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

# Sampling Graphs without Forbidden Subgraphs and Almost-Explicit Unbalanced Expanders

Benny Applebaum* Eliran Kachlon*

**Abstract**

We initiate the study of the following hypergraph sampling problem: Sample a $d$-uniform hypergraph over $n$ vertices and $m$ hyperedges from some pseudorandom distribution $\mathcal{G}$ conditioned on not having some small predefined $t$-size hypergraph $H$ as a subgraph. The algorithm should run in poly$(n)$-time even when the size of the subgraph $H$ is super-constant.

We solve the problem by carefully designing a sampling algorithm for $k$-wise independent hypergraphs $\mathcal{G}$ that supports efficient testing for subgraph-freeness. We use our algorithm 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 $d > c$, 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 argue that such an "almost-explicit" 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 long-standing open problem.

*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" [15]). Motivated by the wide range of applications, the computational problem of efficiently constructing $\mathcal{H}$-free graphs has attracted a huge amount of research. 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/almost-explicit 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 from any inverse polynomial, i.e., $n^{-\omega(1)}$. Such a construction is referred to as an* almost-explicit 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.*

Almost-explicit constructions form 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. Following the standard cryptographic tradition, we insist on a *negligible* error probability of $n^{-\omega(1)}$ in order to guarantee a tiny failure probability even after polynomially-many repetitions.[1] Throughout the paper, we typically assume that an $\alpha$-fraction of all $(n,m,d)$-hypergraphs are $\mathcal{H}$-free, where the density $\alpha$ 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 [21].[2] 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 [21], the *implementation* $G$ must be *truthful* to the $\mathcal{H}$-freeness *specification*. Conditioned on this, $G$ should be distributed uniformly or close to uniformly under some metric.

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.

[1] 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.

[2] The work of [21] 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 the GGN setting.

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.[3]

**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 polynomial. 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 (cf. [16]). We refer to this hardness-of-testing as the *testing barrier*. Jumping ahead, we will show that a 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 task of generating $\mathcal{H}$-free hypergraphs can be roughly ranked from easy to hard as follows: almost-explicit constructions, zero-error constructions, explicit constructions, pseudorandom constructions.

# 2 Our Results

We partially resolve Problems 1.1 and 1.2. Our main 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 almost-explicit constructions of such graphs (Section 2.3), and use them to derive various applications (Section 2.4).

## 2.1 $k$-wise independent $H$-free graphs

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 are distributed uniformly over all possible $d$-uniform hyperedges. The use of $k$-wise independent distributions as a good model for pseudo-random graphs was advocated by Naor, Nussboim and Tromer [40] and by Alon and Nussboim [1]. These works further show that a large family of natural graph-theoretic properties that hold whp over random graphs (with a given edge density) also hold whp 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 given distribution $D$, and the tester $T$ examines the *coins* of the tester 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* poly($n$)*-time sampler/tester pair $(S, T)$ with the following properties:*

[3]The work of [12] provides a good example for such a case in the context of *financial derivatives*.

- *Given $1^n$, the randomized sampler $S$ uses its internal random coins $r$ to sample an $(n, m = n^c, d)$ hypergraph $G_r$ whose hyperedges are $k(n)$-wise independent.*
- *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 [40, 1] hold. (A more detailed version of Theorem 2.1 appears in Theorem 6.4.)

**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 a $k$-wise independent $(n, m, d)$-hypergraph $G$ 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 $\exp(t^d) < \text{poly}(n)$ 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 a $k(n)$-wise independent $(n, m, d)$-hypergraph is $\mathcal{H}$-free with noticeable probability of $1/\text{poly}(n)$. Then, there exists a probabilistic algorithm that runs in expected $\text{poly}(n)$-time and samples an $\mathcal{H}$-free hypergraph from some $k$-wise independent distribution over $(n, m, d)$-hypergraphs.*

**Remark 2.3** (Hardness of Sampling uniform $\mathcal{H}$-free hypergraphs)**.** *It is natural to try and sample a* uniform *$\mathcal{H}$-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 show that sampling uniform $\mathcal{H}$-free hypergraphs is likely to be infeasible. In particular, suppose that, 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, under this assumption, sampling uniform $\mathcal{H}$-free hypergraphs implies the existence of* one-way functions*. Put differently, a sampler would allow us to convert average-case hardness (of testing) to one-wayness, or, in the language of Impagliazzo [26], to move from Pessiland to Minicrypt. The hardness of certifying $\mathcal{H}$-freeness is closely related to previous intractability assumptions (cf. [6, 12, 8]). 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 amendable to efficient analysis (i.e., subgraph testing). The fact that such implementation is computationally-cheap or randomness-efficient is not really needed. (Nevertheless, these properties will be used in the next subsection.)*

**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 $\text{poly}(n)$-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 $\mathsf{prg}$ that fools $n^a$-time algorithms with logarithmic size seed [27]. 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.[4]

## 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 represented by an identifier $z$ of length $\mathrm{polylog}(n)$ such that given $z$, an 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 $\mathrm{polylog}(n)$. (Here we assume that the hyperedges are ordered and can be represented by $d$-tuples.) We prove a succinct version of Theorem 2.1 that applies to constant-size subgraphs $H$ and $\mathrm{polylog}(n)$-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 $k(n) \leq \mathrm{polylog}(n)$, there exists a $\mathrm{polylog}(n)$-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 $G_r$ whose hyperedges are $k(n)$-wise independent.*
- *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 a $k(n)$-wise independent $(n, m, d)$-hypergraph is $\mathcal{H}$-free with probability of $1/\mathrm{polylog}(n)$. Then, there exists a probabilistic algorithm that runs in expected $\mathrm{polylog}(n)$-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 [21, 1, 40, 39]. 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 Almost-Explicit 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 $(\alpha, t)$-expander if every set $S$ of hyperedges of size at most $t$ "touches" at least $\alpha|S|$ vertices.[5] Equivalently, an $(\alpha, t)$-expander is an

[4] There are cases in which derandomization assumptions can be easily used to turn an almost-explicit construction into an explicit construction [32]. 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 somewhere low in the polynomial-hierarchy; 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.

[5] 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 $\alpha|S|$ right vertices.

$(n, m, d)$-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 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 $\delta$ is a constant that depends on $\alpha, c$ and $d$. Unfortunately, the problem of *efficiently constructing* highly-unbalanced constant-degree expanders is wide open. Existing constructions either have only linearly many hyperedges $m = O(n)$ [14] or suffer from a super-constant (actually polylogarithmic) degree [23]. Motivated by the numerous applications of constant-degree highly-unbalanced expanders (to be discussed later), we present an almost-explicit construction of such graphs.

We begin by giving 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 $(\alpha, t)$-expansion where $t = O(\frac{\log\log\log n}{\log\log\log\log n})$.*

Next, we show that Theorem 2.8 gives rise to an almost-explicit 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** (almost-explicit unbalanced expanders)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d > c$, and $\alpha < d - c$, there exists an almost-explicit construction of $(n, m = n^c, 2d)$-hypergraph with $(\alpha, t)$-expansion where $t = \Omega(n^{1-\delta})$ and $\delta = (c-1)/(d-\alpha-1)$.*

Recall that an almost-explicit constructions guarantees the existence of a poly$(n)$-time randomized algorithm that outputs, except with negligible probability of $n^{-\omega(1)}$, an $(\alpha, t)$-expanding $(n, m, 2d)$-hypergraph. (See Theorem 7.15 for a more detailed version.)

Theorem 2.9 provides an $(n, m, D = 2d)$-hypergraph whose expansion parameters $(\alpha, t)$ 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 $\alpha$ 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. Formally, a hypergraph is a *$(\beta, t)$-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 an almost-explicit construction of unique expanders.

**Theorem 2.10** (almost-explicit unbalanced unique-expanders)**.** *For every log-density parameter $c > 1$, edge-uniformity parameter $d > 2c$, and $\beta < d - 2c$, there exists an almost-explicit construction of $(n, m = \Omega(n^c), 2d)$-hypergraph with $(\beta, t)$ 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 almost-explicit constructions of highly-unbalanced constant-degree expanders.

## 2.4 Applications

In Section 8, we use our almost-explicit unbalanced expanders to obtain the first almost-explicit 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

LDPC codes [18] (see also [35, 43, 44])) are $[m, k]$ linear error-correcting codes whose $(m-k)\times m$ parity check matrix is *sparse* in the sense that it contains only $dm$ non-zero entries for some *sparsity constant* $d = O(1)$.[6] 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 $(\beta, \gamma)$ then the resulting code has a distance of $\gamma$.

Theorem 2.10 leads to the first almost-explicit construction of high-rate LDPC code that tolerates polynomially small number of errors. In particular, 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 [43]. 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 an almost explicit 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 $O(\log^2 m)$ parallel steps and corrects up to $\Omega(n^{1-\delta})$ errors where $\delta = (c-1)/(d-\alpha-1)$.*

### 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 [30, 7] and general-purpose obfuscation based on constant-degree multilinear maps (cf. [33, 34]).

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 $m = n + o(n)$ [9], and even linear stretch $m = n + \Omega(n)$ [10, 3, 5] under various intractability assumptions, we currently have only partial solutions to the polynomial-stretch regime. In particular, in [3] the first author constructed a locally-computable polynomial-stretch *weak-PRG*. Here *weak* means that the distinguishing advantage $\varepsilon$ 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 [3] is based on the one-wayness of random local functions with polynomially-long output length – a variant of Goldreich's one-wayness assumption [19].

We show that our almost-explicit 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*

[6] Recall that an $[m, k]$-code is a linear code with codewords of length $m$ and information words of length $k$, and an $[m, k, \Delta]$-code has, in addition, an absolute distance of $\Delta$.

*$n$ 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 does not affect the applications.

Combined with the weak-PRG of [3], Theorem 2.12 yields the first construction of local PRG with polynomial stretch based on a one-wayness assumption. We mention that there is a second heuristic approach for constructing such pseudorandom generators (see [37, 11] and the survey [4]). This approach also requires the existence of explicit (or almost-explicit) highly-unbalanced constant degree expanders, and one can instantiate it using our constructions as well. In fact, it is known that such explicit expanders are *necessary* for *any* construction of locally-computable PRG with large-stretch [10].[7] 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 $c \geq 1$.

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 [31, 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.

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

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

[7] Indeed, prior works on expander-based cryptography (cf. [2, 3, 7, 11, 19, 30, 33, 34]) 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, cf. [30, Remark 5.7]).

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 $(\alpha, t)$-expansion for small threshold $t = O(\frac{\log\log\log n}{\log\log\log\log n})$ into an almost-explicit $(\alpha, T)$ 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 $\alpha$. 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 $\alpha$, 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 an almost-explicit 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 $\alpha s$ 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 an almost-explicit 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 $(\beta, t)$ unique-expander and $G_2$ expands by $\beta$ for sets of size $t < s \leq T$, then $G$ is a $(\beta, t)$ 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 $\ell$ independent copies of the seed and XOR the results. By Yao's XOR-lemma (cf. [22]), 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 $k - 1/\mathrm{poly}(k)$. Finally, we extract the randomness from each column using randomness extractor. This approach was used by [3] (following a more general transformation from [24]) 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 $k - 1/\mathrm{poly}(k)$ 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 an almost-explicit construction can be achieved based on almost-explicit highly-unbalanced constant-degree expanders. (Similar connections between expanders and locally-computable extractors were established, for a different regime of parameters, in related contexts [10, 3]). See Sections 9 and 8.3.

## 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 an 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 $\pi$ 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 $\pi$ 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 an 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 $\Lambda_i$ 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 $\ell$ and $k < \ell(q-1)$, let $\mathcal{P}_{\ell,k,q}$ denote the uniform distributions over multivariate polynomials with $\ell$ 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 $\mathbb{F}_q$.) 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 $\ell$ and $k < \ell(q-1)$, the distribution $\mathcal{P}_{\ell,k,q}$ is $k$-wise independent.*

*Sketch.* 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 $\ell$ 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 [42, Proposition 5.4.14].

In particular, the distance of the dual code is $(\rho+1)q^\sigma$ where $\rho$ is the reminder after division of $k+1$ by $q-1$ with quotient $\sigma$ (that is, $k+1 = \sigma(q-1)+\rho$ for $\rho < q-1$). One can show that for every $k < \ell(q-1)$, the dual code has distance 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 $\mathrm{poly}(k^\ell \cdot \log q)$.

# 5 The Sampler/Tester Frameowrk

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 gap between the $n$-th is upper-bounded by some polynomial in $n$.*

**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

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

**Definition 5.4.** *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a function with $n < m$. We say that $G$ $\epsilon$-fools probabilistic algorithms with running $T = T(n)$ if for every deterministic Turing-machine $\mathcal{A}$ of time $T$ it holds that*

$$|\Pr[\mathcal{A}(U_m) = 1] - \Pr[\mathcal{A}(G(U_n)) = 1]| < \epsilon.$$

**Theorem 5.5** ([27][47])**.** *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 $b \cdot t$, which $(1/t)$-fools every deterministic algorithm running in time $t$.*

In fact, the assumption in Theorem 5.5 implies a PRG that fools non-uniform adversaries, but all we need is a PRG that fools (or actually *hit*) uniform adversaries.[8]

**Corollary 5.6.** *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 5.5, 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 5.5 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 5.5. 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 $\text{poly}(n)$ binary strings in $\{0,1\}^{a \log n}$ it follows that the running time of $A$ is $\text{poly}(n)$. □

# 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 $\ell$-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 $\ell$ and $r$, and that $k < \ell(q-1)$. Correspondingly, we index the vertices by elements from $\mathbb{F}_q^r$ and index the hyperedges by elements in $\mathbb{F}_q^\ell$.*

[8] One can get such a generator under weaker uniform hardness assumptions [28, 46]. 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.

- *(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 $P$ represent 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 $\ell$-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 vertices 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 $r \cdot d \cdot \binom{k+\ell}{\ell} \log q$, and given its representation, one can find the $i$-th element of the $j$-th hyperedge in time $\mathrm{poly}(r \cdot k^\ell \cdot \log d \cdot \log q)$.*

*Proof.* The fact that every $k$ hyperedges are distributed uniformly and independently over $([n]^d)^k$ follows from Claim 4.5 and from the fact that the $d \cdot r$ polynomials are sampled independently from $\mathcal{P}_{\ell,k,q}$. The complexity of the sampling algorithm follows from the fact that we can sample a field element in time $\mathrm{poly}(\log q)$, and that every $\ell$-variate polynomial of degree $k$ has at most $\binom{k+\ell}{\ell}$ monomials. The size of the representation of the graph follows from the fact that a field element requires $\log q$ 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 $d$ be a constant positive integer. Let $c$ be a constant positive rational number, represented in lowest terms by $\ell/r$. Let $q$ be a prime power, $n = q^r$, $m = n^c$, $n' = O(\frac{\log\log\log n}{\log\log\log\log n})$ and $m' = O(\frac{\log\log\log n}{\log\log\log\log n})$. Let $k \leq \min(\ell(q-1) - 1, \mathrm{poly}(n^{1/e(n',m')}))$, where $e(n', m') = (2m'\ell + 2n'r)^{O(2m'\ell + 2n'r)}$. Then, there is a $\mathrm{poly}(n)$ algorithm $\mathcal{S}$ that samples a graph $G$ from $\mathcal{G}_{q,n,m,d,k}$ and a $\mathrm{poly}(n)$ 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.*

**Remark 6.5.** *If we are not interested in succinct hypergrah, then we can even take $d = \mathrm{poly}(n)$, and still get a polynomial-time testing algorithm. For simplicity, we stick to a constant $d$.*

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 $2^{\log n / \log\log n} \gg \mathrm{poly}(\log n)$.

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) of 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 6.6.** *Let $c, \ell, r, q, m, n, d, n', m'$ 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 $\mathcal{G}_{q,n,m,d,k}$ is also succinct.

## 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 $G_P$ (for ease of readability we will drop the subscript $P$). We then employ the algorithm of Kayal [31, 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 $\ell$-variables $x_h = (x_h(1), \ldots, x_h(\ell))$ (intuitively, think of $x_h$ as taking values in $\mathbb{F}_q^\ell$). For every vertex $v$ of $H$ define a tuple of $r$-variables $x_v = (x_v(1), \ldots, x_v(r))$ (intuitively, think of $x_v$ as taking values in $\mathbb{F}_q^r$). 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 6.7.** *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 mappings 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 $\pi$ 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 $\sigma$ is injective. □

**Claim 6.8.** *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 6.7). Then $H$ is a subgraph of $G$.*

*Proof.* Denote the value assigned to the variable $x_v$ by $a_v$, for every vertex $v$, and the value assigned to the variable $x_h$ by $a_h$ for every hyperedge $h$. 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 $x_{h_i} = e_j$, 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 $\pi$ is injective, and as all the $a_h$'s are distinct, it follows that $\sigma$ is injective. Hence, by Definition 4.1 it follows that $H$ is a subgraph of $G$. □

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

**Claim 6.9** (Distinctness gadget)**.** *Let $X = (X_1, \ldots, X_\eta)$ be $\eta$ formal variables taking values from some finite field $\mathbb{F}$ of size at least $\eta$. Let $Y = (Y_0, \ldots, Y_{\eta-1})$ be additional formal variables over $\mathbb{F}$, and consider the polynomial system of $\eta$ equations whose $i$-th equation is*

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

*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 if and only if $x_1, \ldots, x_\eta$ are all distinct values in $\mathbb{F}$.*

*Proof of Claim.* Let $Z$ be a formal variable over $\mathbb{F}$, let $f(Z; Y) = \sum_{j=0}^{\eta-1} Y_j Z^j$ be a polynomial over $\mathbb{F}$, and let $x$ be a fixed assignment for $X$. For the first direction, let $y$ be an assignment to $Y$ such that $(x, y)$ satisfies the system, and assume towards contradiction that there exist $i \neq j$ such that $x_i = x_j$. Then $f(x_i, y) = f(x_j, y)$ while $f(x_i, y) = a_i \neq a_j = f(x_j, y)$, in contradiction. Hence all the $x_i$'s are distinct.

For the other 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. □

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 $\mathcal{L}_2$.** Let $a_1, \ldots, a_{n'}$ be some fixed 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 $\mathcal{L}_3$.** Let $b_1, \ldots, b_{m'}$ be some fixed 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 $\ell$ formal variables over $\mathbb{F}_n$. Similarly, we can view each $z_i$ as a vector of $\ell$ 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.** The following claims prove that $H$ is a subgraph of $G$ if and only if there exists a satisfying assignment to the set of equations $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$.

**Claim 6.10.** *If $H$ is a subgraph of $G$ then there exists a satisfying assignment to the set of equations $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$.*

*Proof.* By Claim 6.7, $\mathcal{L}_1$ 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 6.9 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 6.9 implies that there exists an assignment to the $z_i$'s that satisfies $\mathcal{L}'_3$ (over $\mathbb{F}_{q^\ell}$). Fact 4.3 implies that there exists an assignment to the $z_i(j)$'s that satisfies $\mathcal{L}_3$ (over $\mathbb{F}_q$). Fix this assignment to the $y_i(j)$'s.

Altogether, we get an assignment that satisfies $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$. □

**Claim 6.11.** *If there exists a satisfying assignment to the set of equations $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$, then there exists a satisfying assignment to $\mathcal{L}_1$ that assigns distinct values to all the vertex variables and distinct values to all the hyperedge variables (in the sense of Claim 6.7).*

*Proof.* Denote the value assigned to the variable $x_v$ by $a_v$, for every vertex $v$, and the value assigned to the variable $x_h$ by $a_h$ for every hyperedge $h$. It is enough to show that all the $a_v$'s are distinct, and that all the $a_h$ are distinct.

Fix the assignment $(x_v = a_v)_{v \in V(H)}$. As there exists an assignment to the $y_i(j)$'s that satisfies $\mathcal{L}_2$ (over $\mathbb{F}_q$), Fact 4.3 implies that there exists an assignment to the $y_i$'s that satisfies $\mathcal{L}'_2$ (over $\mathbb{F}_{q^r}$). Claim 6.9 implies that all the $a_v$'s are distinct.

Similarly, fix the assignment $(x_h = a_h)_{h \in V(H)}$. As there exists an assignment to the $z_i(j)$'s that satisfies $\mathcal{L}_2$ (over $\mathbb{F}_q$), Fact 4.3 implies that there exists an assignment to the $z_i$'s that satisfies $\mathcal{L}'_2$ (over $\mathbb{F}_{q^\ell}$). Claim 6.9 implies that all the $a_h$'s are distinct. □

Combining Claim 6.10 with Claim 6.11 and Claim 6.8, we get that $H$ is a subgraph of $G$ if and only if there exists a satisfying assignment to the set of equations $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$.

Finally, we need 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 $\text{poly}(d, (m')^\ell, (n')^r, k^\ell, r, \log q)$, while, by [31, Theorem 6.1.1], checking whether the set of equations is solvable takes time $\text{poly}(D^{(2\ell m' + 2rn')^{O(2\ell m' + 2rn')}} \cdot (dm'r + rn' + \ell m') \cdot \log q)$.

Altogether, we completed 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 $\sigma$. See Definition 4.1.) A useful relaxation of this notion is obtained by allowing $\pi$ to be non-injective. In particular, two vertices of $H$ can be mapped to the same vertex in $G$. Since the edge-mapping $\sigma$ 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 6.12.** *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 6.12 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 6.12 we derive the following theorem (that can be viewed as a version of Theorem 6.4 that applies to edge-disjoint homomorphism).

**Theorem 6.13.** *Let $d$ be a constant. Let $c$ be a constant positive rational number, represented in lowest terms by $\ell/r$. Let $q$ be a prime power, $n = q^r$, $m = n^c$, $n' = O(\frac{\log\log n}{\log\log\log n})$, and $m' = O(\frac{\log\log\log n}{\log\log\log\log n})$. Let $k \leq \min(\ell(q-1)-1, \text{poly}(n^{1/e(n',m')}))$, where $e(n',m') = (2m'\ell + n'r)^{O(2m'\ell+n'r)}$. Then, there is a $\text{poly}(n)$ algorithm $\mathcal{S}$ that samples a graph $G$ from $\mathcal{G}_{q,n,m,d,k}$ and a $\text{poly}(n)$ 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 $k \leq \text{polylog}(n)$, 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 $\text{polylog}(n)$ time.*

Note that if we are not interested in succinct hypergraphs, we can even take $d = \text{poly}(n)$ and still get a polynomial-time testing algorithm. (See Remark 6.5.) In this case, with edge-disjoint homomorphism we can test for more vertices in comparison to the notion of subgraph.

The following theorem is an adaptation of Corollary 6.6 to the case of edge-disjoint homomorphism.

**Corollary 6.14.** *Let $c, \ell, r, q, m, n, d, n', m'$ and $k$ be as in Theorem 6.13. 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 $\text{poly}(n)/(1-\epsilon(n))$ 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 sub-graph of $G$ (resp., $H$ is edge-disjoint homomorphic to $G$).

**Observation 6.15.** *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. Observation 6.15 together with Lemma 6.3 and Lemma 6.12 imply Lemma 6.16.

**Lemma 6.16.** *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 $\mathcal{P}_{\ell,k,q}$ (where $k < \ell(q-1)$), tests if $H$ is a subgraph of $G_P$ (defined as in Construction 6.1), and its running time polynomial in*

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

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

*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 $\mathcal{P}_{\ell,k,q}$ (where $k < \ell(q-1)$), tests if $H$ is a edge-disjoint homomorphic of $G_P$, and its running time is polynomial in*

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

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

*Proof.* From Observation 6.15 it follows that the non-oriented hypergraph $H$ is a subgraph of the non-oriented hypergraph $G_P$ (resp., edge-disjoint homomorphic to $G_P$) if and only if there exists an orientation of $H$ such that the oriented hypergraph $H$ is a subgraph of the oriented hypergraph $G_P$ (resp., edge-disjoint homomorphic to $G_P$), where $G_P$ has it's natural orientation (that is, the orientation that was defined in Construction 6.1).

Fix the natural orientation for $G_P$, and iterate over all $(d!)^{m'}$ possible orientations of $H$. From Lemma 6.3 (resp., Lemma 6.12) it follows that for every such orientation we can check whether $H$ is a subgraph of $G_P$ (resp., edge-disjoint homomorphic to $G_P$) in time $\text{poly}(D^{(2\ell m' + 2rn')^{O(2\ell m' + 2rn')}} \cdot (dm'r + rn' + \ell m') \cdot \log q)$ for $D := \max\{k, m', n'\}$ (resp., $\text{poly}(D^{(2\ell m' + rn')^{O(2\ell m' + rn')}} \cdot (dm'r + \ell m') \cdot \log q)$ for $D := \max\{k, m'\}$). The claim follows. □

The following theorem follows immediately from Claim 6.2 and Lemma 6.16.

**Theorem 6.17.** *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 $\text{poly}(n)$ algorithm $S$ that samples $\mathcal{G}_{q,n,m,d,k}$ and a $\text{poly}(n)$ 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 6.13 and let $d$ be a constant. Then, there is a $\text{poly}(n)$ algorithm $S$ that samples $\mathcal{G}_{q,n,m,d,k}$ and a $\text{poly}(n)$ 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 6.17 we can no longer take $d$ to be polynomial in $q$.

**Corollary 6.18.** *Let $c, \ell, r, q, m, n, d, n', m'$ and $k$ be as in the first part of Theorem 6.17 (resp., the second part of Theorem 6.17). 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 $\text{poly}(n)/(1 - \epsilon(n))$ 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 $k$-wise independent *expanders*. We will be interested in the following notions of expansion.[9]

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 contained in 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 emphasize that in the following section all hypergraphs are non-oriented.

[9] Expanders are usually presented as bipartite graphs. In the following, we will stick to the presentation with hypergraphs.

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

We will usually refer to $\alpha$ 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* $\alpha$-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* $(\beta, \gamma)$-unique-neighbor expander *if every set $S \subseteq [m]$ of hyperedges of size at most $\gamma$ is $\beta$-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 $\rho$ 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 $\alpha$-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 $\alpha$ and $d$. Consequently, the probability that the sampled graph is not $(\alpha, \gamma)$-expander is at most $1/2$.*

It is well known that vertex expansion implies unique-neighbor expansion (cf. [14]). In particular, if $\alpha = (1-\epsilon)d$ and $\beta = (1-2\epsilon)d$ for some $\epsilon < 0.5$, then every set which is $\alpha$-expanding, is also $\beta$-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 $\rho$ 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 $\beta$-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 $\beta$ and $d$. Consequently, the probability that the sampled graph is not $(\beta, \gamma)$-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.*

2. *The tester tests if the sampled hypergraph is $(\alpha, \gamma)$-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 $k \leq \min(\text{poly}(n^{1/e(\gamma)}), n^\epsilon)$, 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 $k \leq \text{poly}(\log n)$ 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 $k = \min(\text{poly}(n^{1/e(\gamma)}), n^\epsilon)$ 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 $(\alpha, \gamma)$-expansion. Consequently, there is a zero-error randomized construction of such expanders. Moreover, if $k \leq \text{poly}(\log n)$ 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 $\text{poly}(n)$-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 $(\beta, \gamma)$-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 $k = \min(\text{poly}(n^{1/e(\gamma)}), n^\epsilon)$, 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 $k \leq \text{poly}(\log n)$ 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 $k = \min(\text{poly}(n^{1/e(\gamma)}), n^\epsilon)$ 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 $(\alpha, \gamma)$-unique-expansion. Consequently, there is a zero-error randomized construction of such expanders. Moreover, if $k \leq \text{poly}(\log n)$ 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 6.17), we will show that we can use it to construct a tester for the expansion properties of the sampled graph.

### 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 6.17 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 $(\alpha, \gamma)$-expander with probability at least $1/2$.

Moreover, Theorem 6.17 implies a 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 $(\alpha, \gamma)$-expander.

**Claim 7.9.** *Let $G$ be an $(n, m, d)$-hypergraph. Fix some constant $\alpha$. 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 $\alpha$-expanding.*

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

For the other direction, let $S$ be a set of $m'$ hyperedges which is not $\alpha$-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 a $k$-wise independent $(n, m, d)$-graph $G$, 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 $(\alpha, \gamma)$-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 $\gamma$ hyperedges which is not $\alpha$-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 poly$(n)$-time algorithm $\mathcal{T}'$ that tests for the event that $G$ is not an $(\alpha, \gamma)$-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 6.17 implies a sampling algorithm $\mathcal{S}$, that, given as input $1^n$ for a prime power $n$, samples in time 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 $(\beta, \gamma)$-unique-neighbor expander with probability at least $1/2$.

Moreover, Theorem 6.17 implies a 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 $(\beta, \gamma)$-unique-neighbor expander.

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

*Proof.* For the first direction, let $\sigma$ and $\pi$ be the mappings promised in Definition 4.1, and take $S$ to be the image of $\sigma$. Then $S$ is not $\beta$-uniquely-expanding in $G$. For the other direction, take $H$ to be the subgraph that contains the hyperedges of the set that is not $\beta$-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 a $k$-wise independent $(n, m, d)$-graph $G$, 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 $(\beta, \gamma)$-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 bad for $\beta$, 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 $\beta$-uniquely-expanding if and only if some $H$ which is not $\beta$-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 $(\beta, \gamma)$-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 $\gamma'$ hyperedges of $G_1$ has expansion $\alpha_1$ (that is, $|\Gamma(S)| \geq \alpha_1 |S|$). In addition, assume that every set $S$ of hyperedges of $G_2$, of size at least $\gamma'$ and at most $\gamma$ has expansion $\alpha_2$. Let $\alpha = \min\{\alpha_1, \alpha_2\}$. Then $G$, defined as in Construction 7.13, is an $(n, m, 2d)$-hypergraph and an $(\alpha, \gamma)$-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** (Almost-Explicit 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 $(\alpha, \gamma)$-expander, where $\gamma = \rho n^{1-\delta}$ for a sufficiently small constant $\rho$ 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 $(\alpha, \gamma')$-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 $\gamma'$ and at most $\gamma$ which is not $\alpha$-expanding is negligible. Moreover, note that $G_2$ can be sampled in time $\text{poly}(n)$.

Conditioned on the events that $G_1$ is an $(\alpha, \gamma')$-expander, and that in $G_2$ every set $S \subseteq [m]$ of size $\gamma' \leq s \leq \gamma$ is $\alpha$-expanding, Let $G$ be as in Construction 7.13. Then, by Claim 7.14 $G$ is a $(n, m, 2d)$-hypergraph which is $(\alpha, \gamma)$-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 $\gamma'$ hyperedges of $G_1$ is $\beta$-uniquely-expanding (that is, $G_1$ is a $(\beta, \gamma')$-unqiue-neighbor expander). In addition, assume that every set $S$ of hyperedges of $G_2$, of size at least $\gamma'$ and at most $\gamma$, is $\beta$-uniquely-expanding. Then $G$, defined as in Construction 7.16, is a $(2n, m, 2d)$-hypergraph and a $(\beta, \gamma)$-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 $(\beta, \gamma)$-unique-neighbor expander. □

We can now prove the following version of Theorem 2.10.

**Theorem 7.18** (Almost-Explicit 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 $\text{poly}(n)$-time probabilistic algorithm that, except with negligible probability $n^{-\omega(1)}$, samples an $(2n, m = n^c, 2d)$-hypergraph with $(\beta, \gamma)$-unique-neighbor-expander, where $\gamma = \rho n^{1-\delta}$ for a sufficiently small constant $\rho$ 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 $(\beta, \gamma')$-unique-neighbor expander, promised in Corollary 7.8. Note that $G_1$ can be sampled in time $\text{poly}(n)$ 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 $\gamma'$ and at most $\gamma$ which is not $\beta$-uniquely-expanding is negligible. Moreover, note that $G_2$ can be sampled in time $\text{poly}(n)$.

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

# 8 Applications

## 8.1 Batch Codes

Batch codes, introduced by Ishai, Kushilevitz, Ostrovsky and Sahai in [29], allow us to distribute an $m$-bit information word $x$ over $n$ devices such that every subset of $\gamma$ 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 $(m, M, \gamma, n)$ *batch code* is a pair of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ such that:

- The encoder $\mathsf{Enc}$ 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 $\mathsf{Dec}$ takes as an input a $\gamma$-subset $S \subset [n]$ 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)[45]. If each bit is stored in precisely $d$ buckets the code is referred to as *$d$-uniform* [45]. 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 [29] 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 an almost-explicit construction of an $(m, M, \gamma, n)$-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.*

The algorithm $A$ 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 $\gamma$ subset $S \subseteq [n]$ can be done by finding a perfect matching between the required bits $S$ and the some $\gamma$ buckets, and reading exactly one bit from each of the buckets. This can be done in time $\tilde{O}(m^{10/7})$ (see [36]).

Theorem 8.1 yields the first almost-explicit 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 $\gamma$ is polynomially smaller than $n$. Boyle Couteau, Gilboa and Ishai [13] 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. Theorem 8.1 yields a constructive version of this result.

## 8.2 High-Rate LDPC Codes

As already mentioned, the $n \times m$ incidence matrix of an $(n, m, d)$-hypergraph $G$ with $(\beta, \gamma)$-unique-expansion forms a $d$-sparse parity-check matrix of an $[m, m-n, \gamma]$-linear code. 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 $\Omega$-notation depend on $d$ and $c$. For every prime power $n$, there exists an almost-explicit construction of a $2n \times m$ parity-check matrix with at most $2d$ ones in each columns which describes an $[m, m-2n, \gamma]$-code.*

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

**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. Let $m = n^c$ and let $\gamma = \Omega(n^{1-\delta})$ where $\delta = (c-1)/(d-\alpha-1)$ and the constants in the $\Omega$-notation depend on $d$ and $\alpha$. For every prime power $n$ there exists an almost-explicit construction of a $2n \times m$ parity-check matrix with at most $2d$ ones in each column, which represents an $[m, m-2n, \gamma]$-code. Moreover, the sampled code has a decoding algorithm that corrects at least $0.6\gamma$ errors with $O(\log^2 n)$ parallel steps, where the constants in the $O$-notation depend on $\alpha$.*

(A larger constant $\alpha$ improves the number of parallel steps, but reduces the distance.) We sketch the proof of Theorem 8.3 in Section 8.2.1.

### 8.2.1 Proof Sketch of Theorem 8.3

First, we present some notation. Then, to gain intuition, we present a decoding algorithm for an LDPC code which is defined by a good vertex-expander. Finally, we show how to sample a code for which Theorem 8.3 holds.

**Notation.** 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 think of the hyperedges of $G$ as representing the word, where the value of the $i$-th hyperedge $e_i$ is $x_i$. We think of the vertices as constraints, where the value of the $i$-th vertex $v_i$ is the parity bit of all the hyperedges that contain it (with multiplicity). We say that a constraint is satisfied if its value is 0, and otherwise we say that it is not satisfied. Note that the vector of values of the vertices is exactly $Hx$. This implies that $x$ is a codeword if and only if all its constraints are satisfied.

The analysis of the decoder will relies on the following lemma.

**Lemma 8.4** ([14])**.** *Let $G$ be an $(n, m, d)$-hypergraph, let $\alpha = (1-\epsilon)d$ for $\epsilon \leq 1/2$, and assume that every set $S \subseteq [m]$, where $\gamma' \leq |S| \leq \gamma$, is $\alpha$-expanding. Then for every such subset $S$ we have:*

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

One can show that for every $\epsilon < 0.1$, one can efficiently compute $1/2 < \eta < 1$ and $\delta \geq 2\epsilon$ for which

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

**The Sipser-Spielman Decoder for Good Expander.** Let $G$ be an $(n, m, d)$-hypergraph, let $\alpha = (1-\epsilon)d$ for $\epsilon < 0.1$, and assume that $G$ is an $(\alpha, \gamma)$-expander. First, recall that the expansion-property of $G$ implies that $G$ is also a $(\beta, \gamma)$-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 Eq. (1). The decoding algorithm gradually updates the noisy codeword until it reaches a valid codeword. In each iteration, the decoder flips the value of all the hyperedges that have at least $\eta d$ unsatisfied constraints.

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 constrained is in $\Gamma(S)$. The first part of Lemma 8.4 implies that in each iteration at least $1-\delta$ fraction of the hyperedges in $S$ have at least $\eta d$ 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 $\eta d$ 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 $\xi$, 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.

We show that a modified version of this decoder can decode our construction. But first, we have to slightly modify the sampling algorithm.

**Sampling Algorithm.** Fix some constants $c, d$ so that $10c < d$ and choose some constant $\varepsilon < 0.1$ for which $(1-\epsilon)d < d - c$. Set $\alpha = (1-\epsilon)d$ and $m = n^c$. Let $\gamma = \Omega(n^{1-\delta})$ where $\delta = (c-1)/(d-\alpha-1)$, and $\gamma' = \Theta(\frac{\log\log\log n}{\log\log\log\log n})$. Use Corollary 7.6 to get an almost-explicit construction of an $(\alpha, \gamma')$-expanding $(n, m, d)$-hypergraph $G_1 = (V_1, E_1)$. Sample a random $(n, m, d)$-hypergraph $G_2 = (V_2, E_2)$ and note that, by Claim 7.3, with all but negligible probability, $G_2$ $\alpha$-expands every set of size $s$ for $0.6\gamma' < s < \gamma$. We let $G$ be the $(2n, m, 2d)$-graph defined in Construction 7.13. Note that $G$ can be sampled in time $\text{poly}(n)$ with all but negligible probability.

**Distance of Sampled Code.** To prove that $G$ defines an $[m, m-2n, \gamma]$-code, it suffices to show that $G$ is $(\beta, \gamma)$-expander for some $\beta > 0$. This follows from the fact that if a set $S$ is $\alpha$-expanding for $\alpha = (1-\epsilon)d$, then it is also $\beta$-uniquely-expanding for $\beta = (1-2\epsilon)d$, and from Claim 7.17.

**Decoding Algorithm.** First, for simplicity, assume that the decoding algorithm has an oracle that given a noisy codeword $c$ returns 1 only if the distance of $c$ from the closest codeword is less than $0.6\gamma'$. 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 decoding algorithm with an oracle call as follows. At the beginning of each iteration, on noisy codeword $c$, the decoding algorithm calls the oracle to check if $c$ has less than $0.6\gamma'$ errors. If so, it looks only at the restriction of $G$ to $V_1$ (that is, it looks at $G_1$), and otherwise it looks only at the restriction of $G$ to $V_2$ (that is, it looks at $G_2$). Then, when restricted to $V_i$, it flips (in parallel) the value of every hyperedge that has more than $\eta d$ 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, and there is at most one transition from $G_2$ to $G_1$

Finally, since the decoding algorithm does not have an oracle, we do not know in which of the $O(\log n)$ iterations the transition from $G_2$ to $G_1$ occurs. Hence we simply try all $O(\log n)$ possibilities, thus having $O(\log^2 n)$ parallel steps.

## 8.3 Local Non-Cryptographic Generators and Randomness Extractors

We rely on standard transformations to turn our almost-explicit unbalanced expanders into almost-explicit 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 an almost-explicit construction of a $2d$-local function (represented as a circuit) $f : \{0,1\}^n \to \{0,1\}^{\Omega(m)}$ 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 $\Omega$-notation depend on $c$ and $d$.*

*Proof.* Sample the $2d$-sparse $n \times m$ 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 $\gamma$ is the distance of the sampled LDPC code. □

**Low-bias generators.** The notion of low-bias generators is due to Naor and Naor [38]. A random variable $X = (X_1, \ldots, X_m)$, distributed over $\{0,1\}^m$, is $\epsilon$-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 $\epsilon$-biased generator if the random variable $f(U_n)$ is $\epsilon$-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 an almost-explicit construction of a $d$-local function (represented as a circuit) $f : \{0,1\}^{3n} \to \{0,1\}^{n^c}$ which is an $\epsilon$-biased generator for $\epsilon = e^{-\Omega(n^\delta)}$, for any $\delta < \frac{r-2c}{(r-2)(2c+1)}$ for $r = (d-(2c+1)^2)/2$.*

*Proof.* Let $d' > 2c$ and let $f_1 : \{0,1\}^{2n} \to \{0,1\}^{n^c}$ be the $2d'$-local $t$-wise independent generator promised in Theorem 8.5, for $t = \Omega(n^{1-\xi})$, for every $\xi > 2(c-1)/(d'-2)$. Let $f_2$ be a $(2c+1)^2$-local low-bias generator promised in [37, Lemma 23], where, for a linear test $S$, the bias is at most $\exp\left(-|S|^{1/(2c+1)}/2^{2c+1}\right)$. Let $d = 2d' + (2c+1)^2$, 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 $\epsilon$ by the low-bias property of $f_2$. Hence $f$ is an $\epsilon$-biased generator. □

**Randomness extractors [41].** 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 $Ext : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ be a function. We say that $Ext$ is a $(k,\epsilon)$-extractor if for every distribution $X$ with min-entropy at least $k$ it holds that $\Delta(Ext(X,U_k),U_m) \leq \epsilon$, that is, the statistical distance between the random variable $Ext(X,U_k)$ and $U_m$ is at most $\epsilon$.

**Theorem 8.7.** *Let $c,d$ be constants such that $d > 4c^2+8c+1$. Let $\delta = \delta(n) < 1$. For a prime power $n$ there exists an almost-explicit construction of a local function (represented as a circuit) $Ext : \{0,1\}^{n^c} \times \{0,1\}^{3n} \to \{0,1\}^{n^c}$ which is a $(k,\epsilon)$-extractor, for $k = (1-\delta)n^c$ and $\epsilon = e^{-\Omega(n^\xi)} \cdot 2^{\delta \cdot n^c/2 - 1/2}$, for any $\xi < \frac{r-2c}{(r-2)(2c+1)}$, for $r = (d-(2c+1)^2)/2$. Moreover, each bit of the output depends on exactly one bit of the input and $d$ bits of the seed.*

*Proof.* The proof follows immediately from Theorem 8.6 and [10, Lemma 5.7]. □

# 9 Local PRG with Polynomial Stretch

## 9.1 Definitions

**Indistinguishability.** We say that a pair of distribution ensembles $Y = \{Y_n\}$ and $Z = \{Z_n\}$ are $\epsilon$-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 $\epsilon = neg(n)$ we say that the two ensembles are computationally-indistinguishable.

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

**Collection of Functions.** Let $s = s(n)$ and $m = m(n)$ be integer-valued functions. A collection of functions $\{F_k\}$ is formally defined via a mapping $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ which takes an index $k \in \{0,1\}^s$ and an input point $x \in \{0,1\}^n$, and outputs the evaluation $F_k(x)$ of the point $x$ under the $k$-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 $k \in \{0,1\}^s$, and an evaluation algorithm that given $(1^n, k, x)$ outputs $F_k(x)$. 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 $k$ the function $F_k$ 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 *$\epsilon$-pseudorandom generator* (PRG) if the ensemble $(K(1^n), F_{K(1^n)}(U_n))$ is $\epsilon$ indistinguishable from the ensemble $(K(1^n), U_m)$, where $U_n$ is the uniform distribution over $\{0,1\}^n$. When $\epsilon$ is negligible, we refer to $F$ as a *pseudorandom generator*.

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

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

for all sufficiently large $n$'s.

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 $\epsilon$ is negligible, and as a weak-PPRG in the case that $\epsilon$ 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.

**Fact 9.1** ([48]). *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

**Proof strategy.** Given a weak-PPRG $G_t : \{0,1\}^n \to \{0,1\}^m$, where $t$ is the index of the function, we apply it on $k$ uniform strings $U_n^{(1)}, \ldots, U_n^{(k)}$, to get a $k \times m$ matrix $Y$, whose $i$-th row is $G_t(U_n^{(i)})$. We show that, even conditioned on all previous columns, the $j$-th column has high pseudo-entropy, for every $j \in [m]$. Hence, the $j$-th column is computationally-indistinguishable from a distribution with high min-entropy. We then apply the locally-computable extractor from Theorem 8.7 on each column of $Y$ to get a distribution which is computationally-indistinguishable from uniform. We proceed with a formal proof.

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 for $\epsilon = n^{-b}$ for $b = c - a$, and without loss of generality we may assume that $a, b > 1$ are sufficiently large (see, [3, Fact 6.5]).

We present the index-sampling algorithm $K$, which will sample a local circuit. First, $K$ samples a circuit $G_t$, for $t \in \{0,1\}^s$ using the index-sampling algorithm of $G$. As we always assume that $t$ is known to the adversary, then, for ease of read, we omit the subscript $t$. Let $Ext : \{0,1\}^k \times \{0,1\}^{3k^\alpha} \to \{0,1\}^k$ be the local extractor promised at Theorem 8.7 with required min-entropy $(1-\delta)k$ and error $e^{-\Omega(k^{\alpha/4})} \cdot 2^{O(\delta k)}$, for some $\alpha < 1$. Think of $k = \text{poly}(n)$ and $\delta = 4m\epsilon$ and note that $K$ can sample $Ext$ with all but negligible probability. Finally, $K$ outputs circuit of the following construction.

**Construction 9.2.** *Given a PRG $G$ and an extractor $Ext$ as above, we construct the following function $G^*$:*

- *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 $i$th column of $Y$, and output $(Ext(Y_1; z^{(1)}), \ldots, Ext(Y_m; z^{(m)}))$.*

Notice that the locality of $G^*$ is at most the multiplication of the localities of $G$ and $Ext$, and so it is constant.

First, we show that every column of $Y$ has high pseudo-entropy, even conditioned on all previous columns. To do this, we need to show a strong unpredictability property of every row of $Y$.

Let $r = \lfloor \log m \rfloor + 1$ be the number of bits required to represent an index in $[m]$, and let $\ell = n + r$. Define the following functions

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

Note that both $\mathsf{PRE}$ and $\mathsf{N}$ are efficiently computable algorithms, and 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 [24, Proposition 4.7]) it follows that for every oracle-aided $P$ running in time $T_P = \text{poly}(n)$ and all sufficiently large $n$, there exists $L_{\ell,P} \subseteq \{0,1\}^\ell$ of density at least $1 - 2\epsilon$ such that

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

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

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

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

**Lemma 9.3** (Proposition 4.8 at [24])**.** *For every oracle-aided distinguisher $D$ running in time $T_D = \text{poly}(n)$, there is a set $L_{\ell,D} \subseteq \{0,1\}^\ell$ of density at least $1 - 2\epsilon$ for which the following holds. Define the probabilistic function $\mathsf{R}_{L_{\ell,D}}$ by*

$$\mathsf{R}_{L_{\ell,D}}(i,x) = \begin{cases} \mathsf{N}(i,x) & \textit{if } (i,x) \notin L_{\ell,D} \\ \textit{random bit} & \textit{if } (i,x) \in L_{\ell,D} \end{cases}.$$

*Then, given an oracle $\mathcal{O}$ that samples according to the joint distribution $(\mathsf{PRE}(i,x), \mathsf{N}(i,x), \mathsf{R}_{L_{\ell,D}}(i,x))$, 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 negligible advantage.*

*Proof.* Assume that there exists a distinguisher $D$ of time $T_D = \text{poly}(n)$ such that for infinitely many $n$'s,

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

for every $L$ of density $1 - 2\epsilon$ and some polynomial $p(\cdot)$. 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 $\chi_L$ and time $T_P = T_D + O(1) = \text{poly}(n)$ (note that $P$ can simulate the oracle calls of $D$ efficiently, using $\chi_L$) such that for every $L$ 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 2. □

Now, we show that even condintioned on all previous columns, one cannot distinguish the $i$-th column of $Y$ from a distribution with high min-entropy. For this we need the following definitions. Let $L \subseteq \{0,1\}^\ell$. Define

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

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

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

For every algorithm $D$ with input of length $(r+m+1)\times k$, define the oracle-aided algorithm $\mathcal{A}_D$ in the following way. On input $i,y,z$, for $i\in[m]$, $y\in\{0,1\}^m$ and $z\in\{0,1\}$, the algorithm outputs

$$D\left(\begin{array}{c} \mathsf{PRE}(i,x_1),\mathsf{N}(i,x_1) \\ \vdots \\ \mathsf{PRE}(i,x_{j-1}),\mathsf{N}(i,x_{j-1}) \\ i,y,z \\ \mathsf{PRE}(i,x_{j+1}),\mathsf{R}_{L_{\ell,\mathcal{A}_D}}(i,x_{j+1}) \\ \vdots \\ \mathsf{PRE}(i,x_k),\mathsf{R}_{L_{\ell,\mathcal{A}_D}}(i,x_k) \end{array}\right)$$

for $j\leftarrow[k]$ and $x_1,\ldots,x_k\leftarrow\{0,1\}^{n\times k}$, and $\mathcal{A}_D$ samples $\mathsf{R}_{L_{\ell,\mathcal{A}_D}}(x)$ using the oracle call to $\mathcal{O}$ which samples from $(\mathsf{PRE},\mathsf{N},\mathsf{R}_{L_{\ell,\mathcal{A}_D}})$. Note that if $D$ is polynomial-time algorithm then so is $\mathcal{A}_D$.

**Claim 9.4.** *Every polynomial-time algorithm $D$ cannot distinguish*

$$(\mathsf{PRE}^k(U_{r+nk}),\mathsf{N}^k(U_{r+nk})) \quad \textit{from} \quad (\mathsf{PRE}^k(U_{r+nk}),\mathsf{R}^k_{L_{\ell,\mathcal{A}_D}}(U_{r+nk}))$$

*with more than negligible advantage. Consequently, for every polynomial-time algorithm $D$ and every $i\in[m]$, $D$ cannot distinguish $(\mathsf{PRE}^k(i,U_{nk}),\mathsf{N}^k(i,U_{nk}))$ from $(\mathsf{PRE}^k(i,U_{nk}),\mathsf{R}^k_{L_{\ell,\mathcal{A}_D}}(i,U_{nk}))$ with more than negligible advantage.*

*Proof.* Assume towards contradiction that there exists a PPT $D$ that distinguishes the above distributions with advantage $1/p(n)$ for some polynomial $p(\cdot)$. Then by a standard hybrid argument $\mathcal{A}_D$ distinguishes $(\mathsf{PRE}(U_\ell),\mathsf{N}(U_\ell))$ from $(\mathsf{PRE}(U_\ell),\mathsf{N}_{L_{\ell,\mathcal{A}_D}}(U_\ell))$ with advantage $1/(p(n)k)$, in contradiction to Lemma 9.3. □

It remains to show that the random variable $(\mathsf{R}^k_{L_{\ell,\mathcal{A}_D}}(U_{r+n\times k})|\mathsf{PRE}^k(U_{r+n\times k}))$ has high min-entropy.

**Claim 9.5.** *Except for probability $e^{-\Omega(m\epsilon k)}$ over $(i,x_1,\ldots,x_k)\leftarrow\{0,1\}^{r+nk}$, for every $z\in\{0,1\}^k$ and every PPT $D$,*

$$\Pr[\mathsf{R}^k_{L_{\ell,\mathcal{A}_D}}(i,x_1,\ldots,x_k)=z|\mathsf{PRE}^k(i,x_1,\ldots,x_k)] \le 2^{-k(1-\delta)}.$$

*Proof.* Fix some PPT $D$. First, note that if $L$ is of density $1-2\epsilon$ then for every $i$ there is at least $(1-2\epsilon m)$ fraction of the $x$'s for which $(i,x)\in L$. Say that the sample $(i,x_1,\ldots,x_k)$ is "good" if $|\{j:(i,x_j)\notin L_{\ell,\mathcal{A}_D}\}| < 4km\epsilon = k\delta$. Note that the expected size of the set $\{j:(i,x_j)\notin L_{\ell,\mathcal{A}_D}\}$ is at most $2km\epsilon$, and since the $x_j$'s are independent, by multiplicative Chernoff bound $\Pr[(i,x_1,\ldots,x_k)$ is "bad"$] \le e^{-\Omega(m\epsilon k)}$.

Finally, for a "good" $(i, x_1, \ldots, x_k)$ and every $z$ we have

$$\Pr[\mathsf{R}^k_{L_{\ell,A_D}}(i, x_1, \ldots, x_k) = z | \mathsf{PRE}^k(i, x_1, \ldots, x_k)] \leq 2^{-k(1-4m\epsilon)} = 2^{-k(1-\delta)}.$$

since $\mathsf{R}^k_{L_{\ell,A_D}}(i, x_1, \ldots, x_k)$ has at least $k(1 - 4m\epsilon)$ random bits. □

Set $\epsilon = 1/(m \cdot n^{8/\alpha})$ and $k = \log^2(n) \cdot n^{8/\alpha}$. Note that $G^*$ has polynomial expansion. Furthermore, both the error of $Ext$ and the probability that Claim 9.5 does not hold are negligible.

It follows that the distributions $(\mathsf{PRE}^k(U_{r+nk}), Ext(\mathsf{R}^k_{L_{\ell,A_D}}(U_{r+nk}); r))$ and $(\mathsf{PRE}^k(U_{r+nk}), U_k)$ are statistically indistinguishable, where $r$ is the random seed for the extractor (of length $3k^\alpha$). This implies that for every $i \in [m]$, the the distributions $(\mathsf{PRE}^k(i, U_{nk}), Ext(\mathsf{R}^k_{L_{\ell,A_D}}(i, U_{nk}); r))$ and $(\mathsf{PRE}^k(i, U_{nk}), U_k)$ are statistically indistinguishable.

Denote by $Y[i : j]$ for $j \geq i$ the submatrix of $Y$ that includes columns $i$ to $j$. We claim that for every efficiently samplable family of indices $\{i_n\}$, the distributions $(Y[1 : i_n - 1], Ext(Y_{i_n}; r))$ and $(Y[1 : i_n - 1], U_k)$ are indistinguishable.

Assume towards contradiction that there exists some polynomial-time adversary $\mathcal{B}$ that for infinitely many $n$'s distinguishes the above two distributions with non-negligible advantage. Construct adversary $\mathcal{C}$ that distinguishes the distribution $(\mathsf{PRE}^k(i_n, U_{nk}), \mathsf{N}^k(i_n, U_{nk}))$ from $(\mathsf{PRE}^k(i_n, U_{nk}), \mathsf{R}^k_{L_{\ell,A_D}}(i_n, U_{nk}))$ in the following way. On input $(\mathsf{PRE}^k(i_n, x_1, \ldots, x_k), u)$ return $\mathcal{B}(\mathsf{PRE}^k(i_n, x_1, \ldots, x_k)[1 : i_n - 1], Ext(u; r))$. If $u$ is sampled from $\mathsf{N}^k(i_n, U_{nk})$ then the input to $\mathcal{B}$ is distributed according to $(Y[1 : i_n - 1], Ext(Y_{i_n}; r))$. If $u$ is sampled from $\mathsf{R}^k_{L_{\ell,A_D}}(i_n, U_{nk})$ then the input to $\mathcal{B}$ is statistically indistinguishable from $(Y[1 : i_n - 1], U_k)$. Hence $\mathcal{C}$ distinguishes $(\mathsf{PRE}^k(i_n, U_{nk}), \mathsf{N}^k(i_n, U_{nk}))$ from $(\mathsf{PRE}^k(i_n, U_{nk}), \mathsf{R}^k_{L_{\ell,A_D}}(i_n, U_{nk}))$ in contradiction to Claim 9.4.

Hence, for every family $\{i_n\}$, the distribution $G^*(U_{nk})[1 : i_n]$ is indistinguishable from $(G^*(U_{nk})[1 : i_n - 1], U_1)$. Therefore $G^*$ is unpredictable, and from Yao's theorem it follows that $G^*$ is a PRG.

Finally, we use [3, Fact 6.5], to get a local-PPRG with output of length $n^{c'}$. The completes the proof of Theorem 2.12. □

# References


[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

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

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

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

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

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

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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`.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On $\varepsilon$-biased generators in $\mathrm{NC}^0$. *Random Structures & Algorithms*, 29(1):56–81, 2006.

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

[39] 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.

[40] 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.

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

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

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

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

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

[46] 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.

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

[48] 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.

# A On Sampling Uniform $H$-free Hypergraphs

An interesting question is 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* 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* 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. [20]).

Assume, towards a contradiction, that there exists a 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 $n < m(n) \leq$ 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* 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 $\epsilon$ 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 that each of their vertices have *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 [25, 17]. Closely related average-case hardness assumptions were also used in [6, 12, 8].

# 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 $\alpha$-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 $\alpha$ and $d$.

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



ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il