Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 11 (2019)

Sampling Graphs without Forbidden Subgraphs and Unbalanced Expanders with Negligible Error*

Benny Applebaum† Eliran Kachlon†

Abstract

Suppose that you wish to sample a random graph G over n vertices and m edges conditioned on the event that G does not contain a "small" t-size graph H (e.g., clique) as a subgraph. Assuming that most such graphs are H-free, the problem can be solved by a simple rejected-sampling algorithm (that tests for t-cliques) with an expected running time of $n^{O(t)}$. Is it possible to solve the problem in running time that does not grow polynomially with n^t?

In this paper, we introduce the general problem of sampling a "random looking" graph G with a given edge density that avoids some arbitrary predefined t-size subgraph H. As our main result, we show that the problem is solvable with respect to some specially crafted k-wise independent distribution over graphs. That is, we design a sampling algorithm for k-wise independent graphs that supports efficient testing for subgraph-freeness in time $f(t)\cdot n^c$ where f is a function of t and the constant c in the exponent is independent of t. Our solution extends to the case where both G and H are d-uniform hypergraphs.

We use these algorithms to obtain the first probabilistic construction of constant-degree polynomially-unbalanced expander graphs whose failure probability is *negligible* in n (i.e., $n^{-\omega(1)}$). In particular, given constants $d > c \geq 1$, we output a bipartite graph that has n left nodes, n^c right nodes with right-degree of d so that any right set of size at most $n^{\Omega(1)}$ expands by factor of $\Omega(d)$. This result is extended to the setting of unique expansion as well.

We observe that such a negligible-error construction can be employed in many useful settings, and present applications in coding theory (batch codes and LDPC codes), pseudorandomness (low-bias generators and randomness extractors) and cryptography. Notably, we show that our constructions yield a collection of polynomial-stretch locally-computable cryptographic pseudorandom generators based on Goldreich's one-wayness assumption resolving a central open problem in the area of parallel-time cryptography (e.g., Applebaum-Ishai-Kushilevitz, FOCS 2004; and Ishai-Kushilevitz-Ostrovsky-Sahai, STOC 2008).

*A preliminary version of this work appeared in FOCS 2019. The journal version appears in SIAM J. Comput. 2023.

†Tel-Aviv University, {benny.applebaum, elirn.chalon}@gmail.com. Supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under grant agreement no. 639813 ERC-CLC, and the Check Point Institute for Information Security.

ISSN 1433-8092

1 Introduction

Many combinatorial properties of graphs and hypergraphs can be formulated as avoiding some family $\mathcal{H}$ of small subgraphs. Notable examples consist of graphs that avoid short cycles or small cliques, expander graphs (that avoid small non-expanding subgraphs) and even graphical representations of good error-correcting codes (that avoid small "stopping sets" [18]). Motivated by the wide range of applications, the computational problem of efficiently constructing $\mathcal{H}$-free graphs has attracted a huge amount of research (e.g., [41, 26, 17, 1, 14]). In this paper, we consider several natural probabilistic variants of the construction problem.

Setup. Let $\mathcal{G}_{n,m,d}$ be the set of all (n, m, d)-*hypergraphs*, i.e., d-uniform hypergraphs over n vertices with m hyperedges. We typically think of d as a constant that does not grow with n and take $m = \text{poly}(n)$. Let $\mathcal{H}$ be a family of "small" d-uniform hypergraphs of size at most t for some slowly growing function $t(n)$. While our setup is defined with respect to hypergraphs (to match our applications), the following problems make sense even for simple undirected graphs (i.e., $d = 2$) and so, for now, the reader may safely focus on this special case. (Indeed, we are not aware of prior solutions that handle the case of simple graphs.)

Problem 1.1 (Zero-error/negligible-error constructions)**.** *Generate an $\mathcal{H}$-free hypergraph $G \in \mathcal{G}_{n,m,d}$ in probabilistic* poly(n)*-time. The algorithm is allowed to fail with a* negligible *error probability that vanishes faster than any inverse polynomial, i.e., $n^{-\omega(1)}$. Such a construction is referred to as a* negligible-error construction*. We say that this is a* zero-error *(or* **ZPP***) construction if the algorithm outputs a special failure symbol whenever it fails to find an $\mathcal{H}$-free graph.*

Unlike the classical de-randomization literature which typically emphasizes the distinction between *deterministic* and *probabilistic* construction, in Problem 1.1 we focus on the *error level*. We advocate the use of negligible-error constructions as a second-best alternative when explicit constructions are unknown. Indeed, for many applications a randomized construction that almost never fails is almost as good as a fully explicit construction. In particular, if one is planning to plug-in G into some randomized algorithm or system then a negligible error in the construction of G will be swallowed by the overall error probability of the algorithm.[1]

Following the standard cryptographic tradition, we insist on an error that is negligible (i.e., tends to zero faster than any inverse polynomial), in order to guarantee a tiny failure probability even after polynomially-many repetitions.[2] Throughout the paper, we typically assume that an α-fraction of all (n, m, d)-hypergraphs are $\mathcal{H}$-free, where the density α is large but not overwhelming, i.e., $\alpha(n) = 1 - n^{-c}$ for some constant $c > 0$. In this case, the problem is non-trivial when testing $\mathcal{H}$-freeness cannot be done in polynomial-time.

While Problem 1.1 is a relaxation of the explicit-construction problem, our next problem addresses the harder task of generating a random, or pseudorandom, $\mathcal{H}$-free graph.

Problem 1.2 (Quasi-random $\mathcal{H}$-free graphs)**.** *Sample in expected probabilistic* poly(n) *time a random graph G from some "pseudorandom" distribution over $\mathcal{G}_{n,m,d}$* conditioned *on being $\mathcal{H}$-free.*

The general task of generating a pseudorandom object that always satisfies some given property was first studied by Goldreich, Goldwasser and Nussboim [24].[3] In this setting the property (i.e., $\mathcal{H}$-freeness) is viewed as a necessary *worst-case* requirement that should be satisfied by *any* sampled hypergraph G. Using the terminology of [24], the *implementation* G must be *truthful* to the $\mathcal{H}$-freeness *specification* (i.e., G must be H-free *with probability* 1). Conditioned on this, G should be distributed uniformly or close to uniformly under some metric.

[1] This view is implicitly used in other contexts. For example, although the problem of deterministically generating n-bit primes in poly(n)-time is wide open, there are randomized algorithms that generate such primes with negligible (or even zero) error probability. Consequently, applications which employ prime numbers (or prime-order finite fields) rely on negligible-error constructions.

[2] Observe that in our context it is not clear how to reduce the error probability from constant or even inverse polynomial $1/n^{\Omega(1)}$ to negligible.

[3] The work of [24] focuses on *huge* exponential-size random objects. However, the problem remains non-trivial even for polynomial-size objects as long as the required property cannot be tested in polynomial-time. See Section 2.2 for further discussion regarding the applicability of our results to settings of [24].

This combination of requirements arises when one tries to understand the behavior of an $\mathcal{H}$-free random system (e.g., in simulation) or when the hypergraph G is being used as part of a system whose analysis relies on a random choice of G and, in addition, its validity depends on $\mathcal{H}$-freeness. In such a case we cannot use a single explicit construction of $\mathcal{H}$-free hypergraphs since it may fail to achieve some other property of pseudorandom hypergraphs. On the other hand, we cannot use a random sample from $\mathcal{G}_{n,m,d}$ since it fails to be $\mathcal{H}$-free with *positive* (in our case, inverse polynomial) probability.

We further mention that in some cases even a tiny positive failure probability can be problematic. This is the case, for example, when the sampling procedure is invoked by an untrusted party who can benefit from the existence of $\mathcal{H}$-subgraphs. If our sampling algorithm has a positive failure probability, then a cheating party can cheat by selecting "bad coins" that lead to hypergraphs with $\mathcal{H}$-subgraphs. Since general subgraph-testing seems to be computationally-hard such a cheating may be left undetected.[4]

The testing barrier. A natural way to sample $\mathcal{H}$-free random hypergraphs is via rejected sampling. That is, repeatedly sample G until an $\mathcal{H}$-free hypergraph is chosen. Since we work in a regime where most hypergraphs are $\mathcal{H}$-free, the expected number of iterations will be constant. This approach reduces the sampling problem to the subgraph testing problem. If the largest hypergraph in $\mathcal{H}$ is of constant size t, then the problem can be trivially solved in time $f(t)n^{O(t)}$. However, we think of t as a large constant, or as a slowly increasing function of n, and so we would like to have a running time of $f(t)n^c$ where the exponent c is independent of t. Unfortunately, such a running time cannot be achieved for general subgraph-testing (even for simple cases such as cliques) unless the exponential-time hypothesis (ETH) fails (see, e.g., [19]). We refer to this hardness-of-testing as the *testing barrier*. Jumping ahead, we will show that some variant of this barrier arises if one tries to sample a hypergraph that is *uniformly* distributed over all $\mathcal{H}$-free hypergraph in $\mathcal{G}_{n,m,d}$.

Summary: The problem of constructing $\mathcal{H}$-free hypergraphs can be roughly ranked from easy to hard as follows: negligible-error constructions, zero-error constructions, explicit constructions, pseudorandom constructions.

2 Our Results

We partially resolve Problems 1.1 and 1.2. Our results consist of two main parts. We begin by studying pseudorandom constructions of $\mathcal{H}$-free hypergraphs (Sections 2.1 and 2.2). Then we focus on the concrete case of unbalanced expanders, describe negligible-error constructions of such graphs (Section 2.3), and use them to derive various applications (Section 2.4).

2.1 k-Wise Independent Graphs Conditioned on H-Freeness

We show that Problem 1.2 can be solved with respect to some *k-wise independent* distribution over $\mathcal{G}_{n,m,d}$. Here k-wise independence means that every k-subset of the hyperedges, $e_1, \ldots, e_m$, is distributed uniformly over all k-tuples of d-uniform hyperedges. The use of k-wise independent distributions as a good model for pseudorandom graphs was advocated by Naor, Nussboim and Tromer [45] and by Alon and Nussboim [2]. These works further show that a large family of natural graph-theoretic properties that hold with high probability over random graphs (with a given edge density) also hold with high probability over polylog(n)-wise independent distributions with the same density.

We bypass the "testing barrier" by designing a concrete k-wise independent probability distribution $\mathcal{G}_{n,m,d,k}$ in a way that allows us to efficiently test whether a given sample G is H-free. That is, our distribution is amenable to subgraph testing *by design*. To formalize this strategy, we introduce a new notion of *sampler/tester* pair of algorithms. Roughly speaking, the sampler S samples an object according to some

[4]The work of [13] provides a good example for such a case in the context of *financial derivatives*.

given distribution D, and the tester T examines the *coins* of the sampler and checks whether the corresponding object avoids some *bad* event E. The combination of the two allows us to sample the conditional distribution $[D|\neg E]$. (See Section 5 for more details on the sampler/tester framework.)

We prove the following key theorem. Below we define the log-density of an (n, m, d) hypergraph as $c = \log_n m$, and define the size of a hypergraph as the sum of its vertices and hyperedges.

Theorem 2.1 (key theorem)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d \geq 2$, subgraph-size function $t(n) \leq O(\frac{\log\log\log n}{\log\log\log\log n})$ and independence parameter $k(n)$ that satisfies $k(n) \leq O(n^{1/t^{c't}})$ where c' is a constant that depends on c, there exists a* $\mathrm{poly}(n)$*-time sampler/tester pair (S, T) with the following properties:*

- *Given 1^n, the randomized sampler S uses its internal random coins r to sample an $(n, m = n^c, d)$ hypergraph G_r from a $k(n)$-wise independent distribution.*
- *The deterministic tester takes as input a d-uniform hypergraph H of size at most $t(n)$ and a fixed sequence of coin tosses r and accepts the input if and only if H is a subgraph of the hypergraph $G_r = S(1^n, r)$ that is generated by S using coin tosses r.*

Although the size t of the tested subgraph is relatively small, it is still *super-constant*. This property will be crucial for our applications. We further note that the independence parameter $k(n)$ is super-logarithmic (or even "almost" polynomial) in n and so the pseudorandomness properties established by [45, 2] hold. (See Theorem 6.4 for a more detailed version of Theorem 2.1.)

Sampling $\mathcal{H}$-free graphs. It is important to note that our sampler S is independent of the subgraph H, and that the tester T gets H as an input. These properties allow us to partially solve the sampling problem (Problem 1.2) with respect to a *family* of small hypergraphs $\mathcal{H}$. Indeed, we can use the sampler S to sample an (n, m, d)-hypergraph G from a k-wise independent distribution, and use the basic tester to test that G is H-free for all subgraphs $H \in \mathcal{H}$. If one of the tests fails, we repeat the process from the beginning. Since $\mathcal{H}$ contains at most $t^{t^d} < \mathrm{poly}(n)$ hypergraphs, the expected running time will be polynomial, assuming that a random k-wise independent (n, m, d)-hypergraph is $\mathcal{H}$-free with noticeable probability.

Corollary 2.2 (pseudorandom $\mathcal{H}$-free hypergraphs)**.** *Let $c, d, t(n), k(n)$ and $m = n^c$ be as in Theorem 2.1. Let $\mathcal{H}$ be an efficiently constructible family of hypergraphs each of size at most $t(n)$ such that ($\star$) every $k(n)$-wise independent (n, m, d)-hypergraph is $\mathcal{H}$-free with noticeable probability of* $1/\mathrm{poly}(n)$*. Then, there exists a probabilistic algorithm that runs in expected* $\mathrm{poly}(n)$*-time and samples an $\mathcal{H}$-free hypergraph from some k-wise independent distribution over (n, m, d)-hypergraphs.*

It is useful to note that, when $t(n) \leq k(n)$, in order to satisfy the $(\star)$ condition it suffices to show that the expected number of copies of $\mathcal{H}$-subgraphs in a uniformly chosen graph is at most $1 - 1/\mathrm{poly}(n)$. Indeed, for any k-wise independent distribution $\mathcal{G}_{n,m,d,k}$ the first-moment method implies $\Pr[N_{\mathcal{H}}(\mathcal{G}_{n,m,d,k}) \neq 0] \leq \mathbb{E}[N_{\mathcal{H}}(\mathcal{G}_{n,m,d,k})]$, where $N_{\mathcal{H}}(G)$ is the number of copies of $\mathcal{H}$-subgraphs in G. Furthermore, it is not hard to verify that $t(n) \leq k(n)$ implies $\mathbb{E}[N_{\mathcal{H}}(\mathcal{G}_{n,m,d,k})] = \mathbb{E}[N_{\mathcal{H}}(\mathcal{G}_{n,m,d})]$, where $\mathcal{G}_{n,m,d}$ is the uniform distribution over (n, m, d)-graphs. Therefore $\Pr[N_{\mathcal{H}}(\mathcal{G}_{n,m,d,k}) \neq 0] \leq \mathbb{E}[N_{\mathcal{H}}(\mathcal{G}_{n,m,d})]$.

Remark 2.3 (Hardness of uniformly sampling $\mathcal{H}$-free hypergraphs)**.** *It is natural to try and* uniformly *sample an $\mathcal{H}$-free (n, m, d)-hypergraph, i.e., to replace the k-wise independent distribution in Corollary 2.2 with the uniform distribution over $\mathcal{G}_{n,m,d}$. We conjecture that sampling $\mathcal{H}$-free hypergraphs from the uniform distribution over $\mathcal{G}_{n,m,d}$ is computationally infeasible and present some evidence towards this conjecture. In particular, suppose that:*

> *(H) For some families of hypergraphs, it is infeasible to* certify *$\mathcal{H}$-freeness over the uniform distribution. That is, there is no 1-sided error tester that accepts most (n, m, d)-hypergraphs and never accepts a hypergraph that is not $\mathcal{H}$-free.*

We show that if one can efficiently sample $\mathcal{H}$-free hypergraphs from the uniform distribution, then hypothesis H implies the existence of one-way functions*. Basing one-way functions on "hardness of certification" type assumptions may be considered a breakthrough result in the foundations of cryptography, and so we interpret this as an evidence against the existence of an efficient sampler for uniform $\mathcal{H}$-free hypergraphs. Put differently, a sampler would allow us to convert average-case hardness (of testing) to one-wayness, or, in the language of Impagliazzo [30], to move from Pessiland to Minicrypt.*

The H assumption (hardness of certifying $\mathcal{H}$-freeness) is closely related to previous intractability assumptions (cf. [7, 13, 9]). We further relate this assumption to the problem of certifying that a random low-density parity-check code has a high distance. (See Appendix A for details.)

Remark 2.4 (On k-wise independence)**.** *It is instructive to note that Theorem 2.1 employs k-wise independence in an unconventional way. Typically, the notion of k-wise independence is useful due to the combination of pseudorandomness with computationally-cheap and randomness-efficient implementations. In contrast, the proof of Theorem 2.1 exploits the simple algebraic structure of k-wise independence constructions to force a structure on the sampled object (the hypergraph G) in a way that makes it amenable to efficient analysis (i.e., subgraph testing). The fact that such implementation is computationally-cheap or randomness-efficient is not really needed for our main applications.*

ZPP and explicit constructions. Corollary 2.2 immediately leads to a **ZPP**-construction of $\mathcal{H}$-free hypergraphs. We further observe that, under standard worst-case de-randomization assumptions, any **ZPP**-construction implies an explicit construction.

Corollary 2.5 (explicit $\mathcal{H}$-free hypergraphs)**.** *Let $c, d, t(n), k(n)$ and $m = n^c$ and $\mathcal{H}$ be as in Corollary 2.2. Assuming that the class of functions computable in $2^{O(n)}$ uniform-time requires $2^{\Omega(n)}$-size circuits, there exists a deterministic* $\mathrm{poly}(n)$*-time algorithm that always outputs an $\mathcal{H}$-free (n, m, d)-hypergraph.*

The above assumption is known to imply, for any constant a, a pseudorandom generator prg that fools n^a-time algorithms with logarithmic size seed [31]. Such a generator can be used to fully de-randomize the **ZPP** construction A and derive a fully explicit construction A'. (The algorithm A' just outputs the first seed s for which $A(\mathsf{prg}(s))$ does not output "failure".) This makes a crucial use of the ability to recognize bad outputs (which trivially holds for **ZPP** samplers). We are not aware of a similar transformation that applies to "Monte-Carlo" constructions that have a positive failure probability.[5]

2.2 The Succinct Setting

So far we assumed that the computational complexity of the sampler is allowed to grow polynomially in the size of the hypergraph G. In some scenarios, it is more natural to think of the hypergraph as a huge object, and require a running time that is polynomial in $\log n$. In particular, we say that an (n, m, d) hypergraph G has a succinct representation if it can be described by a short binary string z of length $\mathrm{polylog}(n)$ such that given z, a hyperedge $e \in [m]$, and an index $i \in [d]$, it is possible to compute the i-th member of the hyperedge e in time $\mathrm{polylog}(n)$.[6] (Here we assume that the hyperedges are ordered and can be represented by d-tuples.) Our goal is to obtain a $\mathrm{polylog}(n)$-time sampler, that samples a succinct representation z of an (n, m, d) hypergraph G from a k-wise independent distribution, as well as a $\mathrm{polylog}(n)$-time tester that

[5] There are cases in which derandomization assumptions can be easily used to turn a negligible-error construction into an explicit construction [36]. This typically happens when one of the following holds: (1) It is "easy" to recognize a "bad" object (i.e., to detect a violation of the desired property) in polynomial-time; or (2) There is an efficient way to combine a "bad" instance with several "good" instances into a single "good" instance. As far as we know, in general, both conditions fail for $\mathcal{H}$-freeness.

[6] This is in contrast to the (more common) notion of succinctness (in the context of standard graphs), where, given a vertex v and an index i, we can compute the i-th neighbor of v in time $\mathrm{poly}\log n$ (this notion is also called *fully-explicit*). Our notion of succinctness is better suited for our applications, in which a hypergraph represents the dependencies graph of some function $f : \{0,1\}^n \to \{0,1\}^m$ (e.g., low-biased generator) where inputs correspond to vertices, outputs correspond to hyperedges, and the i-th hyperedge contains the vertices on which the i-th output depends. Our notion of succinctness guarantees that each output of f can be computed in $\mathrm{polylog}(n)$-time (e.g., in the RAM model).

can test for small subgraphs in G. We prove a succinct version of Theorem 2.1 that applies to constant-size subgraphs H and polylog(n)-wise independence.

Theorem 2.6. *For every log-density parameter $c > 1$, edge-uniformity parameter $d \geq 2$, constant subgraph size t and independence parameter $k(n) \leq$ polylog(n), there exists a* polylog(n)*-time sampler/tester pair (S, T) with the following properties:*

- *Given n (in binary representation), the randomized sampler uses its internal random coins r to sample a succinct $(n, m = n^c, d)$ hypergraph G_r from a $k(n)$-wise independent distribution.*
- *The deterministic tester takes as input a d-uniform hypergraph H of size at most t and a fixed sequence of coin tosses r and accepts the input if and only if H is a subgraph of the hypergraph $G_r = S(n, r)$ that is generated by S using coin tosses r.*

Theorem 2.6 leads to the following succinct version of Corollary 2.2.

Corollary 2.7 (pseudorandom $\mathcal{H}$-free succinct hypergraphs)**.** *Let $c, d, t, k(n)$ and $m = n^c$ be as in Theorem 2.6. Let $\mathcal{H}$ be a family of hypergraphs each of size at most t, such that every $k(n)$-wise independent (n, m, d)-hypergraph is $\mathcal{H}$-free with probability of* $1/\text{polylog}(n)$*. Then, there exists a probabilistic algorithm that runs in expected* polylog(n)*-time and samples a succinct $\mathcal{H}$-free hypergraph from some k-wise independent distribution over (n, m, d)-hypergraphs.*

As already mentioned the problem of constructing huge k-wise independent graphs that satisfy some given property (specification) was studied in [24, 2, 45, 44]. Corollary 2.7 provides a zero-error (aka "truthful") solution for this problem with respect to $\mathcal{H}$-free hypergraphs of given density. To the best of our knowledge, prior to our work no solution was known even for the case of undirected graphs and concrete fixed-size forbidden subgraphs.

2.3 Negligible-Error Construction of Constant-Degree Unbalanced Expanders

We move back to the non-succinct setting, and consider the problem of explicitly constructing a single, not necessarily random, *expander* graph. We say that an (n, m, d)-hypergraph is an (α, t)-expander if every set S of hyperedges of size at most t "touches" at least $\alpha|S|$ vertices.[7] Equivalently, an (α, t)-expander is an (n, m, d)-hypergraph that avoids small "dense" subgraphs, i.e., (n', m', d)-hypergraphs with $n' \leq \alpha m'$ for $m' \leq t$.

We focus on the setting of *constant-degree highly unbalanced* expanders. That is, we let d be a constant, and assume that the number of hyperedges m is polynomially larger than n, i.e., $m = n^c$ for some constant log-density $1 < c < d$. A standard probabilistic calculation shows that in this regime most (n, m, d)-hypergraphs achieve a good expansion factor of $\alpha = \Omega(d)$ (or even $\alpha = d - O(1)$) for polynomially-small subsets of size at most $t = n^{1-\delta}$ where δ is a constant that depends on α, c and d. Unfortunately, the problem of *efficiently constructing* highly-unbalanced constant-degree expanders is wide open. Existing constructions either provide only slightly unbalanced expanders (that have only linearly many hyperedges $m = O(n)$) [17] or suffer from a super-constant (actually polylogarithmic) degree [26].[8] In fact, it is not even known how to achieve a negligible-error construction. The main problem with natural strategies such as random sampling is that small sets fail to expand with non-negligible probability. Motivated by the numerous applications of constant-degree highly-unbalanced expanders (to be discussed later), we present a negligible-error construction of such graphs.

[7] This formulation is equivalent to the more standard notion of bipartite expanders over n left vertices and m right vertices where the degree of each right vertex is d, and every set S of right vertices of size at most t is connected to at least $\alpha|S|$ left vertices.

[8] Insisting on both constant degree and polynomially-unbalanced hypergraph, the work of [55] implicitly yields an expander with a weak, yet non-trivial, expansion factor $\alpha < 1$ that decreases polynomially with n. We thank the anonymous referee for pointing this out.

We begin by providing a **ZPP**-construction of constant-degree highly-unbalanced hypergraphs that expand well for small sets of super-constant size. The following theorem follows from Corollary 2.2 by instantiating the class $\mathcal{H}$ of forbidden subgraphs with the class of small non-expanding hypergraphs. (See Corollary 7.6 in Section 7.2.)

Theorem 2.8 (**ZPP**-construction of small-set expanders)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d > c$, and $\alpha < d - c$ there exists a* **ZPP***-construction of $(n, m = n^c, d)$-hypergraph with (α, t)-expansion where $t = O(\frac{\log\log\log n}{\log\log\log\log n})$.*

Next, we show that Theorem 2.8 gives rise to a negligible-error construction of hypergraphs that expand well for polynomial-size subsets. That is, we downgrade the level of explicitness (from zero-error construction to negligible-error construction) and upgrade the expansion threshold t to polynomial.

Theorem 2.9 (negligible-error construction of unbalanced expanders)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d > c$, and $\alpha < d - c$, there exists a negligible-error construction of $(n, m = n^c, 2d)$-hypergraph with (α, t)-expansion where $t = \Omega(n^{1-\delta})$ and $\delta = (c-1)/(d-\alpha-1)$.*

Recall that a negligible-error construction guarantees the existence of a poly(n)-time randomized algorithm that outputs, except with negligible probability of $n^{-\omega(1)}$, an (α, t)-expanding $(n, m, 2d)$-hypergraph. (See Theorem 7.15 for a more detailed version.) Unfortunately, the negligible function that we obtain decreases relatively slowly at a rate of $n^{-\Omega(t)}$ where $t = \frac{\log\log\log n}{\log\log\log\log n}$ is the expansion parameter from Theorem 2.8. This caveat is common to all our negligible-error constructions. We emphasize that this limitation follows from the concrete parameters of our explicit $\mathcal{H}$-free hypergraphs and we hope that future instantiations of our framework would lead to better parameters and to smaller error rates.

Theorem 2.9 provides an $(n, m, D = 2d)$-hypergraph whose expansion parameters (α, t) match the parameters of a random (n, m, d)-hypergraph. While this factor-2 gap in the degree has a relatively minor effect on the expansion threshold t (which is still polynomial in n), it limits the expansion factor α to be at most $D/2 - O(1)$. Such an expansion factor suffices for many applications, but in some cases it is useful to expand by a factor larger than $D/2$. Notably, expansion beyond half the degree guarantees the useful *unique expansion* property.

Constructing unique expanders. A hypergraph is a (β, t)-*unique expander* if for every set S of at most t hyperedges there exists a set U of at least $\beta|S|$ vertices such that each vertex in U appears in a unique hyperedge e in S. Perhaps surprisingly, although we cannot expand by a factor better than $D/2$, we can still get a negligible-error construction of unique expanders.

Theorem 2.10 (negligible-error construction of unbalanced unique-expanders)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d > 2c$, and $\beta < d - 2c$, there exists a negligible-error construction of $(n, m = \Omega(n^c), 2d)$-hypergraph with (β, t) unique-expansion where $t = \Omega(n^{1-\delta})$ and $\delta = 2(c-1)/(d-\beta-2)$.*

Theorems 2.8, 2.9 and 2.10 (whose proofs appear in Section 7) provide the first negligible-error constructions of highly-unbalanced constant-degree expanders.

2.4 Applications

In Section 8, we use our negligible-error construction of unbalanced expanders to obtain the first negligible-error constructions of several useful objects including batch codes (Section 8.1), and locally-computable k-wise independent generators, low-bias generators and randomness extractors (Section 8.3). These applications follow immediately from our expanders via standard techniques. Below we briefly describe two non-trivial applications: high-rate low-density parity-check (LDPC) codes (Section 8.2), and locally-computable cryptographic pseudorandom generators (PRGs) with polynomial stretch (Section 9).

2.4.1 High-Rate LDPC Codes

An $[m, k]$-code is a linear code with codewords of length m and information words of length k. Such a code can be always represented by an $(m-k) \times m$ parity check matrix. LDPC codes [21] (see also [39, 49, 50] and [48]) can be represented by a *sparse* parity check matrix that contains only dm non-zero entries for some *sparsity constant* $d = O(1)$. Any (n, m, d)-hypergraph G defines an $[m, m-n]$-binary LDPC by letting the parity-check matrix be the $n \times m$ incidence matrix of G. The parity-check matrix has md ones, and is therefore sparse when $d = O(1)$. Moreover, it is well known that if, for some $\beta > 0$, the hypergraph G achieves unique-neighbor expansion of (β, γ) then the resulting code has a distance of γ.

LDPC codes have numerous applications and accordingly, such codes have been extensively studied from various perspectives (see, e.g., [48] and references therein). Here we focus on the problem of providing a negligible-error construction of such codes in the somewhat non-standard regime where the code's rate is very close to one (up to an inverse polynomial) and the relative distance is small but non-trivial (inverse polynomial).[9] Specifically, Theorem 2.10 leads to the first negligible-error construction of such codes: For every constants $1 < c < d/2$ we get an LDPC with sparsity $2d$ that maps k bits of information into $k + O(k^{1/c})$-bit codeword with a distance of $n^{1-O(c/d)}$.

Sipser and Spielman showed that an LDPC code whose underlying graph has a very good expansion factor (well beyond half the degree) can be efficiently decoded by a linear time decoding algorithm with $O(\log n)$ parallel steps [49]. Unfortunately, the hypergraph given by Theorem 2.10 does not satisfy such a strong expansion property. Nevertheless, we show that our construction can be tweaked in a way that still allows for highly efficient decoding via a variant of the Sipser-Spielman decoder. In particular, we prove the following theorem. (See also Theorem 8.3.)

Theorem 2.11. *For every constant $c > 1$, integer $d > 10c$ and constant $0.9d < \alpha < d - c$, there exists a negligible-error construction of an $[m, m - 2m^{1/c}]$-LDPC code with sparsity of $2d$ that admits a decoder that runs in quasi-linear time $O(m \log^2 m)$ and $O(\log m)$ parallel steps and corrects up to $\Omega(n^{1-\delta})$ errors where $\delta = (c-1)/(d-\alpha-1)$.*

While the parameters of our LDPC code are somewhat not standard, it will be extremely useful as a building block in our next application of *polynomial-stretch locally computable PRGs*.

2.4.2 Polynomial-Stretch Locally-Computable PRGs

A cryptographic pseudorandom generator stretches a short random n-bit seed into a longer m-bit pseudorandom string that is computationally indistinguishable from a truly random string. We say that a PRG is locally-computable if each of its output bits depends on at most $d = O(1)$ input bits. Locally-computable PRGs were extensively studied in the past two decades. In particular, locally-computable PRGs that *polynomially* stretch their input (i.e., $m = n^c$ for $c > 1$) have shown to have remarkable applications. This includes secure-computation with constant computational overhead [34, 8] and general-purpose obfuscation based on constant-degree multilinear maps (cf. [37, 38]).

Unfortunately, constructing locally-computable PRGs with polynomial-stretch turns out to be a challenging task. Indeed, while there are good constructions of local PRGs with sub-linear stretch $m = n + o(n)$ [10], and even linear stretch $m = n + \Omega(n)$ [11, 4, 6] under standard assumptions, we currently have only partial solutions to the polynomial-stretch regime. In particular, in [4] the first author constructed a locally-computable polynomial-stretch *weak-PRG*. Here *weak* means that the distinguishing advantage ε of any polynomial-time adversary is upper-bounded by some fixed inverse polynomial $1/\mathrm{poly}(n)$, whereas the standard cryptographic definition requires a negligible distinguishing advantage of $n^{-\omega(1)}$. The construction of [4]

[9] The status of existing explicit/negligible-error constructions is the same as the status of unbalanced expanders. In fact, any $[m, m-n, t]$ LDPC with sparsity md implies an (n, m)-hypergraph with average rank of d such that any set of t hyperedges has some "odd-expansion" property. We do not have better ways to construct such expanders compared to standard expanders. The situation is similar for all the applications discussed in this paper. That is, unbalanced constant-degree hypergraphs with some expansion property for polynomial-size subsets of hyperedges are also necessary for all these applications, and accordingly so far we had no explicit or negligible-error constructions.

is based on the one-wayness of random local functions with polynomially-long output length – a variant of Goldreich's one-wayness assumption [22].

We show that our negligible-error construction of expanders can be used to upgrade any weak-PRG into standard PRG while preserving constant locality and polynomial stretch.

Theorem 2.12 (local-PRG with polynomial-stretch: weak-to-strong)**.** *For every constants $d \in \mathbb{N}, a > 0$ and $c, c' > 1$ there exists a constant d' for which the following holds. Any ensemble of d-local PRGs that stretches n bits to n^c bits and achieves indistinguishability parameter of $\varepsilon = 1/n^a$ can be converted into an ensemble of d'-local (standard) PRGs that stretches n bits to $n^{c'}$ bits.*

The term ensemble here means that, given 1^n, we can sample in polynomial-time a circuit that implements a locally computable function f from n-bits to m bits so that except with negligible probability f is a PRG. This use of ensembles is standard in the context of parallel cryptography and typically has at most a minor effect on the applications.

Combined with the weak-PRG of [4], Theorem 2.12 yields the first construction of local PRG with polynomial stretch based on a one-wayness assumption, resolving an important open question in the theory of parallel cryptography [42, 10, 34, 4]. We mention that there is a second heuristic approach for constructing such pseudorandom generators, due to [34] (see also [42, 12] and the survey [5]). This approach also requires the existence of explicit (or negligible-error) construction of highly-unbalanced constant degree expanders, and one can instantiate it using our constructions as well. In fact, it is known that such expanders are *necessary* for *any* construction of locally-computable PRG with large-stretch [11].[10] Theorem 2.12 shows that, up to some extent, such expanders are also sufficient for this task.

3 Technical Overview

We briefly sketch some of the main techniques.

3.1 Sampler/Tester for H-free hypergraphs

We present a k-wise independent distribution over (n, m, d) hypergraphs that admits efficient subgraph-testing for hypergraphs of size $t = O(\frac{\log\log\log n}{\log\log\log\log n})$ (as in Theorem 2.1). For simplicity let us focus on the case of directed *graphs* ($d = 2$). Let us further assume that the number of vertices n is prime, and that the number of edges m is an integer power of n, i.e., $m = n^c$ for some integer $c \geq 1$. Later, we discuss the case where c is non-integral.

We identify every vertex with an element of the field $\mathbb{F} = \mathrm{GF}(n)$, and index the edges with c-tuples of elements of $\mathbb{F}$. We sample the graph by uniformly sampling a pair (A, B) of c-variate polynomials over $\mathbb{F}$ of total degree k. For every tuple $h = (h_1, \ldots, h_c) \in \mathbb{F}^c$, we define the h-th edge to be $(A(h), B(h))$. That is, h leaves the source vertex $A(h)$ and enters the target vertex $B(h)$.

It is not hard to show that every set of k edges are uniformly distributed. (This follows by a simple extension of the well-known fact that random degree-k univariate polynomials are k-wise independent.) We reduce the problem of subgraph testing to the following polynomial satisfiability problem: Check whether a system of $O(t)$ polynomial equations of degree $D = O(k+t)$ and $O(t)$ variables over the field $\mathbb{F}$ has a solution. The latter problem can be solved by an algorithm of Kayal [35, Theorem 6.1.1] in time $\mathrm{poly}(D^{t^{O(t)}} t \log|\mathbb{F}|)$ which is polynomial in n for our choice of parameters.[11]

[10] Indeed, prior works on expander-based cryptography (cf. [3, 4, 8, 12, 22, 34, 37, 38]) assumed, either explicitly or implicitly, the existence of explicit constant-degree unbalanced vertex-expander, or at least that such expanders can be sampled efficiently with negligible error, even though it was unknown how to do so, see for example [34, Remark 5.7]).

[11] On a high-level, Kayal's algorithm decomposes the algebraic closed set X, defined by the input polynomials, into closed sets X_i, such that each X_i is birational to a hypersurface Y_i. If some Y_i has an absolutely irreducible $\mathbb{F}_q$-factor, then by Weil's theorem there exist rational points in Y_i, and by the birational correspondence also in X_i, so the algorithm outputs "yes". Otherwise, if X_i contains a rational point it has to lie on a closed proper subset of X_i. The subset is computed and the algorithm is applied on it recursively. See [35] for full details.

We describe a simplified version of the reduction for the special case of detecting a directed rectangle (4-cycle). First observe that any sequence of edges indexed by $x_1, x_2, x_3, x_4 \in \mathbb{F}^c$ that form a rectangle must satisfy the system $\mathcal{L}_1$ of equations

$$B(X_1) = A(X_2), \quad B(X_2) = A(X_3) \quad B(X_3) = A(X_4) \quad B(X_4) = A(X_1)$$

where the formal variables X_1, X_2, X_3 and X_4 correspond to indices of edges and so they take values from $\mathbb{F}^c$. However, a moment of inspection suggests that the system $\mathcal{L}_1$ can be also solved by a 2-cycle: Assign the first edge to X_1 and X_3 and the second edge to X_2 and X_4. We therefore need a mechanism for excluding solutions that assign the same value to different variables. Fortunately, this can be achieved by introducing few more auxiliary variables and few more low-degree equations.

In particular, we add four new variables $Y = (Y_0, Y_1, Y_2, Y_3)$ which take values from $\mathbb{F}^c$ and define a new system $\mathcal{L}_2$ of four equations

$$\sum_{j=0}^{3} Y_j X_1^j = 1, \quad \sum_{j=0}^{3} Y_j X_2^j = 2, \quad \sum_{j=0}^{3} Y_j X_3^j = 3, \quad \sum_{j=0}^{3} Y_j X_4^j = 4,$$

where arithmetic is over the extension field $\mathrm{GF}(n^c)$ and 1,2,3,4 represent four distinct constants from this field. Observe that the variables Y define a degree-3 univariate polynomial $P_Y(\cdot)$, and the system is satisfiable if this polynomial evaluates to i over the input X_i. Clearly, any solution to $\mathcal{L}_1$ that assigns non-distinct values to the X variables violates $\mathcal{L}_2$. On the other hand, any solution to $\mathcal{L}_1$ that assigns distinct values to the X variables can be extended by an assignment to Y in a way that satisfies $\mathcal{L}_2$. (Such an assignment can be found via polynomial interpolation). Hence, by combining $\mathcal{L}_2$ with $\mathcal{L}_1$ we get a new system that excludes solutions in which the same edge is being used twice.

To complete the reduction one has to deal with few additional minor technicalities. Firstly, the system $\mathcal{L}_1$ is over the field $\mathbb{F} = \mathrm{GF}(n)$ whereas the second system is over the extension field $\mathrm{GF}(n^c)$. This is solved by projecting down the second system to the base field, and checking the satisfiability of the combined system over $\mathbb{F}$. Secondly, an additional distinctness gadget should be used to further force distinct vertices. (Otherwise, a system for detecting a 4-path will be fooled by a 4-cycle.)

The construction extends to d-uniform hypergraphs in a straightforward way (use d polynomials instead of 2), and to the case of *non-integral* log-density $c = \log_n m$ by working over appropriate extension fields. (See Section 6 for full details.) Finally, observe that the sampled graph has a *succinct* representation: An edge query can be implemented by evaluating a low-degree polynomial. Moreover, for polylogarithmic k and constant t, the polynomial-satisfiability algorithm can be implemented in polylogarithmic time, and so we get a succinct version of the theorem.

3.2 Expanding the Expansion: From Small-Sets to Large Sets

Theorem 2.8 converts a zero-error construction of (n, m, d)-hypergraphs G_1 with (α, t)-expansion for small threshold $t = O(\frac{\log\log\log n}{\log\log\log\log n})$ into a negligible-error construction of (α, T)-expander with polynomial threshold of $T = n^{1-\delta}$. The proof is based on two observations.

First, suppose that, in addition to G_1, we are given an (n, m, d)-hypergraph G_2 with the property that any "medium-size" set S of hyperedges $t \leq |S| \leq T$ expands by a factor of α. Then, we can combine G_1 and G_2 into a single $(n, m, 2d)$-hypergraph G by letting the i-th hyperedge of G be the union of the i-th hyperedge of G_1 and G_2. (Both hyperedges should be viewed as d-size multisets over $[n]$.) Every hyperedge set S of size at most T now expands by a factor of α, either due to the expansion of G_1 (when $|S| \leq t$) or due to the expansion of G_2 (when $t < |S| \leq T$). As a result, the expansion factor remains unchanged, but the overall degree doubles.

The second observation is that a random (n, m, d)-hypergraph forms a negligible-error construction of medium-size expanders. Indeed, in our regime of parameters, the probability that a random (n, m, d)-hypergraph contains an s-tuple of hyperedges that violate expansion (touch less than αs vertices) is $n^{-\Omega(s)}$ which is negligible when $s \geq t > \omega(1)$. (The constants in the big-Omega depend on $d, c = \log_n m$ and

the exponent of the expansion threshold $\log_n T$.) Indeed, the only reason for which a random (n, m, d)-hypergraph does not qualify as a negligible-error expander is the existence of small non-expanding sets which appear with *noticeable* probability.

Unique-expansion. Unique-expansion is achieved via a similar approach except that the merging procedure is slightly different. As before we merge a pair of (n, m, d)-hypergraphs G_1 and G_2 into a single hypergraph G by defining the i-hyperedge of G to be the union of the i-th hyperedge of G_1 and G_2. However, now we treat the vertices of G_1 and the vertices of G_2 as distinct sets. (E.g., the vertices of G_1 are indexed from 1 to n and the vertices of G_2 are indexed by $n + 1$ to $2n$.) As a result, G is a $(2n, m, 2d)$-hypergraph. It is not hard to verify that if G_1 is a (β, t) unique-expander and G_2 expands by β for sets of size $t < s \leq T$, then G is a (β, T) unique-expander.

Coding perspective. Recall that unbalanced hypergraphs can be viewed as parity-check matrices of error-correcting codes where t-weight codewords correspond to "bad" t-size subgraphs (that violate unique expansion). Using this terminology the above transformation defines a code by taking the intersection of the code G_1 (that has no nontrivial codewords of weight smaller than t) with the code G_2 (that has no codewords of weight $s \in [t, T]$). The efficient decoding algorithm presented in Theorem 8.3 further exploits this view, and shows that, in our setting, a noisy codeword of the intersection code G can be decoded by combining the decoders of G_1 and G_2. In particular, the G_2 decoder "reduces" the number of noisy coordinates from T to (roughly) t, and the G_1 decoder further reduces the noise from t to zero.

3.3 Local Hardness Amplification: From weak-PRGs to strong-PRGs

Theorem 2.12 converts a weak-PRG $g : \{0,1\}^n \to \{0,1\}^m$ into a standard PRG while preserving polynomial stretch and constant locality. Such hardness amplification theorems are typically based on a direct sum construction: Apply g on ℓ independent copies of the seed and XOR the results. By Yao's XOR-lemma (cf. [25]), if we start with an inverse polynomial indistinguishability, it suffices to take a super-constant number of copies $\ell = \omega(1)$. Unfortunately, this leads to a super-constant growth in the locality. We therefore take a different approach based on randomness extractors.

We generate polynomially-many pseudorandom strings (using independent seeds) and place them as rows of a $k \times m$ matrix. Since the rows are independent and the indistinguishability parameter is a small inverse polynomial, one can guarantee that each column has an almost full pseudo-entropy of $k - 1/\text{poly}(k)$. Finally, we extract the randomness from each column using randomness extractor. This approach was used by [4] (following a more general transformation from [27]) to obtain a linear-stretch local-PRG.

The success of this approach depends, however, on the existence of a suitable locally-computable randomness extractor. The extractor should take a k-bit source with an almost-full entropy of $k - 1/\text{poly}(k)$ and a polynomially-short random seed of length $k^{1-\varepsilon}$ and output an almost-uniform k-bit string with negligible statistical error. The main new observation is that such extractors exist, and a negligible-error construction can be achieved based on negligible-error construction of highly-unbalanced constant-degree expanders. (Similar connections between expanders and locally-computable extractors were established, for a different regime of parameters, in related contexts [11, 4]). See Sections 8.3 and 9.

4 Preliminaries

4.1 Hypergraphs

An (n, m, d) *oriented hypergraph* $G = (V, E)$ is a hypergraph with a vertex set V of size n (by default, $V = [n]$) that consists of a multiset of m hyperedges E. Although the order of hyperedges is not important for us, it will be convenient to think about E as an m-tuple $(e_1, \ldots, e_m)$. We further assume that each hyperedge e_i is fully oriented and is therefore represented by an ordered d-tuple $(e_i[1], \ldots, e_i[d]) \in V^d$. We refer to e_i as the i-th hyperedge, to $e_i[j]$ as the j-th entry of the i-th hyperedge, and to the index $j \in [d]$

as the location of the vertex $e_i[j]$ in the i-th hyperedge. We allow repetitions both between hyperedges and inside hyperedges. (For example, e_1 may be equal to e_2 and $e_i[1]$ may be equal to $e_i[2]$). When the internal order of the hyperedges is not important (i.e., e_i is a multiset) we refer to G as an non-oriented hypergraph. (By default, all hypergraphs are oriented.)

An $(n, m, \leq d)$ oriented hypergraph is defined similarly except that the size of the hyperedges $E = (e_1, \ldots, e_m)$ can vary as long as it does not exceed d. We still assume that the hyperedges are d-oriented by viewing each hyperedges as a d-tuple over the set $V \cup \{\perp\}$. For example, in an $(n, m, \leq 4)$-hypergraph one can have a hyperedge of arity 3 represented as $(1, 5, \perp, 4)$.

We extend the standard notion of subgraphs to hypergraphs as follows.

Definition 4.1 (sub-hypergraphs)**.** *A pair of d-oriented hyperedges e, e'* match *if for every $i \in [d]$ either $e[i] = \perp$ or $e'[i] = \perp$ or $e[i] = e'[i]$ (That is, null entries are viewed as wild-chars). Let $H = (V(H), E(H))$ be an $(n, m, \leq d)$ oriented hypergraph. For a mapping π from $V(H)$ to some set U we let $\pi(H) = (U, E')$ denote the hypergraph obtained by replacing the i-th hyperedge e_i of H with the hyperedge $e'_i = (\pi(e_i[1]), \ldots, \pi(e_i[d]))$ where π is extended to map $\perp$ to a $\perp$. We say that an $(n', m', \leq d)$ oriented hypergraph H is a* subgraph *of an (n, m, d) oriented hypergraph G if there exists a pair of injective mapping $\pi : V(H) \to V(G)$ and $\sigma : [m'] \to [m]$ such that i-th hyperedge of $\pi(H)$ matches to the $\sigma(i)$-th hyperedge of G. (Note that this implies that $n' \leq n$ and $m' \leq m$.)*

Ensembles of hypergraphs. Let $m(\cdot)$ and $d(\cdot)$ be integer-valued functions. An *efficiently samplable $(n, m(n), d(n))$-hypergraph ensemble* is defined by a probabilistic polynomial-time sampling algorithm S that given 1^n outputs an $(n, m(n), d(n))$-hypergraph using some canonical representation. When S uses only $O(\log n)$ coins (i.e., the distribution is supported on polynomially many hypergraphs), we refer to the corresponding sequence of hypergraphs as an *efficiently constructible family of hypergraphs*. Note that in this case we can construct all hypergraphs in the family in time $\text{poly}(n)$. A deterministic algorithm that outputs a single hypergraph per output length is treated as a special case.

We say that the ensemble is *succinct* if S can be "partitioned" into two algorithms: a probabilistic $\text{polylog}(n)$-time index-sampler I and a deterministic $\text{polylog}(n)$-time edge-evaluation algorithm G with the following syntax.

- Given n, the algorithm I outputs a string z of length $\text{polylog}(n)$ that represents an $(n, m(n), d(n))$-hypergraph G_z.
- Given n, a graph identifier z, an index of a hyperedge $e \in [m(n)]$ and an internal index $i \in [d]$, the evaluation algorithm outputs a vertex $v \in [n]$ that defines the i-th entry of the hyperedge e in G_z.

In some cases we consider an intermediate setting where the evaluation algorithm runs in $\text{polylog}(n)$-time but the index-sampler runs in time $\text{poly}(n)$. In this case we say that the ensemble has a succinct representation (and emphasize the polynomial complexity of the sampler).

4.2 Finite Fields

Let q be a prime power. We will denote the finite field of size q by $\mathbb{F}_q$, and assume that we are given a description of $\mathbb{F}_q$ that enables computation of field operations and sampling of field elements in time $\text{poly}(\log q)$. We will rely on the following standard facts: (1) elements in an extension field $\mathbb{F}'$ of $\mathbb{F}$ can be represented as $\mathbb{F}$-vectors such that arithmetic operations over $\mathbb{F}'$ can be emulated by arithmetic operations over $\mathbb{F}$. (See Fact 4.2.) (2) Correspondingly, a low-degree system of equations over $\mathbb{F}'$ can be represented by a low-degree system of equations over $\mathbb{F}_q$. (See Fact 4.3.)

Fact 4.2. *Let q be a prime power and let c be an integer. Then, elements of the extension field $\mathbb{F}_{q^c}$ can be viewed as vectors of length c over $\mathbb{F}_q$. For two elements of $\mathbb{F}_q^c$, $a = (a_1, \ldots, a_c)$ and $b = (b_1, \ldots, b_c)$, the operations over $\mathbb{F}_{q^c}$ are defined as follows.*

- *Addition: The sum of a and b over $\mathbb{F}_{q^c}$ is simply the entry-wise sum of a and b over $\mathbb{F}_q$.*

- *Multiplication: The product, $z = (z_1, \ldots, z_c)$, of a and b is defined by $z_i = a\Lambda_i b^T$, for every $i \in [c]$, where Λ_i is a fixed $c \times c$ matrix over $\mathbb{F}_q$, and the arithmetic is over $\mathbb{F}_q$.*

Moreover, the matrices $\Lambda_1, \ldots, \Lambda_c$ can be generated in time $\text{poly}(c \cdot \log q)$.

Fact 4.3. *Let q be a prime power and let c be an integer. Then, elements of the extension field $\mathbb{F}_{q^c}$ can be represented as vectors of length c over $\mathbb{F}_q$ such that the following holds. For any m-variate polynomial $f(X_1, \ldots, X_m)$ of total-degree D over $\mathbb{F}_{q^c}$ we can generate in time $\text{poly}(c \cdot D^m \cdot \log q)$ a vector, $(g_1, \ldots, g_c)$, of c polynomials over $\mathbb{F}_q$ each with $c \cdot m$ variables, $((X_{1,i})_{i\in[c]}, \ldots, (X_{m,i})_{i\in[c]})$ and total-degree of at most D such that for any f-input $(x_1, \ldots, x_m)$ the corresponding f-output $y = f(x_1, \ldots, x_c)$ satisfies*

$$(y_1, \ldots, y_c) = (g_1(x_{1,1}, \ldots, x_{m,c}), \ldots, g_c(x_{1,1}, \ldots, x_{m,c})),$$

where $x_{i,j}$ (resp., y_j) is the j-th component of the vector representation of x_i (resp., y).

4.3 k-wise Independent Polynomials

Definition 4.4. *A probability distribution $\mathcal{F}$ over functions $f : X \to Y$ is k-wise independent if for every k-tuple $(x_1, \ldots, x_k)$ of distinct elements of X, the random variable $(\mathcal{F}(x_1), \ldots, \mathcal{F}(x_k))$ is uniform over Y^k.*

For a prime power q and positive integers ℓ and $k < q-1$, let $\mathcal{P}_{\ell,k,q}$ denote the uniform distributions over multivariate polynomials with ℓ variables and total degree at most k over the field $\mathbb{F}_q$. That is, a polynomial is chosen by sampling each coefficient uniformly and independently from $\mathbb{F}_q$, and each polynomial is viewed as a function from $\mathbb{F}_q^\ell$ to $\mathbb{F}_q$. It is not hard to show that $\mathcal{P}_{\ell,k,q}$ is k-wise independent.

Claim 4.5. *For every prime power q and positive integers ℓ and $k < q-1$, the distribution $\mathcal{P}_{\ell,k,q}$ is k-wise independent.*

Proof. The truth table of a randomly chosen $f \leftarrow \mathcal{P}_{\ell,k,q}$ is just a random code-word of a Reed-Muller code with parameters ℓ and k over $\mathbb{F}_q$. To prove the claim we should show that the marginal distribution of any subset of k coordinates of such a random code-word is uniformly distributed over $\mathbb{F}_q^k$. Since the code is linear, this is equivalent to showing that the dual distance of the code is $k+1$. Indeed, it is known that the dual distance of such Reed-Muller code is at least $k+1$, see [47, Proposition 5.4.14].

More generally, the distance of the dual code is $(\rho+1)q^\sigma$ where ρ and σ are the reminder and quotient obtained by dividing $k+1$ by $q-1$ (that is, $k+1 = \sigma(q-1)+\rho$ for $\rho < q-1$). In our case, $k+1 < q$, and therefore $\rho = k+1$ and $\sigma = 0$, which yields a dual distance of at least $k+1$. □

Note that, given the description of $\mathbb{F}_q$, we can sample an element of $\mathcal{P}_{\ell,k,q}$ in time $\text{poly}(k^\ell \cdot \log q)$.

5 The Sampler/Tester Framework

Let $\mathcal{D}$ be a probability distribution and let E be some "bad event" that happens with low-probability (e.g., $o(1)$). The following mechanism allows us to sample from the conditional distribution $[\mathcal{D}|\neg E]$.

Definition 5.1 (Sampler/Tester)**.** *We say that a pair of algorithms (S, T) is a* sampler/tester pair *for a probability distribution $\mathcal{D}$ and an event E if the following holds.*

1. *The random variable $S(r)$, induced by feeding S with fresh random string $r \leftarrow \{0,1\}^s$, is identically distributed to $\mathcal{D}$.*
2. *Given a string $r \in \{0,1\}^s$, the deterministic algorithm $T(r)$ outputs 1 if and only if the sampler's corresponding output $S(r)$ satisfies the event E.*

Asymptotically, we consider an infinite sequence of pairs $\{(\mathcal{D}_n, E_n)\}_{n\in\mathbb{N}}$ and allow S, T to get 1^n as an additional input. In this setting, $S(1^n, \cdot), T(1^n, \cdot)$ should run in polynomial-time in n and should form a sampler/tester pair for $(\mathcal{D}_n, E_n)$ for every $n \in \mathbb{N}$.

Remark 5.2. *We will usually relax the requirement that the family $\{(\mathcal{D}_n, E_n)\}_{n\in\mathbb{N}}$ is defined for every $n \in \mathbb{N}$ and instead allow the family to be defined only with respect to an infinite set N of integers as long as the set N is not too sparse, i.e., the i-th element of N is upper-bounded by some polynomial in i.*

Observation 5.3. *Given a sampler/tester pair (S, T) for $\{(\mathcal{D}_n, E_n)\}_{n\in\mathbb{N}}$, we can sample from $[\mathcal{D}_n|\neg E_n]$ in expected time $\mathrm{poly}(n)/(1-\epsilon(n))$ where $\epsilon(n)$ denotes the probability that a sample from $\mathcal{D}_n$ satisfies the event E_n.*

Proof. Repeatedly sample $z = S(1^n; r)$ using fresh randomness r and output the result only if $T(1^n, r) = 0$. The expected number of repetition is $1/(1-\epsilon(n))$ as required. □

Note that by publishing the random seed r, everyone can verify that the output of the sampling algorithm is not in E.

5.1 Explicit Constructions under De-Randomization Assumptions

Observation 5.3 implies a **ZPP**-sampler for elements in the distribution $[\mathcal{D}|\neg E]$ whenever the failure probability $\epsilon(n)$ is upper-bounded by $1 - 1/p(n)$ for some polynomial $p(\cdot)$. In the following we show that, under standard worst-case de-randomization assumptions, any **ZPP**-construction implies a fully-explicit construction.

Theorem 5.4 ([31], see also Theorem 7.63 in [53])**.** *Assume that the class of functions computable in $2^{O(n)}$ uniform-time requires $2^{\Omega(n)}$ size circuits. Then, for every $t = t(n)$ there exist constants a, b and a pseudorandom generator $G : \{0,1\}^{a\cdot\log n} \to \{0,1\}^t$, computable in time t^b, that $(1/t)$-fools every deterministic Turing $\mathcal{A}$ machine running in time t, in the sense that*

$$\left|\Pr[\mathcal{A}(U_{t(n)}) = 1] - \Pr[\mathcal{A}(G(U_{a\cdot\log n})) = 1]\right| < 1/t(n),$$

where U_ℓ denotes the uniform distribution over $\{0,1\}^\ell$.

In fact, the assumption in Theorem 5.4 implies a PRG that fools non-uniform adversaries, but all we need is a PRG that fools (or actually *hit*) uniform adversaries.[12]

Corollary 5.5. *Let $\mathcal{D} = \{\mathcal{D}_n\}$ be a family of distributions. Let A be a* **ZPP** *construction for $\mathcal{D}$. Then, under the assumption in Theorem 5.4, there exists a polynomial time algorithm that on input 1^n outputs an element from $\mathcal{D}_n$.*

Proof. We show that the assumption in Theorem 5.4 implies a derandomization of the **ZPP**-constructions. Let A be a **ZPP** construction algorithm of some distribution $\mathcal{D} = \{\mathcal{D}_n\}$, running in time $t(n)$, and let $G : \{0,1\}^{a\cdot\log n} \to \{0,1\}^t$ be the PRG promised in Theorem 5.4. We define an explicit construction algorithm A' in the following way. On input 1^n the algorithm A' iterates over all binary strings $s \in \{0,1\}^{a\log n}$ and outputs the first seed s for which $A(G(s))$ does not output failure. The seed s represents the element $A(G(S))$ of $\mathcal{D}_n$.

Note that since A fails with negligible probability and since G $(1/t)$-fools A, it follows that there exists at least one seed s for which $A(G(s))$ does not output failure. Since A is a **ZPP**-construction it follows that if A does not fail then it outputs an element from the distribution $\mathcal{D}_n$, so A' always outputs an element of $\mathcal{D}_n$. Finally, since there are only $\mathrm{poly}(n)$ binary strings in $\{0,1\}^{a\log n}$ it follows that the running time of A' is $\mathrm{poly}(n)$. □

[12] One can get such a generator under weaker uniform hardness assumptions [32, 52]. However, to the best of our knowledge, in this setting all known generators work only for infinitely many inputs lengths. We thank Ronen Shaltiel for pointing us to the relevant literature.

6 k-wise Independent H-Free Hypergraphs

In the following section we present a sampler/tester for k-wise oriented hypergraphs and forbidden subgraphs. That is, the sampler samples an oriented (n, m, d)-hypergraphs from a k-wise independent distribution, while the tester tests whether some hypergraph H is a subgraph of the sampled graph. Recall that a distribution over (n, m, d) oriented hypergraph is said to be *k-wise independent* if every set of k hyperedges $(e_{i_1}, \ldots, e_{i_k})$ is uniformly distributed over $([n]^d)^k$. Extensions of the main result will be discussed in Section 6.2.

Below, we assume that n and m are both integer powers of some prime power q, and let $\mathcal{P}_{\ell,k,q}$ denote the uniform distribution over ℓ-variate polynomials of total degree k over the field $\mathbb{F}_q$.

Construction 6.1 (Sampler for $\mathcal{G}_{q,n,m,d,k}$)**.** *The distribution $\mathcal{G}_{q,n,m,d,k}$ is parameterized by a prime power q, and integers n, m, d and k, that correspond to the number of nodes, number of hyperedges, hyperedge arity, and independence parameter. We further assume that $n = q^r$, $m = q^\ell$ for some positive integers ℓ and r, and that $k < q - 1$. Correspondingly, we index the vertices by elements from $\mathbb{F}_q^r$ and index the hyperedges by elements in $\mathbb{F}_q^\ell$.*

- *(Index sampler) Uniformly sample $d \cdot r$ independent polynomials $P = (P_{i,1}, \ldots, P_{i,r})_{i\in[d]}$ from $\mathcal{P}_{\ell,k,q}$.*
- *(Edge evaluation) The vector of polynomials P represents an (n, m, d) oriented hypergraph G_P as follows. Each vertex is associated with an r-tuple $\alpha \in \mathbb{F}_q^r$ and each hyperedge is associated with an ℓ-tuple $\beta \in \mathbb{F}_q^\ell$. Every hyperedge $\beta \in \mathbb{F}_q^\ell$ is of arity d and its i-th vertex is defined to be $(P_{i,1}(\beta), \ldots, P_{i,r}(\beta))$.*

For a first reading, it will be convenient to think of d as a constant and of c as an integer. In this case we get $\ell = c$ and $r = 1$, so $n = q$, $m = q^c$ and there is exactly one polynomial P_i which represents the i-th vertex of the hyperedges.

Claim 6.2. *The distribution $\mathcal{G}_{q,n,m,d,k}$ over (n, m, d)-oriented hypergraphs is k-wise independent. Moreover, the complexity of the sampling algorithm is $r \cdot d \cdot \binom{k+\ell}{\ell} \cdot \mathrm{poly}(\log q)$, for $r := \log_q n$ and $\ell := \log_q m$. Furthermore, the graph G has representation of size $O(r \cdot d \cdot \binom{k+\ell}{\ell} \log q)$, and given its representation, one can find the i-th element of the j-th hyperedge in time $\mathrm{poly}(r \cdot \binom{k+\ell}{\ell} \cdot \log d \cdot \log q)$.*

Proof. Since the $d \cdot r$ polynomials are sampled independently from $\mathcal{P}_{\ell,k,q}$, Claim 4.5 implies that every k hyperedges are distributed uniformly and independently over $([n]^d)^k$. In order to sample a random ℓ-variate polynomial of degree k one has to sample at most $\binom{k+\ell}{\ell}$ random field elements (one coefficient for each possible monomial), which can be done in time $\mathrm{poly}(\log q)$ per field element. Hence, the sampling algorithm has total complexity of $r \cdot d \cdot \binom{k+\ell}{\ell} \cdot \mathrm{poly}(\log q)$. The size of the representation of the graph follows from the fact that a field element requires $O(\log q)$ space, and the complexity of computing the i-th element of the j-th hyperedge follows from the fact that field operations can be computed in time $\mathrm{poly}(\log q)$. □

The following key-lemma describes a tester for H-subgraphs.

Lemma 6.3 (key-lemma)**.** *There exists an algorithm A that given an $(n', m', \leq d)$ oriented hypergraph H and an index $P = ((P_{1,1}, \ldots, P_{1,r}), \ldots, (P_{d,1}, \ldots, P_{d,r}))$ of a hypergraph G_P from $\mathcal{G}_{q,n,m,d,k}$, tests if H is a subgraph of G_P in time polynomial in*

$$D^{(2\ell m' + 2rn')^{O(2\ell m' + 2rn')}} \cdot (dm'r + rn' + \ell m') \cdot \log q,$$

for $D := \max\{k, m', n'\}$.

The proof of the lemma is deferred to Section 6.1. The following theorem follows immediately from Claim 6.2 and Lemma 6.3.

Theorem 6.4. *Let $d \geq 2$ be a constant positive integer and let c be a constant positive rational number, represented in lowest terms by ℓ/r, and let A be a positive constant. For every prime power q define $n = q^r$ and $m = n^c$ and consider any*

$$n' \leq A \cdot \frac{\log\log\log n}{\log\log\log\log n} \qquad \text{and} \qquad m' \leq A \cdot \frac{\log\log\log n}{\log\log\log\log n}.$$

Then there exists a positive constant B such that for every

$$k \leq \min(q-2, n^{A/e(n',m')}), \quad \text{where } e(n',m') = (2m'\ell + 2n'r)^{B\cdot(2m'\ell+2n'r)},$$

the following holds. There is a $\mathrm{poly}(n)$*-time algorithm $\mathcal{S}$ that samples a graph G from $\mathcal{G}_{q,n,m,d,k}$ and a* $\mathrm{poly}(n)$*-time testing algorithm $\mathcal{T}$ that given an $(n',m',\leq d)$ oriented hypergraph H tests for the event that H is a subgraph of G. Moreover, if $k \leq \mathrm{polylog}(n)$, the ensemble $\mathcal{G}_{q,n,m,d,k}$ is succinct, and if, in addition, the hypergraph H is of constant size, i.e., $n'+m' = O(1)$, then the tester also runs in* $\mathrm{polylog}(n)$ *time.*

The first part of the theorem implies Theorem 2.1, and the "Moreover" part implies Theorem 2.6. We further mention that if succinctness is not needed, we can take the independence parameter k to be at least $2^{\log n/\log\log n} \gg \mathrm{poly}(\log n)$, and the degree d to be polynomial in n. For simplicity, we focus on $d = O(1)$.

Let $\mathcal{H} = \{H_n\}$ be an *efficiently constructible family of oriented hypergraphs*, that is, there exists an algorithm that on input 1^n outputs all the hypergraphs in H_n in time $poly(n)$. Further assume that H_n consists of hypergraphs of size (vertices plus hyperedges) at most $O(\frac{\log\log\log n}{\log\log\log\log n})$. Then Theorem 6.4 implies a sampler/tester for forbidden hypergraphs from $\mathcal{H}$.

Corollary 6.5. *Let c,ℓ,r,q,m,n,d,n',m' and k be as in Theorem 6.4. Let $\mathcal{H} = \{H_n\}$ be a efficiently constructible family of oriented hypergraphs, where H_n consists of hypergraphs with m' hyperedges and n' vertices. Then, there exists a sampler/tester pair for $\{\mathcal{G}_{q,n,m,d,k}, E_n\}$, where E_n is the event that some hypergraph in H_n is a subgraph of $\mathcal{G}_{q,n,m,d,k}$. Consequently, we can sample from $[\mathcal{G}_{q,n,m,d,k}|\neg E_n]$ in expected time of* $\mathrm{poly}(n)/(1-\epsilon(n))$ *where $\epsilon(n)$ is the probability that E_n occurs.*

Recall that if $k \leq \mathrm{poly}(\log n)$, the ensemble $\mathcal{G}_{q,n,m,d,k}$ has succinct representation.

6.1 Proof of Lemma 6.3

Given H and P we construct a set $\mathcal{L}$ of at most $dm'r + rn' + \ell m'$ polynomial equations with $2\ell m' + 2rn'$ variables over the field $\mathbb{F}_q$ where each equation is of degree at most $D := \max\{k, n', m'\}$. We show that the system has a solution if and only if H is a subgraph of $G := G_P$. We then employ the algorithm of Kayal [35, Theorem 6.1.1] which determines if such a system is solvable in time

$$\mathrm{poly}(D^{(2\ell m'+2rn')^{O(2\ell m'+2rn')}} \cdot (dm'r + rn' + \ell m') \cdot \log q).$$

Our system $\mathcal{L}$ will be composed of three sub-systems $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$.

The hypergraph equations $\mathcal{L}_1$. For every hyperedge h of H we define a tuple of ℓ-variables $x_h = (x_h(1), \ldots, x_h(\ell))$. For every vertex v of H define a tuple of r-variables $x_v = (x_v(1), \ldots, x_v(r))$. We will sometimes think of x_h and x_v as "super-variables" that take values in $\mathbb{F}_q^\ell$ and $\mathbb{F}_q^r$, respectively. The first part of our system $\mathcal{L}_1$ consists of the following equations. For every hyperedge h of H and every $i \in [d]$ for which $h[i] \neq \perp$, add r equations of the form

$$P_{i,j}(x_h) = x_v(j),$$

for every $j \in [r]$, where v is the i-th vertex of h in H, i.e., $h[i] = v$.

Let us analyze this part of our system.

Claim 6.6. *If H is a subgraph of G then $\mathcal{L}_1$ has a satisfying assignment that assigns distinct values to all the vertex variables and distinct values to all the hyperedge variables, i.e., for every two distinct vertices u and v the values assigned to x_u and x_v, when viewed as elements of $\mathbb{F}_q^r$, are distinct, and for every two distinct hyperedges h and f, the values assigned to x_h and x_f, when viewed as elements of $\mathbb{F}_q^\ell$, are distinct.*

Proof. Suppose that H is a subgraph of G, and let $\pi : V(H) \to V(G)$ and $\sigma : [m'] \to [m]$, be the injective maps guaranteed by Definition 4.1. We define the following assignment. For every $i \in [m']$ set the variable x_{h_i}, that corresponds to the i-th hyperedge h_i of H, to $x_{h_i} = e_{\sigma(i)}$ where $e_j \in \mathbb{F}_q^\ell$ is the j-th hyperedge

of G. For every vertex v of H set $x_v = \pi(v)$. (Recall that we identify the vertices of G with $\mathbb{F}_q^r$.) Since the i-th hyperedge of $\pi(H)$ *matches* to the $\sigma(i)$-th hyperedge of G, it follows that this assignment satisfies $\mathcal{L}_1$. Moreover, note that for every two distinct vertices u and v of H, the values assigned to x_u and x_v are distinct, since π is injective. Similarly, for every two distinct hyperedges h and f of H, the values assigned to x_h and x_f are distinct, since σ is injective. □

Claim 6.7. *Suppose that $\mathcal{L}_1$ has a satisfying assignment that assigns distinct values to all the vertex variables and distinct values to all the hyperedge variables (in the sense of Claim 6.6). Then H is a subgraph of G.*

Proof. For every vertex v and every hyperedge h, denote the value assigned to the variable x_v by a_v and the value assigned to the variable x_h by a_h. Define the mapping $\pi : V(H) \to V(G)$ by $\pi(v) = a_v$, and the mapping $\sigma : [m'] \to [m]$ by $\sigma(i) = j$ for a $j \in [m]$ such that $a_{h_i} = e_j$, where e_j is the j-th edge of G.

By construction, the assignment satisfies $\mathcal{L}_1$ if and only if the i-th hyperedge of $\pi(H)$ matches the $\sigma(i)$-th hyperedge of G. Moreover, as all the a_v's are distinct, it follows that π is injective, and as all the a_h's are distinct, it follows that σ is injective. Hence, by Definition 4.1 it follows that H is a subgraph of G. □

In order to prove that π and σ are injective, we crucially relied on the distinctness property of the assignment. Indeed, a violation of these additional properties results in a non-injective π or σ. We solve this problem by adding more constraints (and some auxiliary variables). Specifically, we use the following gadget.

Claim 6.8 (Distinctness gadget)**.** *Let $X = (X_1, \ldots, X_\eta)$ be η formal variables taking values from some finite field $\mathbb{F}$ of size at least η. Let $Y = (Y_0, \ldots, Y_{\eta-1})$ be additional formal variables over $\mathbb{F}$, and consider the polynomial system of η equations whose i-th equation is*

$$\sum_{j=0}^{\eta-1} Y_j X_i^j = a_i, \tag{1}$$

where $a_1, \ldots, a_\eta$ are some fixed distinct values from $\mathbb{F}$. Then, for any fixed assignment x for X there exists an assignment y for Y such that (x, y) satisfies the system (1) if and only if $x_1, \ldots, x_\eta$ are all distinct values in $\mathbb{F}$.

Proof of Claim. For the "if" direction, assume that the values $x_1, \ldots, x_\eta$ are all distinct. Then, using Lagrange's interpolation, we can construct a univariate polynomial $g(Z) = \sum_{j=0}^{\eta-1} b_j Z^j$ of degree at most $\eta - 1$ over $\mathbb{F}$, such that $g(x_i) = a_i$ for every $i \in [\eta]$. Then, by taking $y = (b_0, \ldots, b_{\eta-1})$, the assignment (x, y) satisfies the system.

For the "only-if" direction, assume towards a contradiction that the assignment (x, y) satisfies the system (1) but $x_i = x_j$ for some $i \neq j$. In this case, the LHS of the i-th equation equals to the LHS of the j-th equation, and therefore at least one of these equations is violated since $a_i \neq a_j$. □

In the following we define two additional set of equations, $\mathcal{L}_2$ and $\mathcal{L}_3$. The former makes sure that any assignment that satisfies $\mathcal{L}_1$ has distinct values for the x_v's, while the latter makes sure that any assignment that satisfies $\mathcal{L}_1$ has distinct values for the x_h's. We first define those equations over an extension field, and then show how to translate them to equations over the base field.

The vertices equations $\mathcal{L}_2$. Fix some arbitrary sequence $a_1, \ldots, a_{n'}$ of distinct values from $\mathbb{F}_{q^r}$, and let $y_0, \ldots, y_{n'-1}$ be additional formal variables over $\mathbb{F}_{q^r}$. First, we define the system $\mathcal{L}_2'$ as follows. For every vertex v_i of H add the equation $\sum_{j=0}^{n'-1} y_j x_{v_i}^j = a_i$ over $\mathbb{F}_{q^r}$, where we think of x_{v_i} as a formal variable over $\mathbb{F}_{q^r}$ (see Fact 4.2). Note that $\mathcal{L}_2'$ consists of n' polynomial equations over $\mathbb{F}_{q^r}$, where each equation has total-degree at most n' and variables $(x_v)_{v \in V(H)}$ and $(y_i)_{i \in [n']}$.

It remains to translate $\mathcal{L}_2'$ to equations over $\mathbb{F}_q$. Recall that for every vertex v of H, x_v is a vector of r formal variables over $\mathbb{F}_q$. Similarly, we can view each y_i as a vector of r formal variables over $\mathbb{F}_q$, that is $y_i = (y_i(1), \ldots, y_i(r))$ for every $i \in [n']$. Then, from Fact 4.3, there exist $r \cdot n'$ equations over $\mathbb{F}_q$, each of

total degree at most n', over the variables $(x_v(i))_{v\in V(H),i\in[r]}$ and $(y_i(j))_{i\in[n'],j\in[r]}$, for which the following holds. Every assignment $x_v(i)$ and $y_i(j)$ satisfies the new set of equations if and only if it satisfies $\mathcal{L}'_2$, when taking $x_v = (x_v(1),\ldots,x_v(r))$ and $y_i = (y_i(1),\ldots,y_i(r))$. We take the new set of equations to be $\mathcal{L}_2$.

The hyperedges equations $\mathcal{L}_3$. Fix some arbitrary sequence $b_1,\ldots,b_{m'}$ of distinct values from $\mathbb{F}_{q^\ell}$, and let $z_0,\ldots,z_{m'-1}$ be additional formal variables over $\mathbb{F}_{q^\ell}$. First, we define the system $\mathcal{L}'_3$ as follows. For every hyperedge h_i of H add the equation $\sum_{j=0}^{m'-1} z_j x_{h_i}^j = b_i$ over $\mathbb{F}_{q^\ell}$, where we think of x_{h_i} as a formal variable over $\mathbb{F}_{q^\ell}$ (see Fact 4.2). Note that $\mathcal{L}'_3$ consists of m' polynomial equations over $\mathbb{F}_{q^\ell}$, where each equation has total-degree at most m' and variables $(x_h)_{h\in E(H)}$ and $(z_i)_{i\in[m']}$.

It remains to translate $\mathcal{L}'_3$ to equations over $\mathbb{F}_q$. Recall that for every hyperedge h of H, x_h is a vector of ℓ formal variables over $\mathbb{F}_n$. Similarly, we can view each z_i as a vector of ℓ formal variables over $\mathbb{F}_q$, that is $z_i = (z_i(1),\ldots,z_i(\ell))$ for every $i\in[m']$. Then, from Fact 4.3, there exist $\ell\cdot m'$ equations over $\mathbb{F}_q$, each of total degree at most m', over the variables $(x_h(i)_{h\in E(H),i\in[\ell]})$ and $(z_i(j))_{i\in[m'],j\in[\ell]}$, for which the following holds. Every assignment $x_h(i)$ and $z_i(j)$ satisfies the new set of equations if and only if it satisfies $\mathcal{L}'_3$, when taking $x_h = (x_h(1),\ldots,x_h(\ell))$ and $z_i = (z_i(1),\ldots,z_i(\ell))$. We take the new set of equations to be $\mathcal{L}_3$.

Completing the proof. We can now prove that H is a subgraph of G if and only if there exists a satisfying assignment to the set of equations $\mathcal{L}_1\cup\mathcal{L}_2\cup\mathcal{L}_3$.

If direction: Suppose that we can satisfy $\mathcal{L}_1\cup\mathcal{L}_2\cup\mathcal{L}_3$ by some assignment σ that assigns the values $(a_v)_{v\in V(H)}$ to the vertex variables $(x_v)_{v\in V(H)}$ and the values $(a_h)_{h\in E(H)}$ to the hyperedge variables $(x_h)_{h\in E(H)}$. By Claim 6.7, it suffices to show that all the a_v's are distinct, and that all the a_h's are distinct. Indeed, since σ assigns values to the $y_i(j)$ variables that satisfy $\mathcal{L}_2$ (over $\mathbb{F}_q$) and to the $z_i(j)$ variables that satisfy $\mathcal{L}_3$ (over $\mathbb{F}_q$), Fact 4.3 implies that there exists an assignment to the "super-variables" y_i's that satisfies $\mathcal{L}'_2$ (over $\mathbb{F}_{q^r}$) and to the "super-variables" z_i's that satisfies $\mathcal{L}'_3$ (over $\mathbb{F}_{q^\ell}$). We can therefore use Claim 6.8 and conclude that the values $(a_v)_{v\in V(H)}$ are all distinct as well as the values $(a_h)_{h\in E(H)}$.

Only-If direction: Suppose that H is a subgraph of G. Then, by Claim 6.6, $\mathcal{L}_1$ has a satisfying assignment to the x_v's and x_h's, that assigns distinct values to all the vertex variables and distinct values to all the hyperedge variables. Fix this assignment to the x_v's and the x_h's. Since the values assigned to the x_v's are distinct, Claim 6.8 implies that there exists an assignment to the y_i's that satisfies $\mathcal{L}'_2$ (over $\mathbb{F}_{q^r}$). Fact 4.3 implies that there exists an assignment to the $y_i(j)$'s that satisfies $\mathcal{L}_2$ (over $\mathbb{F}_q$). Fix this assignment to the $y_i(j)$'s. Similarly, since the values assigned to the x_h's are distinct, Claim 6.8 implies that there exists an assignment to the z_i's that satisfies $\mathcal{L}'_3$ (over $\mathbb{F}_{q^\ell}$), and Fact 4.3 implies that there exists an assignment to the $z_i(j)$'s that satisfies $\mathcal{L}_3$ (over $\mathbb{F}_q$). By fixing this assignment to the $z_i(j)$'s, we derive an assignment that satisfies the system $\mathcal{L}_1\cup\mathcal{L}_2\cup\mathcal{L}_3$.

Running time. To analyze the running time, note that we have at most $dm'r + rn' + \ell m'$ polynomial equations with $2\ell m' + 2rn'$ variables over the field $\mathbb{F}_q$, where each equation is of degree at most $D := \max\{k, m', n'\}$. The total running-time for generating the equations, including the translation of $\mathcal{L}'_2$ and $\mathcal{L}'_3$ (see Fact 4.3), is $\mathrm{poly}(d, (m')^\ell, (n')^r, k^\ell, r, \log q)$, while, by [35, Theorem 6.1.1], checking whether the set of equations is solvable takes time $\mathrm{poly}(D^{(2\ell m'+2rn')^{O(2\ell m'+2rn')}}\cdot(dm'r + rn' + \ell m')\cdot\log q)$. This completes the proof of Lemma 6.3. □

6.2 Extensions

6.2.1 Edge-Disjoint Homomorphism

Recall that H is a subgraph of G if the vertices of H can be renamed, using an injective mapping $\pi : V(H)\to V(G)$, such that each of the hyperedges of H matches to a unique hyperedge of G. (The mapping between H hyperedges to G hyperedges is denoted by σ. See Definition 4.1.) A useful relaxation of this

notion is obtained by allowing π to be non-injective. In particular, two vertices of H can be mapped to the same vertex in G. Since the edge-mapping σ is still required to be injective, we refer to this notion as *edge-disjoint homomorphism* from H to G. For example, under this notion, a k-path digraph H is edge-disjoint homomorphic to a k-directed cycle G. The following lemma restates Lemma 6.3 to the case of edge-disjoint homomorphism.

Lemma 6.9. *There exists an algorithm A that given an $(n', m', \leq d)$ oriented hypergraph H and a vector of polynomials in $P = ((P_{1,1}, \ldots, P_{1,r}), \ldots, (P_{d,1}, \ldots, P_{d,r}))$ from $\mathcal{P}_{\ell,k,q}$ (where $k < \ell(q-1)$), tests if H is edge-disjoint homomorphic to G_P (defined as in Construction 6.1), and its running time is polynomial in*

$$D^{(2\ell m' + rn')^{O(2\ell m' + rn')}} \cdot (dm'r + \ell m') \cdot \log q,$$

for $D := \max\{k, m'\}$.

The proof of Lemma 6.9 follows from the analysis in Section 6.1, and noting that H is edge disjoint homomorphic to G_P if and only if the set of equations $\mathcal{L}_1 \cup \mathcal{L}_3$ has a solution.

Based on Claim 6.2 and Lemma 6.9 we derive the following theorem (that can be viewed as a version of Theorem 6.4 that applies to edge-disjoint homomorphism).

Theorem 6.10. *Let $d \geq 2$ be a constant integer and let c be a constant positive rational number, represented in lowest terms by ℓ/r, and let A be a positive constant. For every prime power q define $n = q^r$, $m = n^c$, and consider any*

$$n' \leq A \cdot \frac{\log\log n}{\log\log\log n} \qquad \text{and} \qquad m' \leq A \cdot \frac{\log\log\log n}{\log\log\log\log n}.$$

Then there exists a positive constant B such that for every

$$k \leq \min(q-2, n^{A/e(n',m')}), \quad \text{where } e(n', m') = (2m'\ell + n'r)^{B \cdot (2m'\ell + n'r)},$$

the following holds. There is a $\mathrm{poly}(n)$-time algorithm $\mathcal{S}$ that samples a graph G from $\mathcal{G}_{q,n,m,d,k}$ and a $\mathrm{poly}(n)$-time testing algorithm $\mathcal{T}$ that given an $(n', m', \leq d)$ oriented hypergraph H tests for the event that H is edge-disjoint homomorphic to G. Moreover, if $k \leq \mathrm{polylog}(n)$, the ensemble $\mathcal{G}_{q,n,m,d,k}$ is succinct, and if, in addition, the hypergraph H is of constant size, i.e., $n' + m' = O(1)$, then the tester also runs in $\mathrm{polylog}(n)$ time.

Note that if we are not interested in succinct hypergraphs, we can even take $d = \mathrm{poly}(n)$ and still get a polynomial-time testing algorithm. Observe that if $d = \omega(1)$ then the number of vertices might be $\omega\left(\frac{\log\log\log n}{\log\log\log\log n}\right)$ even if the number of edges is $O\left(\frac{\log\log\log n}{\log\log\log\log n}\right)$. In this case, edge-disjoint homomorphism allows us to test for $O\left(\frac{\log\log n}{\log\log\log n}\right)$ vertices. (Recall that for subgraphs we can only cope with $O\left(\frac{\log\log\log n}{\log\log\log\log n}\right)$ vertices.)

The following theorem is an adaptation of Corollary 6.5 to the case of edge-disjoint homomorphism.

Corollary 6.11. *Let $c, \ell, r, q, m, n, d, n', m'$ and k be as in Theorem 6.10. Let $\mathcal{H} = \{H_n\}$ be an efficiently constructible family of oriented hypergraphs, where H_n consists of hypergraphs with m' hyperedges and n' vertices. Then, there exists a sampler/tester pair for $\{\mathcal{G}_{q,n,m,d,k}, E_n\}$, where E_n is the event that some hypergraph in H_n is edge-disjoint homomorphic to $\mathcal{G}_{q,n,m,d,k}$. Consequently, we can sample from $[\mathcal{G}_{q,n,m,d,k} | \neg E_n]$ in expected time of $\mathrm{poly}(n)/(1-\epsilon(n))$ where $\epsilon(n)$ is the probability that E_n occurs.*

6.2.2 Non-Oriented Hypergraphs

Both the notion of oriented subgraph, and the notion of edge-disjoint homomorphism can be extended to the case of non-oriented hypergraphs. In particular, we say that an $(n', m', \leq d)$ non-oriented hypergraph H is a *subgraph* of an (n, m, d) non-oriented hypergraph G (resp., edge-disjoint homomorphic to G) if one can orient H and G (i.e., turn each hyperedge into a d-tuple) so that H is a subgraph of G (resp., H is edge-disjoint homomorphic to G).

Observation 6.12. *If a non-oriented hypergraph H is a subgraph of a non-oriented hypergraph G (resp., edge-disjoint homomorphic to G), then for any fixed orientation of G there exists an orientation of H such that H is an oriented subgraph of G (resp. edge-disjoint homomorphic to G).*

In the following section we will think of $\mathcal{G}_{q,n,m,d,k}$ (defined in Construction 6.1) as a distribution over non-oriented hypergraphs, unless stated otherwise. Similarly, for a vector of polynomials

$$P = ((P_{1,1}, \ldots, P_{1,r}), \ldots, (P_{d,1}, \ldots, P_{d,r}))$$

from $\mathcal{P}_{\ell,k,q}$, we will think of G_P (defined in Construction 6.1) as a non-oriented hypergraph, unless stated otherwise. In order to check if a non-oriented $(n', m', \leq d)$-hypergraph H is a subgraph of G_P, one can view G_P as an oriented graph G'_P and check, for each of all $(d!)^{m'}$ possible orientations H' of H, if H' appears as a subgraph of G_P. A similar approach applies to the case of testing whether H is edge-disjoint homomorphic to G_P. Therefore, by combining Lemmas 6.3 and 6.9 with Observation 6.12, we derive the following lemma.

Lemma 6.13. *There exists an algorithm A that given a non-oriented $(n', m', \leq d)$ hypergraph H and a vector of polynomials in $P = ((P_{1,1}, \ldots, P_{1,r}), \ldots, (P_{d,1}, \ldots, P_{d,r}))$ from $\mathcal{P}_{\ell,k,q}$, tests if H is a subgraph of G_P (defined as in Construction 6.1), and its running time is polynomial in*

$$(d!)^{m'} \cdot D^{(2\ell m' + 2rn')^{O(2\ell m' + 2rn')}} \cdot (dm'r + rn' + \ell m') \cdot \log q, \quad \text{for } D := \max\{k, m', n'\}.$$

Similarly, there exists an algorithm A that given a non-oriented $(n', m', \leq d)$ hypergraph H and a vector of polynomials in $P = ((P_{1,1}, \ldots, P_{1,r}), \ldots, (P_{d,1}, \ldots, P_{d,r}))$ from $\mathcal{P}_{\ell,k,q}$, tests if H is a edge-disjoint homomorphic to G_P, and its running time is polynomial in

$$(d!)^{m'} \cdot D^{(2\ell m' + rn')^{O(2\ell m' + rn')}} \cdot (dm'r + \ell m') \cdot \log q, \quad \text{for } D := \max\{k, m'\}.$$

The following theorem follows immediately from Claim 6.2 and Lemma 6.13.

Theorem 6.14. *Let $c, \ell, r, q, m, n, n', m'$ and k be as in Theorem 6.4 and let d be a constant. Then, there is a $\mathrm{poly}(n)$-time algorithm S that samples $\mathcal{G}_{q,n,m,d,k}$ and a $\mathrm{poly}(n)$-time testing algorithm T that given an $(n', m', \leq d)$ non-oriented hypergraph H tests for the event that H is a subgraph of G.*

Let $c, \ell, r, q, m, n, n', m'$ and k be as in Theorem 6.10 and let d be a constant. Then, there is a $\mathrm{poly}(n)$-time algorithm S that samples $\mathcal{G}_{q,n,m,d,k}$ and a $\mathrm{poly}(n)$-time testing algorithm T that given an $(n', m', \leq d)$ non-oriented hypergraph H tests for the event that H is edge-disjoint homomorphic to G.

Note that in Theorem 6.14 we can no longer take d to be polynomial in q.

Corollary 6.15. *Let $c, \ell, r, q, m, n, d, n', m'$ and k be as in the first part of Theorem 6.14 (resp., the second part of Theorem 6.14). Let $\mathcal{H} = \{H_n\}$ be a efficiently constructible family of non-oriented hypergraphs, where H_n consists of hypergraphs with m' hyperedges and n' vertices. Then, there exists a sampler/tester pair for $\{\mathcal{G}_{q,n,m,d,k}, E_n\}$, where E_n is the event that some hypergraph in H_n is a subgraph of $\mathcal{G}_{q,n,m,d,k}$ (resp., edge-disjoint homomorphic to $\mathcal{G}_{q,n,m,d,k}$). Consequently, we can sample from $[\mathcal{G}_{q,n,m,d,k} | \neg E_n]$ in expected time of $\mathrm{poly}(n)/(1 - \epsilon(n))$ where $\epsilon(n)$ is the probability that E_n occurs.*

7 Unbalanced Expanders

7.1 Definitions

In the following section we will be interested in the problem of sampling *expanders* from a k-wise independent distribution. We will be interested in the following notions of expansion.[13]

[13]Expanders are usually presented as bipartite graphs, as discussed in Footnote 7. In the following, we will stick to the presentation with hypergraphs.

Let $G = (V, E)$ be a non-oriented (n, m, d)-hypergraph. For a set $S \subseteq [m]$ of hyperedges, let $\Gamma(S)$ be the set of all vertices that are an element of at least one hyperedge of S. Let $\Gamma_1(S)$ be the set of all vertices that have multiplicity 1 in the sum of all hyperedges of S. (Recall that the sum of two multisets e and f is a multiset g in which the multiplicity of an element x is the sum of the multiplicity of x in e with the multiplicity of x in f.) For example, if $S = \{1, 2, 5\}$ and $e_1 = \{1, 2, 1\}, e_2 = \{2, 3, 4\}$ and $e_5 = \{5, 2, 3\}$ then $\Gamma(S) = \{1, 2, 3, 4, 5\}$, and $\Gamma_1(S) = \{4, 5\}$. Note that although 1 is contained only in e_1, its multiplicity is 2, so it is not contained in $\Gamma_1(S)$.

We begin with a formal definition of a vertex-expander and a unique-neighbor expander. We emphasize that in the following section all hypergraphs are non-oriented.

Definition 7.1 (vertex-expander)**.** *Let G be an (n, m, d)-hypergraph. We say that a set $S \subseteq [m]$ of hyperedges is* α-expanding *if $|\Gamma(S)| \geq \alpha|S|$. We say that G is an* (α, γ)-expander *if every set $S \subseteq [m]$ of hyperedges of size at most γ is α-expanding.*

We will usually refer to α as the expansion-factor. Note that we must have $\alpha \leq d$ and $\gamma \leq m/\alpha$.

Definition 7.2 (unique-neighbor expander)**.** *Let G be an (n, m, d)-hypergraph. We say that a set $S \subseteq [m]$ of hyperedges is* α-uniquely-expanding *if $|\Gamma_1(S)| \geq \alpha|S|$. Every $v \in \Gamma_1(S)$ is called a* unique-neighbor *in S. We say that G is a* (β, γ)-unique-neighbor expander *if every set $S \subseteq [m]$ of hyperedges of size at most γ is β-uniquely-expanding.*

Organization of the section. In Section 7.2 we show that a random (n, m, d)-hypergraph is likely to be a good expander. In Section 7.3 we will show how to sample an (n, m, d)-hypergraph from a k-wise independent distribution of hypergraphs, conditioned on the event that the hypergraph expands small sets. In Section 7.4 we will show how to compose the expanders that we sampled with a random (n, m, d)-hypergraph in order get an expander that also expands large sets.

7.2 The Expansion of k-wise Independent Hypergraphs

It is well known that a random (n, m, d) hypergraph is likely to be a good expander. We extend this fact to the case of k-wise independent (n, m, d)-hypergraphs. The proof of the claim is deferred to Appendix B.

Claim 7.3. *Let $c > 1$ be an arbitrary constant, and let $d > c$ be an integer. Let $\alpha < d - c$ be a constant, and let $m = n^c$. Let $\mathcal{G}_{n,m,d,k}$ be any k-wise independent distribution over (n, m, d)-hypergraphs. Let $\gamma \leq \min\{\rho n^{1-\delta}, k\}$ for sufficiently small constant ρ and $\delta = (c-1)/(d-\alpha-1)$. Let $s \leq \gamma$. Then the probability that there exists a set of s hyperedges which is not α-expanding is at most*

$$\left(a_{\alpha,d} \cdot \frac{s^{d-\alpha-1}}{n^{d-c-\alpha}}\right)^s,$$

for a constant $a_{\alpha,d}$ that depends only on α and d. Consequently, the probability that the sampled graph is not (α, γ)-expander is at most $1/2$.

It is well known that vertex expansion implies unique-neighbor expansion (cf. [17]). In particular, if $\alpha = (1 - \epsilon)d$ and $\beta = (1 - 2\epsilon)d$ for some $\epsilon < 0.5$, then every set which is α-expanding, is also β-uniquely-expanding. The following claim follows.

Claim 7.4. *Let $c > 1$ be an arbitrary constant and let $d > 2c$ be an integer. Let $\beta < d - 2c$ be a constant, and let $m = n^c$. Let $\mathcal{G}_{n,m,d,k}$ be any k-wise independent distribution over (n, m, d)-hypergraphs. Let $\gamma \leq \min\{\rho n^{1-\delta}, k\}$ for sufficiently small constant ρ and $\delta = 2(c-1)/(d-\beta-2)$. Let $s \leq \gamma$. Then the probability that there exists a set of s hyperedges which is not β-uniquely-expanding is at most*

$$\left(a_{\beta,d} \cdot \frac{s^{(d-\beta-2)/2}}{n^{(d-2c-\beta)/2}}\right)^s,$$

for a constant $a_{\beta,d}$ that depends only on β and d. Consequently, the probability that the sampled graph is not (β, γ)-unique-neighbor expander is at most $1/2$.

7.3 k-wise Independent Small-Set Expanders

In this section we prove the following theorem.

Theorem 7.5. *Let $c > 1$ be an arbitrary constant and let $d > c$ be some integer. Let $\alpha < d - c$ be a constant and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. There exists a $\mathrm{poly}(n)$-time sampler/tester pair with the following properties:*

1. *For every prime power n, the sampler samples $(n, m = n^c, d)$-hypergraph.*[14]
2. *The tester tests if the sampled hypergraph is (α, γ)-expander. Moreover, the sampled graph passes the test with probability $\frac{1}{2}$.*
3. *The hypergraph is sampled from a k-wise independent distribution, for $k \leq \min(\mathrm{poly}(n^{1/e(\gamma)}), n^\epsilon)$, where $e(\gamma) = \gamma^{O(\gamma)}$, the constants in the big-O notation depend on c, and $\epsilon > 0$ is a constant that depends on c. Moreover, if $k \leq \mathrm{poly}(\log n)$ then the sampled graph has a succinct representation.*

Combining Theorem 7.5 with Observation 5.3 we get the following version of Theorem 2.8.

Corollary 7.6. *Let $c > 1$ be an arbitrary constant and let $d > c$ be some integer. Let $\alpha < d - c$ be a constant and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. Then, for a prime power n and $k = \min(\mathrm{poly}(n^{1/e(\gamma)}), n^\epsilon)$ it is possible to sample in expected polynomial-time an $(n, m = n^c, d)$-hypergraph from a k-wise independent distribution, conditioned on having (α, γ)-expansion. Consequently, there is a zero-error randomized construction of such expanders. Moreover, if $k \leq \mathrm{poly}(\log n)$ then the resulting graph has a succinct representation (but polynomial-time generation algorithm).*

For the case of unique-neighbor expander, we prove the following.

Theorem 7.7. *Let $c > 1$ be some arbitrary constant and let $d > 2c$ be some integer. Let $\beta < d - 2c$ be a constant, and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. There exists a $\mathrm{poly}(n)$-time sampler/tester pair with the following properties:*

1. *For every prime power n, the sampler samples $(n, m = n^c, d)$-hypergraph.*
2. *The tester tests if the sampled hypergraph is (β, γ)-unique-neighbor expander. Moreover, the sampled graph passes the test with probability $\frac{1}{2}$.*
3. *The hypergraph is sampled from a k-wise independent distribution, for $k = \min(\mathrm{poly}(n^{1/e(\gamma)}), n^\epsilon)$, where $e(\gamma) = \gamma^{O(\gamma)}$, the constants in the big-O notation depend on c, and $\epsilon > 0$ is a constant that depends on c. Moreover, if $k \leq \mathrm{poly}(\log n)$ then the sampled graph is succinct.*

Combining Theorem 7.7 with Observation 5.3 we get the following theorem.

Corollary 7.8. *Let $c > 1$ be some arbitrary constant and let $d > 2c$ be some integer. Let $\beta < d - 2c$ be a constant, and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. Then, for a prime power n and $k = \min(\mathrm{poly}(n^{1/e(\gamma)}), n^\epsilon)$ it is possible to sample in expected polynomial-time an $(n, m = n^c, d)$-hypergraph from a k-wise independent distribution conditioned on have (β, γ)-unique-expansion. Consequently, there is a zero-error randomized construction of such expanders. Moreover, if $k \leq \mathrm{poly}(\log n)$ then the resulting graph has a succinct representation (but polynomial-time generation algorithm).*

To prove the theorems we observe that expanders and unique-neighbor expanders occur with high probability when sampling from a k-wise independent distribution of hypergraphs. As we have already seen a sampler for such a distribution, and an algorithm that tests for subgraphs and edge-disjoint homomorphism (see Theorem 6.14), we will show that we can use it to construct a tester for the expansion properties of the sampled graph.

[14] If n^c is not an integer then $m = \lfloor n^c \rfloor$. For clarity, we always omit the floor symbol.

7.3.1 Proof of Theorem 7.5

Let $c > 1$ be an arbitrary constant and let $d > c$ be some integer. Let $\alpha < d - c$ be a constant and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. Let $k = \min(\text{poly}(n^{1/e(\gamma)}), n^{\epsilon})$.

Note that we can assume without loss of generality that c is rational, as for any irrational number c and every $\alpha < d - c$, there exists a rational number $c^* > c$ such that $\alpha < d - c^*$.

Theorem 6.14 implies a sampling algorithm $\mathcal{S}$, that, given as input 1^n for a prime power n, samples in time $\text{poly}(n)$ an $(n, m = n^c, d)$-hypergraph G from a k-wise independent distribution. Note that if $k \leq \text{poly}(\log n)$ then the sampled graph is succinct, and that Claim 7.3 implies that G is an (α, γ)-expander with probability at least $1/2$.

Moreover, Theorem 6.14 implies a $\text{poly}(n)$-time testing algorithm $\mathcal{T}$, that, given a non-oriented $(n', m', \leq d)$-hypergraph H, with $n' = O(\gamma)$ and $m' = O(\gamma)$, tests for the event that H is edge-disjoint homomorphic to G. It remains to show that we can use $\mathcal{T}$ in order to construct a tester $\mathcal{T}'$ for the event that the sampled graph is not an (α, γ)-expander.

Claim 7.9. *Let G be an (n, m, d)-hypergraph. Fix some constant α. Then, there exists an (n', m', d)-hypergraph H, for $n' < \alpha m'$, which is edge-disjoint homomorphic to G if and only if G has a set of m' hyperedges which is not α-expanding.*

Proof. Let H be an (n', m', d)-hypergraph, for $n' < \alpha m'$ that is edge-disjoint homomorphic to G. Let σ and π be the corresponding mappings. Let S be the image of σ (recall that σ is injective, hence its image is of size m'). Then $\Gamma(S)$, corresponds to the image of π, that has at most $n' < \alpha m'$ vertices.

For the other direction, let S be a set of m' hyperedges which is not α-expanding, and denote $n' = |\Gamma(S)|$. Denote the edges in S by $e_1, \ldots, e_{m'}$ and the vertices by $v_1, \ldots, v_{n'}$. Let H be the (n', m', d)-graph whose vertices are $v_1, \ldots, v_{n'}$ and whose edges are $e_1, \ldots, e_{m'}$. Then H is edge-disjoint homomorphic to G. □

The following observation shows that a tester for edge-disjoint homomorphism can be used for testing expansion.

Observation 7.10. *Let $\mathcal{S}$ be a sampling algorithm that samples an (n, m, d)-graph G from a k-wise independent distribution, and let $\mathcal{T}$ be a testing algorithm that, given an $(n', m', \leq d)$-graph, test for the event that H is edge-disjoint homomorphic to G. Then, using only $\gamma \cdot \alpha\gamma \cdot (\alpha\gamma)^{d\gamma}$ applications of $\mathcal{T}$ on hypergraphs with $m' \leq \gamma$ and $n' < \alpha\gamma$, we can test whether the sampled graph is an (α, γ)-expander.*

Proof. Iterate over all possible $m' \leq \gamma$ and $n' < \alpha m'$. For every pair (m', n') we go over all $(n')^{dm'}$ possible (n', m', d)-hypergraphs. For every such (n', m', d)-hypergraph H, check if H is edge-disjoint homomorphic to G using $\mathcal{T}$. From Claim 7.9 it follows that there exists a set at most γ hyperedges which is not α-expanding if and only if some H is edge-disjoint homomorphic to G. □

Finally, Observation 7.10, together with the existence of $\mathcal{T}$ implies the existence of a $\text{poly}(n)$-time algorithm $\mathcal{T}'$ that tests for the event that G is not an (α, γ)-expander. This concludes the proof of Theorem 7.5.

7.3.2 Proof of Theorem 7.7

The proof is very similar to the one presented in Section 7.3.1. Let $c > 1$ be some constant and let $d > 2c$ be an integer. Let $\beta < d - 2c$ be a constant, and let $\gamma = O(\frac{\log\log\log n}{\log\log\log\log n})$ be an efficiently-computable function. Let $k = \min(\text{poly}(n^{1/e(\gamma)}), n^{\epsilon})$.

Again, we assume without loss of generality that c is rational.

Theorem 6.14 implies a sampling algorithm $\mathcal{S}$, that, given as input 1^n for a prime power n, samples in time $\text{poly}(n)$ an $(n, m = n^c, d)$-hypergraph G from a k-wise independent distribution. Note that if $k \leq \text{poly}(\log n)$ then the sampled graph is succinct, and that Claim 7.4 implies that G is a (β, γ)-unique-neighbor expander with probability at least $1/2$.

Moreover, Theorem 6.14 implies a $\text{poly}(n)$-time testing algorithm $\mathcal{T}$, that, given a non-oriented $(n', m', \leq d)$-hypergraph H, with $n' = O(\gamma)$ and $m' = O(\gamma)$, tests for the event that H is a subgraph of G. It remains

to show that we can use $\mathcal{T}$ in order to construct a tester $\mathcal{T}'$ for the event that the sampled graph is not an (β, γ)-unique-neighbor expander.

Claim 7.11. *Let G be an (n, m, d)-hypergraph. Fix some constant β. Then, there exists an (n', m', d)-hypergraph H, for which the set $S = [m']$ is not β-uniquely-expanding and H is a subgraph of G if and only if G has a set of m' hyperedges which is not β-uniquely-expanding.*

Proof. For the first direction, let σ and π be the mappings promised in Definition 4.1, and take S to be the image of σ. Then S is not β-uniquely-expanding in G. For the other direction, take H to be the subgraph that contains the hyperedges of the set that is not β-uniquely-expanding. □

The following observation shows that a tester for subgraphs can be used for testing unique-neighbors expansion.

Observation 7.12. *Let $\mathcal{S}$ be a sampling algorithm that samples an (n, m, d)-graph G from a k-wise independent distribution, and let $\mathcal{T}$ be a testing algorithm that, given an $(n', m', \leq d)$-graph, test for the event that H is a subgraph of G. Then, using only $\gamma \cdot (d\gamma) \cdot (d\gamma)^{d\gamma}$ applications of $\mathcal{T}$ on hypergraphs with $m' \leq \gamma$ and $n' \leq d\gamma$, we can test whether the sampled graph is a (β, γ)-unique-neighbor expander.*

Proof. Iterate over all possible $m' \leq \gamma$ and $n' \leq dm'$. For every pair (m', n') we go over all $(n')^{dm'}$ possible (n', m', d)-hypergraphs. For every (n', m', d)-hypergraph H for which the set $T = [m']$ is not β-uniquely expanding, check if H is a subgraph of G. From Claim 7.11 it follows that there exists a set S of size $\leq \gamma$ in G that is not β-uniquely-expanding if and only if some H which is not β-uniquely-expanding is a subgraph of G. □

Finally, Observation 7.12, together with the existence of $\mathcal{T}$ implies the existence of a poly(n)-time algorithm $\mathcal{T}'$ that tests for the event that G is not an (β, γ)-unique-neighbor expander. This concludes the proof of Theorem 7.7.

7.4 Expanding Large Sets

7.4.1 Composing Vertex-Expander with Random Graphs

The following construction shows how to compose two hypergraphs, G_1 and G_2, where in G_1 every "small" set is good (that is, it has a good expansion), and in G_2 the "big" sets are good.

Construction 7.13. *Given two (n, m, d)-hypergraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ we define the following $(n, m, 2d)$-hypergraph $G = (V, E)$. For the vertices, identify both V_1 and V_2 with $[n]$. The set of vertices of G is simply $[n]$. For the hyperedges, assume that $E_1 = (f_1, \ldots, f_m)$ and $E_2 = (h_1, \ldots, h_m)$. Then the i-th hyperedge of G is $e_i = f_i + h_i$, where $f_i + h_i$ is the sum of the multisets f_i and h_i.*

Recall that the sum of two multisets f and h is a multiset e in which the multiplicity of an element x is the sum of the multiplicity of x in f with the multiplicity of x in h. For example, if $f_i = \{1, 3, 5\}$ and $h_i = \{1, 2, 4\}$ then $e_i = \{1, 1, 2, 3, 4, 5\}$ (recall that in this context, $\{\}$ represents a multiset).

Claim 7.14. *Let G_1 and G_2 be an (n, m, d)-hypergraphs and let $\gamma > \gamma'$. Assume that every set S of at most γ' hyperedges of G_1 has expansion α_1 (that is, $|\Gamma(S)| \geq \alpha_1 |S|$). In addition, assume that every set S of hyperedges of G_2, of size at least γ' and at most γ has expansion α_2. Let $\alpha = \min\{\alpha_1, \alpha_2\}$. Then G, defined as in Construction 7.13, is an $(n, m, 2d)$-hypergraph and an (α, γ)-expander.*

Proof. It is clear that G is an $(n, m, 2d)$-hypergraph. For the expansion property, fix some set S of s hyperedges from G. If $s \leq \gamma'$ then, by the expansion property of G_1 it holds that $\Gamma(S) \geq \alpha_1 s$. If $\gamma' \leq s \leq \gamma$, then by the expansion property of G_2 it holds that $\Gamma(S) \geq \alpha_2 s$. The claim follows. □

We can now prove the following version of Theorem 2.9 from the introduction.

Theorem 7.15 (Negligible-Error Construction of Expanders)**.** *Let $c > 1$ be some constant, let $d > c$ be an integer, and let $\alpha < d - c$ be a constant. There exists a* poly(n)*-time probabilistic algorithm that, except with negligible probability $n^{-\omega(1)}$, samples an $(n, m = n^c, 2d)$-hypergraph which is an (α, γ)-expander, where $\gamma = \rho n^{1-\delta}$ for a sufficiently small constant ρ and $\delta = (c-1)/(d-\alpha-1)$.*

Proof. Let $\gamma' = \Theta(\frac{\log\log\log n}{\log\log\log\log n})$. Let G_1 be the (n, m, d)-hypergraph which is a (α, γ')-expander, promised in Corollary 7.6. Note that G_1 can be sampled in time poly(n) with negligible error.

Let G_2 be a random (n, m, d) hypergraph (that is, every hyperedge is distributed uniformly and independently over $[n]^d$). Let $\gamma' \leq s \leq \gamma$. By Claim 7.3, the probability that there exists a bad set of size s is negligible. Taking union-bound over at most $\rho n^{1-\delta}$ possible assignments to s, we get that the probability that there exists a set of size more than γ' and at most γ which is not α-expanding is negligible. Moreover, note that G_2 can be sampled in time poly(n).

Conditioned on the events that G_1 is an (α, γ')-expander, and that in G_2 every set $S \subseteq [m]$ of size $\gamma' \leq s \leq \gamma$ is α-expanding, Let G be as in Construction 7.13. Then, by Claim 7.14 G is a $(n, m, 2d)$-hypergraph which is (α, γ)-expander, which completes the proof. □

7.4.2 Composing Unique-Neighbor Expander with Random Graphs

The following construction shows how to compose two hypergraphs, G_1 and G_2, where in G_1 every "small" set is uniquely-expanding, while in G_2 the "big" sets are uniquely-expanding.

Construction 7.16. *Given two (n, m, d)-hypergraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ we define the following $(2n, m, 2d)$-hypergraph $G = (V, E)$. For the vertices, assume that $V_1 = \{v_1, \ldots, v_n\}$ and $V_2 = \{u_1, \ldots, u_n\}$. The vertices of G is the union of the set of vertices of G_1 with the set of vertices of G_2 (that is, $V = V_1 \cup V_2$). For the hyperedges, assume that $E_1 = (f_1, \ldots, f_m)$ and $E_2 = (h_1, \ldots, h_m)$. Then the i-th hyperedge of G is $e_i = f_i + h_i$, where $f_i + h_i$ is the sum of the multisets f_i and h_i.*

For example, if $f_i = \{v_1, v_1, v_7\}$ and $h_i = \{u_2, u_5, u_7\}$ then $e_i = \{v_1, v_1, v_7, u_2, u_5, u_7\}$ (recall that in this context, $\{\}$ represents a multiset).

Claim 7.17. *Let G_1 and G_2 be an (n, m, d)-hypergraphs and let $\gamma > \gamma'$. Assume that every set S of at most γ' hyperedges of G_1 is β-uniquely-expanding (that is, G_1 is a (β, γ')-unqiue-neighbor expander). In addition, assume that every set S of hyperedges of G_2, of size at least γ' and at most γ, is β-uniquely-expanding. Then G, defined as in Construction 7.16, is a $(2n, m, 2d)$-hypergraph and a (β, γ)-unique-neighbor expander.*

Proof. It is clear that G is a $(2n, m, 2d)$-hypergraph. For the expansion property, fix a set S of s hyperedges of G. First, note that if the set S has a unique neighbor v in G_1 (resp., G_2) then this vertex is also a unique neighbor in G, since no hyperedge in G_2 (resp., G_1) touches v. Now, if $s \leq \gamma'$, then, by the expansion property of G_1, it holds that $\Gamma_1(S) \geq \beta s$. If $\gamma' \leq s \leq \gamma$ then, by the expansion property of G_2 it holds that $\Gamma_1(S) \geq \beta s$. Hence G is a (β, γ)-unique-neighbor expander. □

We can now prove the following version of Theorem 2.10.

Theorem 7.18 (Negligible-Error Construction of Unique-Neighbor Expaner)**.** *Let $c > 1$ be some constant, let $d > 2c$ be an integer and let $\beta < d - 2c$ be a constant. There exists a* poly(n)*-time probabilistic algorithm that, except with negligible probability $n^{-\omega(1)}$, samples an $(2n, m = n^c, 2d)$-hypergraph with (β, γ)-unique-neighbor-expander, where $\gamma = \rho n^{1-\delta}$ for a sufficiently small constant ρ and $\delta = 2(c-1)/(d-\beta-2)$.*

Proof. Let $\gamma' = \Theta(\frac{\log\log\log n}{\log\log\log\log n})$. Let G_1 be the (n, m, d)-hypergraph which is a (β, γ')-unique-neighbor expander, promised in Corollary 7.8. Note that G_1 can be sampled in time poly(n) with negligible error.

Let G_2 be a random (n, m, d) hypergraph (that is, every hyperedge is distributed uniformly and independently over $[n]^d$). Let $\gamma' \leq s \leq \gamma$. By Claim 7.4, the probability that there exists a bad set of size s is negligible. Taking union-bound over at most $\rho n^{1-\delta}$ possible assignments to s, we get that the probability that there exists a set of size more than γ' and at most γ which is not β-uniquely-expanding is negligible. Moreover, note that G_2 can be sampled in time poly(n).

Conditioned on the events that G_1 is a (β, γ')-unique-neighbor expander, and that in G_2 every set $S \subseteq [m]$ of size $\gamma' \leq s \leq \gamma$ is β-uniquey-expanding, let G be as in Construction 7.16. Then, by Claim 7.17, G is a $(2n, m, 2d)$-hypergraph which is (β, γ)-unique-neighbor expander, which completes the proof. □

8 Applications

8.1 Batch Codes

Batch codes, introduced by Ishai, Kushilevitz, Ostrovsky and Sahai in [33], allow us to distribute an m-bit information word x over n devices such that every subset of γ bits of x ("batch") can be decoded by retrieving at most a single bit from each device while using a total space of at most M bits. Formally, an (m, M, γ, n) *batch code* is a pair of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ such that:

- The encoder Enc maps an information word $x \in \{0,1\}^m$ into an n-tuple of strings, $y = (y_1, \ldots, y_n)$, where $y_i \in \{0,1\}^*$ and the total length of the y_i's is M.
- The decoder Dec takes as an input a γ-subset $S \subset [m]$ and, by querying a single bit from each y_i, it recovers $(x_i)_{i \in S}$.

The y_i's are called *buckets*, and the *rate* of the code is m/M.

When every bucket is simply a multiset of the bits of x, such a code is called *combinatorial batch code* (CBC)[51]. If each bit is stored in precisely d buckets the code is referred to as *d-uniform* [51]. A CBC can be represented by a hypergraph with n vertices (representing buckets) and m hyperedges (representing the bits of x), where the i-th hyperedge contains all the buckets in which the i-th bit is contained in (with multiplicity). In [33] it is noted that such a hypergraph is a batch code if and only if it has expansion factor 1. Hence, the explicitness of such codes is tighly connected to the explicitness of unbalanced expanders. By Theorem 7.15 we get the following theorem.

Theorem 8.1. *For every constant $c > 1$ and integer $d > 1 + c$, there exists a negligible-error construction of an (m, M, γ, n)-CBC with information words of length m, codeword of length $M = 2dm$, $n = m^{1/c}$ buckets and batch-size of $\Omega(n^{1-\frac{c-1}{d-2}})$ where the constant in the Omega depends on d. Moreover, the code is $2d$-uniform.*

Theorem 8.1 guarantees an algorithm A that outputs a description of the CBC encoder and decoder. Since this is a CBC code, the encoding algorithm can be implemented in linear-time of $O(m)$ in the RAM model (assuming some initial data-independent prepossessing that can be implemented by A). Moreover, since the code is $2d$-uniform every bit-change in the information word requires only $2d$ modifications in the codeword. Decoding of a γ-subset $S \subseteq [m]$ can be done by finding a perfect matching between the required bits S and some γ buckets, and reading exactly one bit from each of the buckets. This can be done in time $\tilde{O}(m^{10/7})$ (see [40]).

Theorem 8.1 yields the first negligible-error construction of constant-rate CBC in the regime where the number of devices n is polynomially smaller than the length m of the information word and the batch-size γ is polynomially smaller than n.[15] Boyle, Couteau, Gilboa and Ishai [16] recently showed that such CBC's can be used to construct a useful cryptographic primitive (Multi-Point Function Secret Sharing) with an optimal computational cost. Due to the lack of negligible-error constructions, they had to rely on heuristic assumptions or to tolerate an inverse polynomial error. (See the discussion in [16, page 17].) Theorem 8.1 avoids these caveats and yields a constructive version of [16].

[15] As we have mentioned, batch codes require a weak notion of expansion, that is, an expansion factor 1. To the best of our knowledge, there are no known constructions for such expanders in this regime.

8.2 High-Rate LDPC Codes

As already mentioned, the $n \times m$ incidence matrix of an (n, m, d)-hypergraph G with (β, γ)-unique-expansion forms a d-sparse parity-check matrix of an $[m, m-n]$-linear code of distance γ. Hence, by Theorem 7.18, we get the following.

Theorem 8.2. *Let $c > 1$ be some arbitrary constant and let $d > 2c$ be an integer. Let $m = n^c$ and let $\gamma = \Omega(n^{1-\delta})$, where $\delta > 2(c-1)/(d-2)$ and the constants in the Ω-notation depend on d, c and δ. For every prime power n, there exists a negligible-error construction of a $2n \times m$ parity-check matrix with at most $2d$ ones in each columns which describes an $[m, m-2n]$-code of distance at least γ.*

We tweak the construction in order to get efficient decoding. Formally, we prove the following theorem (a restatement of Theorem 2.11) in Section 8.2.1.

Theorem 8.3. *Let $c > 1$ be an arbitrary constant, let $d > 10c$ be an integer and let $0.9d < \alpha < d - c$ be a constant. For every prime power n there exists a negligible-error construction of a $2n \times m$ parity-check matrix with at most $2d$ ones in each column where $m = n^c$. The corresponding code has an efficient decoding algorithm that corrects at least $\Omega(n^{1-\frac{c-1}{d-\alpha-1}})$ errors within $O(\log n)$ parallel steps, where the constants in the Ω-notation depend on d and α, and the constants in the O-notation depend on α.*

The constant α provides a trade-off between the distance and the decoding complexity: A larger α improves the number of parallel steps, but reduces the distance.

8.2.1 Proof of Theorem 8.3

As a warm-up, we start by presenting the Sipser-Spielman Decoder that works under the assumption that the underlying graph satisfies strong expansion properties (stronger than the ones that we actually achieve). Then, we present our construction and explain how to adopt the decoding algorithm to our setting.

The decoder relies on the following simple combinatorial lemma taken from [17]. For completeness, we prove the lemma in Appendix C.

Lemma 8.4 ([17])**.** *Let G be an (n, m, d)-hypergraph, let $\alpha = (1-\epsilon)d$ for $\epsilon \leq 1/2$. Let $\delta, \eta > 2\epsilon$ be constants and $0 \leq \gamma' \leq \gamma \leq m$ be some integers. If every set $S \subseteq [m]$ of size $|S| \in (\gamma', \gamma)$ is α-expanding, then for every such set S it holds that:*

1. *At least $1 - \delta$ fraction of the hyperedges in S each have more than $(1 - 2\epsilon/\delta)d$ unique neighbors.*
2. *At most $\frac{2\epsilon}{\eta - 2\epsilon}|S|$ hyperedges not in S each have at least ηd vertices in $\Gamma(S)$, provided that $|S| \in (\gamma', (1 - \frac{2\epsilon}{\eta})\gamma)$.*

Decoding as solving a constraint-satisfaction problem. Let G be an (n, m, d)-graph and let H be its incidence matrix that will be used as the parity-check matrix of the code. Given a word $x \in \{0,1\}^m$, we label the ith hyperedges, e_i, of G by x_i, the ith bit of x. We associate a linear constraint with each vertex v_i which is satisfied if the sum of all the labels of the hyperedges that contain v_i (with multiplicity) is even. By definition, x is a codeword if and only if all its constraints are satisfied (i.e., $Hx = 0^n$).

The Sipser-Spielman decoder for good expanders. Assume that the (n, m, d)-hypergraph G is an (α, γ)-expander where $\alpha = (1-\epsilon)d$ for $\epsilon < 0.1$. The expansion-property of G implies that G is also a (β, γ)-unique-neighbor expander, for $\beta = (1-2\epsilon)d$. Hence, G defines an $[m, m-n, \gamma]$-code. Let $1/2 < \eta < 1$ and $\delta \geq 2\epsilon$ be some constants that satisfy

$$(1 - 2\epsilon/\delta) > \eta \qquad \text{and} \qquad 1 - \delta > 2\epsilon/(\eta - 2\epsilon), \tag{2}$$

e.g., $\eta = 0.6$ and $\delta = 0.5$. The decoding algorithm gradually updates the noisy codeword until it reaches a valid codeword in the following way. In each iteration, the decoder flips the value of all the hyperedges that have at least ηd unsatisfied constraints, until all constraints are satisfied.

The analysis proceeds as follows. Fix some noisy codeword and let S be the set of hyperedges that correspond to the coordinates in which errors occur. We assume that S is sufficiently small (the maximal number of errors that we can correct will be specified later). The key observation is that every unique-neighbor in S is an unsatisfied constraint, and every unsatisfied constraint is in $\Gamma(S)$. Now let us apply Lemma 8.4 with $\gamma' = 0$ for now. The first part of the lemma implies that in each iteration at least $1-\delta$ fraction of the hyperedges in S have at least ηd unique-neighbors, which are unsatisfied constraints. Hence, in each iteration, at least $1-\delta$ fraction of the hyperedges in S get the correct value. The second part of Lemma 8.4 implies that at most $2\epsilon/(\eta-2\epsilon)|S|$ hyperedges outside of S (that is, hyperedges with correct value), have at least ηd vertices in $\Gamma(S)$. Since all unsatisfied constraints are in $\Gamma(S)$, it follows that at each iteration the algorithm flips the correct value of at most $2\epsilon/(\eta-2\epsilon)|S|$ hyperedges. Let $\xi = (1-\delta) - 2\epsilon/(\eta-2\epsilon) > 0$. It follows that at each iteration the total number of error decreases by a constant factor of $(1-\xi)$, and therefore the number of iterations is $O(\log n)$. Finally, note that the decoding algorithm can fix at least $(1-2\epsilon/\eta)\gamma > 0.6\gamma$ errors.

Our construction. We show that a modified version of this decoder can decode our construction. Let $c > 1$, $d > 10c$ and $\alpha \in (0.9d, d-c)$ be constants as in the statement of Theorem 8.3. We sample a hypergraph G as follows. Use Corollary 7.6 to sample an $(n, m = n^c, d)$-hypergraph $G_1 = (V_1, E_1)$ which is (α, γ')-expander for $\gamma' = \Theta(\frac{\log\log\log n}{\log\log\log\log n})$. Let ϵ be a constant for which $\alpha = (1-\epsilon)d$ and note that, by assumption, $\epsilon < 0.1$. Let $G_2 = (V_2, E_2)$ be a random $(n, m = n^c, d)$-hypergraph, and recall that, by Claim 7.3, with all but negligible probability G_2 α-expands every set of size s for $0.6\gamma' < s < \gamma$, for $\gamma = \Omega(n^{1-\delta})$ where $\delta = (c-1)/(d-\alpha-1)$. We let G be the $(2n, m, 2d)$-graph defined in Construction 7.13. Note that G can be sampled in time $\mathrm{poly}(n)$ with all but negligible probability.

Decoding algorithm. We describe a decoding algorithm for the $[m, m-2n]$-code whose parity-check matrix is the incidence matrix of G. Let us begin with the simplified assumption that we have an oracle that given a noisy codeword y returns 1 if the distance of y from the closest codeword is less than $0.6\gamma'$, and 0 otherwise. We will later show how to get rid of this assumption.

As before, fix $1/2 < \eta < 1$ and $\delta \geq 2\epsilon$ for which $(1-2\epsilon/\delta) > \eta$ and $1-\delta > 2\epsilon/(\eta-2\epsilon)$. We define the oracle-aided decoder as follows. At the beginning of each iteration, on noisy codeword y, the decoding algorithm calls the oracle with input y. If there are less than $0.6\gamma'$ errors, then in this iteration the decoder considers only the restriction of G to V_1 (that is, it looks at G_1), and otherwise, if there are more than $0.6\gamma'$ errors, then in this iteration the decoder considers only the restriction of G to V_2 (that is, it looks at G_2). Now, when restricted to V_i, the decoder flips (in parallel) the value of every hyperedge that has more than ηd unsatisfied constraints in V_i.

Note that, by construction, both G_1 and G_2 are good expanders. As before, this implies that at each iteration the number of errors is reduced by a constant factor. Hence there are at most $O(\log n)$ iterations.

In order to get rid of the oracle calls, we observe that there is at most one transition from G_2 to G_1. That is, if $r \cdot \log n$ is a bound on the number of iterations of the oracle-aided algorithm, where r is some constant, then there exists $i \in \{0, 1, \ldots, r \cdot \log n\}$ such that for all $j > i$, the oracle returns 1 in the j-th iteration. We get rid of the oracle by trying all possible i's, and taking the valid codeword which is closest to y.

Formally, on input x the decoding algorithm executes in parallel $r \log n + 1$ instances of the oracle-aided decoder on input x, each for at most $r \log n + 1$ rounds, such that in the i-th instance (for $i \in \{0, 1, \ldots, r \cdot \log n\}$) the oracle returns 1 in round j if and only if $j > i$. At the end of the simulation the decoding algorithm discards any instance in which there are unsatisfied constraints at the end of round $r \log n$. For all remaining instances, let $x_1, \ldots, x_\ell$ be the corresponding codewords, for $\ell \leq r \log n + 1$. For each x_i the decoder computes the distance of x_i from x (observe that this can be computed in $O(\log n)$ parallel iterations), and outputs the codeword x_i which is closest to x (again, this can be done in $O(\log n)$ parallel iterations). Finally, the correctness of the decoding algorithm follows by observing that there exists $i \in \{0, 1, \ldots, r \cdot \log n\}$ such that the i-th instance is executed exactly like the oracle-aided decoder. □

8.3 Local Non-Cryptographic Generators and Randomness Extractors

We rely on standard transformations to turn our negligible-error constructions of unbalanced expanders into negligible-error constructions of locally computable functions with various pseudorandom properties. Here "locally-computable" means that every output of the function depends on at most d inputs where d is a constant that does not grow with the input length.

t-wise independent generators. A t-wise independent generator is a function $G : \{0,1\}^n \to \{0,1\}^m$ such that for every subset $I \subseteq [m]$ of size t, where $I = \{i_1, \ldots, i_t\}$, the random variable $(G(U_n)_{i_1}, \ldots, G(U_n)_{i_t})$ is uniformly distributed over $\{0,1\}^t$, where U_n is the uniform distribution over $\{0,1\}^n$.

Theorem 8.5. *Let $c > 1$ be an arbitrary constant, $m = n^c$, and $d > 2c$ be an integer. For every prime power n there exists a negligible-error construction of a $2d$-local function (represented as a circuit) $f : \{0,1\}^{2n} \to \{0,1\}^m$ which is a t-wise independent generator for $t = \Omega(n^{1-\delta})$, for every $\delta > 2(c-1)/(d-2)$, where the constants in the Ω-notation depend on c and d.*

Proof. Sample the $2d$-sparse $2n \times m$ parity-check matrix promised by Theorem 8.2, and let $f_H(x) : x \mapsto H^T x$. This mapping is $2d$-local. Moreover, a standard linear-algebraic argument shows that the output is $(\gamma - 1)$-wise independent where γ is the distance of the sampled LDPC code. □

Low-bias generators. The notion of low-bias generators is due to Naor and Naor [43]. A random variable $X = (X_1, \ldots, X_m)$, distributed over $\{0,1\}^m$, is ϵ-biased if for every non-empty set $S \subseteq [m]$, it holds that

$$\left| \Pr\left[\bigoplus_{i \in S} X_i = 0 \right] - \frac{1}{2} \right| \leq \epsilon.$$

The set S is called a *linear test*.

A function $f : \{0,1\}^n \to \{0,1\}^m$ with $m > n$ is an ϵ-biased generator if the random variable $f(U_n)$ is ϵ-biased, where U_n is the uniform distribution over $\{0,1\}^n$.

Theorem 8.6. *Let $c > 1$ be an arbitrary constant and let $d > 4c^2 + 8c + 1$ be an integer. For every prime power n there exists a negligible-error construction of a d-local function (represented as a circuit) $f : \{0,1\}^{3n} \to \{0,1\}^{n^c}$ which is an ϵ-biased generator for $\epsilon = e^{-\Omega(n^\xi)}$, for any $\xi < \frac{r-2c}{(r-2)(2c+1)}$ for $r = (d - (2c+1)^2)/2$.*

Proof. Let $d_1 > 2c$ and let $f_1 : \{0,1\}^{2n} \to \{0,1\}^{n^c}$ be the $2d_1$-local t-wise independent generator promised in Theorem 8.5, for $t = \Omega(n^{1-\delta})$, for every $\delta > 2(c-1)/(d_1 - 2)$. Next, we employ [42, Lemma 23] that provide, for every integers p and ℓ, an ℓ^2-local generator $G : \{0,1\}^{p^2} \to \{0,1\}^m$ for $m = \binom{p}{\ell}$ so that for every linear test S the bias is at most $\exp(-|S|^{1/\ell}/2^\ell)$. We instantiate their construction with $p = \lfloor \sqrt{n} \rfloor$ and an integer $\ell \in (2c, 2c+1]$, and obtain a low-bias generator $f_2 : \{0,1\}^n \to \{0,1\}^{n^c}$ whose locality d_2 satisfies $(2c)^2 < d_2 \leq (2c+1)^2$, and its output length is n^c. (Lemma 23 of [42] gives an output length of $\binom{p}{\ell} > n^c$ but we can always omit some output bits if needed.) For a linear test S, the bias of f_2 is at most $\exp\left(-|S|^{1/(2c+1)}/2^{2c+1}\right)$.

Let $d = 2d_1 + d_2$, and let $f : \{0,1\}^{3n} \to \{0,1\}^{n^c}$ be the d-local function that is defined by $f(x,y) = f_1(x) \oplus f_2(y)$, for $x \in \{0,1\}^{2n}$ and $y \in \{0,1\}^n$. Note that since f_1 is t-wise independent, f perfectly fools every linear test of size at most t. For every larger linear test, the bias is at most ϵ by the low-bias property of f_2. Hence f is an ϵ-biased generator. □

Randomness extractors [46]. Let X be a random variable, distributed over $\{0,1\}^N$. Define the *min-entropy* of X by $H_\infty(X) := \log \frac{1}{\max_x \Pr[X=x]}$. Let $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^s \to \{0,1\}^m$ be a function. We say that Ext is a (ℓ, μ)-extractor if for every distribution X with min-entropy at least ℓ it holds that $\Delta(\mathsf{Ext}(X, U_s), U_m) \leq \mu$, that is, the statistical distance between the random variable $\mathsf{Ext}(X, U_s)$ and U_m

is at most μ. We refer to the first entry of Ext as the *source* and to the second entry as the *seed*. We show that one can locally extract N bits with negligible statistical error from an N-bit source with "almost full" min-entropy using a polynomially-small seed.

Theorem 8.7. *Let $c > 1$ be an arbitrary constant, and let $d > 4c^2 + 8c + 1$ be an integer. Let $r = r(c,d) = (d - (2c+1)^2)/2$, and let $0 < \xi < \frac{r-2c}{(r-2)(2c+1)}$ and $\gamma > 0$ be a positive constants. For a prime power n there exists a negligible-error construction of a local function Ext (represented as a circuit) that takes a source of length $N = n^c$ and a seed of length $3n$ and outputs an N-bit string which forms an (ℓ, μ)-extractor, for*

$$\ell = (1 - n^{\xi - c - \gamma})n^c \quad \textit{and} \quad \mu = \exp(-\Omega(n^\xi)).$$

Moreover, each bit of the output depends on exactly one bit of the input and d bits of the seed.

The theorem is meaningful since $\frac{r-2c}{(r-2)(2c+1)}$ is positive. (This follows by noting that $r > 2c > 2$ which is implied by the constraint $d > 4c^2 + 8c + 1$.) Also observe that $\xi < 1 < c$ and so the entropy gap $n^c - \ell$ is inverse polynomial in the source length $N = n^c$.

Remark 8.8. *In the next section, we will use the following setting of parameters. Given a constant $c > 1$, take d to be a sufficiently large constant for which $\frac{r-2c}{(r-2)(2c+1)} > \frac{1}{4c+2}$ where $r = r(c,d)$. (Such a d exists since $r = \Theta(d)$.) Take $\xi = \frac{1}{4c+2}$ and $\gamma = \xi/2 = \frac{1}{8c+4}$. Then, we obtain an (ℓ, μ)-extractor that extracts $N = n^c$ bits from a source of length N and a seed of length $3n$ for*

$$\ell = \left(1 - N^{-(1-\Omega_c(1))}\right) N \quad \textit{and} \quad \mu = \mathrm{negl}(N).$$

We continue with the proof of Theorem 8.7.

Proof. Let $f : \{0,1\}^{3n} \to \{0,1\}^{n^c}$ be the d-local generator whose bias is $e^{-\Omega(n^\xi)}$ that is promised by Theorem 8.6. Our extractor Ext takes a source X of bit-length n^c and a $3n$-bit seed S and outputs $f(S) \oplus X$. For the analysis, we employ [11, Lemma 5.7] that guarantees the following: for every $\alpha, \beta > 0$, every integers p, q, every α-biased generator $g : \{0,1\}^p \to \{0,1\}^q$, and every random variable X_q taking values in $\{0,1\}^q$ that has min-entropy at least $(1-\beta)q$, the statistical distance between $g(U_p) \oplus X_q$ and U_q is at most $\alpha \cdot 2^{\beta \cdot q/2 - 1/2}$. This immediately implies that Ext is an (ℓ, μ)-extractor with $\ell = (1 - n^{\xi - c - \gamma})n^c$ and $\mu = e^{-\Omega(n^\xi)} \cdot 2^{n^{\xi - c - \gamma} n^c / 2} = \exp(-\Omega(n^\xi))$, as required. □

9 Local PRG with Polynomial Stretch

In this section we prove Theorem 2.12, and show how to convert a local weak-PRG with polynomial-stretch to a local strong-PRG with polynomial-stretch. Combined with the weak-PRG of [4], Theorem 2.12 yields the first construction of local PRG with polynomial stretch based on a one-wayness assumption. In Section 9.1 we present the required cryptographic preliminaries, and prove the theorem in Section 9.2.

9.1 Cryptographic Preliminaries

The following is taken, with minor changes, from [4].

Indistinguishability. We say that a pair of distribution ensembles $Y = \{Y_n\}$ and $Z = \{Z_n\}$ are ϵ-indistinguishable if for every efficient adversary $\mathcal{A}$, the *distiguishability-gap*

$$|\Pr[\mathcal{A}(1^n, Y_n) = 1] - \Pr[\mathcal{A}(1^n, Z_n) = 1]|$$

is at most $\epsilon(n)$. If $\epsilon = \mathrm{negl}(n)$ we say that the two ensembles are computationally-indistinguishable.[16]

If the above holds for computationally unbounded adversaries, we say that the ensembles are ϵ-statistically-close, and in the case that $\epsilon = \mathrm{negl}(n)$ we say that the ensembles are statistically-indistinguishable.

[16] A more traditional definition (see, e.g., [23, Definition 3.2.2]), asserts that Y and Z are computationally-indistinguishable if for every efficient adversary $\mathcal{A}$ the distiguishability-gap is bounded by some negligible function $\varepsilon_{\mathcal{A}}$. In [15] it is shown that this variant is equivalent to our variant (in which $\varepsilon_{\mathcal{A}}$ is replaced with a universal negligible quantity).

Collection of functions. Let $s = s(n)$ and $m = m(n)$ be integer-valued functions. A collection of functions $\{F_\sigma\}$ is formally defined via a mapping $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ which takes an index $\sigma \in \{0,1\}^s$ and an input point $x \in \{0,1\}^n$, and outputs the evaluation $F_\sigma(x)$ of the point x under the σ-th function in the collection. We always assume that the collection is equipped with two efficient algorithms: and index-sampling algorithm K which given 1^n samples a index $\sigma \in \{0,1\}^s$, and an evaluation algorithm that given $(1^n, \sigma, x)$ outputs $F_\sigma(x)$. We say that the collection is in $\mathbf{NC^0}$ if there exists a constant d (which does not grow with n), such that for every fixed σ the function F_σ has output locality of d.

As we will always be interested in collections in $\mathbf{NC^0}$, it will be convenient to think of the index-sampling algorithm K as an algorithm that samples a circuit that implements a locally computable function f from n bits to m bits, and of the evaluation algorithm as the evaluation of the sampled circuit on input x.

Pseudorandom and unpredictability generators. Let $m = m(n) > n$ be a length parameter. A collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is *ϵ-pseudorandom generator* (PRG) if the ensemble $(K(1^n), F_{K(1^n)}(U_n))$ is ϵ-indistinguishable from the ensemble $(K(1^n), U_m)$, where U_n is the uniform distribution over $\{0,1\}^n$. When ϵ is negligible, we refer to F as a *pseudorandom generator*.

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

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

for all sufficiently large n's.

Remark 9.1. *Note that we quantified over all index sequences $\{i_n\}$, and so our notion of unpredictability is somewhat non-uniform. One may think of a uniform variant, in which $\{i_n\}$ must be* efficiently samplable*. However, it is known that the two notions are essentially equivalent, see [4, Remark 3.2]. We continue with the non-uniform variant which is easier to work with.*

We will only be interested in the case where $m = n^c$ for some $c > 1$. In this case we will refer to F as a polynomial pseudorandom generator (PPRG) when ϵ is negligible, and as a weak-PPRG in the case that ϵ is not negligible but $\epsilon < 1/n^a$ for some positive constant a.

We will always assume that the adversary that tries to break the generator gets the collection index (which we think of as a circuit) as a public parameter.

It is well known that an ϵ-PRG is also an $(\frac{1}{2} + \epsilon)$-UG. For the other direction, Yao [54] proved that a $(\frac{1}{2} + \epsilon)$-unpredictable generator of output length $m(n)$ is an $(m \cdot \epsilon)$-PRG.

9.2 Proof of Theorem 2.12

In this section we prove Theorem 2.12 (restated here for ease of reading).

Theorem 9.2 (Theorem 2.12 restated)**.** *For every constants $d \in \mathbb{N}, a > 0$ and $c, c' > 1$ there exists a constant d' for which the following holds. Any ensemble of d-local PRGs that stretches n bits to n^c bits and achieves indistinguishability parameter of $\varepsilon = 1/n^a$ can be converted into an ensemble of d'-local (standard) PRGs that stretches n bits to $n^{c'}$ bits.*

The proof of the theorem follows the analysis of [4] which is a special case of [27]. As Theorem 2.12 presents a *uniform* reduction, its proof is somewhat technical and lengthy (see Sections 9.2.1–9.2.4). For ease of reading, we first present a detailed proof sketch for a *non-uniform* reduction, that conveys the main ideas of the proof.

Proof sketch. For simplicity, let us focus on the task of constructing a local PRG G^* with *some* polynomial stretch.[17] The first ingredient of the construction is a weak PPRG $G : \{0,1\}^n \to \{0,1\}^m$ with constant

[17] This is, in fact, without loss of generality, since by using standard unpredictability and stretch amplification techniques, we may amplify the stretch to an arbitrary polynomial, at the expanse of increasing the locality, see, e.g., [4, Fact 6.5].

locality d, output-length $m = \text{poly}(n)$ and indistinguishability-gap $\epsilon = 1/\text{poly}(n)$, where, by using standard unpredictability and stretch amplification techniques, we may assume without loss of generality that ϵ is a sufficiently small inverse polynomial. The second ingredient is the locally-computable $((1-\delta(n))n, \text{negl}(n))$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{n^\alpha} \to \{0,1\}^n$ from Theorem 8.7 and Remark 8.8, for $\delta(n) = 1/n^{1-\Omega(1)}$ and some constant $0 < \alpha < 1$. We construct G^* in the following way: First, we apply G on n uniform strings $U_n^{(1)}, \ldots, U_n^{(n)}$, to obtain an $n \times m$ matrix Y, whose ith row is $G(U_n^{(i)})$, and then apply Ext to each column of Y, each time with a fresh random seed of length n^α, to obtain the output of G^*. Overall, the input length of G^* is $n^2 + m \cdot n^\alpha$, and the output length is mn, and therefore G^* has polynomial stretch. In addition, since both G and Ext are locally computable, so is G^*.

To prove that the output of G^* is indistinguishable from uniform, we proceed in three steps: First, for each row of Y, which is just the output of G on a random seed, we prove that for every index $j \in [m]$, the jth output-bit of G is computationally-indistinguishable from a random variable that has high min-entropy even conditioned on the first $j-1$ output bits of G. We then show that this implies that the jth column of Y is computationally-indistinguishable from a random variable that has high min-entropy even conditioned on the first $j-1$ columns of Y. We conclude that the output of the jth extractor is computationally-indistinguishable from uniform even conditioned on the outputs of the first $j-1$ extractors, which completes the proof.

In more details, let $S(n)$ be a circuit-size bound that is polynomial in n and let $p(n)$ be some polynomial in n. Fix some n and let $S = S(n)$ and $m = m(n)$. For the first step, let $j \in [m]$ be an index, and observe that G is a $(1/2+\epsilon)$-UG, and therefore, for every adversary $\mathcal{A}$ of size S, we have $\Pr_{x \leftarrow \{0,1\}^n}\left[\mathcal{A}(G(x)_{[1,\ldots,j-1]}) = G(x)_j\right] \leq \frac{1}{2} + \epsilon$. By the (non-uniform) hardcore lemma of Impagliazzo [30] as appears in [28], there exists a set $L_{S,p} \subseteq \{0,1\}^n$ of density $1-2\epsilon$ so that

$$\Pr_{x \leftarrow L_{S,p}}\left[\mathcal{A}(G(x)_{[1,\ldots,j-1]}) = G(x)_j\right] \leq \frac{1}{2} + \frac{1}{p(n)} \tag{3}$$

for every adversary $\mathcal{A}$ of size $S' := S/(100n \cdot p(n)^2)$. Let $R_j(x)$ be the probabilistic function that returns a random bit if $x \in L_{S,p}$, and $G(x)_j$ otherwise. We claim that

$$\left|\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(G(x)_{[1,\ldots,j]}) = 1] - \Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(G(x)_{[1,\ldots,j-1]}, R_j(x)) = 1]\right| \leq 1/p(n) \tag{4}$$

for every adversary $\mathcal{A}$ of size S'/n. Indeed, conditioned on $x \notin L_{S,p}$ both terms are equal, and therefore it is enough to analyse them conditioned on $x \in L_{S,p}$, for which the claim follows from Equation (3) and a standard unpredictability-to-indistinguishability argument.

We move on to the second step. For an input $(x_1, \ldots, x_n) \in (\{0,1\}^n)^n$, let Y be the $n \times m$ matrix defined by $G^*(x_1, \ldots, x_n)$ and let $Y[1, \ldots, j]$ denote the first j columns of Y. We also define the randomized function $C_j(x_1, \ldots, x_n)$ that returns a column-vector in $\{0,1\}^n$ whose ith entry is a random bit if $x_i \in L_{S,p}$, and $G(x_i)_j$ otherwise. By Equation (4) and a standard hybrid argument, we have that

$$\left|\Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n}[\mathcal{A}(Y[1,\ldots,j]) = 1] - \Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n}[\mathcal{A}(Y[1,\ldots,j-1], C_j(x_1,\ldots,x_n)) = 1]\right| \leq 1/(np(n))$$

for every adversary $\mathcal{A}$ of size S'/n.

We continue by proving that for all but a negligible fraction of the tuples $(x_1, \ldots, x_n)$, the random variable $C_j(x_1, \ldots, x_n)$ has very high entropy even conditioned on $(x_1, \ldots, x_n)$. Indeed, since $L_{S,p}$ has density $1-2\epsilon$, the expected number of x_i's that do not belong to $L_{S,p}$ is $2\epsilon n$. Assuming that ϵ is sufficiently small so that $2\epsilon \leq \delta/2$, a standard concentration bound implies that the probability that δ-fraction of the x_i's do not belong to $L_{S,p}$ is at most $\exp(-\Omega(\delta n)) = \exp(-n^{\Omega(1)}) = \text{negl}(n)$. Since for every $i \in [m]$ for which $x_i \in L_{S,p}$ the ith bit of $C_j(x_1, \ldots, x_n)$ is a random bit, we conclude that for all but a negligible fraction of the tuples $(x_1, \ldots, x_n)$ the random variable $C_j(x_1, \ldots, x_n)$ has min-entropy of at least $(1-\delta)n$ even conditioned on $(x_1, \ldots, x_n)$. As a consequence, the random variable $(x_1, \ldots, x_n, \mathsf{Ext}(C_j(x_1, \ldots, x_n), U_{n^\alpha}))$ is statistically-indistinguishable from $(x_1, \ldots, x_n, U_n)$, where $x_1, \ldots, x_n$ are uniformly distributed over $(\{0,1\}^n)^n$. As the

matrix Y is just a deterministic function of $x_1, \ldots, x_n$, we conclude that $(Y[1, \ldots, j-1], \mathsf{Ext}(C_j(x_1, \ldots, x_n), U_{n^\alpha}))$ is statistically-indistinguishable from $(Y[1, \ldots, j-1], U_n)$. We therefore conclude that the term

$$\left| \Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n} [\mathcal{A}(Y[1, \ldots, j-1], \mathsf{Ext}(Y[j], U_{n^\alpha})) = 1] - \Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n} [\mathcal{A}(Y[1, \ldots, j-1], U_n) = 1] \right|$$

is at most $1/(np(n)) + \mathrm{negl}(n)$ for every adversary $\mathcal{A}$ of size $S'/(nq(n))$, where $q(n)$ is some fixed polynomial that depends on the computational complexity of Ext. This immediately implies that the term

$$\left| \Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n} [\mathcal{A}(\mathsf{Ext}(Y[1]), \ldots, \mathsf{Ext}(Y[j])) = 1] - \Pr_{x_1,\ldots,x_n \leftarrow \{0,1\}^n} [\mathcal{A}(\mathsf{Ext}(Y[1]), \ldots, \mathsf{Ext}(Y[j-1]), U_n) = 1] \right|$$

is at most $1/(np(n)) + \mathrm{negl}(n)$ for every adversary $\mathcal{A}$ of size $S'/(nmq(n))$. Since this is true for every choice of polynomials $S(n)$ and $p(n)$, we conclude that $(\mathsf{Ext}(Y[1]), \ldots, \mathsf{Ext}(Y[j]))$ is computationally indistinguishable from $(\mathsf{Ext}(Y[1]), \ldots, \mathsf{Ext}(Y[j-1]), U_n)$. Since this is true for every choice of $j \in [m]$, we conclude that the output of G^* is computationally indistinguishable from uniform, and the claim follows. In the next sections we proceed with a formal proof of a *uniform* reduction.

9.2.1 The Construction

Let $G : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ be a local weak-PPRG with $m = n^c$ and distinguishability-gap $\epsilon = n^{-a}$. Then G is $(\frac{1}{2} + \epsilon)$-unpredictable, and without loss of generality we may assume that $a, c > 1$ are sufficiently large constants, at the expense of increasing the locality d to some larger constant (this follows by a standard unpredictability and stretch amplification argument, e.g., [4, Fact 6.5]). We begin by presenting the index-sampling algorithm K.

The index-sampling algorithm. The index-sampling algorithm K first uses the index-sampling algorithm of G to sample an index σ which defines a weak PPRG G_σ. Since we always assume that σ is known to the adversary, we typically omit the subscript σ for ease of notation. In addition, based on Theorem 8.7 and Remark 8.8, the index-sampler K samples a local extractor $\mathsf{Ext} : \{0,1\}^k \times \{0,1\}^{3k^\alpha} \to \{0,1\}^k$ that extracts from sources with min-entropy $(1-\delta)k$ with negligible error of $\mathrm{negl}(k)$, for some constant $0 < \alpha < 1$ and $\delta(k) = k^{-(1-\Omega(1))}$. We set $k = n$. Moreover, we assume that $\delta \geq 4m\epsilon$ which can be guaranteed by amplifying the pseudorandomness parameter ϵ to be sufficiently small inverse polynomial (at the expense of increasing the locality d). Throughout, we condition on the event that K sampled Ext successfully, which occurs with all but negligible probability. The new PRG is defined via the following evaluation algorithm.

Construction 9.3. *Given a PRG $G = G_\sigma$ and an extractor Ext, we define the function G^* as follows:*

- *Input: k independent seeds $x = (x^{(1)}, \ldots, x^{(k)}) \in (\{0,1\}^n)^k$ for the generator, and m independent seeds for the extractor $z = (z^{(1)}, \ldots, z^{(m)}) \in (\{0,1\}^{3k^\alpha})^m$.*
- *Output: compute the $k \times m$ matrix Y whose i-th row is $G(x^{(i)})$, let Y_i denote the ith column of Y, and output $(\mathsf{Ext}(Y_1; z^{(1)}), \ldots, \mathsf{Ext}(Y_m; z^{(m)}))$.*

Notice that the locality of G^* is at most the multiplication of the localities of G and Ext, and so it is constant. Furthermore, G^* has polynomial stretch. Indeed, the output length of G^* (which is mk) is polynomial in the input length (which is $nk + 3k^\alpha m$).

We continue the proof according to the following steps. First, we show a strong unpredictability property of every row of Y, and use this property to show that every column of Y is indistinguishable from a random variable with high min-entropy, even conditioned on all previous columns. Then, we show that this implies that G^* is a local-UG, and so, by Yao's theorem, G^* is a local-PRG. Finally, a standard unpredictability and stretch amplification argument (see, e.g., [4, Fact 6.5]) implies a local-PPRG with output of length $n^{c'}$.

9.2.2 Unpredictability in a row

Let $r = \lfloor \log m \rfloor + 1$ be the number of bits required to represent an index in $[m]$, and let $\ell = n + r$. Observe that $r = r(n)$ and $\ell = \ell(n)$ are functions of n, but throughout the proof we stick to the notation ℓ and r for ease of read. Define the following functions

$$\begin{aligned} &\mathsf{PRE} : \{0,1\}^\ell \to \{0,1\}^{r+m}, && \mathsf{PRE}(i,x) = i, G(x)_{1,\ldots,i-1} 0^{m-i+1}, \\ &\mathsf{N} : \{0,1\}^\ell \to \{0,1\}, && \mathsf{N}(i,x) = G(x)_i. \end{aligned}$$

That is, $\mathsf{PRE}(i,x)$ outputs the index i and the prefix of length $i-1$ of $G(x)$ (padded by zeroes), while $\mathsf{N}(i,x)$ outputs the i-th bit of $G(x)$. Note that both PRE and N are efficiently computable. From G's unpredictability it follows that for every PPT $\mathcal{A}$ and all sufficiently large n,

$$\Pr_{\substack{i \leftarrow [m] \\ x \leftarrow \{0,1\}^n}} [\mathcal{A}(\mathsf{PRE}(i,x)) = \mathsf{N}(i,x)] \leq \frac{1}{2} + \epsilon.$$

From the hardcore lemma (see [27, Proposition 4.7]) it follows that for every polynomial $p(\cdot)$ and every oracle-aided P running in time $T_P = \mathrm{poly}(n)$ and all sufficiently large n, there exists a set $L_{\ell,P} \subseteq \{0,1\}^\ell$ of density at least $1 - 2\epsilon$ such that

$$\Pr_{(i,x) \leftarrow L_{\ell,P}} \left[P^{\chi_{L_{\ell,P}}(\cdot)}(\mathsf{PRE}(i,x)) = \mathsf{N}(i,x) \right] \leq \frac{1}{2} + \frac{1}{p(n)}, \tag{5}$$

where $\chi_{L_{\ell,P}}$ is the characteristic function of $L_{\ell,P}$, provided that the queries of P to $\chi_{L_{\ell,P}}$ are computed independently of the input $\mathsf{PRE}(i,x)$.

For every set $L \subseteq \{0,1\}^\ell$ define the probabilistic function R_L by

$$\mathsf{R}_L(i,x) = \begin{cases} \mathsf{N}(i,x) & \text{if } (i,x) \notin L \\ \text{random bit} & \text{if } (i,x) \in L \end{cases}.$$

The following lemma shows that for every adversary D there is a set L of density $1 - 2\epsilon$ such that $(\mathsf{PRE}(U_\ell), \mathsf{N}(U_\ell))$ is indistinguishable from the distribution $(\mathsf{PRE}(U_\ell), \mathsf{R}_L(U_\ell))$. The lemma follows from the hardcore lemma, similarly to [27, Proposition 4.8]. We prove it here for completeness.

Lemma 9.4. *For every polynomial $p(\cdot)$ and every oracle-aided distinguisher D running in time $T_D = \mathrm{poly}(n)$, and for all sufficiently large n, there is a set $L_{\ell,D} \subseteq \{0,1\}^\ell$ of density at least $1 - 2\epsilon$ for which the following holds. Let $\mathcal{O}$ be an oracle that samples $i \leftarrow [m]$ and $x \leftarrow \{0,1\}^n$ and returns a sample from $(\mathsf{PRE}(i,x), \mathsf{N}(i,x), \mathsf{R}_{L_{\ell,D}}(i,x))$. Then, given an oracle access to $\mathcal{O}$, the distinguisher $D^{\mathcal{O}}$ cannot distinguish*

$$(\mathsf{PRE}(U_\ell), \mathsf{N}(U_\ell)) \qquad \textit{from} \qquad (\mathsf{PRE}(U_\ell), \mathsf{R}_{L_{\ell,D}}(U_\ell))$$

with more then $1/p(n)$ advantage.

Proof. Assume that there exists a distinguisher D of time $T_D = \mathrm{poly}(n)$ and a polynomial $p(\cdot)$, such that for infinitely many n's, and every L of density $1 - 2\epsilon$,

$$\left| \Pr_{(i,x) \leftarrow \{0,1\}^\ell} [D^{\mathcal{O}}(\mathsf{PRE}(i,x), \mathsf{N}(i,x)) = 1] - \Pr_{(i,x) \leftarrow \{0,1\}^\ell} [D^{\mathcal{O}}(\mathsf{PRE}(i,x), \mathsf{R}_L(i,x)) = 1] \right| > 1/p(n).$$

Since both terms are equal for $(i,x) \notin L$, we have

$$\left| \Pr_{(i,x) \leftarrow L} [D^{\mathcal{O}}(\mathsf{PRE}(i,x), \mathsf{N}(i,x)) = 1] - \Pr_{(i,x) \leftarrow L} [D^{\mathcal{O}}(\mathsf{PRE}(i,x), \mathsf{R}_L(i,x)) = 1] \right| > 1/p(n).$$

Since $\mathsf{R}_L(i,x)$ is uniformly distributed for $(i,x) \in L$, by the standard distinguishing to predicting reduction, there exists a predictor P with oracle call to χ_L and time $T_P = \mathrm{poly}(T_D) = \mathrm{poly}(n)$ (note that P can

simulate the oracle calls of D efficiently, using χ_L) such that for infinitely many n's, and every set L of density at least $1-2\epsilon$,

$$\Pr_{(i,x)\leftarrow L}\left[P^{\chi_L(\cdot)}(\mathsf{PRE}(i,x))=\mathsf{N}(i,x)\right]\geq\frac{1}{2}+\frac{1}{p(n)},$$

in contradiction to Equation 5. □

9.2.3 Indistinguishability in columns

In this section we prove that, even conditioned on the first $i-1$ columns of Y, the output of the i-th extractor is indistinguishable from uniform (Lemma 9.8). In order to prove this, we first show that, even conditioned on all previous columns, the i-th column of Y is indistinguishable from a random variable with high min-entropy (Claim 9.5 and Corollary 9.7).

We denote the first $i-1$ columns of Y by $Y[1:i-1]$, and as before, we denote the i-th column of Y by Y_i. We begin with the following definitions. For any $S\subseteq\{0,1\}^\ell$ define

$$\mathsf{PRE}^k:\{0,1\}^{r+n\times k}\to\{0,1\}^{k\times(r+m)},\qquad \mathsf{PRE}^k(i,x_1,\ldots,x_k)=\begin{bmatrix}\mathsf{PRE}(i,x_1)\\ \vdots\\ \mathsf{PRE}(i,x_k)\end{bmatrix},$$

$$\mathsf{N}^k:\{0,1\}^{r+n\times k}\to\{0,1\}^{k},\qquad \mathsf{N}^k(i,x_1,\ldots,x_k)=\begin{bmatrix}\mathsf{N}(i,x_1)\\ \vdots\\ \mathsf{N}(i,x_k)\end{bmatrix},$$

$$\mathsf{R}^k_S:\{0,1\}^{r+n\times k}\to\{0,1\}^{k},\qquad \mathsf{R}^k_S(i,x_1,\ldots,x_k)=\begin{bmatrix}\mathsf{R}_S(i,x_1)\\ \vdots\\ \mathsf{R}_S(i,x_k)\end{bmatrix}.$$

That is, $\mathsf{PRE}^k(i,x_1,\ldots,x_k)$ corresponds to the first $i-1$ columns of Y (padded with zeroes), and N^k corresponds to the i-th column of Y. Also, each entry of $\mathsf{R}^k_S(i,x_1,\ldots,x_k)$ uses independent internal randomness. The following claim shows that for every adversary D there exists a set S of high density such that $(\mathsf{PRE}^k(U_{r+nk}),\mathsf{N}^k(U_{r+nk}))$ is indistinguishable from $(\mathsf{PRE}^k(U_{r+nk}),\mathsf{R}^k_S(U_{r+nk}))$.

Claim 9.5. *For every polynomial $p(\cdot)$, every probabilistic polynomial-time algorithm D and all sufficiently large n, there is a set $S_{\ell,D}\subseteq\{0,1\}^\ell$ of density at least $1-2\epsilon$, such that D cannot distinguish*

$$(\mathsf{PRE}^k(U_{r+nk}),\mathsf{N}^k(U_{r+nk}))\quad\textit{from}\quad(\mathsf{PRE}^k(U_{r+nk}),\mathsf{R}^k_{S_{\ell,D}}(U_{r+nk}))$$

with more than $1/p(n)$ advantage.

Proof. Assume towards contradiction that there exists a polynomial $p(\cdot)$ and a distinguisher D such that for infinitely many n's, and any $S\subseteq\{0,1\}^\ell$ of density at least $1-2\epsilon$, the adversary D distinguishes $(\mathsf{PRE}^k(U_{r+nk}),\mathsf{N}^k(U_{r+nk}))$ from $(\mathsf{PRE}^k(U_{r+nk}),\mathsf{R}^k_S(U_{r+nk}))$ with more than $1/p(n)$ advantage. We will show that this implies a violation of Lemma 9.4.

Define the following oracle-aided adversary $\mathcal{A}_D$. Let $\mathcal{O}$ be an oracle that samples $i\leftarrow[m]$ and $x\leftarrow\{0,1\}^n$ and returns a sample from $(\mathsf{PRE}(i,x),\mathsf{N}(i,x),\mathsf{R}_L)$, where L will be determined later. On input $(i,y,z)\in[m]\times\{0,1\}^m\times\{0,1\}$, the algorithm $\mathcal{A}_D$ samples m^2k^2 triples $\{(\mathsf{PRE}(i_j,x_j),\mathsf{N}(i_j,x_j),\mathsf{R}_L(i_j,x_j))\}_{j\in[m^2k^2]}$ using the oracle calls to $\mathcal{O}$. If the number of indices j such that $i_j=i$ is less than k, then $\mathcal{A}_D$ fails. Otherwise, denote the corresponding triples by $\{(\mathsf{PRE}(i,x_j),\mathsf{N}(i,x_j),\mathsf{R}_L(i,x_j))\}_{j\in[k]}$. Then $\mathcal{A}_D$ samples

$j \leftarrow [k]$, simulates D on input

$$\begin{pmatrix} \mathsf{PRE}(i,x_1), \mathsf{N}(i,x_1) \\ \vdots \\ \mathsf{PRE}(i,x_{j-1}), \mathsf{N}(i,x_{j-1}) \\ i, y, z \\ \mathsf{PRE}(i,x_{j+1}), \mathsf{R}_L(i,x_{j+1}) \\ \vdots \\ \mathsf{PRE}(i,x_k), \mathsf{R}_L(i,x_k) \end{pmatrix},$$

and outputs the same value as D. Set L to be the set $L_{\ell,\mathcal{A}_D}$ promised in Lemma 9.4, with respect to the polynomial $2p(n)k$. By Chernoff bound the probability that $\mathcal{A}_D$ fails is negligible, and so, by a standard hybrid argument $\mathcal{A}_D$ distinguishes $(\mathsf{PRE}(U_\ell), \mathsf{N}(U_\ell))$ from $(\mathsf{PRE}(U_\ell), \mathsf{R}_{L_{\ell,\mathcal{A}_D}}(U_\ell))$ with advantage $1/(p(n)k) - \mathrm{negl}(n) > 1/(2p(n)k)$, in contradiction to Lemma 9.4. □

Next, we prove that for any set S of density $1-2\epsilon$, the random variable $\mathsf{R}^k_S(U_{r+nk})$ has high min-entropy even conditioned on $\mathsf{PRE}^k(U_{r+nk})$.

Claim 9.6. *For every n, every set $S \subseteq \{0,1\}^\ell$ of density $1-2\epsilon$, and every $i \in [m]$, there exists a set $\mathbf{BAD} \subseteq \{0,1\}^{k\times n}$ such that*

- $\Pr[(x_1,\ldots,x_k) \in \mathbf{BAD}] \leq e^{-\Omega(k\delta)} = \mathrm{negl}(n)$,
- *If $(x_1,\ldots,x_k) \notin \mathbf{BAD}$ then the random variable $R^k_S(i,x_1,\ldots,x_k)$ has min-entropy at least $k(1-\delta)$ even conditioned on $x_1,\ldots,x_k$.*

Proof. Since S is of density $1-2\epsilon$ then for every $i \in [m]$ there is at least $(1-2\epsilon m)$ fraction of the x's for which $(i,x) \in S$. Say that the sample $(i,x_1,\ldots,x_k)$ is "good" if $|\{j : (i,x_j) \notin S\}| < k\delta$. Note that the expected size of the set $\{j : (i,x_j) \notin S\}$ is at most $2km\epsilon \leq k\delta/2$, and since the x_j's are independent, by multiplicative Chernoff bound $\Pr[(i,x_1,\ldots,x_k)$ is "bad"$] \leq e^{-\Omega(k\delta)} = e^{-k^{\Omega(1)}} = \mathrm{negl}(n)$. Finally, for a "good" $(i,x_1,\ldots,x_k)$, the random variable $\mathsf{R}^k_S(i,x_1,\ldots,x_k)$ has at least $k(1-\delta)$ random bits, and so its min-entropy is at least $k(1-\delta)$, as required. □

The following corollary follows immediately from Claim 9.6 and the properties of our extractor. (From now on, we omit the uniform seed from the extractor's description and, for a k-bit source X, write $\mathsf{Ext}(X)$ as a shorthand for $\mathsf{Ext}(X, U_{3k^\alpha})$.)

Corollary 9.7. *For every set S of density $1-2\epsilon$ and every $i \in [m]$, the random variables $(\mathsf{PRE}^k(i,U_{nk}), U_k)$ and $(\mathsf{PRE}^k(i,U_{nk}), \mathsf{Ext}(\mathsf{R}^k_S(i,U_{nk})))$ are statistically indistinguishable.*

Finally, we show that for every $i \in [m]$, the output of the i-th extractor is indistinguishable from uniform, even conditioned on the first $i-1$ columns of Y.

Lemma 9.8. *For every polynomial $p(\cdot)$, probabilistic polynomial-time adversary D, (non-uniform) family of indices $\{i_n\}$ and all sufficiently large n, D cannot distinguish $(Y[1:i_n-1], \mathsf{Ext}(Y_{i_n}))$ from $(Y[1:i_n-1], U_k)$ with more than $1/p(n)$ advantage.*

Proof. Assume towards contradiction that there exist polynomial $p(\cdot)$ and adversary D such that for infinitely many n's D distinguishes $(Y[1:i_n-1], \mathsf{Ext}(Y_{i_n}))$ from $(Y[1:i_n-1], U_k)$ with advantage at least $1/p(n)$, where $\{i_n\}$ is a non-uniform sequence of indices. For each $i \in [m]$ let

$$\mu(i) := \Pr[D(Y[1:i-1], \mathsf{Ext}(Y_i)) = 1] - \Pr[D(Y[1:i-1], U_k) = 1],$$

and note that, without loss of generality, $\mu(i_n) > 1/p(n)$. We show that this implies a violation Claim 9.5.

Consider the following adversary $\mathcal{A}$. On input $(i,g,u) \in [m] \times \{0,1\}^{k\times m} \times \{0,1\}^k$, the adversary $\mathcal{A}$ first approximates $\mu(i)$ within additive error $1/4p(n)$ and confidence of $1-\mathrm{negl}(n)$. By a standard Chernoff

bound, this can be done efficiently. Denote the approximation by $\tilde{\mu}(i)$. If $\tilde{\mu}(i) > 3/4p(n)$ then $\mathcal{A}$ simulates D on input $(g[1:i-1], \mathsf{Ext}(u))$ and outputs the same bit as D. Otherwise $\mathcal{A}$ outputs a random bit.

We claim that $\mathcal{A}$ distinguishes $(\mathsf{PRE}^k(U_{r+nk}), \mathsf{N}^k(U_{r+nk}))$ from $(\mathsf{PRE}^k(U_{r+nk}), \mathsf{R}^k_{S_{\ell,\mathcal{A}}}(U_{r+nk}))$ with more than $1/2mp(n)$ advantage, where $S_{\ell,\mathcal{A}}$ is the set promised in Claim 9.5 with respect to adversary $\mathcal{A}$ and the polynomial $1/2mp(n)$.

Let us condition on the event that the approximation of $\mathcal{A}$ does not fail, i.e., $\tilde{\mu}(i) \in [\mu(i) \pm 1/4p(n)]$. Fix some i. First, observe that if $(i, g, u) \leftarrow (\mathsf{PRE}^k(i, U_{nk}), \mathsf{N}^k(i, U_{nk}))$, then $(g[1:i-1], \mathsf{Ext}(u))$ is distributed according to $(Y[1:i-1], \mathsf{Ext}(Y_i))$, and if $(i, g, u) \leftarrow (\mathsf{PRE}^k(i, U_{nk}), \mathsf{R}^k_{S_{\ell,\mathcal{A}}}(i, U_{nk}))$, then $(g[1:i-1], \mathsf{Ext}(u))$ is statistically indistinguishable from $(Y[1:i-1], U_k)$ by Corollary 9.7. Next, let us distinguish between three cases depending on the value of $\mu(i)$. If $\mu(i) > 1/p(n)$, the adversary $\mathcal{A}$ always calls to D, and so the advantage of $\mathcal{A}$ is at least $1/p(n) - \mathrm{negl}(n) > 0$, and indeed, for $i = i_n$, the adversary $\mathcal{A}$ gets this advantage. If $1/2p(n) < \mu(i) < 1/p(n)$, the adversary might either output a random bit or call to D. Since this choice depends only on i (and is independent of the other inputs of $\mathcal{A}$), the adversary has an advantage of 0 in the former case, and of $\mu(i) - \mathrm{negl}(n)$ in the latter case, and overall the advantage is non-negative in any case. Finally, if $\mu(i) < 1/2p(n)$, the adversary always returns a random bit, and so its advantage is 0.

Since the total advantage of $\mathcal{A}$ is a uniform mixture of the above terms, we conclude that the total advantage is at least $1/mp(n) - \mathrm{negl}(n) > 1/2mp(n)$. This contradicts Claim 9.5. □

9.2.4 From Columns Indistinguishability to PRG

We show that the above notion of columns indistinguishability implies that G^* is indeed a UG. Recall that $m = m(n)$ and $k = k(n)$ are functions of n, let $n'(n) := nk + 3k^\alpha m$ be the input-length of G^*, and let $m'(n) := mk$ be the output-length of G^*.

Lemma 9.9. *For every polynomial $p(\cdot)$, probabilistic polynomial-time predictor $\mathcal{A}$, non-uniform sequence $\{i_n\}$, and all sufficiently large n,*

$$\Pr_{x \leftarrow \{0,1\}^{n'}}[\mathcal{A}(G^*(x)_{[1\ldots i_n-1]}) = G^*(x)_{i_n}] < 1/2 + 1/p(n).$$

Proof. Assume towards contradiction that there exists a polynomial $p(\cdot)$, a predictor $\mathcal{A}$ and a non-uniform sequence $\{i_n\}$ such that for infinitely many n's, $\Pr[\mathcal{A}(G^*(x)_{[1\ldots i_n-1]}) = G^*(x)_{i_n}] > 1/2 + 1/p(n)$. Let

$$\mu(i) := \Pr[\mathcal{A}(G^*(x)_{[1\ldots i-1]}) = G^*(x)_i],$$

and note that $\mu(i_n) > 1/2 + 1/p(n)$. Let $\{j_n\}$ be a non-uniform sequence of columns' indices, so that i_n corresponds to the j_n-th column of Y (that is, $k \cdot (j_n - 1) < i_n \leq k \cdot j_n$). We show that there exists an adversary $\mathcal{B}$ that violates Lemma 9.8.

Given an input (g, u), where $g = (g_1, \ldots, g_{j_n-1})$ is a $k \times (j_n - 1)$ matrix and u is a vector of length k, the adversary $\mathcal{B}$ first approximates $\mu(i)$ for each $k \cdot (j_n - 1) < i \leq k \cdot j_n$, within additive error $1/4p(n)$ and confidence of $1 - \mathrm{negl}(n)$. (By standard Chernoff bounds, this can be done efficiently.) Denote the approximation by $\tilde{\mu}(i)$. If there exists $k \cdot (j_n - 1) < i^* \leq k \cdot j_n$ such that $\tilde{\mu}(i^*) > 1/2 + 3/4p(n)$, then $\mathcal{B}$ computes $w := (\mathsf{Ext}(g_1), \ldots, \mathsf{Ext}(g_{j_n-1}), u)$, and $z := \mathcal{A}(w_{[1\ldots i^*-1]})$, and outputs 1 if z is equal to w_{i^*}, and 0 otherwise. If there is no such i^* then $\mathcal{B}$ outputs a random bit.

Observe that when (g, u) is sampled from $(Y[1:j_n-1], U_k)$, then the probability that $\mathcal{A}$ predicts the i^* bit is exactly $1/2$. Therefore, when the input is sampled from $(Y[1:j_n-1], U_k)$ the output of $\mathcal{B}$ is uniformly distributed. It remains to show that when (g, u) is sampled from $(Y[1:j_n-1], \mathsf{Ext}(Y_{j_n}))$ then $\mathcal{B}$ outputs 1 with probability at least $1/2 + 1/\mathrm{poly}(n)$.

When $(g, u) \leftarrow (Y[1:j_n-1], \mathsf{Ext}(Y_{j_n}))$, the random variable w is distributed exactly like $G^*(x)_{[1\ldots k\cdot j_n]}$. Condition on the event that the approximation of $\mathcal{B}$ does not fail, and observe that (1) if there exists i^* such that $\tilde{\mu}(i^*) > 1/2 + 3/4p(n)$ then necessarily $\mu(i^*) > 1/2 + 1/2p(n)$; and (2) such i^* necessarily exists since $\tilde{\mu}(i_n) > 1/2 + 3/4p(n)$. Therefore, $\mathcal{B}$ outputs 1 with probability at least $1/2 + 1/2p(n)$. We conclude that $\mathcal{B}$ has advantage of at least $1/2 + 1/2p(n) - 1/2 = 1/2p(n)$ in distinguishing $(Y[1:j_n-1], U_k)$ from $(Y[1:j_n-1], \mathsf{Ext}(Y_{j_n}))$, in contradiction to Lemma 9.8. □

By Yao's theorem [54] that unpredicatability implies pseudorandomness, we conclude that for every polynomial $p(\cdot)$, probabilistic polynomial-time distinguisher $\mathcal{A}$, and all sufficiently large n,

$$|\Pr[\mathcal{A}(G^*(U_{n'})) = 1] - \Pr[\mathcal{A}(U_{m'}) = 1]| < 1/p(n).$$

Finally, by Footnote 16, we conclude that there exists a negligible function $\nu(\cdot)$ so that G^* is ν-PRG. This completes the proof of Theorem 2.12. □

References

[1] Noga Alon. Explicit ramsey graphs and orthonormal labelings. *the electronic journal of combinatorics*, 1(1):12, 1994.

[2] Noga Alon and Asaf Nussboim. k-wise independent random graphs. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 813–822. IEEE Computer Society, 2008.

[3] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 152–181. Springer, 2017.

[4] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM Journal on Computing*, 42(5):2008–2037, 2013.

[5] Benny Applebaum. Cryptographic hardness of random local functions - survey. *Computational Complexity*, 25(3):667–722, 2016.

[6] Benny Applebaum. Exponentially-hard gap-csp and local PRG via local hardcore functions. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 836–847. IEEE Computer Society, 2017.

[7] 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.

[8] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *Annual International Cryptology Conference (CRYPTO)*, pages 223–254. Springer, 2017.

[9] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in NC^0. *Computational Complexity*, 17(1):38–69, 2008.

[12] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. *SIAM Journal on Computing*, 47(1):52–79, 2018.

[13] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. *Commun. ACM*, 54(5):101–107, 2011.

[14] Boaz Barak, Anup Rao, Ronen Shaltiel, and Avi Wigderson. 2-source dispersers for sub-polynomial entropy and ramsey graphs beating the frankl-wilson construction. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 671–680. ACM, 2006.

[15] Bellare. A note on negligible functions. *J. Cryptol.*, 15(4):271284, 2002.

[16] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector ole. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 896–912, New York, NY, USA, 2018. ACM.

[17] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2002.

[18] Changyan Di, David Proietti, I. Emre Telatar, Thomas J. Richardson, and Rüdiger L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Information Theory*, 48(6):1570–1579, 2002.

[19] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[20] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.

[21] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962.

[22] Oded Goldreich. Candidate one-way functions based on expander graphs. *IACR Cryptology ePrint Archive*, 2000:63, 2000.

[23] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[24] Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM J. Comput.*, 39(7):2761–2822, 2010.

[25] Oded Goldreich, Noam Nisan, and Avi Wigderson. On yao's xor-lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(50), 1995.

[26] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009.

[27] Iftach Haitner, Omer Reingold, and Salil Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM Journal on Computing*, 42(3):1405–1430, 2013.

[28] Thomas Holenstein. Key agreement from weak bit agreement. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 664–673, 2005.

[29] Xiao-Yu Hu, Marc PC Fossorier, and Evangelos Eleftheriou. On the computation of the minimum distance of low-density parity-check codes. In *Communications, 2004 IEEE International Conference on*, volume 2, pages 767–771. IEEE, 2004.

[30] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[31] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[32] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 734–743. IEEE Computer Society, 1998.

[33] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 262–271. ACM, 2004.

[34] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 433–442. ACM, 2008.

[35] Neeraj Kayal. *Derandomizing some number-theoretic and algebraic algorithms*. Phd, Indian Institute of Technology, 2007. Available in `https://eccc.weizmann.ac.il/resources/pdf/kayal.pdf`.

[36] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[37] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 28–57. Springer, 2016.

[38] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 11–20. IEEE, 2016.

[39] Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, Daniel A Spielman, et al. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on information Theory*, 47(2):585–598, 2001.

[40] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 253–262. IEEE, 2013.

[41] Grigorii A Margulis. Explicit constructions of graphs without short cycles and low density codes. *Combinatorica*, 2(1):71–78, 1982.

[42] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ε-biased generators in NC^0. *Random Structures & Algorithms*, 29(1):56–81, 2006.

[43] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM journal on computing*, 22(4):838–856, 1993.

[44] Moni Naor and Asaf Nussboim. Implementing huge sparse random graphs. In Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, volume 4627 of *Lecture Notes in Computer Science*, pages 596–608. Springer, 2007.

[45] Moni Naor, Asaf Nussboim, and Eran Tromer. Efficiently constructible huge graphs that preserve first order properties of random graphs. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 66–85. Springer, 2005.

[46] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[47] Ruud Pellikaan, Xin-Wen Wu, Stanislav Bulygin, and Relinde Jurrius. *Codes, Cryptology and curves with computer algebra*, volume 1. Cambridge University Press, 2017.

[48] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[49] Michael Sipser and Daniel A Spielman. Expander codes. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 566–576. IEEE, 1994.

[50] Daniel A Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[51] DR Stinson, Ruizhong Wei, and Maura B Paterson. Combinatorial batch codes. *Advances in Mathematics of Communications*, 3(1):13–27, 2009.

[52] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, pages 129–138. IEEE Computer Society, 2002.

[53] Salil P. Vadhan. *Pseudorandomness*, volume 7. Foundations and Trends in Theoretical Computer Science, 2012.

[54] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.

[55] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

A On Uniformly Sampling H-free Hypergraphs

It is interesting to ask whether one can extend the results of this paper to random (n, m, d) hypergraphs, i.e., hypergraphs where each edge is uniformly distributed over $[n]^d$ and the hyperedges are independent. In particular, is it possible to sample a random (n, m, d)-hypergraph which is $\mathcal{H}$-free for general family $\mathcal{H}$ of small hypergraphs?

Clearly, this problem can be solved with the aid of an efficient tester algorithm that determines whether an (n, m, d)-hypergraph is $\mathcal{H}$-free or not. Unfortunately, the testing problem seems hard over the uniform distribution. In fact, we conjecture that even the seemingly easier task of *certifying* $\mathcal{H}$-freeness is hard, and show that, under this assumption, the existence of a sampler for $\mathcal{H}$-free hypergraphs implies the existence of one-way functions. Before stating this formally, we need the following definitions.

Certification and sparsity. Fix some distribution D over some objects (without loss of generality strings) and let P be some property of strings which is *common* over D in the sense that $\Pr[D \notin P] < 0.1$. A *certification* algorithm for P over D is an algorithm that, given a string, either outputs "good" or "I don't know". The algorithm should never err (i.e., it can output "good" only on inputs that satisfy P) and should output "good" with probability at least half on a sample from D.

In our context, D will be the uniform distribution over (n, m, d)-hypergraphs, for some constant d and function $m = m(n)$. The property P will be $\mathcal{H}$-freeness where $\mathcal{H}$ is some family of forbidden subgraphs. That is, a certification algorithm provides a certificate for $\mathcal{H}$-freeness for at least half of the hypergraphs. To guarantee that $\mathcal{H}$-freeness is common over random (n, m, d)-hypergraphs, we introduce the following notion of sparsity. Let $\mathcal{H} = \{\mathcal{H}_i\}_{i\in\mathbb{N}}$ be a family of hypergraphs where $\mathcal{H}_i$ is a set of $\leq d$-uniform hypergraphs. Let $h : \mathbb{N} \to \mathbb{N}$. We say that the family $\mathcal{H}$ is *h-sparse* over (n, m, d)-hypergraphs if for every n and every $t \leq h(n)$, a randomly chosen hypergraph $G \leftarrow \mathcal{G}_{n,m(n),d}$ is $\mathcal{H}_t$-free with probability at least 0.9.

Theorem A.1. *Let d be a constant, $m = m(n)$ be a polynomial, and let $\mathcal{H} = \{\mathcal{H}_i\}_{i\in\mathbb{N}}$ be a family of d-uniform hypergraphs. Assume that $\mathcal{H}$ is h-sparse over (n, m, d)-hypergraphs for some sparsity function $h(n)$, and suppose that the following conditions hold for some function $t(n) \leq h(n)$.*

1. *($\mathcal{H}$-freeness is hard to certify): There is no $\mathrm{poly}(n)$-time algorithm that, for infinitely many n's, certifies $\mathcal{H}_{t(n)}$-freeness over $(n, m(n), d)$-hypergraphs.*
2. *(Random $\mathcal{H}$-free graphs are easy to sample): There exists a $\mathrm{poly}(n)$-time algorithm $\mathcal{S}$ that samples a random $(n, m(n), d)$-hypergraph conditioned on the event that the graph is $\mathcal{H}_{t(n)}$-free (and outputs "Error" with negligible probability).*

Then, one-way functions exist.

Proof. Let f_n be the function that maps the random coins r consumed by $\mathcal{S}(1^n)$ to the output of the sampler, i.e., a string that represents an (n, m, d)-hypergraph G or a special failure symbol. We claim that $f = \{f_n\}$ cannot be inverted with probability more than $2/3$ by an efficient adversary. (Such a weak one-wayness property can be amplified to strong one-wayness via standard transformations, cf. [23]).

Assume, towards a contradiction, that there exists a $\mathrm{poly}(n)$-time adversary $\mathcal{A}$ and an infinite set of integer N such that, for all $n \in N$, the adversary $\mathcal{A}$ inverts f_n with probability at least $2/3$ where the probability is taken over the random input r and the internal coins of $\mathcal{A}$. We construct an efficient algorithm $\mathcal{R}$ that certifies $\mathcal{H}_{t(n)}$-freeness for all $n \in N$, in contradiction to our assumption. The certification algorithm $\mathcal{R}$ is given an $(n, m(n), d)$-hypergraph G as an input, and asks the inverter $\mathcal{A}$ to find a preimage of G under f_n. Given the result r, the algorithm $\mathcal{R}$ outputs "free" if and only if $f_n(r) = G$. Otherwise, $\mathcal{R}$ outputs "I don't know".

Since $\mathcal{H}_t$ is h-sparse and $\mathcal{A}$ inverts f with probability at least $2/3$, it follows that, on a random (n, m, d)-hypergraph, $\mathcal{R}$ outputs "I don't know" with probability at most $1/3 + 0.1 < 1/2$. Furthermore, $\mathcal{R}$ outputs "free" only if G is in the support of $\mathcal{S}$, and since $\mathcal{S}$ samples only $\mathcal{H}_t$-free hypergraphs, it follows that G must be $\mathcal{H}_t$-free, and $\mathcal{R}$ never errs. The theorem follows. □

Hardness of certifying $\mathcal{H}$-free. We propose a candidate for a family $\mathcal{H}$ for which $\mathcal{H}$-freeness is hard to certify based on the following coding-related assumption:

Assumption A.2 (random LDPC's distance is hard to certify)**.** *For every polynomial $n < m(n) \leq \mathrm{poly}(n)$, constant $d > 2\log_n m$ and every super-constant function $t(n) > \omega(1)$, there is no efficient algorithm A that given a random $n \times m(n)$ binary parity-check matrix M with d ones in each column, certifies that the distance of the corresponding code is at least $t(n)$. That is, a $\mathrm{poly}(n)$-time algorithm fails on all sufficiently large n's.*

It is known that for $d > 2\log_n m$, such a random LDPC code is likely to have a minimal-distance of $h = n^\epsilon$, where ϵ is a constant that depends on d and $2\log_n m$. Using graph-theoretic terminology, we can think of M as the incidence matrix of a random (n, m, d)-hypergraph G_M. The distance of the code is (at least) t if and only if G_M is $\mathcal{H}_t$-free, where $\mathcal{H}_t$ is the family of all d-uniform hypergraphs with at most t hyperedges in which each vertex has an *even* degree. Since most codes have a distance of at least h, the family $\mathcal{H}$ is h-sparse over $(n, m(n), d)$-hypergraphs.

Assumption A.2 therefore yields a family $\mathcal{H}$ that satisfies the first condition of Theorem A.1. We mention that related worst-case problems (e.g., computing the minimal-distance of a linear code or approximating it within a multiplicative constant) are known to be NP-hard [29, 20]. Closely related average-case hardness assumptions were also used in [7, 13, 9].

B Proof of Claim 7.3

Let $S \subseteq [m]$ be a set of s hyperedges, and let $T \subseteq [n]$ be a set of $\alpha \cdot s$ vertices. The probability that $\Gamma(S) \subseteq T$ is exactly $(\alpha s/n)^{ds}$. Taking union bound over all S of size s and T of size $\alpha \cdot s$, the probability that there

exists a set S of size s which is not α-expanding is at most

$$\binom{n^c}{s}\binom{n}{\alpha s}\left(\frac{\alpha s}{n}\right)^{ds} \leq \left(\frac{en^c}{s}\right)^s \left(\frac{en}{\alpha s}\right)^{\alpha s}\left(\frac{\alpha s}{n}\right)^{ds} = \left(a_{\alpha,d}\frac{s^{d-\alpha-1}}{n^{d-c-\alpha}}\right)^s,$$

where we used the known inequality $\binom{n}{k} \leq (en/k)^k$ and $a_{\alpha,d}$ is a constant that depends on α and d.

Finally, for sufficiently small ρ (e.g. $\rho = (1/3a_{\alpha,d})^{1/(d-\alpha-1)}$), by taking union-bound we get that the probability the sampled graph is not (α,γ)-expander is at most $1/2$. □

C Proof of Lemma 8.4

Let G be an (n,m,d) hypergraph, $\alpha = (1-\epsilon)d$ for $\epsilon < 1/2$, and assume that every set $S \subseteq [m]$, where $\gamma' \leq |S| \leq \gamma$ is α-expanding.

- For the first part, assume that there exist $\delta > 2\epsilon$ and a set S for which the claim does not hold. We will prove that the set S is not α-expanding, in contradiction. Indeed, since there are less than $(1-\delta)$ fraction of the vertices of S that have at least $(1-2\epsilon/\delta)d$ unique neighbors, we conclude that the maximal number of neighbors of S is

  $$|\Gamma(S)| < d\cdot(1-\delta)|S| + (1-2\epsilon/\delta)d\cdot\delta|S| = (1-2\epsilon)d|S| < \alpha|S|,$$

  in contradiction to the expanding property of S.

- For the second part, assume that there exist $\eta > 2\epsilon$ and a set S for which the claim does not hold. Let F be the set of hyperedges not in S that have at least ηd vertices in $\Gamma(S)$, so that $|F| \geq (2\epsilon/(\eta-2\epsilon))|S|$. Let $F' \subseteq F$ be a subset of F of size $(2\epsilon/(\eta-2\epsilon))|S|$. Observe that

  $$\gamma' \leq |S\cup F'| = |S| + \frac{2\epsilon}{\eta-2\epsilon}|S| = \frac{\eta}{\eta-2\epsilon}|S| \leq \frac{\eta}{\eta-2\epsilon}\cdot\frac{\eta-2\epsilon}{\eta}\cdot\gamma = \gamma.$$

  We conclude that $S\cup F'$ is α-expanding, so $|\Gamma(S\cup F')| \geq (1-\epsilon)d\cdot|S\cup F'| = (1-\epsilon)d\cdot\frac{\eta}{\eta-2\epsilon}|S|$. On the other hand $|\Gamma(S)| \leq d|S|$ and $|\Gamma(F')\setminus\Gamma(S)| \leq d|F'| - \eta d|F'| = (1-\eta)d|F'| = (1-\eta)d\cdot\frac{2\epsilon}{\eta-2\epsilon}|S|$. We conclude that

  $$(1-\epsilon)d\cdot\frac{\eta}{\eta-2\epsilon}|S| \leq \Gamma(S\cup F') \leq d|S| + (1-\eta)d\cdot\frac{2\epsilon}{\eta-2\epsilon}|S|,$$

  which contradicts the fact that ϵ and η are both positive.

This concludes the proof of Lemma 8.4. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il