Pseudorandom Bit Generators that Fool Modular Sums

Shachar Lovett[1], Omer Reingold[2], Luca Trevisan[3], and Salil Vadhan[4]

[1] Department of Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. shachar.lovett@weizmann.ac.il ⋆

[2] Department of Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. omer.reingold@weizmann.ac.il ⋆⋆

[3] Computer Science Division, University of California, Berkeley, CA, USA. luca@cs.berkeley.edu ⋆⋆⋆

[4] School of Engineering and Applied Science, Harvard University, Cambridge, MA 02138. salil@eecs.harvard.edu †

Abstract. We consider the following problem: for given n, M, produce a sequence $X_1, X_2, \ldots, X_n$ of *bits* that fools every linear test modulo M. We present two constructions of generators for such sequences. For every constant prime power M, the first construction has seed length $O_M(\log(n/\epsilon))$, which is optimal up to the hidden constant. (A similar construction was independently discovered by Meka and Zuckerman [MZ]). The second construction works for every M, n, and has seed length $O(\log n + \log(M/\epsilon)\log(M\log(1/\epsilon)))$.

The problem we study is a generalization of the problem of constructing *small bias* distributions [NN], which are solutions to the $M = 2$ case. We note that even for the case $M = 3$ the best previously known constructions were generators fooling general bounded-space computations, and required $O(\log^2 n)$ seed length.

For our first construction, we show how to employ recently constructed generators for sequences of elements of $\mathbb{Z}_M$ that fool small-degree polynomials (modulo M). The most interesting technical component of our second construction is a variant of the derandomized graph squaring operation of [RV]. Our generalization handles a product of two distinct graphs with distinct bounds on their expansion. This is then used to produce pseudorandom-walks where each step is taken on a different regular directed graph (rather than pseudorandom walks on a single regular directed graph as in [RTV,RV]).

⋆ Research supported by the Israel Science Foundation (grant 1300/05).

⋆⋆ Research supported by US-Israel BSF grant 2006060.

⋆⋆⋆ This material is based upon work supported by the National Science Foundation under grant No. CCF-0729137 and by the US-Israel BSF grant 2006060.

† Work done in part while visiting U.C. Berkeley, supported by the Miller Institute for Basic Research in Science and a Guggenheim Fellowship. Also supported by US-Israel BSF grant 2006060.

1 Introduction

Pseudorandomness is the theory of generating objects that "look random" despite being constructed using little or no randomness. A primary application of pseudorandomness is to address the question: *Are randomized algorithms more powerful than deterministic ones?* That is, how does randomization trade off with other computational resources? Can every randomized algorithm be converted into a deterministic one with only a polynomial slowdown (*i.e.,* does $\mathbf{BPP} = \mathbf{P}$) or with only a constant-factor increase in space (*i.e.,* does $\mathbf{RL} = \mathbf{L}$)? The study of both these questions has relied on pseudorandom bit generators that fool algorithms of limited computational powers. In particular, generators that fool space-bounded algorithms [AKS,BNS,Nis,INW] were highly instrumental in the study of the **RL** vs. **L** problem (e.g. used in the best known derandomization of **RL** [SZ]).

While the currently available space-bounded generators are extremely powerful tools, their seed length is still suboptimal. For example, if we want to fool a $\log n$-space algorithm then known generators require $\log^2 n$ truly random bits (the seed) in order to generate up to polynomially many pseudorandom bits. On the other hand, for several interesting special cases we do know generators with almost optimal seed length. The special case which serves as a motivation for our work is that of small-biased generators [NN]. These generators produce n bits $X_1, X_2, \ldots, X_n$ that fool all linear tests modulo 2. In other words, for each subset T of the bits, the sum $\Sigma_{i \in T} X_i \mod 2$ is uniformly distributed up to bias ϵ. Explicit constructions of ϵ-biased generators are known with seed-length $O(\log(n/\epsilon))$, which is optimal up to the hidden constant [NN]. Even though linear tests may seem very limited, ϵ-biased generators have turned out to be very versatile and useful derandomization tools [NN,MNN,HPS,Nao,AM,AR,BSVW,BV,Lov,Vio].

Given the several applications of distributions that fool linear tests modulo 2, it is natural to consider the question of fooling modular sums for larger moduli. It turns out that the notion of small-biased generators can be generalized to larger fields. Such generators produce a sequence $X_1, X_2, \ldots, X_n$ of elements in a field $\mathbb{F}$ that fool every linear test over $\mathbb{F}$ [Kat,AIK$^+$,RSW,EGL$^+$,AM].[5] In this work, instead, we consider a different generalization of ϵ-biased generators where we insist on *bit*-generators. Namely we would like to generate a sequence $X_1, X_2, \ldots, X_n$ of bits that fool every linear test modulo a given number M. For every sequence $a_1, a_2, \ldots, a_n$ of integers in $\mathbb{Z}_M = \{0, 1, \ldots, M-1\}$ we want the sum $\sum_i a_i X_i \mod M$ to have almost the same distribution (up to statistical distance at most ϵ) as in the case where the X_i's are uniform and independent random bits. (Note that this distribution may be far from the uniform distribution over $\mathbb{Z}_M$, particularly when only a few a_i's are nonzero.) It turns out that

[5] More generally, an *ϵ-bias space* over a finite abelian group G is a distribution D on elements of G such that for every nontrivial character $\chi : G \to \mathbb{C}$, $|\mathbb{E}[\chi(D)]| \leq \epsilon$. The aforementioned results correspond to the special case $G = \mathbb{F}^n$, using the fact that the characters of $\mathbb{F}^n$ are in one-to-one correspondence with linear functions $\mathbb{F}^n \to \mathbb{F}$.

even for $M = 3$ and even if we limit all the a_i's to be either ones or zeros, the best generators that were known prior to this work are generators that fool general space-bounded computations [Nis,INW], and required a seed of length $O(\log^2 n)$. Therefore, obtaining better pseudorandom bit generators that fool modular sums may be considered a necessary step towards improved space-bounded generators. In addition, we consider this notion to be a natural generalization of that of a small-bias generator, which is a central derandomization tool.

Our Results

We give two constructions of pseudorandom bit generators that fool modular sums. Similarly to [MST], each construction is actually comprised of two generators: one that fools summations $\sum_i a_i X_i$ in which only relatively few coefficients a_i are nonzero (the "low-weight" case) and one that fools summations $\sum_i a_i X_i$ in which many coefficients a_i are nonzero (the "high weight" case). The motivation is that fooling low-weight sums and fooling high-weight sums are tasks of a different nature. In the high-weight case, if R_i are truly random bits, then $\Sigma_i a_i R_i \mod M$ is almost uniformly distributed in $\mathbb{Z}_M$ (at least when M is prime). Thus, in analyzing our generator, we just need to argue that $\Sigma_i a_i X_i \mod M$ is close to uniform, where $X_1, \ldots, X_n$ is the output of the generator.

On the other hand, in the low-weight case the distribution may be far from uniform and therefore we may need to imitate the behavior of a random sequence of bits more closely.

Thus, in each construction, we shall present two generators: one that is pseudorandom against low-weight sums, and one that is pseudorandom against high-weight sums. We shall then combine them by evaluating them on independently chosen seeds and XORing the two resulting sequences.

Construction Based on Pseudorandom Generators for Polynomials

In our first construction, we handle the case of $M = 3$ and any other fixed prime modulus M (in fact, our construction works also for any fixed prime power). For these cases, our seed length is $O(\log(n/\epsilon))$ as in the case of ϵ-biased generators (but the hidden constant depends exponentially on M).

As mentioned above, for every fixed finite field $\mathbb{F}$, there are nearly-optimal known generators that construct a small-bias distribution $X_1, \ldots, X_n$ of *field elements*, while our goal is to generate *bits*. A natural approach to construct a bit generator would be to sample a sequence of field elements $X_1, \ldots, X_n$ from a small-bias distribution, and output a bit-sequence $g(X_1), \ldots, g(X_n)$ for an appropriate function $g : \mathbb{F} \to \{0, 1\}$. Unfortunately the pseudorandomness of $g(X_1), \ldots, g(X_n)$ against $\mathbb{F}$-linear tests does not seem to follow from the small-bias property of $X_1, \ldots, X_n$. Indeed, when $|\mathbb{F}|$ is odd, then g cannot be balanced, so at best we could hope is for $g(X_1), \ldots, g(X_n)$ to be indistinguishable by linear tests from a sequence of independent *biased* bits. But even this is not achievable in general, if we only assume the pseudorandomness of $X_1, \ldots, X_n$ against $\mathbb{F}$-linear tests(as per the definition of small-bias space).[6]

If, however, we start from a sequence of field elements $X_1, \ldots, X_n$ that fools *polynomials* over $\mathbb{F}$, then we can indeed show that $g(X_1), \ldots, g(X_n)$ is indistinguishable by linear tests from independent biased bits. The reason is that g can be chosen to be itself a polynomial (of degree $d = \Theta(|\mathbb{F}|)$), and thus any $\mathbb{F}$-linear test distinguisher on $g(X_1), \ldots, g(X_n)$ yields a degree d distinguisher on $X_1, \ldots, X_n$. Since we still only have indistinguishability from *biased* coins, we only apply this approach when the coefficient vector has sufficiently high weight so that both biased and unbiased random bits will yield a sum that is almost uniformly distributed over $|\mathbb{F}|$. Specifically, we need at least k non-zero coefficients a_i, where $k = O(M^2 \log 1/\epsilon)$. For fixed M, there are known constructions [BV,Lov,Vio] of pseudorandom generators that fool polynomials of degree d over $\mathbb{F} = \mathbb{Z}_M$, M prime, and which only require seed length $O_{M,d}(\log n/\epsilon)$.

In order to fool low-weight sums, we observe that a bit generator $X_1, \ldots, X_n$ which is ϵ-almost k-wise independent fools, by definition, every sum $\sum_i a_i X_i \bmod M$ of weight at most k, and that such generators are known which require only seed length $O(\log n + k + \log 1/\epsilon)$.

A similar construction was independently discovered by Meka and Zuckerman [MZ].

Construction Based on the INW Generator

In our second construction, we give a pseudorandom bit generator that fools sums modulo *any* given M (not necessarily prime) with seed length $O(\log n + \log(M/\epsilon)\log(M \log(1/\epsilon)))$. In both the low-weight and high-weight cases, this generator relies on versions of the Impagliazzo–Nisan–Wigderson [INW] pseudorandom generator for space-bounded computation. Of course, modular sums are a special case of space-bounded computations, and thus we could directly apply the INW generator. But this would require seed length larger than $\log^2 n$. We obtain better bounds by more indirect use of the INW generator inside our construction.

The most interesting technical contribution underlying this construction is a new analysis of the derandomized graph squaring operation of [RV], which captures the effect of using the INW generator to derandomize random walks on graphs. Here we study the analogue of derandomized squaring for taking products of two distinct Cayley graphs over an abelian group (namely $\mathbb{Z}_M$). The advantage of the new analysis is that it handles graphs that have distinct bounds on their expansion, and works for bounding each eigenvalue separately. This is then used to produce pseudorandom walks where each step is taken on

[6] Let $\mathbb{F} = \mathbb{Z}_3$, and $g : \mathbb{Z}_3 \to \{0, 1\}$ be any nonconstant function. Let a be the element of $\mathbb{Z}_3$ such that a is the unique preimage of $g(a)$. Let $(X_1, \ldots, X_n)$ be uniformly distributed over all elements of $\mathbb{Z}_3^n$ where the number of a's is divisible by 3. Then $\sum_i g(X_i) \mod 3$ is constant, but it can be shown that $(X_1, \ldots, X_n)$ is a $2^{-\Omega(n)}$-biased space.

a different abelian Cayley graph (rather than pseudorandom walks on a single graph as in [RTV,RV]).

For the purpose of this informal discussion we will assume that M is prime. (The idea for handling composite M's is to analyze each Fourier coefficient of the distribution of the sum separately. We defer further details to Section 2.1.)

Low-Weight Case. Let us first consider the case where the number of non-zero a_i's is at most $M' \cdot \log(1/\epsilon)$, for $M' = \mathrm{poly}(M)$.[7] As before, we could use an almost k-wise independent distribution, but then our seed length would depend polynomially on M, while our goal is a polylogarithmic dependency.

First, we use a hash function to split the index set $[n] = \{1, 2, \ldots, n\}$ into $B = O(M')$ disjoint subsets T_j such that with high probability (say, $1 - \epsilon/10$) over the splitting, each set T_j contains at most $k = \log(1/\epsilon)$ indices i such that $a_i \neq 0$. We show that the selection of the hash function that determines the splitting can be done using $O(\log n + (\log M/\epsilon) \cdot \log(M \log 1/\epsilon))$ random bits.

Once we have this partition, it is sufficient to independently sample in each block from an ϵ/B-almost k-wise independent distribution, which requires $s = O(\log n + k + \log(B/\epsilon)) = O(\log n + \log(M/\epsilon))$ random bits per block. Then we argue that it is not necessary for the sampling in different blocks to be independent, and instead they can be sampled using a pseudorandom generator for space-bounded computation [Nis,INW]. (This relies on the fact the computation $\sum_i a_i X_i \bmod M$ can be performed in any order over the i's, in particular the order suggested by $\sum_j \sum_{i \in T_j} a_i \cdot X_i \bmod M$.) Using the INW generator, we can do all the sampling using $O(s + \log B \cdot (\log(B/\epsilon) + \log M)) = O(\log n + \log M \cdot \log(M/\epsilon))$ random bits.

High-Weight Case. We now discuss the generator that fools sums with more than $M' \cdot \log 1/\epsilon$ non-zero coefficients a_i, for $M' = \mathrm{poly}(M)$. Here, we can think of the computation $\sum_i a_i X_i \mod M$ as an n-step walk over $\mathbb{Z}_M$ that starts at 0. Unlike standard walks, *each step is taken on a different graph* (over the same set of vertices, namely $\mathbb{Z}_M$). Specifically, step i is taken on the (directed) Cayley graph where every node v has two outgoing edges. The first edge is labeled 0 and goes into v itself (*i.e.,* this edge is a self loop). The second edge is labeled 1 and goes into $v + a_i \mod M$. Following the walk along the labels $X_1, X_2, \ldots, X_n$ arrives at the vertex $\sum_i a_i X_i \mod M$. If the X_i's are uniform (*i.e.,* we are taking a random walk) then the end vertex will be almost uniformly distributed (because the number of steps is larger than $M^2 \cdot \log(1/\epsilon)$). What we are seeking is a pseudorandom walk that is generated using much fewer truly random bits but still converges to the uniform distribution (possibly slower, e.g. using $M' \cdot \log(1/\epsilon)$ steps).

Pseudorandom walk generators were constructed in [RTV,RV] for walks on a single regular and connected graph. In our case, we are walking not on a

[7] In this preliminary version we did not try to optimize the various constants. In particular, in our analysis $M' = O(M^{24})$. We note that it can be made as small as $O(M^{2+\alpha})$ for any $\alpha > 0$.

single graph but rather on a sequence of graphs, each of which is indeed regular. It turns out that the pseudorandom generators of [RTV,RV] still work for a sequence of graphs rather than a single graph. The more difficult aspect is that in our walk there is no uniform bound on the expansion of the graphs. Indeed, the graphs that correspond to $a_i = 0$ are not connected at all (they consist solely of self loops). In our setting, where the graphs are directed Cayley graphs for the abelian group $\mathbb{Z}_M$, we show how to generate pseudorandom walks on graphs with varying bounds on expansion.

We do this by a generalization of the derandomized graph product of [RV]. There, expanders are used to generate two steps on a degree-D graph using less than $2 \log D$ random bits, yet the (spectral) expansion of the resulting graph is almost as good as the square of the original graph. We analyze the analogous derandomization of two steps on two distinct (abelian Cayley) graphs for which we may have distinct bounds on their expansion. Moreover, to handle composite M, we show that the expansion can be analyzed in each eigenspace separately. (For example, for $\mathbb{Z}_6 = \mathbb{Z}_2 \times \mathbb{Z}_3$, a sequence of even coefficients a_i will yield a random walk that does not mix in the $\mathbb{Z}_2$ component, but may mix in the $\mathbb{Z}_3$ component, and our pseudorandom generator needs to preserve this property.)

To obtain our pseudorandom walk generator, we first randomly reorder the index set $[n]$ so that the nonzero coefficients are well-spread out, and then derandomize the walk by a recursive application of our aforementioned derandomized product. As discussed in [RV], the resulting pseudorandom walk generator is the same as the Impagliazzo–Nisan–Wigderson [INW] generator for space-bounded computation, with a different setting of parameters that enables a much smaller seed length than their analysis requires for general space-bounded algorithms.

Discussion

The natural open problem left by our work is to reduce the seed length further, ideally to $O(\log(nM/\epsilon))$, which can be shown to be possible via a nonconstructive probabilistic argument. For achieving such optimal parameters, the modular reduction is actually insignificant — it is equivalent to construct generators such that for every bounded coefficient vector $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ where each $|a_i| \leq M$, $\sum_i a_i X_i$ is statistically close to $\sum_i a_i R_i$ as distributions on $\mathbb{Z}$, where $(X_1, \ldots, X_n)$ is the output distribution of the generator, and $(R_1, \ldots, R_n)$ is the uniform distribution on $\{0,1\}^n$. [8] As a result, such generators would also “fool” linear threshold functions (halfspaces) whose coefficients are polynomially bounded. Pseudorandom generators and related objects for threshold functions (with no bound on the coefficients) have recently been studied in [RS,DGJ$^+$], with the latter achieving seed length $O((\log n) \cdot \log^2(1/\epsilon)/\epsilon^2)$.

[8] Indeed, given any coefficient vector $(a_1, \ldots, a_n) \in \mathbb{Z}^n$, where each $|a_i| \leq M$, we can apply the generator for modulus $M' = M \cdot n$ so that no modular reduction occurs.

2 Definitions and Tools

We denote by U_n the uniform distribution over $\{0,1\}^n$. We fix an integer $M \geq 2$ for the rest of the paper. We will be interested in constructing pseudorandom bit generators that fool sums modulo M. We denote by $\mathbb{Z}_M$ the set $\{0, 1, \ldots, M-1\}$ with arithmetic modulo M. Due to space limitations, we defer many of the proofs to the full version of the paper.

Definition 2.1. *The* statistical distance *between two random variables* X, Y *taking values in* $\mathbb{Z}_M$ *is* $\mathrm{dist}(X,Y) = \frac{1}{2}\sum_{i=0}^{M-1} |\Pr[X = i] - \Pr[Y = i]|$. *The variables* X *and* Y *are said to be* ϵ-close *if their statistical distance is at most* ϵ.

Definition 2.2. *A random variable* $X = (X_1, \ldots, X_n)$ *taking values in* $\{0,1\}^n$ *is* ϵ-pseudorandom against sums modulo M *if for any* $a_1, \ldots, a_n \in \mathbb{Z}_M$*, the distribution of* $a_1X_1 + \cdots + a_nX_n$ modulo M*, is* ϵ*-close (in statistical distance) to the distribution* $a_1R_1 + \cdots + a_nR_n$ modulo M*, where* $R_1, \ldots, R_n$ *are uniform and independent random bits.*

Definition 2.3. *A function* $G : \{0,1\}^r \to \{0,1\}^n$ *is an* ϵ-pseudorandom bit generator against sums modulo M *if the distribution* $G(U_r)$ *is* ϵ*-pseudorandom against sums modulo* M*.*

Note that ϵ-biased generators is a special case of the definition of pseudorandom bit generators against sums modulo M, for $M = 2$.

Our goal is to build generators that fool sums modulo M, where M can be either prime or composite. Handling prime modulus is somewhat easier, and the approach in the following section allows handling both cases simultaneously. We will show that it is enough to construct pseudorandom generators which fools the bias of a sum modulo M, and under this approach, there is no major difference between primes and composites.

2.1 Small Bias Bit Generators

First we define the *bias* of a linear combination with coefficients $a_1, \ldots, a_n \in \mathbb{Z}_M$, given some distribution of $X = (X_1, \ldots, X_n) \in \{0,1\}^n$:

Definition 2.4. *Let* $X = (X_1, \ldots, X_n)$ *be a distribution over* $\{0,1\}^n$*, and* $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ *a coefficient vector. We define the* bias *of* $a_1, \ldots, a_n$ *according to* X *to be*

$$\mathrm{bias}_X(a_1, .., a_n) = \mathbb{E}\left[\omega^{\sum a_i X_i}\right]$$

where $\omega = e^{2\pi i/M}$ *is a primitive* M*-th root of unity.*

Notice that the bias can in general be a complex number, of absolute value at most 1.

Definition 2.5. *We say a distribution $X = (X_1, \ldots, X_n)$ over n bits is* ϵ-bit-biased against sums modulo M *if for every coefficient vector* $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$,

$$|\mathrm{bias}_X(a_1, \ldots, a_n) - \mathrm{bias}_{U_n}(a_1, \ldots, a_n)| \leq \epsilon$$

Let $G : \{0,1\}^r \to \{0,1\}^n$ be a bit generator. We shorthand $\mathrm{bias}_G(a_1, \ldots, a_n)$ for $\mathrm{bias}_{G(U^r)}(a_1, \ldots, a_n)$.

Definition 2.6. $G : \{0,1\}^r \to \{0,1\}^n$ *is an* ϵ-bit-biased generator against sums modulo M *if the distribution $G(U_r)$ is ϵ-bit-biased against sums modulo M. That is, for every coefficient vector $(a_1, \ldots, a_n)$,*

$$|\mathrm{bias}_G(a_1, \ldots, a_n) - \mathrm{bias}_{U_n}(a_1, \ldots, a_n)| \leq \epsilon$$

The name "bit-biased" in the above definitions is meant to stress the difference from standard ϵ-biased generators modulo M. Here we compare the bias under the generator to the bias under uniformly selected bits (rather than uniformly selected elements in $\mathbb{Z}_M$).

We first reduce the problem of constructing pseudorandom modular generators to that of constructing ϵ-bit-biased modular generators.

Lemma 2.7. *Let $X = (X_1, \ldots, X_n)$ be an ϵ-bit-biased distribution against sums modulo M. Then X is $(\epsilon\sqrt{M})$-pseudorandom against sums modulo M.*

From now on, we focus on constructing ϵ-bit-biased generators. We will need to differentiate two types of linear combinations, based on the number on non-zero terms in them.

Definition 2.8. *The* weight *of a coefficient vector $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ is the number of non-zero coefficients a_i.*

We will construct two generators: one fooling linear combination with small weights, and the other fooling linear combinations with large weight. Our final generator will be the be the bitwise-XOR of the two, where each is chosen independently. The following lemma shows this will result in an ϵ-bit-biased generator fooling all linear combinations.

Lemma 2.9. *Fix a weight threshold W. Let $X' = (X'_1, \ldots, X'_n)$ be a distribution over $\{0,1\}^n$ such that for any vector coefficient $a_1, \ldots, a_n$ of weight at most W,*

$$|\mathrm{bias}_{X'}(a_1, \ldots, a_n) - \mathrm{bias}_{U_n}(a_1, \ldots, a_n)| \leq \epsilon.$$

Let $X'' = (X''_1, \ldots, X''_n)$ be a distribution over $\{0,1\}^n$ such that for any vector coefficient $a_1, \ldots, a_n$ of weight at least W,

$$|\mathrm{bias}_{X'}(a_1, \ldots, a_n) - \mathrm{bias}_{U_n}(a_1, \ldots, a_n)| \leq \epsilon.$$

Let X be the bitwise-XOR of two independent copies of X' and X'', i.e.

$$X = X' \oplus X'' = (X'_1 \oplus X''_1, \ldots, X'_n \oplus X''_n).$$

Then X is ϵ-bit-biased against sums modulo M.

Convergence of the bias for large weights The bias of a coefficient vector with respect to the uniform distribution can be large if there are only a few non-zero elements in the vector. However, when the weight is large, the bias is guaranteed to be small.

Lemma 2.10. *Let $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector of weight w. Then*

$$|\mathrm{bias}_U(a_1, \ldots, a_n)| \leq \left(1 - \frac{1}{M^2}\right)^w$$

In particular, for $w \geq M^2 \log(1/\epsilon)$ the bias is at most $\epsilon/2$.

Notice that the above lemma holds for all coefficient vectors $(a_1, \ldots, a_n)$ and moduli M, even when M is composite and the coefficients are not relatively prime to M. For example, when $M = 6$ and $(a_1, \ldots, a_n) = (2, \ldots, 2)$. In such a case, $\sum_i a_i R_i \mod M$ does not converge to the uniform distribution on $\mathbb{Z}_M^n$, but the above lemma still says that the bias tends to zero.

A similar result holds if we consider the bias of a large weight coefficient vector under a skewed distribution.

Lemma 2.11. *Let $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector of weight w. Let $Z_1, \ldots, Z_n \in \{0, 1\}$ be independently distributed with $\Pr[Z_i = 0] = (1 + \alpha)/2$. Then*

$$|\mathrm{bias}_{Z_1,\ldots,Z_n}(a_1, \ldots, a_n)| \leq \left(1 - \Omega\left(\frac{1-\alpha^2}{M^2}\right)\right)^w$$

In particular, for $w \geq cM^2 \log(1/\epsilon)/(1 - \alpha^2)$ for a sufficiently large constant c, the bias is at most $\epsilon/2$.

2.2 Hashing

We use hashing as one of the ingredients in our construction. A family (multiset) of functions $\mathcal{H} = \{h : [n] \to [k]\}$ is called a family of hash functions, if a randomly chosen function from the family behaves pseudorandomly under some specific meaning. We consider a hash function $H : [n] \to [k]$ to be a random variable depicting a randomly chosen function from the family. We say H can be generated *efficiently and explicitly* using s random bits, if a random function in the family can be sampled by a randomized polynomial-time algorithm using s random bits, and this function can be evaluated using a deterministic polynomial-time algorithm.

Fix $S \subset [n]$. We define the j-th bucket of H with respect to S, to be the set of elements of S mapped by H into j, i.e. $\{s \in S : H(s) = j\} = H^{-1}(j) \cap S$.

We will use the following three constructions of hash functions.

Lemma 2.12. *Assume k is a power of 2. There exists a hash function $H_1 : [n] \to [k]$ such that for every set $S \subset [n]$ of size at most $k \log(1/\epsilon)$, the probability that H_1 has a bucket $H_1^{-1}(j) \cap S$ with more than $100 \log(1/\epsilon)$ elements is at most $\epsilon/100$. Moreover, H_1 can be generated explicitly and efficiently using $O(\log n + \log(k/\epsilon) \log(k \log(1/\epsilon)))$ random bits.*

Lemma 2.13. *Assume k is a power of 2. There exists a hash function $H_2 : [n] \to [k]$ such that for every $S \subset [n]$ of size at least $100k^2$, the probability that H_2 has an empty bucket $H_2^{-1}(j) \cap S$ is at most $1/100$. Moreover, H_2 can be generated explicitly and efficiently using $O(\log n + \log^2 k)$ random bits.*

Lemma 2.14. *There exists a hash function $H_3 : [n] \to [16 \log(1/\epsilon)]$ such that for every $S \subset [n]$ of size at least $800k \log(1/\epsilon)$, the probability that H_3 has at least $\log(1/\epsilon)$ buckets $H_3^{-1}(j) \cap S$ with at most k elements is at most $\epsilon/100$. Moreover, H_3 can be generated explicitly and efficiently using $O(\log n + \log(1/\epsilon) \log(k \log(1/\epsilon)))$ random bits.*

The constructions of the hashes in Lemmas 2.12, 2.13 and 2.14 are based on almost t-wise independence. A sequence of random variables $X_1, \ldots, X_n \in \{0, 1\}$ is said to be *t-wise independent* if any t random variables in it are independent. It is said to be δ-almost t-wise independent if any t random variables in it are δ-close in statistical distance to independent. Explicit constructions of δ-almost t-wise independent distributions are known, with nearly optimal seed length [NN,AGHP].

We identify a function $h : [n] \to [\ell]$, where ℓ is a power of 2, by a sequence of $n \log \ell$ bits. We construct the hash functions by choosing the sequence of bits according to an δ-almost t-wise independent distribution, where the values of δ and t differ in the three constructions. The main tool in our analysis is a tail bound on t-wise independent distributions, due to Bellare and Rompel [BR], extended to the case of δ-almost t-wise distributions. We defer further details to the full version of the paper.

2.3 Pseudorandom generators for small space

An ingredient in our construction is the small-space pseudorandom generator of Impagliazzo, Nisan, and Wigderson [INW]. We first define branching programs, which form a non-uniform model of small-space computations.

Definition 2.15. *A* (read-once, oblivious) branching program *of length n, degree d and width w is a layered graph with $n+1$ layers, where each layer contains at most w vertices. From each vertex in the i-th layer ($1 \le i \le n$) there are d outgoint edges, numbered $0, 1, \ldots, d-1$. A vertex in the first layer is designated as the* start vertex. *Running the branching program on an input $x_1, \ldots, x_n \in [d]$ is done by following the path according to the inputs, starting at the start vertex. The* output *of the branching program is the vertex reached in the last layer.*

Definition 2.16. *A* pseudorandom generator for branching programs *of length n, degree d and width w with error ϵ is a function $G : \{0, 1\}^r \to [d]^n$, such that for every branching program of length n, degree d and width w, the statistical distance between the output of the branching program when run on uniform element in $[d]^n$, and the output when run on $G(U_r)$, is at most ϵ.*

Lemma 2.17. *[INW] There exists an explicit pseudorandom generators for branching programs of length n, degree d, width w with error ϵ, which uses $r = O(\log d + (\log n)(\log(n/\epsilon) + \log w))$ truly random bits.*

3 Construction using PRG for low-degree polynomials

We present in this section a simple construction for prime powers M, based on pseudorandom generators for low-degree polynomials. This construction is optimal for constant M, achieving a pseudorandom generator with seed length $O_M(\log(1/\epsilon))$ (where the constant depends exponentially on M).

Let $W = \Omega(M^3 \log 1/\epsilon)$. We will construct two generators: one for coefficient vectors of weight at most W, and one for coefficient vectors of weight at least W. Lemma 2.9 shows that the bitwise-XOR of the two generators is a pseudorandom generator for all coefficient vectors.

For small weights, we will use a distribution that is ϵ-almost W-wise independent. Such a distribution trivially fools coefficient vectors of weight at most W. It can be explicitly generated using $O(\log n + W + \log 1/\epsilon) = O_M(\log n/\epsilon)$ random bits [NN].

For large weights, let $(a_1, \dots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector of weight at least W. Consider first the distribution of $a_1 R_1 + \dots a_n R_n$ for independent and uniform bits $R_1, \dots, R_n$. By Lemma 2.10, $|\text{bias}_{U_n}(a_1, \dots, a_n)| < \epsilon/2$.

Consider now $Z_i \in \{0,1\}$, where $\Pr[Z_i = 0] = c/M$ for some integer $1 \le c \le M-1$. By Lemma 2.11,

$$|\text{bias}_{Z_1,\dots,Z_n \sim (c/M, 1-c/M)}(a_1, \dots, a_n)| < \epsilon/4,$$

given that $W = \Omega(M^3 \log(1/\epsilon))$ with a large enough hidden constant.

The benefit of using this skewed distribution, is that it can be simulated by low-degree polynomials modulo M. Since we assume M is a prime power, there is a polynomial $g : \mathbb{Z}_M \to \mathbb{Z}_M$ that maps some c elements of $\mathbb{Z}_M$ to 0, and the rest to 1. For example, if $M = p^k$, the polynomial $g(x) = x^{(p-1)p^{k-1}}$ maps elements divisible by p to 0, and the rest to 1. The degree of this g is at most $M-1$.

Let $Z_1, \dots, Z_n \in \{0,1\}^n$ be generated by $g(Y_1), \dots, g(Y_n)$, where $Y_1, \dots, Y_n \in \mathbb{Z}_M$ are uniform and independent. We thus have:

$$|\text{bias}_{Z_1,\dots,Z_n \sim g(U_{Z_M})^n}(a_1, \dots, a_n)| < \epsilon/4$$

Note that

$$\text{bias}_{Z_1,\dots,Z_n \sim g(U_{Z_M})^n}(a_1, \dots, a_n) = \mathbb{E}_{Y_1,\dots,Y_n \in \mathbb{Z}_M}[\omega^{a_1 g(Y_1) + \dots + a_n g(Y_n)}],$$

and that $a_1 g(Y_1) + \dots + a_n g(Y_n)$ is a polynomial of degree $\deg(g)$ in $Y_1, \dots, Y_n$. Thus we can derandomize the choice of $Y_1, \dots, Y_n$ using a a pseudorandom generator for low-degree polynomials [BV,Lov,Vio]. We note the results in these papers are stated for polynomials over prime finite fields, but they hold also for polynomials over $\mathbb{Z}_M$, using small-bias spaces for $\mathbb{Z}_M^n$ [Kat,AIK+,RSW,EGL+,AM] as a building block.

Lemma 3.1. *For every $M, n, d \in \mathbb{N}$, there is an explicit generator $G : \{0,1\}^r \to \mathbb{Z}_M^n$ such that for every polynomial $f : \mathbb{Z}_M^n \to \mathbb{Z}_M$ of degree at most d, the distribution of $f(\mathbb{Z}_M^n)$ and $f(G(U_r))$ are ϵ-close in statistical distance. The number of random bits required is $r = O(d2^d \log(M/\epsilon) + d\log(nM))$.*

We use the generator of Lemma 3.1 for error $\epsilon/4$ and degree $d = M - 1$. We thus get an explicit generator whose output distribution $(Y_1', \dots, Y_n') \in \mathbb{Z}_M^n$, such that:

$$|\mathbb{E}_{(Y_1',\dots,Y_n')}[\omega^{a_1 g(Y_1')+\dots+a_n g(Y_n')}] - \mathbb{E}_{Y_1,\dots,Y_n \in \mathbb{Z}_M^n}[\omega^{a_1 g(Y_1)+\dots+a_n g(Y_n)}]| < \epsilon/4$$

Thus, if we define our generator G' to output $g(Y_1'), \dots, g(Y_n')$, we have $Y_1', \dots, Y_n'$ are the output of G, we get an explicit generator,such that $|\text{bias}_{G'}(a_1, \dots, a_n)| < \epsilon/2$. Hence, we get that

$$|\text{bias}_{G'}(a_1, \dots, a_n) - \text{bias}_G(a_1, \dots, a_n)| < \epsilon$$

The randomness requirement of our generator comes directly from that of G, which is $O(M2^{M-1} \log(M/\epsilon) + M \log(nM)) = O_M(\log(n/\epsilon))$ for constant M.

4 Construction Based on Pseudorandom Walk Generators

4.1 A generator for small sums

We construct an ϵ-bit-biased generator for weights at most $W = 10^5 M^{24} \log(1/\epsilon)$. Let $(a_1, \dots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector of weight at most W.

The construction has three stages:

1. Partitioning the set of indices $[n]$ into W buckets using the hash function H_1. Lemma 2.12 guarantees that with probability at least $1 - \epsilon/100$, each bucket contains at most $O(\log(1/\epsilon))$ non-zero coefficients.
2. For each bucket j, generate the X_i's for i's in the j'th bucket using an almost $O(\log(1/\epsilon))$-wise independent distribution.
3. Use the INW generator given by Lemma 2.17 to generate the W seeds for the $O(\log(1/\epsilon))$-wise independent distributions used for the different buckets.

Lemma 4.1. *The above construction is an ϵ-bit-biased generator against coefficient vectors of weight at most W, using $O(\log n + \log(M/\epsilon) \log(M \log(1/\epsilon)))$ random bits.*

4.2 A generator for large sums

In this section we construct an ϵ-bit-biased distribution for coefficient vectors of weight at least $W = 10^5 M^{24} \log(1/\epsilon)$,

Recall that by Lemma 2.10, when the weight is large, the bias under the uniform distribution is small. Thus, to prove that a distribution is ϵ-bit-biased against large weight sums modulo M, it is enough to show that its bias is also small. We construct our ϵ-bit-biased generator in three steps:

- G_1: a generator that has bias at most $1 - 1/M^2$ on every coefficient vector which is not all zeros.

– G_2: a generator that has bias at most 0.91 on every coefficient vector of weight at least $100M^{24}$.
– G_3: a generator that has bias at most $\epsilon/2$ on every coefficient vector of weight at least $10^5 M^{24} \log 1/\epsilon$.

The generator G_3 will be our ϵ-bit-biased generator for large weights. We will sketch the constructions of G_1, G_2 and G_3, deferring full details and proofs to the full version of the paper. The main ingredient in the construction will be a derandomized expander product, which we now define and analyze.

Derandomized expander products

Definition 4.2. *We say an undirected graph H is a $(2^r, 2^d, \lambda)$-expander if H has 2^r vertices, it is regular of degree 2^d and all eigenvalues but the first have absolute value at most λ. We will identify the vertices of H with $\{0,1\}^r$, and the edges exiting each vertex with $\{0,1\}^d$ in some arbitrary way.*

We will need explicit constructions of expanders, which can be obtained from various known constructions.

Lemma 4.3. *For some constant $Q = 2^q$, there exist an efficient sequence H_k of $(Q^k, Q, 1/100)$-expanders.*

Impagliazzo, Nisan, and Wigderson [INW] compose two pseudorandom generators using an expander as follows:

Definition 4.4. *Let $G', G'' : \{0,1\}^r \to \{0,1\}^t$ be two bit generators. Let H be a $(2^r, 2^d, \lambda)$-expander. We define $G' \otimes_H G'' : \{0,1\}^{r+d} \to \{0,1\}^{2t}$ to be the concatenation $(G'(x), G''(y))$, where x is a random vertex in H, and y is a random neighbor of x in H.*

Our main lemma relates the bias of $G' \otimes_H G''$ to the biases of G' and G'':

Lemma 4.5. *Let $G', G'' : \{0,1\}^r \to \{0,1\}^t$ be two bit generators and let H be a $(2^r, 2^d, \lambda)$-expander. Let $(a_1, \ldots, a_t)$,$(b_1, \ldots, b_t)$ be two coefficient vectors. Then:*

$$\begin{aligned}&|\mathrm{bias}_{(G' \otimes_H G'')(U_{r+d})}(a_1, \ldots, a_t, b_1, \ldots, b_t)|\\ &\le f_\lambda(|\mathrm{bias}_{G'(U_r)}(a_1, \ldots, a_t)|, |\mathrm{bias}_{G''(U_r)}(b_1, \ldots, b_t)|)\end{aligned}$$

where $f_\lambda(x, y) = xy + \lambda\sqrt{1-x^2}\sqrt{1-y^2}$.

The bounds of [RV] imply that if $\max_{k \in \mathbb{Z}_M \setminus 0} |\mathrm{bias}_{G'(U_r)}(ka_1, \ldots, ka_t)| \le x$ then $\max_{k \in \mathbb{Z}_M \setminus 0} |\mathrm{bias}_{(G' \otimes_H G')(U_{r+d})}(a_1, \ldots, a_t, a_1, \ldots, a_t)| \le x^2 + \lambda \cdot (1 - x^2) = f_\lambda(x, x)$. If also $\max_{k \in \mathbb{Z}_M \setminus 0} |\mathrm{bias}_{G''(U_r)}(kb_1, \ldots, kb_t)| \le y$, then [RV] proof can be extended to show $\max_{k \in \mathbb{Z}_M \setminus 0} |\mathrm{bias}_{(G' \otimes_H G')(U_{r+d})}(ka_1, \ldots, ka_t, kb_1, \ldots, kb_t)| \le xy + \lambda \cdot (1 - xy)$, which is a worse than our bound $f(x, y)$ in case $x \ne y$ and does not suffice for our purposes. In addition, our result only requires a bound on the bias for the specific coefficient vectors $(a_1, \ldots, a_t), (b_1, \ldots, b_t)$ of interest, and not multiples of those coefficient vectors; this is crucial for our analysis when M is composite (cf., discussion after Lemma 2.10). On the other hand, the results of [RV] are more general in that they apply to generators G',G'' that correspond to random walks on any expander, not just Cayley graphs of $\mathbb{Z}_M$.

Construction of G_1 As in [INW,RV], we iterate the above product. Like [RV] we can use the constant-degree expander graphs $H_1, H_2, \ldots$ of Lemma 4.3 (as opposed to the expanders of degree poly(nw/ϵ) used by [INW] to prove Lemma 2.17). We define $G'_\ell : \{0,1\}^{\ell q} \to \{0,1\}^{2^{\ell-1}q}$ iteratively. $G'_1 : \{0,1\}^q \to \{0,1\}^q$ is the identity mapping, and $G'_\ell = G'_{\ell-1} \otimes_{H_{\ell-1}} G'_{\ell-1}$. We set $G_1 = G'_\ell$ for the minimal ℓ such that $2^{\ell-1}q \geq n$. We have:

Lemma 4.6. *Let $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector, which is not all zeros. Then:*

$$\text{bias}_{G_1}(a_1, \ldots, a_n) \leq 1 - \frac{1}{M^2}.$$

The seed-length of G_1 is $O(\log n)$.

Construction of G_2 We will construct G_2 based on G_1. Let $(a_1, \ldots, a_n)$ be a coefficient vector. Assume first a special case: Let $n = k2^s$, and partition the set of coefficients into 2^s consecutive parts, each of size k. Assume that each part contain at least one non-zero coefficient. By Lemma 4.6, applying G_1 to each part independently gives bias of at most $1 - 1/M^2$. We use this to analyze the bias of G_1 when applied in the special case:

Lemma 4.7. *Let $n = k2^s$. Let $a_1, \ldots, a_n$ be a coefficient vector such that for every $j \in [2^s]$,* weight$(a_{jk+1}, a_{jk+2}, \ldots, a_{(j+1)k}) > 0$. *Then:*

$$\text{bias}_{G_1}(a_1, \ldots, a_n) \leq \min\left(1 - \left(\frac{9}{8}\right)^s \frac{1}{M^2}, 0.9\right).$$

In particular if $s \geq 12 \log M$, we have $\text{bias}_{G_1}(a_1, \ldots, a_n) \leq 0.9$.

We now construct the generator G_2 in three steps:

- Obliviously partition the coefficients, using the hash function H_2. Re-order the coefficients according to the partition. This guarantees that with probability at least 0.99, the conditions of Lemma 4.7 hold.
- Use G_1 on the re-ordered coefficients.
- Return the pseudorandom bits back to the original order.

We have:

Lemma 4.8. *Let $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector, of weight at least $100M^{24}$. Then:*

$$\text{bias}_{G_2}(a_1, \ldots, a_n) \leq 0.91.$$

The seed length of G_2 is $O(\log n + \log^2 M)$.

Construction of G_3 We use G_2 to build our final ϵ-bit-biased generator G_3. The construction of G_3 has three parts:

- Use H_3 to partition the inputs to $O(\log(1/\epsilon))$ buckets, such that with probability $1-\epsilon/100$, most buckets contain at least $100M^{24}$ non-zero coefficients.
- Use G_2 on each bucket.
- Combine the generators for the separate buckets using expander products, with expanders of growing degree as in [RV].

Lemma 4.9. *Let $(a_1, \ldots, a_n) \in \mathbb{Z}_M^n$ be a coefficient vector, of weight at least $10^5 M^{24} \log(1/\epsilon)$. Then:*

$$\text{bias}_{G_3}(a_1, \ldots, a_n) \leq \epsilon/2.$$

The randomness required by G_3 is $O(\log n + \log(M/\epsilon)\log(M\log(1/\epsilon)))$.

Acknowledgments

We thank Emanuele Viola for drawing our attention to this problem. We thank Andrej Bogdanov for helpful discussions.

References

[AIK⁺] M. Ajtai, H. Iwaniec, J. Komlós, J. Pintz, and E. Szemerédi. Construction of a thin set with small Fourier coefficients. *Bull. London Math. Soc.*, 22(6):583–590, 1990.

[AKS] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic Simulation in LOGSPACE. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 132–140, New York City, 25–27 May 1987.

[AGHP] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[AM] N. Alon and Y. Mansour. ϵ-discrepancy sets and their application for interpolation of sparse polynomials. *Information Processing Letters*, 54(6):337–342, 1995.

[AR] N. Alon and Y. Roichman. Random Cayley graphs and expanders. *Random Structures Algorithms*, 5(2):271–284, 1994.

[BNS] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, pages 204–232, 15–17 May 1989.

[BR] M. Bellare and J. Rompel. Randomness-Efficient Oblivious Sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287, Santa Fe, New Mexico, 20–22 Nov. 1994. IEEE.

[BSVW] E. Ben-Sasson, M. Sudan, S. Vadhan, and A. Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 612–621 (electronic), New York, 2003. ACM.

[BV] A. Bogdanov and E. Viola. Pseudorandom Bits for Polynomials. In *FOCS*, pages 41–51. IEEE Computer Society, 2007.

[DGJ+] I. Diakonikolas, P. Gopalan, R. Jaiswal, R. A. Servedio, and E. Viola. Bounded Independence Fools Halfspaces. *CoRR*, abs/0902.3757, 2009.

[EGL+] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Efficient approximation of product distributions. *Random Structures Algorithms*, 13(1):1–16, 1998.

[HPS] J. Håstad, S. Phillips, and S. Safra. A well-characterized approximation problem. *Information Processing Letters*, 47(6):301–305, 1993.

[INW] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for Network Algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[Kat] N. M. Katz. An estimate for character sums. *Journal of the American Mathematical Society*, 2(2):197–200, 1989.

[Lov] S. Lovett. Unconditional pseudorandom generators for low degree polynomials. In R. E. Ladner and C. Dwork, editors, *STOC*, pages 557–562. ACM, 2008.

[MZ] R. Meka and D. Zuckerman. Small-Bias Spaces for Group Products. These proceedings, 2009.

[MST] E. Mossel, A. Shpilka, and L. Trevisan. On ϵ-biased generators in NC^0. *Random Structures Algorithms*, 29(1):56–81, 2006.

[MNN] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994.

[NN] J. Naor and M. Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM J. Comput.*, 22(4):838–856, Aug. 1993.

[Nao] M. Naor. Constructing Ramsey graphs from small probability spaces. Technical Report RJ 8810, IBM Research Report, 1992.

[Nis] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[RS] Y. Rabani and A. Shpilka. Explicit construction of a small epsilon-net for linear threshold functions. In M. Mitzenmacher, editor, *STOC*, pages 649–658. ACM, 2009.

[RSW] A. Razborov, E. Szemerédi, and A. Wigderson. Constructing small sets that are uniform in arithmetic progressions. *Combinatorics, Probability and Computing*, 2(4):513–518, 1993.

[RTV] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom Walks in Regular Digraphs and the RL vs. L Problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, 21–23 May 2006. Preliminary version on *ECCC*, February 2005.

[RV] E. Rozenman and S. Vadhan. Derandomized Squaring of Graphs. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM '05)*, number 3624 in Lecture Notes in Computer Science, pages 436–447, Berkeley, CA, August 2005. Springer.

[SZ] M. Saks and S. Zhou. $\mathrm{BP_H SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999. 36th IEEE Symposium on the Foundations of Computer Science (Milwaukee, WI, 1995).

[Vio] E. Viola. The Sum of d Small-Bias Generators Fools Polynomials of Degree d. In *IEEE Conference on Computational Complexity*, pages 124–127. IEEE Computer Society, 2008.