Structured-Seed Local Pseudorandom Generators and Their Applications

Benny Applebaum
Tel-Aviv University, Israel

Dung Bui
LIP6, Université Sorbonne, Paris, France

Geoffroy Couteau
CNRS, IRIF, Université Paris Cité, France

Nikolas Melissaris
CNRS, IRIF, Université Paris Cité, France

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.

2012 ACM Subject Classification Security and privacy → Cryptography; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Pseudorandom Generator, Local Pseudorandom Generators, Secure Computation, Obfuscation, Hardness of Learning, Local Compression

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.63

Category RANDOM

Related Version *Full Version*: `https://eprint.iacr.org/2024/1027`

Funding 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).

Acknowledgements The work was done while Dung Bui was at IRIF.

1 Introduction

We study the existence of cryptographic primitives that minimize parallel time-complexity, continuing a fruitful line of work initiated by [8]. 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 [4, 12, 32], hardness of approximation [1, 2, 9], secure computation [7, 22–24, 39], indistinguishability obfuscation [40, 41], and more [4, 15].

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).

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 [31] 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 [8] 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 [9] 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 [1].

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 [37] 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 [2, 10], or assume that the direct construction is a PRG.[1] The latter assumption and its variants have been extensively studied and analyzed in the past two decades [5, 10, 11, 16, 26, 28, 34, 43, 45–47, 49, 50, 55, 57, 58]. (See [3] for a survey.)

[1] 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.

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 [14], 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 $m = \mathsf{poly}(n)$.

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 [1] 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., [4]). Additionally, it is known [2, 4] 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 [2, 9] and to the field of hardness of approximation [35]. 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. [36] 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 [40, 41] and secure computation [7, 22–24, 39], 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 [29])[2]. 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 [10], 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[3], stretch expansion, and (in a special case) search-to-decision reductions.
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** (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 the full version [6] 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 [2].

---

[2] Other applications, such as PAC learning lower bounds [12, 32] 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.

[3] 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.

▶ **Corollary 2.** *Under the assumption of Theorem 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 3.** *Under the assumption of Theorem 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 with the subexponential hardness of* decisional *sparse-LPN with* regular *noise, there exists a (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits.*

▶ **Corollary 4.** *Under the assumption of Theorem 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 5.** *Under the assumption of Theorem 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 [52]. The full version of their work [53] 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 [52]. Below, we expand on the differences between the current work and the work of [52].

The work of [52] focuses on indistinguishability obfuscation (iO). Their main observation is that the polynomial-stretch local PRG used in a recent construction of iO [41] 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 [52], confining ourselves to observing that structured-seed local PRGs can be used as a drop-in replacement in [40], without revisiting the (more involved) construction of [41] 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 [52]). Compared with [52], 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 ([52] relied on the hardness of the decisional problem) for all standard noise distributions (the construction of [52] requires sparse LPN with Bernoulli noise); (3) we explore a broader range of applications, outlined in the previous section.

1.4.1 Subsequent work

After a preliminary version of our work appeared online, a subsequent work [13] 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 $\mathsf{SampleSeed}$) 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 [1] 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 [9] and pseudorandom correlation generators [18, 19], but also more advanced primitives such as multiparty homomorphic secret sharing [33] and somewhat homomorphic encryption [27].

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 [42, 56]. For other distributions, such as Bernoulli noise with parameter $p = 2^{-k}$, local samplers are known when k is constant [9] (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 [56][4]. 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:

- 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 $r' = (r'_{i,j})_{i \leq w, j \leq d}$.

[4] The work of [9] 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.

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 $\prod_{\ell=1}^{d} r'_{i,\ell}[j_\ell] = 1$.

2.1.1 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.

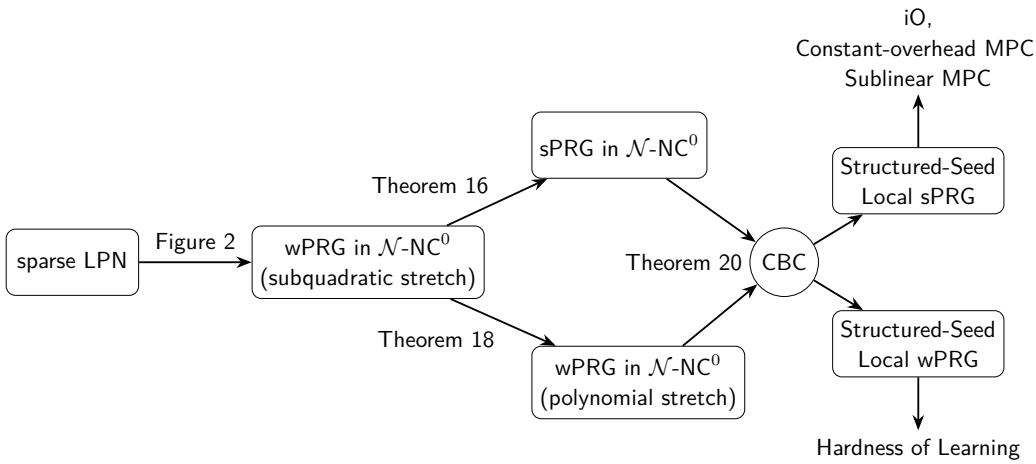

Figure 1 Summary of the implications shown in this work.

2.2 Noisy-local PRGs

We now overview our results on noisy-local PRGs.

2.2.1 Weak-to-strong amplification

A generic weak-to-strong amplification for local PRGs was described in the work of [10, 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 [10] converts an (arbitrary stretch) local PRG into a subquadratic stretch local PRG. 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 [10] 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 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 [10] requires using the fact that a PRG is in particular

an unpredictable generator 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.

2.2.2 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.

2.2.3 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 [56]. 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 [10]. At a high level, the CBC of [10] 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 Defining Noisy-Local PRGs and Structured-Seed Local PRGs

3.1 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.

3.1.1 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 $\alpha < 1$.

3.1.2 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 6.** *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 7.** *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 $\mathcal{B}_p(\mathbb{F}_2)$.*

3.2 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 8** (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.

3.3 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 9** (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* $\mathsf{SampleSeed}$ *(implemented by a uniformly efficiently generatable family of boolean circuits), we say that a structured-seed PRG has* strong polynomial stretch *if* $m(n) = \mathsf{T}_{\mathsf{ss}}^{1+\Omega(1)}$*.*

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 10** (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 9, except that* $\mathsf{Setup}(1^n)$ *is not required to run in polynomial time.*

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

4.1 The sparse-LPN assumption

We refer the reader to the full version [6] for preliminaries on the LPN assumption and standard noise distributions. In this work, we will mostly focus on a variant introduced in [1] (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 [48] and Mossel, Shpilka, and Trevisan [47], and which was explicitly put forth and studied in the context of LPN in [7, 21, 30]. The central observation of this framework is that most known attacks against LPN (such as BKW [14, 44], ISD [51], 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. We are stating 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* $\varepsilon = \frac{\mathsf{poly}(T)}{2^{\Omega(\lambda)}}$ *and* $\delta = \frac{1}{\Omega(n^\eta)}$*.*

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.*

4.2 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 $(\mathcal{M}^c, \mathcal{N}, \mathbb{F}_2)$-LPN$(n, m)$ assumption.

Figure 2: Weak noisy PRGs from sparse-LPN

Parameters. For Parameter Sets 1 or 2
Construction.
Setup(1^n): Sample $A \xleftarrow{\$} \mathcal{M}^c_{m,n}$. Output $\mathsf{pp} = A$.

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. See the full version [6] ◀

5 Local Compression Schemes

In this section, we introduce *local compression schemes*. 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 11** (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* (Param, Comp, Rec) *of efficient algorithms with the following interface:*

- Param(gp)*: on input global parameters* gp *(assumed to include 1^s), returns public parameters* pp *(assumed to include* gp*).*
- Comp(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 $o(s(n))$.

5.1 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 12.** *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 $\mathsf{cf}(d) \leq 1.01/d$.*

Proof. The proof of each property can be found in the full version [6]. ◀

5.2 Compressing sparse vectors

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

5.2.1 Combinatorial batch codes

At the heart of our construction is the existence of a negligible-error procedure, introduced in [10], for sampling *combinatorial batch codes* [38]. Previous works [17,18,54] 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 13** (Batch Codes [38])**.** *Let w, s, T, N be integers such that $w \leq T < s$, $N = O(s)$. 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 will rely on the following lemma:

▶ **Lemma 14** (CBC with negligible-error [10])**.** *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.*

5.2.2 Efficient bucket selection for CBCs

Lemma 14 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 [25], 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 14:

▶ **Lemma 15** (CBC with negligible error [10], 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* $w^{1+o(1)}$*.*

5.2.3 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. The formal algorithm can be found in the full version in [6].

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 5.1.
- 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 $2td$.
The formal lemma as well as arguments for each property can be found in the full version [6].

6 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) 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 6.1 how to convert any weak PRG in noisy-NC^0 into a strong PRG in noisy-NC^0, in Section 6.2 how to amplify the stretch of a noisy-PRG, and in Section 6.3 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 $\mathcal{N}$.

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

▶ **Theorem 16.** *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. See the full version [6] ◀

▶ **Corollary 17.** *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 $\alpha \in (0,1)$, 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.*

6.2 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 [39]. 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. The detailed construction as well as the security argument can be found in the full version [6].

▶ **Theorem 18.** *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 18, using sparse LPN we get:

▶ **Corollary 19.** *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 $w = \lambda \cdot n^\gamma$, 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 18 and $m^* = \sum_0^{t-1} m^{\theta^{i+1}}$, that stretches an m^{θ^t}-bit seed to m bits.*

6.3 Constructions of Structured-Seed Local PRGs

We present our structured-seed local PRG in Figure 3 and its properties are described in Theorem 20. 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.

▶ **Theorem 20.** *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 $\ell(d) = 2td$. 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.*

The full efficiency and security analysis can be found in the full version [6].

7 Applications

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 [40],

Figure 3: 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 $\mathsf{PRG}^*(\mathsf{pp}^*, \mathsf{seed}) \to \mathbf{y}$.

constant-overhead secure computation [39], sublinear secure computation [23], and hardness of learning [32]. 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 [20], 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. For the analysis of the full range of applications see the full version [6].

References

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

2 Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 – 22, 2012*, pages 805–816. ACM, 2012. doi:10.1145/2213977.2214050.

3 Benny Applebaum. Cryptographic hardness of random local functions – Survey. *Comput. Complex.*, 25(3):667–722, 2016. doi:10.1007/s00037-015-0121-8.

4 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 171–180. ACM, 2010. doi:10.1145/1806689.1806715.

5 Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. *J. Cryptol.*, 29(3):577–596, July 2016. doi:10.1007/s00145-015-9202-8.

6 Benny Applebaum, Dung Bui, Geoffroy Couteau, and Nikolas Melissaris. Structured-seed local pseudorandom generators and their applications. Cryptology ePrint Archive, Report 2024/1027, 2024. URL: https://eprint.iacr.org/2024/1027.

7 Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017 – 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 223–254. Springer, 2017. doi:10.1007/978-3-319-63688-7_8.

8 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *45th Symposium on Foundations of Computer Science, FOCS 2004, Rome, Italy, October 17-19, 2004, Proceedings*, pages 166–175. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.20.

9 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc^0. *Comput. Complex.*, 17(1):38–69, 2008. doi:10.1007/s00037-007-0237-6.

10 Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 171–179. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00020.

11 Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1087–1100. ACM, 2016. doi:10.1145/2897518.2897554.

12 Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography – 14th International Conference, TCC 2016-B, Beijing, China, October 31 – November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 27–56, 2016. doi:10.1007/978-3-662-53641-4_2.

13 Kiril Bangachev, Guy Bresler, Stefan Tiegel, and Vinod Vaikuntanathan. Near-optimal time-sparsity trade-offs for solving noisy linear equations. *CoRR*, abs/2411.12512, 2024. doi:10.48550/arXiv.2411.12512.

14 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 435–440. ACM, 2000. doi:10.1145/335305.335355.

15 Andrej Bogdanov, Pravesh K. Kothari, and Alon Rosen. Public-key encryption, local pseudorandom generators, and the low-degree method. In Guy N. Rothblum and Hoeteck Wee, editors, *Theory of Cryptography – 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 – December 2, 2023, Proceedings, Part I*, volume 14369 of *Lecture Notes in Computer Science*, pages 268–285. Springer, 2023. doi:10.1007/978-3-031-48615-9_10.

16 Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 392–405. Springer, 2009. doi:10.1007/978-3-642-03685-9_30.

17 Maxime Bombar, Dung Bui, Geoffroy Couteau, Alain Couvreur, Clément Ducros, and Sacha Servan-Schreiber. FOLEAGE: F_4ole-based multi-party computation for boolean circuits. *IACR Cryptol. ePrint Arch.*, page 429, 2024. URL: https://eprint.iacr.org/2024/429.

18 Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018. doi:10.1145/3243734.3243868.

19 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022 – 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633. Springer, 2022. doi:10.1007/978-3-031-15979-4_21.

20 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Oblivious transfer with constant computational overhead. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 271–302. Springer, 2023. doi:10.1007/978-3-031-30545-0_10.

21 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1069–1080. IEEE, 2020. doi:10.1109/FOCS46700.2020.00103.

22 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 – November 03, 2017*, pages 2105–2122. ACM, 2017. doi:10.1145/3133956.3134107.

23 Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 159–189. Springer, 2023. doi:10.1007/978-3-031-30617-4_6.

24 Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast public-key silent OT and more from constrained naor-reingold. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024 – 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 88–118. Springer, 2024. doi:10.1007/978-3-031-58751-1_4.

25 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 612–623. IEEE, IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

26 James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. On the one-way function candidate proposed by goldreich. *ACM Trans. Comput. Theory*, 6(3):14:1–14:35, 2014. doi:10.1145/2633602.

27 Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Tauman Kalai, and Vinod Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse LPN. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025 – 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 3–33, Cham, 2025. Springer. doi:10.1007/978-3-031-91124-8_1.

28 Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of goldreich's pseudorandom generator. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018 – 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 96–124. Springer, 2018. doi:10.1007/978-3-030-03329-3_4.

29 Geoffroy Couteau and Naman Kumar. 10-party sublinear secure computation from standard assumptions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024 – 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 39–73. Springer, 2024. doi:10.1007/978-3-031-68400-5_2.

30 Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021 – 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534, virtual, 2021. Springer. doi:10.1007/978-3-030-84252-9_17.

31 Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In Jirí Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, volume 2136 of *Lecture Notes in Computer Science*, pages 272–284. Springer, Springer, 2001. doi:10.1007/3-540-44683-4_24.

32 Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 1358–1394. PMLR, 15–19 August 2021. URL: http://proceedings.mlr.press/v134/daniely21a.html.

33 Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023 – 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 315–348. Springer, 2023. doi:10.1007/978-3-031-38545-2_11.

34 Aurélien Dupin, Pierrick Méaux, and Mélissa Rossi. On the algebraic immunity – resiliency trade-off, implications for goldreich's pseudorandom generator. *Des. Codes Cryptogr.*, 91(9):3035–3079, 2023. doi:10.1007/s10623-023-01220-w.

35 Uriel Feige. Relations between average case complexity and approximation complexity. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 534–543. ACM, 2002. doi:10.1145/509907.509985.

36 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006, Berkeley, California, USA, October 21-24, 2006, Proceedings*, pages 563–574. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.51.

37 Oded Goldreich. Candidate one-way functions based on expander graphs. *Electron. Colloquium Comput. Complex.*, TR00-090, 2000. URL: https://eccc.weizmann.ac.il/eccc-reports/2000/TR00-090/index.html.

38 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 262–271. ACM, 2004. doi:10.1145/1007352.1007396.

39 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442. ACM, 2008. doi:10.1145/1374376.1374438.

40 Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. doi:10.1145/3406325.3451093.

41 Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022 – 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 – June 3, 2022, Proceedings, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699. Springer, 2022. doi:10.1007/978-3-031-06944-4_23.

42 Daniel M. Kane, Anthony Ostuni, and Kewen Wu. Locality bounds for sampling hamming slices. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1279–1286. ACM, 2024. doi:10.1145/3618260.3649670.

43 Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography – 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 119–137. Springer, 2017. doi:10.1007/978-3-319-70500-2_5.

44 Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11538462_32.

45 Pierrick Méaux. On the fast algebraic immunity of threshold functions. *Cryptogr. Commun.*, 13(5):741–762, 2021. doi:10.1007/s12095-021-00505-y.

46 Pierrick Méaux. On the algebraic immunity of direct sum constructions. *Discret. Appl. Math.*, 320:223–234, 2022. doi:10.1016/j.dam.2022.05.021.

47 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th Symposium on Foundations of Computer Science, FOCS 2003, Cambridge, MA, USA, October 11-14, 2003, Proceedings*, pages 136–145. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238188.

48 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 213–223. ACM, 1990. doi:10.1145/100216.100244.

49 Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 1–12. IEEE, IEEE Computer Society, 2014. doi:10.1109/CCC.2014.9.

50 Igor Carboni Oliveira, Rahul Santhanam, and Roei Tell. Expander-based cryptography meets natural proofs. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.18.

51 Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory*, 8(5):5–9, 1962. doi:10.1109/TIT.1962.1057777.

52 Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography – 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part IV*, volume 15367 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2024. doi:10.1007/978-3-031-78023-3_1.

53 Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. *IACR Cryptol. ePrint Arch.*, page 856, 2024. URL: https://eprint.iacr.org/2024/856.

54 Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-ole: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1055–1072. ACM, 2019. doi:10.1145/3319535.3363228.

55 Akin Ünal. Worst-case subexponential attacks on prgs of constant degree or constant locality. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023 – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 25–54. Springer, 2023. doi:10.1007/978-3-031-30545-0_2.

56 Emanuele Viola. The complexity of distributions. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA, October 23-26, 2010*, pages 202–211. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.27.

57 Jing Yang, Qian Guo, Thomas Johansson, and Michael Lentmaier. Revisiting the concrete security of goldreich's pseudorandom generator. *IEEE Trans. Inf. Theory*, 68(2):1329–1354, 2022. doi:10.1109/TIT.2021.3128315.

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