,n}$*; that is,* A *is a uniformly random matrix with row-weight* c *over* $\mathbb{F}_2^{m\times n}$*. For a prime-order field* $\mathbb{F}$*, if the distribution is over* $\mathbb{F}$*, we denote as* $\mathcal{M}^c_{\mathbb{F}}$*.*

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

It is not hard to see that δ cannot be negligible for sparse-LPN: with probability at least m^2/n^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 $(\mathcal{M}^c, \mathcal{S}_w)$-$\mathsf{LPN}(n, m)$ is $(\mathsf{poly}(n), \mathsf{negl}(n), \delta)$-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 studied 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.

5.2 Search to decision

The work of [ABW10] proved that the search $(\mathcal{M}^c, \mathcal{S}_w)$-$\mathsf{LPN}$ problems implies the decisional hardness of sparse LPN. More precisely, [ABW10] shows that a if decision sparse-LPN with dimension n and m samples is not (T, ε)-secure[9], then there exists a $\mathsf{poly}(T)$-time adversary solving search sparse-LPN with dimension n and $m' = \tilde{\Theta}(m^3/\varepsilon^2)$ samples (with the same noise rate). It follows that all our results can be based on the search hardness of sparse LPN, at the cost of a blowup in the number of samples (note that the blowup depends on ε, hence the reduction is only meaningful when ε is sufficiently large).

5.3 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 5.1 (Bias of a Distribution). *Given a distribution $\mathcal{D}$ over $\mathbb{F}_2^m$ and a nonzero vector $\mathbf{u} \in \mathbb{F}_2^m$, 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 $\mathbf{u}$.

[9] We say that an LPN instance is (T, ε)-secure if it is $(T, \varepsilon - \delta, \delta)$-secure for some $\delta < \varepsilon$.

Definition 5.2 (Security against Linear Test). *For any integer $n \in \mathbb{N}$, let $m = m(n)$ be a polynomial, and let $\mathcal{M} = \mathcal{M}_{m,n}$ be a distribution over $\mathbb{F}_2^{m \times n}$. Let $\mathcal{E} = \mathcal{E}_m$ be a noise distribution over $\mathbb{F}_2^m$. Then, we say that the $(\mathcal{M}, \mathcal{E})$-LPN problem is* (ε, δ)-secure against linear tests *if for any (possibly inefficient) adversary $\mathcal{A}$ which, on input a matrix $A \in \mathbb{F}_2^{m \times n}$, outputs a nonzero $\mathbf{v} \in \mathbb{F}_2^m$, it holds that*

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

where $\mathcal{D}^A$ denotes the distribution induced by sampling $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^m$, $\mathbf{e} \leftarrow \mathcal{E}_m$, 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 5.3. *Let $\mathcal{M} = \mathcal{M}_{m,n}$ be a distribution over $\mathbb{F}_2^{m \times n}$ and $\mathcal{E}_m$ denote a noise distribution over $\mathbb{F}_2^m$. Then for any $d \in \mathbb{N}$, the $(\mathcal{M}, \mathcal{E})$-LPN problem with dimension n and $m = m(n)$ samples is $(\varepsilon_d, \delta_d)$-secure against linear tests, where*

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

The quantity ε_d in Lemma 5.3 depends solely on the noise distribution and can be computed easily for standard types of noise:

Lemma 5.4. *For any integer d and noise distribution $\mathcal{E}_m$, we denote $\varepsilon_d(\mathcal{E}_m) = \max_{\mathsf{HW}(\mathbf{v}) > d} \mathsf{bias}_{\mathbf{v}}(\mathcal{E}_m)$. Then*

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

The proof of the claim is a straightforward application of the piling-up lemma (Lemma 3.5): for $\mathcal{X}_{m,w}$ or $\mathcal{R}_{m,w}$, the distribution induced by $\langle \mathbf{v}, \mathbf{e} \rangle$ is a xor of w independent Bernoulli samples, each equal to 1 with probability $\mathsf{HW}(\mathbf{v})/m < d/m$. For $\mathcal{B}_{m,w}$, $\langle \mathbf{v}, \mathbf{e} \rangle$ is a xor of $\mathsf{HW}(\mathbf{v}) > d$ Bernoulli sample of rate w/m. The rightmost side of the inequalities follows from the standard inequality $(1 - 1/m)^N \leq \exp(-1)$. We note that a similar bound can be shown (with a slightly more tedious analysis) for $\mathcal{S}_{m,w}$.

5.4 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 5.5 (Most sparse matrices have large dual distance). *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any large enough n, there is a constant $\gamma(c)$ such that for any $m \leq n^{(1-\eta)c/2+\eta}$,*

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

Lemma 5.6 (Large dual distance over prime-order field). *For any constants $c \geq 3$, $\eta \in (0,1)$, for any large enough $m = m(\lambda)$, there is a constant ϵ such that $|\mathbb{F}|^\varepsilon > m$, and for any $m < n^{(1-\eta)(c-2)+1}$,*

$$\Pr\left[A \xleftarrow{\$} \mathcal{M}^c_{\mathbb{F}} \ : \ \mathsf{dd}(A) \geq n^\eta\right] \geq 1 - |\mathbb{F}|^{-1} \sum_{t=1}^{r} \left(\frac{m}{|\mathbb{F}|^\varepsilon}\right)^t$$

For example, setting $c = 3$ and $\eta = 1/5$, Theorem 5.5 yields that for $m = n^{1.4}$, a random c-sparse matrix A over $\mathbb{F}_2^{m \times n}$ has dual distance $\mathsf{dd}(A) = \Omega(n^{0.2})$ with probability at least $1 - O(n^{-0.2})$. For completeness, we provide a proof of Theorem 5.5 in Appendix A.

5.5 A parametrized version of the sparse-LPN assumption

Combining Lemma 5.4 (bounding ε_d) and Theorem 5.5 (to bound δ_d) yields a quantified estimate of the security of sparse-LPN against attacks from the linear test framework:

Lemma 5.7. *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{m,w}, \mathcal{R}_{m,w}, \mathcal{B}_{m,w}\}$, the $(\mathcal{M}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension n, noise $w = w(n)$, and $m = m(n) \leq n^{(1-\eta)c/2+\eta}$ samples is (ε, δ)-secure against linear tests, where*

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

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 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 $r = w/m$, 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). 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 $\mathcal{E} \in \{\mathcal{X}_{m,w}, \mathcal{R}_{m,w}, \mathcal{B}_{m,w}\}$, for every $\lambda = n^\alpha$ (for some constant $\alpha < 1$), $T = 2^{o(\lambda)}$, the $(\mathcal{M}^c, \mathcal{E})$-$\mathsf{LPN}$ problem with dimension n, number of samples $m = m(n) \leq n^{(1-\eta)c/2+\eta}$, and noise $w = w(n) \geq \lambda \cdot n^{(1-\eta)c/2}$ is (T, ε, δ)-secure, with*

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

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

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

Alternatively, under a different choice of distribution for ther matrix A, that we denote $\mathcal{M}_s^c$, sparse LPN is plausibly (ϵ, δ)-secure with a *negligible* δ. The distribution samples matrices with large dual distance with overwhelming probability (hence ensuring security against linear tests), which are defined via the expander graph in Theorem 8.2 of [AK19].

Lemma 5.8. *For any constants $c \geq 3$ and $\eta \in (0,1)$, $n = n(\lambda)$ and $m = m(n) \leq n^{(1-\eta)(c-1)/4}$, with the failure probability of $1 - n^{-\omega(1)}$, there exists a distribution over matrices $\mathcal{M}_s^c$ over $\mathbb{F}_2^{m \times n}$ with row-weight c such that*

$$\Pr\left[A \xleftarrow{\$} \mathcal{M}_s^c : \mathsf{dd}(A) > \Omega(n^{\eta})\right] > 1 - n^{-\omega(1)}.$$

Assumption 2 *For any constants $c \geq 3$ and $\eta \in (0,1)$, for any noise distribution $\mathcal{E} \in \{\mathcal{X}_{m,w}, \mathcal{R}_{m,w}, \mathcal{B}_{m,w}\}$, for every $\alpha < 1$ and $T = 2^{o(n^{\alpha})}$, the $(\mathcal{M}_s^c, \mathcal{E})$-LPN problem with dimension n, number of samples $m = m(n) \leq n^{(1-\eta)(c-1)/4}$, and noise $w = w(n) \geq \lambda \cdot n^{(1-\eta)c/2}$ is (T, ε, δ)-secure, with*

$$\varepsilon = \frac{\mathsf{poly}(T)}{2^{\Omega(n^{\alpha})}}, \qquad \delta = n^{-\omega(1)}.$$

5.6 Weak noisy local PRGs from sparse-LPN

In this section, we exhibit a straightforward construction of weak local PRG in $\mathcal{N}$-NC^0 from the $(\mathcal{M}^c, \mathcal{N}, \mathbb{F}_2)$-$\mathsf{LPN}(n, m)$ assumption.

Fig. 3: Weak noisy PRGs from sparse-LPN

Parameters. For Parameter Sets 1 or 2
Construction.

$\mathsf{Setup}(1^n)$: Sample $A \xleftarrow{\$} \mathcal{M}_{m,n}^c$. Output $\mathsf{pp} = A$.

$\mathsf{PRG}(\mathsf{pp}, \mathsf{seed})$: On input $\mathsf{seed} \xleftarrow{\$} \{0,1\}^n$, sample $e \xleftarrow{\$} \mathcal{E}$, output $y = \mathsf{PRG}(\mathsf{pp}, \mathsf{seed}; e) = A \cdot \mathsf{seed} + e \in \{0,1\}^m$

Before the proof, we provide some intuition: why the construction "should be" pseudorandom. Sparse-LPN samples already look random. The sparse-LPN assumption says that, for a sparse matrix A, the vector $A \cdot \mathsf{seed} + e$ (where seed is a uniform seed and e is a low-weight noise vector) is computationally indistinguishable from a uniform n-bit string. Our PRG does nothing more than output the LPN samples. It takes a fresh seed seed and fresh noise e, computes $A \cdot \mathsf{seed} + e$ and publishes it. So, as long as the sparse-LPN assumption holds, its output should look random to any efficient observer. The proof just formalizes that intuition with a reduction.

Proof. Define two distributions on pairs (A, y):

$$\mathcal{D}_0 : A \xleftarrow{\$} \mathcal{M}^c_{m,n}, \quad \mathsf{seed} \xleftarrow{\$} \{0,1\}^n, \quad r \xleftarrow{\$} \mathcal{E}, \quad y = A \cdot \mathsf{seed} + r;$$
$$\mathcal{D}_1 : A \xleftarrow{\$} \mathcal{M}^c_{m,n}, \quad y \xleftarrow{\$} \{0,1\}^m.$$

Let $\mathcal{A}$ be any adversary that distinguishes $\mathcal{D}_0$ from $\mathcal{D}_1$. We build an LPN distinguisher $\mathcal{B}$:

1. **Input**. Receive (A, y) from the LPN challenger. Here y is either $A \cdot \mathsf{seed} + e$ for random seed, e, or a uniform string in $\{0,1\}^m$.
2. **Call** $\mathcal{A}$. Give (A, y) to $\mathcal{A}$; let $\mathcal{A}$ output a bit.
3. **Output**. Output the same bit as $\mathcal{A}$'s guess.

Correctness.

- If $y = A \cdot \mathsf{seed} + e$ (the *LPN sample* case), then $(A, y) \sim \mathcal{D}_0$; $\mathcal{A}$ faces the true PRG distribution. This is exactly what our PRG would have output on seed seed and noise e.
- If y is uniform (the *random* case), then $(A, y) \sim \mathcal{D}_1$; $\mathcal{A}$ faces the uniform distribution.

Hence

$$\mathsf{Adv}_{\mathcal{B}} = \left|\Pr[\mathcal{B} \text{ outputs } 1 \,|\, \text{LPN}] - \Pr[\mathcal{B} \text{ outputs } 1 \,|\, \text{random}]\right| = \mathsf{Adv}_{\mathcal{A}}.$$

If $\mathcal{A}$ had distinguishing advantage $> \varepsilon$, then $\mathcal{B}$ would break the sparse-LPN assumption with the *same* advantage, contradicting (T, ε, δ)-hardness except with probability δ.

Strong noisy PRGs from a sparse-LPN variant. Alternatively, replacing the distribution $\mathcal{M}^c$ with the distribution $\mathcal{M}^c_s$ from Lemma 5.8 in the above construction immediately yields a *strong* noisy PRG (with distinguishing advantage bounded by $n^{-\omega(1)}$), at the cost of assuming sparse LPN with respect to a non-standard matrix distribution (Assumption 2).

6 Local Compression Schemes

In this section, we introduce *local compression scheme*. Informally, a local compression scheme for a set $\mathcal{S}$ allows compressing the representation of any element x of $\mathcal{S}$ such that given the compressed representation $\mathbf{c}_x$, x can be reconstructed by a constant-locality algorithm.

Definition 6.1 (Local compression). *Let $\mathcal{S} = \{\mathcal{S}_{s(n)}\}_{n\in\mathbb{N}}$ denote a family of sets $\mathcal{S}_s \subseteq \{0,1\}^s$, and let d be an integer. An $\ell(\cdot)$-*local $\mathsf{cf}(\cdot)$*-compression scheme for $\mathcal{S}$ is a 3-tuple* $(\mathsf{Param}, \mathsf{Comp}, \mathsf{Rec})$ *of efficient algorithms with the following interface:*

- $\mathsf{Param}(\mathsf{gp})$*: on input global parameters* gp *(assumed to include* 1^s*), returns public parameters* pp *(assumed to include* gp*).*
- $\mathsf{Comp}(\mathsf{pp}, x, d)$*: on input public parameters* pp*, $x \in \mathcal{S}_s$, and a compression parameter $d \in \mathbb{N}$, deterministically output $\mathbf{c}_x$ (a compressed representation of x).*
- $\mathsf{Rec}(\mathsf{pp}, \mathbf{c}_x, d)$*: on input public parameters* pp*, a compressed representation $\mathbf{c}_x$, and a compression parameter d, deterministically output $x' \in \{0,1\}^s$.*

An ℓ-local compression scheme for $\mathcal{S}$ must be correct*,* compressing*, and* local*:*

- ***Correctness:*** *with overwhelming probability* $1 - \mathsf{negl}(s)$ *over the sampling of* $\mathsf{pp} \xleftarrow{\$} \mathsf{Param}(\mathsf{gp})$*, the following holds: for every $x \in \mathcal{S}_s$ and every $d \in \mathbb{N}$,*

$$\mathsf{Rec}(\mathsf{pp}, \mathsf{Comp}(\mathsf{pp}, x, d), d) = x.$$

- $\mathsf{cf}(\cdot)$-***compression:*** *For all $d \in \mathbb{N}$, all large enough s, every* pp *in the support of* $\mathsf{Param}(\mathsf{gp})$*, and every $x \in \mathcal{S}_s$, denoting $\mathbf{c}_x \leftarrow \mathsf{Comp}(\mathsf{pp}, x, d)$, it holds that $|\mathbf{c}_x| \leq s^{\mathsf{cf}(d)}$. We call $\mathsf{cf}(d)$ the* compression factor*.*
- $\ell(\cdot)$-***local:*** *for every* pp *in the support of* $\mathsf{Param}(\mathsf{gp})$ *and every $d \in \mathbb{N}$, the algorithm* $\mathsf{Rec}(\mathsf{pp}, \cdot, d)$ *is in* $\mathsf{NC}^0_{\ell(d)}$*. We call $\ell(d)$ the* locality parameter*.*

Of course, local compression schemes can only exist for sets that admit efficient compression in the first place. We will typically be interested in local compression schemes for sets that come with a canonical "compressed representation" of the elements $x \in \mathcal{S}_s$, denoted $\mathsf{rep}(x)$, and we will allow Comp to work with the compressed representation of x directly. We denote Comp^* the modified algorithm. While this is without loss of generality – as $\mathsf{Comp}^*(\mathsf{pp}, \mathsf{rep}(x), d)$ could always recompute x from $\mathsf{rep}(x)$ internally – this becomes non-trivial when we require Comp^* to run in time *smaller than* s. Concretely, given a family of sets $\mathcal{S} = \{\mathcal{S}_{s(n)}\}_{n\in\mathbb{N}}$ equipped with a natural compressed representation $\mathsf{rep} : \{0,1\}^* \to \{0,1\}^*$ (where $|\mathsf{rep}(x)| < |x|$ for any large enough x), we say that a local compression scheme $(\mathsf{Param}, \mathsf{Comp}^*, \mathsf{Rec})$ for $(\mathcal{S}, \mathsf{rep})$ is *efficient* if for every large enough n, parameters pp in the support of $\mathsf{Param}(\mathsf{gp})$, every integer d, and every $x \in \{0,1\}^{s(n)}$, the function $\mathsf{rep}(x) \mapsto \mathsf{Comp}^*(\mathsf{pp}, \mathsf{rep}(x), d)$ runs in time $o(s(n))$.

6.1 Compressing unit Vectors

We introduce a simple method to compress a length-s unit vector x into d smaller unit vectors $(x_1, \cdots, x_d)$ of length $s^{1/d}$ such that each entry of x can be reconstructed by retrieving a single entry from each x_i. Let $\mathsf{u}_s(i)$ denote the procedure which, on input $i \in [s]$, outputs a length-s one-hot $\mathbb{F}_2$-vector with a 1 at position i, and on input $\perp$, outputs the all-zero vector 0^s. Conversely, let $\mathsf{nzi}(x)$ denote the procedure which, given as input a length-s one-hot $\mathbb{F}_2$-vector x, returns its non-zero index i, or $\perp$ if $x = 0^s$.

Algorithm 1 $(\mathsf{Param}_\mathsf{u}, \mathsf{Comp}_\mathsf{u}, \mathsf{Rec}_\mathsf{u})$:

$\mathsf{Param}_\mathsf{u}(1^s)$: outputs $\mathsf{pp} \leftarrow 1^s$.
$\mathsf{Comp}_\mathsf{u}(\mathsf{pp}, x, d)$: on input pp, $x \in \{0,1\}^s$, and $d \in \mathbb{N}$,
1. Set $s_d \leftarrow \lceil s^{1/d} \rceil$.
2. Compute $i \leftarrow \mathsf{nzi}(x)$ and write i over the s_d-ary basis as $(i_1, \cdots, i_d) \in [s_d]^d$. If $i = \perp$, set $i_1 = \cdots = i_d = \perp$.
3. Output $\mathbf{c}_x = (x_1, \cdots, x_d) \leftarrow (\mathsf{u}_{s_d}(i_1), \cdots, \mathsf{u}_{s_d}(i_d))$.

$\mathsf{Rec}_\mathsf{u}(\mathsf{pp}, \mathbf{c}_x, d)$: parse $\mathbf{c}_x$ as $(x_1, \cdots, x_d)$ with $x_i \in \{0,1\}^{s_d}$ for $i = 1$ to d.
1. For $j = 1$ to s, write j over the s_d-ary basis as $(j_1, \cdots, j_d) \in [s_d]^d$.
2. Return $x' \leftarrow (\prod_{k=1}^d x_k[j_k])_{j \leq s}$.

For $j \leq s$, we write $\mathsf{Rec}_\mathsf{u}(\mathsf{pp}, \mathbf{c}_x, d, j)$ to denote the projection of $\mathsf{Rec}_\mathsf{u}(\mathsf{pp}, \mathbf{c}_x, d)$ to the j-th coordinate of its output; that is, $\mathsf{Rec}_\mathsf{u}(\mathsf{pp}, \mathbf{c}_x, d, j)$ returns $x'[j] = \prod_{k=1}^d x_k[j_k]$.

Lemma 6.2. *For every $s \in \mathbb{N}$, let $\mathbb{U}_s \subseteq \{0,1\}^s$ denote the set of all strings $x \in \{0,1\}^s$ with $\mathsf{HW}(x) \leq 1$. The scheme $(\mathsf{Param}_\mathsf{u}, \mathsf{Comp}_\mathsf{u}, \mathsf{Rec}_\mathsf{u})$ is an $\ell(\cdot)$-local $\mathsf{cf}(\cdot)$-compression scheme for $\mathbb{U}$ with $\ell(d) = d$ and $\mathsf{cf}(d) \leq 1.01/d$.*

We prove correctness, compression, locality, and efficiency in turn.

Correctness. For every $x \in \{0,1\}^s$ with $\mathsf{HW}(x) \leq 1$, there are two cases:

- $\mathsf{HW}(x) = 1$. In this case, the value $\prod_{k=1}^d x_k[i_k]$ evaluates to 1 (has x_k has a 1 at position i_k). Then, for $(j_1, \cdots, j_d) \neq (i_1, \cdots, i_d)$, there exists k such that $j_k \neq i_k$, hence $x_k[j_k] = 0$ and the product $\prod_{k=1}^d x_k[j_k]$ evaluates to 0.
- $\mathsf{HW}(x) = 0$. In this case, all the x_k's are zero vectors and therefore x' reconstructs to 0^s.

Compression. The size of $\mathbf{c}_x$ is $d \cdot \lceil s^{1/d} \rceil$, which is bounded above by $s^{1.01/d}$ for a large enough s.

Locality. By construction, every output bit $x'[j]$ of $\mathsf{Rec}(\mathsf{pp}, \mathbf{c}_x, d)$ is computed as $\prod_{k=1}^{d} x_k[j_k])_{j \le s}$, which requires reading exactly one bit from each x_k, hence d bits from $\mathbf{c}_x$.

Efficiency. Eventually, any $x \in \{0,1\}^s$ with $\mathsf{HW}(x) \le 1$ can be succinctly represented as $\mathsf{nzi}(x) \in [s] \times \{\perp\}$. It is straightforward to see that $\mathsf{Comp}_{\mathsf{u}}(\mathsf{pp}, x, d)$ can alternatively be passed $i = \mathsf{nzi}(x)$ as input directly (since it computes $i \leftarrow \mathsf{nzi}(x)$ in step 2 and never reuses x afterwards). The runtime of $\mathsf{Comp}^*_{\mathsf{u}}(\mathsf{pp}, \mathsf{nzi}(x), d)$ is $O(d \cdot s^{1/d})$.

6.2 Compressing regular vectors

Regular vectors. Fix integers s, w with $w \mid s$ and let $\ell = s/w$. A vector

$$\mathbf{x} = \mathbf{x}^{(1)} \parallel \cdots \parallel \mathbf{x}^{(w)} \in \{0,1\}^s$$

is (w, s)-*regular* if every block $\mathbf{x}^{(\alpha)} \in \{0,1\}^\ell$ has Hamming weight at most 1. Equivalently, $\mathbf{x}$ is the concatenation of w unit vectors of length ℓ. The regular-noise distribution $R_{s,w}$ (see the paragraph on *types of noise* under Definition 3.2) is uniform over the set $\mathcal{R}_{s,w}$ of (w, s)-regular vectors.

We now give a local compression scheme for the set $\mathcal{R}_{s,w} \subseteq \{0,1\}^s$. Intuitively, each block is a *unit vector of length* ℓ, so we can reuse the unit-vector compressor of Section 6.1.

Algorithm 2 $(\mathsf{Param}_{\mathsf{reg}}, \mathsf{Comp}_{\mathsf{reg}}, \mathsf{Rec}_{\mathsf{reg}})$:

Let $d \in \mathbb{N}$ and set $\ell_d := \lceil \ell^{1/d} \rceil$.

$\mathsf{Param}_{\mathsf{reg}}(1^s, 1^w)$: Output $\mathsf{pp}_{\mathsf{reg}} = (s, w)$.

$\mathsf{Comp}_{\mathsf{reg}}(\mathsf{pp}_{\mathsf{reg}}, \mathbf{x}, d)$:

1. Split $\mathbf{x}$ into blocks $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(w)} \in \{0,1\}^\ell$.
2. For every $\alpha \in [w]$ compute $\mathbf{c}_\alpha := \mathsf{Comp}_{\mathsf{u}}(1^\ell, \mathbf{x}^{(\alpha)}, d)$ where $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$ is the unit-vector compressor of Section 6.1.
3. Output $\mathbf{c} := (\mathbf{c}_1, \ldots, \mathbf{c}_w)$.

$\mathsf{Rec}_{\mathsf{reg}}(\mathsf{pp}_{\mathsf{reg}}, \mathbf{c}, d)$:

1. Parse $\mathbf{c}$ as $(\mathbf{c}_1, \ldots, \mathbf{c}_w)$ and set $\mathbf{x}^{(\alpha)} := \mathsf{Rec}_{\mathsf{u}}(1^\ell, \mathbf{c}_\alpha, d)$ for every $\alpha \in [w]$.
2. Return $\mathbf{x}' := \mathbf{x}^{(1)} \parallel \ldots \parallel \mathbf{x}^{(w)}$.

Lemma 6.3. *For every $s \in \mathbb{N}$, block count w (where $w = s^\alpha$) for some $\alpha < 1$ and compression parameter $d \ge 1$, the triple $(\mathsf{Param}_{\mathsf{reg}}, \mathsf{Comp}_{\mathsf{reg}}, \mathsf{Rec}_{\mathsf{reg}})$ is an $\ell_{\mathsf{reg}}(d) = d$-local compression scheme for the family $\mathcal{R}_{s,w}$ of (w, s)-regular vectors with compression factor*

$$\mathsf{cf}_{\mathsf{reg}}(d) \le \alpha + \frac{1-\alpha}{d}$$

Moreover, $\mathsf{Comp}_{\mathsf{reg}}$ runs in $O\big(wd\ell^{1/d}\big)$.

Proof. Correctness. Each block is independently compressed and recovered by the unit-vector scheme, so concatenation reconstructs $\mathbf{x}$ exactly.

Compression. A single block of length ℓ is turned into $d\ell_d$, so the total compressed length is:

$$|\mathbf{c}| = wd\lceil \ell^{1/d} \rceil = wd(s/w)^{1/d} = w^{1-1/d} s^{1/d} = (s^\alpha)^{1-1/d} s^{1/d} = d s^{\alpha + \frac{1-\alpha}{d}}$$

which gives us the stated bound for sufficiently large s.

Locality. To output bit j determine its block $\alpha = \lceil j/\ell \rceil$ and offset inside the block. The call to $\mathsf{Rec}_{\mathsf{u}}$ reads exactly d bits of $\mathbf{c}_\alpha$, so locality is d. Write[10]

$$\mathbf{c} = \big(\underbrace{\mathbf{c}_1}_{d \text{ subvectors}}, \ldots, \underbrace{\mathbf{c}_w}_{d \text{ subvectors}} \big), \qquad \mathbf{c}_\alpha = \big(c_{\alpha,1}, \ldots, c_{\alpha,d}\big), \quad c_{\alpha,k} \in \{0,1\}^{\ell_d}.$$

Hence $\mathbf{c}$ is divided into w *blocks*, each block containing d small unit vectors of length ℓ_d.

Given $j \in [s]$:

1. Determine its block index $\alpha = \left\lceil \frac{j}{\ell} \right\rceil \in [w]$ and the within-block offset $u = j - (\alpha - 1)\ell \in [\ell]$.
2. Write u in base ℓ_d: $u \mapsto (u_1, \ldots, u_d) \in [\ell_d]^d$.
3. Access exactly the following d bits of $\mathbf{c}_\alpha$: $c_{\alpha,1}[u_1], c_{\alpha,2}[u_2], \ldots, c_{\alpha,d}[u_d]$.
4. Output $x'[j] = \prod_{k=1}^{d} c_{\alpha,k}[u_k]$.

Thus $x'[j]$ depends on *precisely* d coordinates of the compressed representation $\mathbf{c}$, and on no others. Consequently $\ell_{\mathsf{reg}}(d) = d$.

Efficiency. Reading $\mathbf{x}$ in full costs $O(s)$, after which $\mathsf{Comp}_{\mathsf{u}}$ is invoked w times, each in $O\big(d\,\ell^{1/d}\big)$ time. If instead the encoder is handed the at-most-one index per block, the read is not relevant anymore and the total time is $O\big(d\,w\,\ell^{1/d}\big) = o(s)$ for any fixed d and any $w = o(s)$.

6.3 Compressing sparse vectors

In this section, we describe a local compression scheme that compresses arbitrary sparse vectors.

Combinatorial batch codes. At the heart of our construction is the existence of a negligible-error procedure, introduced in [AK19], for sampling *combinatorial batch codes* [IKOS04]. Previous works [BCGI18, SGRR19, BBC+24] also used batch codes to reduce the computational overhead stemming from the use of random sparse noise when distributively generating LPN samples; however, our use of batch codes for *compressing* sparse vectors (with constant locality) is new.

Definition 6.4 (Batch Codes [IKOS04]). *Let w, s, T, N be integers such that $w \le T < s$, $N = O(s)$. An (s, N, w, T)-batch code encodes any string $x \in \{0,1\}^s$ into a T-tuple of strings $(z_1, \ldots, z_T) \in \{0,1\}^*$ (called buckets) of total length N, such that any w-tuple of coordinates of x can be recovered by reading at most a single entry from each bucket.*

A batch code is *combinatorial* if the bits of the z_i's are exactly bits of x. That is, the encoding procedure assigns each bit of x to some number of buckets. A combinatorial batch code (CBC) is *t-uniform* if each bit of x is assigned to at most t buckets.

Equivalently, an (s, N, w, T)-CBC can be viewed as a bipartite graph with s nodes on the top (the input) and T nodes on the bottom (the buckets) such that for every w-tuple of input nodes, there always exists a selection of a single edge for each node in the tuple such that each bucket is connected to at most one selected node (intuitively, the selected edges indicate which bit of the input to read from each bucket to guarantee that all w bits will be recovered by reading a single bit per bucket). Eventually, if the CBC is t-uniform, then each input node has degree at most t. We represent on Figure 4 a 2-uniform bipartite graph (each input is placed is at most two buckets). Given the subset $\{2, 4, 9\}$, the edges represented in red indicate a "collision-free" selection that guarantees that (x_2, x_4, x_9) can be recovered by reading one entry per bucket.

We will rely on the following lemma:

[10] Recall $\ell_d = \lceil \ell^{1/d} \rceil$ and each call to $\mathsf{Comp}_{\mathsf{u}}$ produces d length-ℓ_d unit vectors.

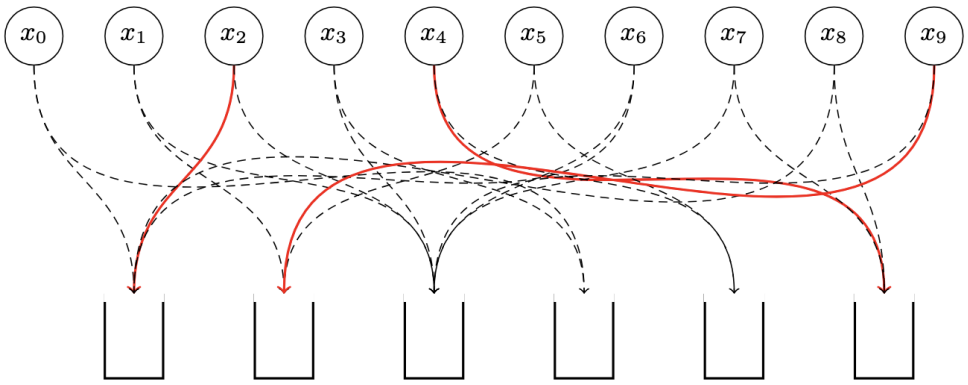

Fig. 4. A 2-uniform bipartite graph with 10 information nodes and 6 buckets. The edges marked in red allow retrieving the information bits with indices in the subset $\{2, 4, 9\}$ by reading at most a single entry from each bucket.

Lemma 6.5 (CBC with negligible-error [AK19]). *For every constant $c > 1$ and integer $t > 1 + c$, there exists a negligible-error construction of an (s, N, w, T)-CBC with information words of length s, codeword of length $N = 2ts$, $T = s^{1/c}$ buckets and batch-size of $w = \Omega(T^{1-\frac{c-1}{t-2}})$ where the constant in the Ω depends on t. Moreover, the code is $2t$-uniform.*

Efficient bucket selection for CBCs. Lemma 6.5 only asserts that for every $\Omega(T^{1-\frac{c-1}{t-2}})$-subset S of the input bits, there exists a decoder that reads at most one entry per bucket and recovers $(x_i)_{i\in S}$. However, finding the decoding procedure amounts to solving the following problem: consider the oriented subgraph induced by removing all input nodes $(x_i)_{i\notin S}$ and all buckets with degree 0, and orienting the edges towards the buckets. This is a bipartite graph with $w = \Omega(T^{1-\frac{c-1}{t-2}})$ input nodes, and at most $2tw$ bucket nodes (by $2t$-uniformity of the code). Then, one must find a maximum matching in the graph – that is, a subset of the edges where no two edges share an endpoint. Due to recent sequence of breakthrough results culminating with [CKL+22], this problem can be solved in almost-linear time $w^{1+o(1)}$ (in a RAM model with word size large enough to store node identifiers).

An (s, N, w, T)-CBC is typically described as a pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$ such that Enc encodes an information word into T strings, and Dec decodes a subset of the bits of the word from the encoding. However, our construction is easier described with a slightly different formalism that nevertheless captures all properties of a CBC outlined above. Concretely, we introduce the following algorithms:

- $\mathsf{SampleCode}(s, c, t)$, as a randomized algorithm, outputs the description of an oriented bipartite graphs with s nodes at the top (indexed from 1 to s) of outdegree at most $2t$ and $T = s^{1/c}$ nodes at the bottom. We view the graph as a function C that, on input $i \in [s]$, output the set S_i of size $|S_i| \neq 2t$ of the neighbors of i; the description size of C is at most $|C| \leq s \cdot 2t\lceil \log T \rceil$.
- $\mathsf{SelectBuckets}(S, C)$, given a subset $S \subset [s]$ of size $w = \Omega(T^{1-\frac{c-1}{d-2}})$ and the description C of a CBC, computes a maximum matching for C, *i.e.*, a list $(B_1, \cdots, B_w) \in [T]^w$ of buckets. If a matching of size exactly w is not found, it outputs $\perp$; else, it outputs $(B_1, \cdots, B_w)$.

Then, we have the following immediate restatement of Lemma 6.5:

Lemma 6.6 (CBC with negligible error [AK19], reformulated). *There exists a pair of algorithms $(\mathsf{SampleCode}, \mathsf{SelectBuckets})$ as described above such that for every constant $c > 1$ and integer $t > 1+c$, with probability $1-\mathsf{negl}(s)$ over the random coins of $C \xleftarrow{\$} \mathsf{SampleCode}(s, c, t)$, it holds that for every $S \subset [s]$ of size $w = O(T^{1-\frac{c-1}{t-2}})$ with $T = s^{1/c}$, $\mathsf{SelectBuckets}(S, C)$ does not*

return $\perp$*, the selected buckets* $(B_1, \cdots, B_w) = \mathsf{SelectBuckets}(S, C)$ *are all pairwise distinct and satisfy that each* B_i *belongs to* $C(i)$ *(that is,* B_i *is a neighbor of the input node* i*). Furthermore,* $\mathsf{SelectBuckets}$ *runs in time* $w^{1+o(1)}$*.*

A local compression scheme for sparse vectors. Equipped with the above lemma, we introduce a scheme for compressing sparse vectors. Fix a weight parameter $w = w(n)$, and let $\mathsf{Sp} = \{\mathsf{Sp}_{s,w}(\mathbb{F}_2)\}_{s\in\mathbb{N}}$ the set of vectors $x \in \mathbb{F}_2^s$ with $\mathsf{HW}(x) \leq w$; we generally omit w and $\mathbb{F}_2$ and write $\mathsf{Sp} = \{\mathsf{Sp}_s\}_{s\in\mathbb{N}}$ when they are clear from the context. We represent our local compression scheme on Algorithm 3.

Algorithm 3 $(\mathsf{Param}_{\mathsf{Sp}}, \mathsf{Comp}_{\mathsf{Sp}}, \mathsf{Rec}_{\mathsf{Sp}})$:

Fix a constant $c > 1$ and a parameter $n \in \mathbb{N}$. Set $\alpha \leftarrow 98/(100c)$, $s = s(n)$ and let the weight parameter w be $w(n) = n^\alpha$. Let $t = 100(c-1)+2$. Let $T \leftarrow s^{1/c}$. Note that $T^{1-\frac{c-1}{t-2}} = s^{99/(100c)} = \omega(w)$, hence for any constant $g(t)$, for a large enough s, $w(n) < g(t) \cdot T^{1-\frac{c-1}{t-2}}$. Set the global parameters $\mathsf{gp} = \mathsf{gp}_s$ to $(1^s, c, w(n), T(n))$.

$\mathsf{Param}_{\mathsf{Sp}}(\mathsf{gp})$**:** sample $C \xleftarrow{\$} \mathsf{SampleCode}(s, c, t)$ and set $\mathsf{pp} = (\mathsf{gp}, C)$.

$\mathsf{Comp}_{\mathsf{Sp}}(\mathsf{pp}, x, d)$**:** on input public parameters pp parsed as $\mathsf{pp} = S$, a vector $x \in \mathsf{Sp}_s = \mathsf{Sp}_{s,w}(\mathbb{F}_2)$, and a compression factor d,

- Let $w_x = \mathsf{HW}(x)$ Construct a set $H_x \subset [s]$ containing the $w_x \leq w$ indices of non-zero coordinates of x. Denote $(h_1, \cdots, h_{w_x})$ the lexicographic ordering of the elements of H_x.
- Run $(B_1, \cdots, B_{w_x}) \leftarrow \mathsf{SelectBuckets}(H_x, C)$.
- For $j = 1$ to T, if there exists $i \leq w_x$ such that $j = B_i$, set $v_j \leftarrow \mathsf{nzi}(h_i)$. Otherwise, set $v_j \leftarrow 0^m$.
- For $j = 1$ to T, set $\mathbf{c}_j \leftarrow \mathsf{Comp}_{\mathsf{u}}(1^s, v_j, d)$.
- Output $\mathbf{c}_x \leftarrow (\mathbf{c}_1, \cdots, \mathbf{c}_T)$.

$\mathsf{Rec}_{\mathsf{Sp}}(\mathsf{pp}, \mathbf{c}_x, d)$**:** On input pp, a compressed noise $\mathbf{c}_x$, and a compression factor d, parse $\mathbf{c}_x = (\mathbf{c}_1, \cdots, \mathbf{c}_T)$. Set

$$x' \leftarrow \left(\bigoplus_{j \in C(i)} \mathsf{Rec}_{\mathsf{u}}(\mathsf{pp}, \mathbf{c}_j, d, i) \right)_{i \leq s}$$

Overview. The high level intuition of the scheme is the following: given a sparse vector x, assume for simplicity that $\mathsf{HW}(x) = w$. By the property of the CBC, we can find using $\mathsf{SelectBuckets}$ an assignment of a unique bucket for each nonzero entry of x such that no two nonzero entries get assigned to the same bucket. Given such an assignment, we compress x as follows:

- For each position i of a 1 in x, assign $\mathsf{u}_m(i)$ to the bucket selected for i.
- Assign the all-zero vector to all empty buckets.
- In each bucket, compress the assigned vector (that is either a weight-one vector or 0^s) using the local compression scheme for unit vectors $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$ given in Section 6.1.
- Return the compressed representation of each bucket.

Observe that this is indeed compressing: for an appropriate choice of parameters, we ensure that the number T of buckets is only slightly larger than w, e.g. $T = w^{1.01}$. Furthermore, each bucket is compressed to less than $s^{1.01/d}$ bits via $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$, hence the output has total length $w^{1.01} \cdot s^{1.01/d}$, which is sublinear in s if $w \ll s^{1/1.01-1/d}$.

For reconstruction, observe that by the $2t$ uniformity of the scheme, each input bit at position i can only belong to $2t$ buckets (whose position is given by the graph of the CBC produced by $\mathsf{SampleCode}$). Reconstruction simply reconstruct the value at position i in each of these candidate buckets (using locality d by the d-locality of the compression scheme for unit vectors)

and XOR the outputs, obtaining $x[i]$ in constant locality $2td$.

Lemma 6.7. *For every $c > 1$, denoting $\alpha = 98/(100c)$, $w(n) = s^\alpha$, and $\mathsf{Sp} = \{\mathsf{Sp}_{s,w(n)}\}_{s\in\mathbb{N}}$, the scheme $(\mathsf{Param}_{\mathsf{Sp}}, \mathsf{Comp}_{\mathsf{Sp}}, \mathsf{Rec}_{\mathsf{Sp}})$ is an $\ell(\cdot)$-local $\mathsf{cf}(\cdot)$-compression scheme for $\mathcal{U}$ with $\ell(d) = 2(100(c-1)+2)d$ and $\mathsf{cf}(d) \le 1/c + 1.01/d$. Furthermore, when given as input a compressed representation of $x \in \mathsf{Sp}_{s,w(n)}$ as a set $H_x \subset [s]$ of the indices of the 1's in x (as a bitstring, $|H_x| = \mathsf{HW}(x) \cdot \log s$), the scheme is* strongly-efficient*: the function $H_x \to \mathsf{Comp}_{\mathsf{Sp}}(\mathsf{pp}, H_x, d)$ runs in time $|H_x|^{1+o(1)}$.*

We argue each of the properties in turn.

Correctness. By construction, each v_j is either 0^s or is a unit vector containing one of the nonzero entries of x, and no two nonzero v_j's are equal; equivalently, $x = \bigoplus_{j=1}^{T} v_j$ and $|\{j \ : \ v_j \neq 0^s\}| = \mathsf{HW}(x)$. The correctness of the CBC guarantees that with probability $1 - \mathsf{negl}(n)$ over the choice of $C \xleftarrow{\$} \mathsf{SampleCode}(s, c, t)$, such a decomposition of x into weight-≤ 1 vectors $(v_1, \cdots, v_T)$ exists for every $x \in \{0,1\}^s$ with $\mathsf{HW}(w) \le x$.

Furthermore, by construction of $\mathsf{SelectBuckets}$ (that computes a maximum matching in the bipartite graph defined by C), for every $x \in \{0,1\}^s$ of weight at most w with non-zero indices $(h_1, \cdots, h_{s_x})$, it holds for all $i \le \mathsf{HW}(x)$ that $B_i \in C(h_i)$. In particular, this implies that there exists $j \in C(h_i)$ such that $v_j[h_i] = 1$ (since this holds for $j = B_i$), and for all other $j \in C(h_i)$, $j \neq B_i$, $v_j[h_i] = 0$ (since all v_j's are distinct vectors of weight at most 1). Hence, $x[h_i] = 1 = \bigoplus_{j\in C(h_i)} v_j[h_i]$. On the other hand, for every $i \notin H_x$, it holds that $v_j[i] = 0$ for $j = 1$ to $\mathsf{HW}(x)$ (as each v_j is either $\mathsf{nzi}(h_k)$ for some k, or 0^s). Hence,

$$x = \left(\bigoplus_{j\in C(i)} v_j[i] \right)_{i\le s},$$

and correctness follows from the correctness of the local compression scheme for unit vectors $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$ (Lemma 6.2).

Compression. By $1.01/d$-compression of $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$ and since $\mathbf{c}_x = (\mathsf{Comp}_{\mathsf{u}}(1^s, v_j, d))_{j\le T}$, we have $|\mathbf{c}_x| \le T \cdot s^{1.01/d} = s^{1/c+1.01/d}$.

Locality. By $2t$-uniformity of the CBC, it holds that $|C(i)| \le 2t$ for every $i \le s$. Therefore, each bit $x'[i]$ of x' is computed by XORing at most $2t$ terms $\mathsf{Rec}_{\mathsf{u}}(\mathsf{pp}, \mathbf{c}_j, d, i)$. By the d-locality of $(\mathsf{Param}_{\mathsf{u}}, \mathsf{Comp}_{\mathsf{u}}, \mathsf{Rec}_{\mathsf{u}})$, it follows that $(\mathsf{Param}_{\mathsf{Sp}}, \mathsf{Comp}_{\mathsf{Sp}}, \mathsf{Rec}_{\mathsf{Sp}})$ is $2td$-local.

Efficiency. Eventually, given H_x instead of x as input, the modified $\mathsf{Comp}_{\mathsf{Sp}}$ procedure works as follows:

$\mathsf{Comp}^*_{\mathsf{Sp}}(\mathsf{pp}, H_x, d)$**:** on input public parameters pp parsed as $\mathsf{pp} = S$, a vector $x \in \mathsf{Sp}_s = \mathsf{Sp}_{s,w}(\mathbb{F}_2)$, and a compression factor d,

- Let $w_x = \mathsf{HW}(x)$. Denote $(h_1, \cdots, h_{w_x})$ the lexicographic ordering of the elements of H_x.
- Run $(B_1, \cdots, B_{w_x}) \leftarrow \mathsf{SelectBuckets}(H_x, C)$.
- For $j = 1$ to T, if there exists $i \le w_x$ such that $j = B_i$, set $v_j \leftarrow h_i$. Otherwise, set $v_j \leftarrow \perp$.
- For $j = 1$ to T, set $\mathbf{c}_j \leftarrow \mathsf{Comp}^*_{\mathsf{u}}(1^s, v_j, d)$.
- Output $\mathbf{c}_x \leftarrow (\mathbf{c}_1, \cdots, \mathbf{c}_T)$.

It is immediate to see that the modified compression procedure is correct. As for efficiency, $\mathsf{SelectBuckets}$ runs in time $w^{1+o(1)}$ in a RAM model with $\log(s)$-sized words (and $(w \cdot \log s)^{1+o(1)}$ in a circuit model). By efficiency of $\mathsf{Comp}^*_{\mathsf{u}}$, the runtime of $\mathsf{Comp}^*_{\mathsf{Sp}}$ is therefore dominated by $(w \cdot \log s)^{1+o(1)} + O(Td \cdot s^{1/d}) < w^{1.01} \cdot s^{1/d}$ for a large enough s.

7 A Structured-Seed Local PRG from Weak PRGs in Noisy-NC^0

In this section, we show how to turn a weak local PRG in noisy-NC^0 into a (strong) structured-seed local PRG. More specifically, we prove two independent results: (1) the existence of weak PRGs in noisy-NC^0 implies the existence of *subquadratic stretch* strong structured-seed local PRGs, and (2) (1) the existence of weak PRGs in noisy-NC^0 implies the existence of *arbitrary stretch* weak structured-seed local PRGs. We leave open the question of generically building strong structured-seed local PRGs with arbitrary stretch from weak PRGs in noisy-NC^0. The organization is as follows: we show in Section 7.1 how to convert any weak PRG in noisy-NC^0 into a strong PRG in noisy-NC^0, in Section 7.2 how to amplify the stretch of a noisy-PRG, and in Section 7.3 how to build a structured-seed local PRG by combining a PRG in $\mathcal{N}$-NC^0 and a local compression scheme for the support of $\mathcal{N}$.

7.1 Weak PRGs to Strong PRGs in Noisy-NC^0

Theorem 7.1. *Let $\mathcal{N} = \{\mathcal{N}_{s(n)}\}_{n\in\mathbb{N}}$ denote a family of distributions over $\{0,1\}^s$. The following hold: For every constant $d \in \mathbb{N}$, $a > 0$, and $\alpha \in (0,1)$, denoting $k = \log^2(n) \cdot n^{8/\alpha}$, any $1/n^a$-secure PRG $(\mathsf{Setup}, \mathsf{PRG})$ with $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ in $\mathcal{N}$-NC^0_d can be converted into a $\mathsf{negl}(n)$-secure PRG $(\mathsf{Setup}', \mathsf{PRG}')$ with $\mathsf{PRG}' : \{0,1\}^{n\cdot k + m\cdot k^\alpha} \to \{0,1\}^{mk}$ in $\mathcal{N}^k$-$\mathsf{NC}^0_{\ell(d,\alpha)}$ where*

$$\ell(d,\alpha) \leq d \cdot \left(\frac{16}{\alpha} + \left(\frac{2}{\alpha} + 1\right)^2\right).$$

In particular, for every constant $\varepsilon > 0$, there exists a choice of $\alpha > 0$ and polynomial $m(n)$ such that PRG' stretches n^t bits into $n^{2t-\varepsilon}$ pseudorandom bits (that is, PRG' achieves near-quadratic stretch).

Proof. We follow the proof of Theorem 2.12 in [AK19] which shows that weak local PRGs can be upgraded into strong local PRGs with polynomial stretch using a local-computable extractor. Because we work in the noisy-NC^0 model, the theorem does not apply *as is*. However, a very similar proof allows to prove a weaker statement, where the main difference with respect to the original proof of [AK19] being that we end up with a subquadratic-stretch (noisy) local PRG. This is due to the fact that, unlike standard PRGs, noisy PRGs do not admit a straightforward generic stretch extension since as we recursively invoke the PRG on itself, we increase the size of the noisy-NC^0 circuit and the noise blows up linearly with the number of steps.

First, a direct calculation shows that except with probability $1 - \mathsf{negl}(n)$, Ext is a $((1 - 4 \cdot n^{8/\alpha}) \cdot k(n'), \mathsf{negl}(n))$-extractor of locality $\ell(1,\alpha)$ by plugging our choice of inputs to $\mathsf{SampleExt}$ to the statement of Lemma 3.9; this also matches the parameters used in the proof of Theorem 2.12 in [AK19]. Second, the resulting noisy PRG $(\mathsf{Setup}', \mathsf{PRG}')$ has nontrivial stretch: PRG' maps $n = \tilde{O}((n')^{1+8/\alpha} + (n')^8 m(n'))$ bits to (more than) $m(n') \cdot n^{8/\alpha}$ bits. Setting $m(n') = (n')^{8/\alpha}$, this translates to stretching $(n')^{8/\alpha+8+o(1)}$ to $(n')^{16/\alpha}$ bits; as α goes to 0, this approaches a quadratic stretch.

As Theorem 2.12 in [AK19] proves the same statement for PRGs in NC^0 instead of PRGs in $\mathcal{N}$-NC^0, we restrict ourselves to extracting from the proof the places where replacing NC^0 with $\mathcal{N}$-NC^0 has an influence on the argument (we note that for most of the argument, it does not). The proof starts by observing that if $(\mathsf{Setup}, \mathsf{PRG})$ is a $1/n^a$-PRG with stretch $m = n^t$ and $t > a$, it is an unpredictable generator (Definition 3.6) with indistinguishability gap $1/n^{t-a}$. A simple observation here is that this property is oblivious to the complexity class of the PRG. Therefore, if we let ρ denote a random tape for $\mathcal{N}$ and write $r = \mathcal{N}(\rho)$ to denote the sample from $\mathcal{N}$ using random tape ρ, the function $G : (\mathsf{seed}, \rho) \mapsto \mathsf{PRG}(\mathsf{seed}; \mathcal{N}(\rho))$ is pseudorandom[11];

[11] We note that in general, G might not be a PRG because it is not necessarily *expanding*, depending on $\mathcal{N}$. However, this condition is not required in the proof of Yao's lemma.

Fig. 5: Strong local PRG $(\mathsf{Setup}', \mathsf{PRG}')$ in $\mathcal{N}^k$-NC^0

Parameters. Let $m = m(n)$ denote a polynomial with $m(n) > n$ and $d \in \mathbb{N}$, $a > 0$, $\alpha \in (0,1)$ denote three constants. Let $k = \log^2(n) \cdot n^{8/\alpha}$ and $\Delta = 16/\alpha + (2/\alpha+1)^2$. Denote $f : n \mapsto k(n) \cdot n + m(n) \cdot k(n)^\alpha$. The construction uses the following building blocks:

1. A $1/n^a$-secure PRGs $(\mathsf{Setup}, \mathsf{PRG})$ with $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ including two algorithms $(\mathsf{Setup}, \mathsf{PRG})$ in $\mathcal{N}$-NC^0, where $\mathcal{N} = \{\mathcal{N}_{s(n)}\}_{n\in\mathbb{N}}$ is a family of noise distributions supported over $\{0,1\}^s$.
2. A sampler $\mathsf{SampleExt}$ that, on input 1^k and two constants, outputs the description of a circuit $\mathsf{Ext} : \{0,1\}^k \times \{0,1\}^{k^\alpha} \to \{0,1\}^k$ $(0 < \alpha < 1)$ such that with probability $1 - \mathsf{negl}(k)$, Ext is a local extractor with parameters as guaranteed by Lemma 3.9.

Construction.

$\mathsf{Setup}'(1^n)$: Set $n' \leftarrow f^{-1}(n)$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{n'})$, and sample $\mathsf{Ext} \xleftarrow{\$} \mathsf{SampleExt}(1^{k(n')}, 1/\alpha, \Delta)$. Output $\mathsf{pp}' = (\mathsf{pp}, \mathsf{Ext})$.

$\mathsf{PRG}'(\mathsf{pp}', \mathsf{seed})$: on input $\mathsf{seed} \in 1^n$,

- set $n' \leftarrow f^{-1}(n)$. Decompose $\mathsf{seed} \in \{0,1\}^{k(n')\cdot n' + m(n')\cdot k(n')^\alpha}$ into $k(n')$ PRG seeds $(\mathsf{seed}_1, \cdots, \mathsf{seed}_{k(n')})$ of length n' bits each, and $m(n')$ extractor seeds $(s_1, \cdots, s_{m(n')})$ of length $k(n')^\alpha$ each.
- Run $(x_1, \cdots, x_{k(n')}) \leftarrow (\mathsf{PRG}(\mathsf{seed}_1; r_1), \cdots, \mathsf{PRG}(\mathsf{seed}_{k'(n)}; r_{k'(n)})$ where the r_i's are sampled from $\mathcal{N}_{s(n')}$ by the $\mathcal{N}$-NC^0 circuit of PRG.
- Let X denote the matrix in $\{0,1\}^{m(n')\times k(n')}$ whose rows are the x_i's, and let $(x'_1, \cdots, x'_{m(n')})$ denote the columns of X. For $i = 1$ to $m(n')$, set $y_i \leftarrow \mathsf{Ext}(x'_i, s_i)$.
- Output $y = (y_1, \cdots, y_{m(n')}) \in \{0,1\}^{k(n')\cdot m(n')}$.

applying Yao's lemma to G yields that it is unpredictable, hence that PRG is an unpredictable generator in $\mathcal{N}$-NC^0.

The rest of the proof proceeds by showing that, even conditioned on all previous columns of X, the i-th column of X has an amount of min-entropy that allows to apply the extraction property of Ext. This proof is identical in our scenario, as it's solely a property of the min-entropy of a distribution defined by the PRG that follows from its pseudorandomness and does not depend on the complexity of the circuit that computes it. Eventually, we conclude with

Corollary 7.2. *For any noise distribution $\mathcal{E} \in \{\mathcal{X}_{m,w}, \mathcal{R}_{m,w}, \mathcal{B}_{m,w}\}$, under the hardness of the $(\mathcal{M}^c, \mathcal{E})$-$\mathsf{LPN}$ assumption with dimension n and $m = n^{1+\eta}$ for $\eta \in (0,1)$, given any constant $\alpha \in (0,1)$ (see Parameter Set 1), denoting $k = \log^2(n) \cdot n^{8/\alpha}$, there exists a $\mathsf{negl}(n)$-secure PRG in $\mathcal{E}^k$-NC^0 that stretches $(n \cdot k + n^{1+\eta} \cdot k^\alpha)$ bits into $n^{(1+\eta)} \cdot k$ bits.*

7.2 Weak PRGs Beyond Quadratic Stretch

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 [IKOS08]. However, this does not hold anymore for a PRG in noisy-NC^0, since the output distribution of the PRG does not match the required seed and noise distributions. Nevertheless, we show in this section that the stretch of any noisy-local PRGs can be extended to an arbitrary polynomial via self-composition. At a high level, given a noisy-local PRG with subquadratic stretch $m(n) = o(n^2)$, we leverage the observation that the input of $\mathsf{PRG}(\mathsf{pp}, \mathsf{seed}; r)$ has two components:

- A random bitstring seed of length n, and
- A random tape $r \xleftarrow{\$} \mathcal{N}_s$.

Since seed is randomly chosen from $\{0,1\}^n$, we can view it as the output of the noisy-local PRG applied itself to a smaller seed along with a random tape sampled from $\mathcal{N}_m$, producing a stretch of m bits. This process can be applied recursively until reaching the desired expansion. The detail is described below.

Construction. Let $\mathcal{N} = \{\mathcal{N}_{s(n)}\}_{n\in\mathbb{N}}$ be a $(w(n), s(n))$-noise distribution. Let $G := (\mathsf{Setup}, \mathsf{PRG})$ be a (ϵ, δ)-secure noisy-local PRG with subquadratic stretch: on input $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^n)$, $\mathsf{seed} \xleftarrow{\$} \{0,1\}^n$, and noise $r \xleftarrow{\$} \mathcal{N}_{s(n)}$, $\mathsf{PRG}(\mathsf{pp}, \mathsf{seed}; r)$ returns $y \in \{0,1\}^{m(n)}$ with $m(n) = n^{1+\gamma}$ for some constant $\gamma < 1$. Denote $\theta = (1+\gamma)^{-1} \in (1/2, 1)$, i.e., $n = m^\theta$. Let $(n_0, m_0, s_0) \leftarrow (m^\theta, m, s(n_0))$. We describe the construction as a sequence of t seed-shrinking steps:

Step 1: Fix a target output length m. We aim to generate a seed $\mathsf{seed}_0 \in \{0,1\}^{m^\theta}$. We do so by sampling seed_0 pseudorandomly using G recursively: set $(n_1, m_1, s_1) = (m^{\theta^2}, m^\theta, s(n_1))$ and sample $\mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^{n_1}$, $r_1 \xleftarrow{\$} \mathcal{N}_{s_1}$. Set $\mathsf{pp}_1 \xleftarrow{\$} \mathsf{Setup}(1^{n_1})$ and $\mathsf{seed}_0 = \mathsf{PRG}(\mathsf{pp}_1, \mathsf{seed}_1; r_1)$.

Step $i < t$: We aim to generate a seed $\mathsf{seed}_{i-1} \in \{0,1\}^{m^{\theta^i}}$. We do so by sampling seed_i pseudorandomly using G recursively: set $(n_i, m_i, s_i) = (m^{\theta^{i+1}}, m^{\theta^i}, s(n_i))$ and sample $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^{n_i}$, $r_i \xleftarrow{\$} \mathcal{N}_{s_i}$. Set $\mathsf{pp}_i \xleftarrow{\$} \mathsf{Setup}(1^{n_i})$ and $\mathsf{seed}_{i-1} = \mathsf{PRG}(\mathsf{pp}_i, \mathsf{seed}_i; r_i)$.

After t seed-shrinking steps, we obtain the following construction of large-stretch noisy-local PRG $G^* = (\mathsf{Setup}^*, \mathsf{PRG}^*)$ in $\mathcal{N}^*$-NC^0, where $\mathcal{N}^*_s := \prod_{i=0}^{t-1} \mathcal{N}_{s_i}$:

- $\mathsf{Setup}^*(1^n)$: for $i = t-1$ to 0, run $\mathsf{pp}_i \xleftarrow{\$} \mathsf{Setup}(1^{n_i})$. Output $\mathsf{pp}^* = (\mathsf{pp}_0, \cdots, \mathsf{pp}_{t-1})$.
- $\mathsf{PRG}^*(\mathsf{pp}^*, \mathsf{seed}^*)$: on input $\mathsf{seed}^* \xleftarrow{\$} \{0,1\}^{n_{t-1}}$, set $\mathsf{seed}_{t-1} := \mathsf{seed}^*$. Sample $(r_0, \cdots, r_t) \xleftarrow{\$} \mathcal{N}^*_{s(n)}$ and iteratively define $\mathsf{seed}_{i-1} = \mathsf{PRG}(\mathsf{pp}_i, \mathsf{seed}_i; r_i)$. Output $y = \mathsf{seed}_0$.

Security follows immediately from a sequence of t hybrids $(H_1, \cdots, H_t)$ that "undo" the t seed-shrinking steps, replacing at hybrid $t - i$ the seed $\mathsf{seed}_{i-1} = \mathsf{PRG}(\mathsf{pp}_i, \mathsf{seed}_i; r_i)$ with a random $\mathsf{seed}_{i-1} \xleftarrow{\$} \{0,1\}^{n_{i-1}}$ (under the (ε, δ)-security of the noisy-local PRG G). The last hybrid replaces $\mathsf{seed}_0 = y$ by random, completing the proof. This incurs a (constant) factor-t loss of security. If the original PRG is d-local, the resulting PRG is d^t-local. The final PRG stretches an m^{θ^t}-bit seed to m bits, yielding an arbitrary polynomial gap.

Theorem 7.3. *Assume the existence of an (ε, δ)-secure noisy-local PRG $G := (\mathsf{Setup}, \mathsf{PRG})$ in $\mathcal{N}$-NC^0, where $\mathcal{N}$ is a (w, s)-noise distribution, and PRG stretches an $n = m^\theta$-bit seed to m bits, for some constant $\theta \in (1/2, 1)$. Let d denote the locality of PRG. Fix any constant $t \in \mathbb{N}^{>2}$. Then there exists a $(t\varepsilon, t\delta)$-secure noisy-local PRG $G^* = (\mathsf{Setup}^*, \mathsf{PRG}^*)$ in $\mathcal{N}^*$-NC^0 where $\mathcal{N}^*$ is a (w^*, s^*)-noise distribution with $w^*(n) = \sum_{i=0}^{t-1} w(m^{\theta^{i+1}})$, $s^*(n) = \sum_{i=0}^{t-1} s(m^{\theta^{i+1}})$. The PRG has the following characteristics:*

- *Stretch: PRG^* stretches an m^{θ^t}-bit seed to m bits.*
- *Locality: PRG^* is d^t-local.*

Plugging our sparse-LPN-based construction of noisy-local PRG in Theorem 7.3, using sparse LPN with Parameter Set 2, we get:

Corollary 7.4. *Fix any noise distribution $\mathcal{E}_m \in \{\mathcal{X}_{m,w}, \mathcal{R}_{m,w}, \mathcal{B}_{m,w}\}$. Then under the (ϵ, δ)-security of the $(\mathcal{M}^c, \mathcal{E}_m)$-$\mathsf{LPN}$ assumption with dimension n, number of samples $m = n^{1+\gamma/2}$ for a constant γ such that $c \geq 2\log\gamma/(\log\gamma - 1)$, and noise $w = \lambda \cdot n^\gamma$ (see Parameter Set 2), denoting $\theta = \frac{1}{1+\gamma/2}$, for any constant $t \in \mathbb{N}^{>1}$, there exists a $((t-1)\varepsilon, (t-1)\delta)$-secure noisy-local PRG $G^* = (\mathsf{Setup}^*, \mathsf{PRG}^*)$ in $\mathcal{N}^*$-NC^0 where $\mathcal{N}^*$ is a (w^*, m^*)-noise distribution, with w^* as in Theorem 7.3 and $m^* = \sum_0^{t-1} m^{\theta^{i+1}}$, that stretches an m^{θ^t}-bit seed to m bits.*

7.3 Constructions of Structured-Seed Local PRGs

We present our structured-seed local PRG in Figure 6 and its properties are described in Theorem 7.5. Our structured-seed local PRG is constructed from a local-noisy PRG and a local compression while its security is leveraged from the security of local-noisy PRG.

Fig. 6: Structured-seed local PRG G^* $(\mathsf{Setup}^*, \mathsf{SampleSeed}, \mathsf{PRG}^*)$

Input:

- A (ϵ, δ)-secure noisy-local PRG $(\mathsf{Setup}, \mathsf{PRG})$ with $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$, where $\mathcal{N} = \{\mathsf{Sp}_{s(n)}\}_{n\in\mathbb{N}}$ is a family of noise distributions supported over $\{0,1\}^s$.
- An $\ell(d)$-local $\mathsf{cf}(d)$-compression scheme $(\mathsf{Param}_{\mathsf{Sp}}, \mathsf{Comp}_{\mathsf{Sp}}, \mathsf{Rec}_{\mathsf{Sp}})$ and d is compression parameter.

PRG G^* is defined by three algorithms as below:

$\mathsf{Setup}^*(1^n) \to \mathsf{pp}^*$:

$$\mathsf{pp}^* = (\mathsf{pp}, \mathsf{pp}_{\mathcal{N}}) \leftarrow (\mathsf{Setup}(1^n), \mathsf{Param}_{\mathsf{Sp}}(1^\lambda))$$

$\mathsf{SampleSeed}(\mathsf{pp}^*) \to \mathsf{seed}$:

1. Sample uniformly $\mathbf{x} \xleftarrow{\$} \{0,1\}^n$, $\mathbf{e} \xleftarrow{\$} \mathcal{N}$.
2. Compute $\mathbf{c} \leftarrow \mathsf{Comp}_{\mathsf{Sp}}(\mathsf{pp}_{\mathcal{N}}, \mathbf{e}, d)$.
3. Define $\mathsf{seed} := (\mathbf{x}, \mathbf{c})$.

$\mathsf{PRG}^*(\mathsf{pp}^*, \mathsf{seed}) \to \mathbf{y}$:

1. Parse $\mathsf{seed} \to (\mathbf{x}, \mathbf{c})$, reconstruct noise $\mathbf{e} \leftarrow \mathsf{Rec}_{\mathsf{Sp}}(\mathsf{pp}_{\mathcal{N}}, \mathbf{c}, d)$.
2. Compute $\mathbf{y} \leftarrow \mathsf{PRG}(\mathsf{pp}, \mathbf{x}, \mathbf{e})$ and define $\mathsf{PRG}^*(\mathsf{pp}^*, \mathsf{seed}) \to \mathbf{y}$.

Theorem 7.5. *Consider the noise distribution $\mathcal{N} = \mathcal{S}_{m,w(n)}$, given a (ϵ, δ)-secure noisy-local PRG $G = (\mathsf{Setup}, \mathsf{PRG})$ with $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ and locality of c bits, an $\ell(d)$-local $\mathsf{cf}(d)$-compression scheme $(\mathsf{Param}_{\mathsf{Sp}}, \mathsf{Comp}_{\mathsf{Sp}}, \mathsf{Rec}_{\mathsf{Sp}})$ over the support of $\mathcal{N}$ with compression parameter d, set $\ell(d) = 2td$ (Algorithm 3). Then, there exists a secure structured-seed local pseudorandom generator $G^* = (\mathsf{Setup}^*, \mathsf{SampleSeed}, \mathsf{PRG}^*)$ with the following characteristics:*

- Parameter size: $m(c \log n + 2t \cdot \lceil \log m^{\frac{100}{t+98}} \rceil)$;
- Seed length: $|\mathsf{seed}| = n + m^{\mathsf{cf}(d)}$;
- Stretch: m;
- Locality: $c + \ell(d)$.

In addition, the algorithms have the following running time:

- $\mathsf{Setup}^*(1^n)$ *runs in* $O(m \log n + w^{1+o(1)})$ *time,*
- $\mathsf{SampleSeed}(\mathsf{pp}^*)$ *runs in* $O(n + w \log m + w^{1.01} \cdot m^{1/d})$ *time.*

Efficiency and Security. 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. In particular:

- Parameter size includes the size of Setup from local-noisy PRG that is dominated by $m \cdot c \cdot \log n$ and the size of Param from local compression scheme, i.e., the description size of $C \leftarrow \mathsf{SampleCode}(m, c, t)$ that is $m \cdot 2t \cdot \lceil \log m^{1/c'} \rceil$ as shown in Section 6.3, plugging parameters from Algorithm 3 where $c' = \frac{100}{t+98}$, we have $|C| < m \cdot 2t \cdot \lceil \log m^{\frac{100}{t+98}} \rceil$.
- Seed length is $n + m^{\mathsf{cf}(d)}$, since $\mathsf{seed} = (\mathbf{x}, \mathbf{c})$ and $\mathbf{x} \in \{0,1\}^n$ while $\mathbf{c} \leftarrow \mathsf{Comp}(\mathsf{pp}_{\mathcal{N}}, \mathbf{e}, d)$ has the size of $m^{\mathsf{cf}(d)}$ bits with a compression factor d.
- Stretch length is m as the stretch of local-noisy PRG G.
- For locality, computing each output bit of PRG requires reading c bits of $\mathbf{x}$ as G is c-local, and recovering each bit of $\mathsf{Rec}(\mathsf{pp}^*, \mathbf{c})$ requires reading $\ell(d)$ bits in total.

The running time of Setup^* is decomposed as follows:

- Setup of G ,i.e., sampling A requires sampling $c \cdot m$ elements of $[n]$ in time $O(m \log n)$.
- $\mathsf{Param}_{\mathsf{Sp}}$ of the local noise compression, i.e., sampling the oriented bipartite graph that is associated with CBC can be done in $w^{1+o(1)}$.

The running time of $\mathsf{SampleSeed}$ is decomposed as follows:

- Sampling $\mathbf{x}$ requires tossing n coins.
- Sampling $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{w,m}$ requires $O(w \log m)$ time.
- Computing the $\mathsf{Comp}(\mathsf{pp}_{\mathcal{N}}, \mathbf{e}, d)$ includes $w^{1.01} \cdot m^{1/d}$ (Lemma 6.7).

Security analysis. The security proof follows directly from the security of the noisy-local PRG G. We note that both the hardness and the stretch of the final structured-seed local PRG G^* depend on the hardness parameter ϵ of the underlying noisy-local PRG G. As a result, we obtain two different cases, corresponding to either stronger security or better stretch:

- By applying Theorem 7.1, if the noisy-local PRG G is a strong PRG, then the resulting G^* is also a strong PRG with subquadratic stretch.
- By applying Theorem 7.3, if the noisy-local PRG G achieves polynomial stretch, then the resulting G^* also achieves polynomial stretch.

8 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) and functional encryption [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 approach breaks down as the seed is not structured anymore.

8.1 Indistinguishability obfuscation and functional encryption

Functional encryption enhances standard public key encryption by enabling fine-grained decryption access: informally, given a function f and a master secret key msk, a functional encryption scheme allows deriving a functional key sk_f that can decrypt any ciphertext $\mathsf{Enc}(\mathsf{pk}, m)$ to $f(m)$. A functional encryption scheme is fully-succinct if, informally, both the ciphertexts and functional keys have size independent of the functions f.

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 $\mathbb{F}_p$,
- the existence of PRGs in NC^0,
- the Decision Linear assumption on symmetric bilinear groups of prime order p.

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 functional encryption (from the polynomial hardness of the above assumptions). Eventually, assuming the subexponential

hardness of a fully-succinct functional encryption scheme (that can be based on the subexponential security of the previous assumptions), there are known constructions of indistinguishability obfuscation [AJS15, BV15]. Above, the local PRG is used for constructing a *perturbation-resilient generator*. It follows by inspection that the construction of perturbation-resilient generator of [JLS21] goes through identically if the local PRG is replaced by a structured-seed local PRG. In the polynomial hardness regime, this immediately gives:

Theorem 8.1 (informal). *Under the following assumptions:*

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

there exists collusion-resistant public-key functional encryption for all polynomial-size circuits.

When moving to the subexponential hardness regime, however, there is an important technicality: the local PRG used in [JLS21] needs to have subexponential security, while our sequence of reductions only guarantees a "mildly" negligible distinguishing advantage $n^{\omega(1)}$. Nevertheless, we make two observations:

- The decisional sparse-LPN assumption with dimension n and m samples can plausibly be conjectured to be (T, ε, δ)-secure for a subexponential $T = 2^{n^\alpha}$ (for some $\alpha < 1$), inverse-subexponential $\varepsilon = 2^{-\Omega(n^\alpha)}$, and inverse-polynomial δ (see Parameter Set 1 and Parameter Set 2). That is, decision sparse-LPN already provides subexponential hardness *except* for a one-time inverse-polynomial failure probability of sampling a "bad LPN matrix".
- While our general local compression scheme for sparse vectors, that builds upon the CBC from [AK19], is probabilistic and has a $n^{-\omega(1)}$ failure probability, the simple local compression scheme for *regular* vectors from Section 6.2 has perfect correctness.

Therefore, combining both observations, there exists under the subexponential hardness of the decisional sparse-LPN assumption with regular noise a (T, ε, δ)-secure SSL-PRG for a subexponential $T = 2^{n^\alpha}$ (for some $\alpha < 1$), inverse-subexponential $\varepsilon = 2^{-\Omega(n^\alpha)}$, and inverse-polynomial δ – that is, a weak SSL-PRG that provides full subexponential security after a one-time failure probability confined to the sampling of $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^n)$.

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 8.2 (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 decisional hardness of sparse-LPN 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.

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. In light of the recent concurrent and independent work of [RVV24a] 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.

8.2 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 8.3 below.

Theorem 8.3. *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 search hardness of sparse-LPN with any standard noise distribution, diversifying the set of assumptions under which extreme efficiency can be achieved in secure computation.

We sketch the proof of Theorem 8.3 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.

8.3 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 8.4 (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: their construction essentially uses the "hybrid encryption" technique with a local PRG (similar to our warmup example in Section 2). More formally, 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 8.5. *Assuming the superpolynomial hardness of DCR and the search hardness of sparse-LPN assumption with any standard noise distribution, 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.[12]

8.4 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 8.6 (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 8.7. 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.

[12] 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.

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

Theorem 8.8 (Theorem 3.1 in [DV21]). *Under the assumption of Corollary 7.4, 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). 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 7.3.

At the heart of the proof of Theorem 3.1 in [DV21] is the following observation: given an NC^0 PRG with output length m, for each seed x (not necessarily sampled uniformly), we can define a formula φ_x that takes as an input the set of locations S_i that affects the ith output bit and outputs the i-th output bit of the PRG. In the specific setting of local PRGs constructed from random local functions, this function is hard to learn over the uniform distribution and can be computed with low complexity, e.g., depth-3 circuit (See [AR16a]). It was further observed in [DV21] that one can write the formula as a DNF, if we push some of the complexity to the input distribution. Specifically, this can be done by representing each index in S_i by a one-hot representation. Looking ahead, our generalization of their result to SSL-PRG will proceed by pushing "even more" of the complexity to the input distribution.

We provide the formal details: 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 $\mathbf{v}_i := \mathsf{u}_n(s_i) \in \{0,1\}^n$ for $i = 1$ to ℓ. Then, consider the following formula ψ:

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

We have

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

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 3 *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 7.3, the assumption above is implied by the search 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 $i \leq k^s$, 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 $(z_i, \psi(z_i))_{i \leq Q}$ (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.

We note that a difference between the result in [DV21] and ours is that $\mathcal{D}$ is a highly artificial distribution, while in [DV21], it simply samples strings of i.i.d. biased bits. As a consequence, we do not see how to derive the corollaries of [AR16b, DV21] that rule out *uniform* PAC-learning of depth-3 AC^0 circuits (where the input distribution is the uniform distribution). We leave the goal of ruling out uniform PAC learning of AC^0 circuits from sparse LPN as an interesting open question.

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 8.8.

Corollary 8.9 (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.

Corollary 8.10. *For all $q(n) = \omega(1)$ there is no efficient algorithm that learns $q(n)$-sparse GF(2) polynomials over $\{0,1\}^n$.*

Acknowledgements

Benny Applebaum was supported by the ISF grant 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Dung Bui, Geoffroy Couteau, and Nikolas Melissaris acknowledge the support of the French Agence Nationale de la Recherche (ANR) under the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. This work is supported by ERC grant OBELiSC (101115790).

References

ABG+14. A. Akavia, A. Bogdanov, S. Guo, A. Kamath, and A. Rosen. Candidate weak pseudorandom functions in $\mathsf{AC}^0 \circ \mathrm{MOD}_2$. 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.

ABW10. B. Applebaum, B. Barak, and A. Wigderson. Public-key cryptography from different assumptions. In *42nd ACM STOC*, pages 171–180. ACM Press, June 2010.

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, Cham, Switzerland, 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.

AJS15. P. Ananth, A. Jain, and A. Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015.

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.

App16. B. Applebaum. Cryptographic hardness of random local functions - survey. *Comput. Complex.*, 25(3):667–722, 2016.

AR16a. B. Applebaum and P. Raykov. Fast pseudorandom functions based on expander graphs. In *TCC 2016-B*, pages 27–56, 2016.

AR16b. B. Applebaum and P. Raykov. Fast pseudorandom functions based on expander graphs. In *TCC 2016-B, Part I*, *LNCS* 9985, pages 27–56. Springer Berlin Heidelberg, Germany, October / November 2016.

Aud07. K. M. R. Audenaert. A sharp continuity estimate for the von neumann entropy. *Journal of Physics A: Mathematical and Theoretical*, 40(28):8127–8136, June 2007.

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.

BBTV24. K. Bangachev, G. Bresler, S. Tiegel, and V. Vaikuntanathan. Near-optimal time-sparsity trade-offs for solving noisy linear equations, 2024.

BCCD23. M. Bombar, G. Couteau, A. Couvreur, and C. Ducros. Correlated pseudorandomness from the hardness of quasi-abelian decoding. In *Advances in Cryptology – CRYPTO 2023, Part IV*, *LNCS* 14084, pages 567–601. Springer, Cham, Switzerland, August 20–24, 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. In *Advances in Cryptology – CRYPTO 2022, Part II*, *LNCS* 13508, pages 603–633. Springer, Cham, Switzerland, August 15–18, 2022.

BCG+23. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Oblivious transfer with constant computational overhead. In *Advances in Cryptology – EUROCRYPT 2023, Part I*, *LNCS* 14004, pages 271–302, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

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. In *TCC 2022: 20th Theory of Cryptography Conference, Part II*, *LNCS* 13748, pages 121–150, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.

BCM23. E. Boyle, G. Couteau, and P. Meyer. Sublinear-communication secure multiparty computation does not require FHE. In *Advances in Cryptology – EUROCRYPT 2023, Part II*, *LNCS* 14005, pages 159–189, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

BCM+24. D. Bui, G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In *Advances in Cryptology – EUROCRYPT 2024, Part VI*, *LNCS* 14656, pages 88–118, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

Bea92. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO'91*, *LNCS* 576, pages 420–432. Springer Berlin Heidelberg, Germany, 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 Berlin Heidelberg, Germany, 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 Berlin Heidelberg, Germany, August 2016.

BKR23. A. Bogdanov, P. K. Kothari, and A. Rosen. Public-key encryption, local pseudorandom generators, and the low-degree method. In *TCC 2023: 21st Theory of Cryptography Conference, Part I*, *LNCS* 14369, pages 268–285, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

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 Berlin Heidelberg, Germany, 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.

BV15. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

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

CGHTKV25. H. Corrigan-Gibbs, A. Henzinger, Y. Tauman Kalai, and V. Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse lpn. In *Advances in Cryptology – EUROCRYPT 2025*, pages 3–33, Cham, 2025. Springer Nature Switzerland.

CK24. G. Couteau and N. Kumar. 10-party sublinear secure computation from standard assumptions. In *Advances in Cryptology – CRYPTO 2024, Part IX*, *LNCS* 14928, pages 39–73. Springer, Cham, Switzerland, August 18–22, 2024.

CKL+22. L. Chen, R. Kyng, Y. P. Liu, R. Peng, M. P. Gutenberg, and S. Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022.

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. In *Advances in Cryptology – CRYPTO 2021, Part III*, *LNCS* 12827, pages 502–534, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

DIJL23. Q. Dao, Y. Ishai, A. Jain, and H. Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In *Advances in Cryptology – CRYPTO 2023, Part II*, *LNCS* 14082, pages 315–348. Springer, Cham, Switzerland, August 20–24, 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, Cham, Switzerland, August 2017.

Fei02. U. Feige. Relations between average case complexity and approximation complexity. In *34th ACM STOC*, pages 534–543. ACM Press, May 2002.

FGKP06. V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. In *47th FOCS*, pages 563–574. IEEE Computer Society Press, October 2006.

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 Berlin Heidelberg, Germany, August 1987.

Gol00. O. Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000.

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

HMS07. D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. In *TCC 2007*, *LNCS* 4392, pages 214–232. Springer Berlin Heidelberg, Germany, 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 Berlin Heidelberg, Germany, 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. In *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73, Virtual Event, Italy, June 21–25, 2021. ACM Press.

JLS22. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In *Advances in Cryptology – EUROCRYPT 2022, Part I*, *LNCS* 13275, pages 670–699, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

Kir11. P. Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011.

KOW24. D. M. Kane, A. Ostuni, and K. Wu. Locality bounds for sampling hamming slices. In *56th Annual ACM Symposium on Theory of Computing*, pages 1279–1286, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

LF06. É. Levieil and P.-A. Fouque. An improved LPN algorithm. In *SCN 06*, *LNCS* 4116, pages 348–359. Springer Berlin Heidelberg, Germany, 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 Berlin Heidelberg, Germany, 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, Cham, Switzerland, 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. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 378–389, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

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 Berlin Heidelberg, Germany, 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. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

RVV24a. S. Ragavan, N. Vafa, and V. Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*, *LNCS* 15367, pages 3–36, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

RVV24b. S. Ragavan, N. Vafa, and V. Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. Cryptology ePrint Archive, Report 2024/856, 2024.

Saa07. M.-J. O. Saarinen. Linearization attacks against syndrome based hashes. In *INDOCRYPT 2007*, *LNCS* 4859, pages 1–9. Springer Berlin Heidelberg, Germany, December 2007.

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.

Üna23. A. Ünal. Worst-case subexponential attacks on PRGs of constant degree or constant locality. In *Advances in Cryptology – EUROCRYPT 2023, Part I*, *LNCS* 14004, pages 25–54, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

Val84. L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984.

Vio10. E. Viola. The complexity of distributions. In *51st FOCS*, pages 202–211. IEEE Computer Society Press, October 2010.

Wag02. D. Wagner. A generalized birthday problem. In *CRYPTO 2002*, *LNCS* 2442, pages 288–303. Springer Berlin Heidelberg, Germany, August 2002.

Wee05. H. Wee. On obfuscating point functions. In *37th ACM STOC*, pages 523–532. ACM Press, May 2005.

Yao82. A. C.-C. Yao. Theory and application of trapdoor functions. *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 80–91, 1982.

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.

Zic17. L. Zichron. Locally computable arithmetic pseudorandom generators. Master's Thesis, Tel Aviv University, 2017. `https://www.bennyapplebaum.sites.tau.ac.il/_files/ugd/f706bf_501515c9cd7744c498935684bd1648a2.pdf`.

A Proof of Theorem 5.5

We restate the Theorem for convenience:

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

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

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 $A \in \mathbb{F}_2^{m\times n}$, we denote by $(\mathbf{a}_1, \cdots, \mathbf{a}_m)$ the rows of A, and by $(\mathbf{a}^1, \cdots, \mathbf{a}^n)$ its columns. For any subset $S \subseteq [m]$ of the rows of A, we call *column-neighbors of S in A*, and write $\mathsf{N}^{\mathsf{c}}_A(S) \subseteq [n]$, 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 $S \subseteq [m]$ 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 $A \in \mathbb{F}_2^{m\times n}$ has d-unique column-neighbor, then $\mathsf{dd}(A) \geq d - 1$.*

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

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 $S \subseteq [m]$ 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 $A \in \mathsf{Supp}(\mathcal{M}^c)$ 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 $S \subseteq [m]$ 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 5.5, it remains to show that a random sample from $\mathcal{M}^c$ is sufficiently expanding with high probability.

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

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

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

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

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

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

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 $d = n^2/(\gamma(c) \cdot n)$, the above is upper bounded by

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

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