M.Sc. Thesis

Notions of Weak Pseudorandomness and $GF(2^n)$-Polynomials

Yoav Tzur

Advisor: Prof. Oded Goldreich

October 29, 2009

Abstract

We study the role of polynomials over the finite field $GF(2^n)$ in constructing and attacking various weak notions of pseudorandomness.

First, we consider the construction of weak pseudorandom generators using $GF(2^n)$-polynomials. We study the generator of [Nisan, STOC'90], that fools space-bounded nonuniform distinguishers, whose standard instantiation can be seen as a $GF(2^n)$-polynomial map. Among other results, we show that its resilience to (nonuniform) space-bounded machines is sensitive to the order of its output bits, and rule out some natural generalizations of the generator. We also consider $GF(2^n)$-polynomial-based generators that have small bias, meaning that they fool all (bit-)linear tests. We present a simple construction of a small-bias generator (called the geometric generator), which is similar (but not identical) to the powering construction presented by [Alon et al., RS&A'92]. A variant of this generator has shorter seed than the classical constructions of Alon et al., if the reciprocal of the bias required is polylogarithmic in the output length. We also use a variant of the geometric generator to construct a separation between small-bias and fooling small space that has good parameters, giving an exponential-stretch generator with exponentially small bias that is $O(1)$-space distinguishable (by a nonuniform machine).

Second, we study how $GF(2^n)$-polynomials can be used to distinguish various distributions from random ones. We provide a full picture of how various notions of $GF(2^n)$-linear distinguishers relate to standard linear (that is, $GF(2)$-linear) tests, and derive a reduction from having small-bias to fooling $GF(2^n)$-linear tests. In fact, this reduction is used to prove the small bias of the geometric generator mentioned above. We also conduct a study of $GF(2^n)$-bilinear and $GF(2^n)$-quadratic forms, and in specific characterize the sets of $GF(2)$-bilinear forms that can be computed exactly, or approximated to various rates, by $GF(2^n)$-bilinear forms. Higher-degree polynomials are also studied, and specifically we consider a recent result of [Viola, CCC'08] that showed that the sum of d independent instances of a small-bias generator fools polynomials of degree at most d. We show the tightness of this result with respect to the number of instances required, showing an explicit degree-$d+1$ polynomial that distinguishes this construction from random with constant gap.

Acknowledgements

Above all, I would like to thank my advisor, Oded Goldreich, for always devoting his time and patience, for guiding and advising me in my research, and for the emphasis he puts on writing skills.

I would also like to thank Igor Shinkar, Shachar Lovett, Or Meir and Noga Alon for countless helpful conversations that led to improving and simplifying many of the proofs and arguments. I thank Shachar Lovett for his major part in Section 4.5.

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 Pseudorandomness

Randomness plays an important role in many fields of computer science, including (but not limited to) algorithm design, complexity theory and cryptography. The randomness complexity of an algorithm, measuring the amount of random coins it requires, is thus considered an important resource. The notion of *derandomization* captures the attempt to reduce the randomness complexity of an algorithm, without significantly affecting its input/output behavior. As a generic method to do this, a distribution is considered *pseudorandom* if it cannot be told apart from a truly random distribution (on strings of similar length). When saying "cannot be told apart", we implicitly fix both a set of limitations on (often, the complexity of) the potential algorithms trying to identify the distribution they are given, and a notion of "telling apart" (e.g., behaving differently with significant probability, called the *distinguishing gap*).

Another issue of importance is the complexity required to sample the pseudorandom distribution. An (efficient) algorithm that uses a short seed of truly random bits to sample this distribution is called a *pseudorandom generator*.

The archtypical case is that of an efficient generator that fools all efficient distinguishers up to a negligible distinguishing gap, that is, a polynomial-time generator, sampling a pseudorandom distribution that cannot be told apart from random by any polynomial-time distinguisher with distinguishing gap

lower bounded by the reciprocal of a positive polynomial. This notion, called a *general-purpose pseudorandom generator*, can be used to reduce the randomness complexity of every efficient algorithm, but always requires some computational hardness assumptions, which are captured by the notion of a one-way function. For background on pseudorandomness, see e.g. Chapter 8 of [Gol08].

1.1.2 Weak pseudorandomness

Despite the appeal of the general-purpose notion above, it is often fruitful to consider other limitations on the class of potential distinguishers. Many interesting results about such *weak notions* of pseudorandomness arise from considering space-complexity, or algebraic assumptions about the very functions trying to distinguish a given distribution from truly random bits. Given such assumptions on the algorithm we try to fool, it is sometimes possible to construct pseudorandom generators without any computational hardness assumptions, and obtain superior parameters. For example, even the notion of fooling linear tests, called *small bias*, studied by [NN] and with simple constructions presented in [AGHP], has found many important applications in various areas of computer science (e.g., [NN, BSSVW]). Higher degree polynomials of the output bits have also been considered as distinguishers, e.g. by [LVW, Bog, BV, Vio].

1.1.3 Using $GF(2^n)$-polynomials

While the basic definitions refer to distributions over bits, several constructions of various weak notions of pseudorandomness rely either directly or indirectly on constructions that take place in a finite field of order 2^n, denoted $GF(2^n)$. As an example, we note that the generator of Nisan ([Nis]) that fools space-bounded distinguishers works with blocks of n bits, and when instantiated with the standard implementation of a universal hash function as an affine $GF(2^n)$-linear function, each output block is the result of applying a $GF(2^n)$-multilinear polynomial to the $GF(2^n)$-elements represented by the seed blocks. Another example, where $GF(2^n)$ is used more directly, is the third small-bias generator of [AGHP], in which a random $GF(2^n)$-element is raised to different powers, with an output bit produced by an inner product of the representation of the resulting $GF(2^n)$-element with a (seed-determined) vector in $\{0,1\}^n$. In fact, this construction is related to the "geometric generator" (considered in this

work), producing the representation of a random $GF(2^n)$-geometric series.

In addition to using $GF(2^n)$-polynomials to construct weak pseudorandom generators, such polynomials can also be considered as distinguishers (of pseudorandom distributions from truly random distributions). This turns out to be useful since fooling various forms of $GF(2^n)$-polynomial tests is often related to resisting various statistical tests of bits: among others, linear tests, bit-polynomial tests, and tests that can be implemented in small space. A methodology for proving weak pseudorandomness thus amounts to proving resistance to various $GF(2^n)$-polynomial-based tests and relating such tests to binary tests.

Given the apparent popularity of using $GF(2^n)$, and specifically different forms of $GF(2^n)$-polynomials, to obtain various weak pseudorandorm generators, we study in this work several aspects of the role of $GF(2^n)$-polynomials in achieving this: both as a tool for constructing pseudorandom generators, or as a tool for distinguishing between different distributions.

1.2 Overview and organization of the thesis

Here we provide a brief overview of the entire work. A more elaborate overview precedes each of the chapters. Highlights of some of our important results appear in the following section.

In Chapter 2 we present the preliminaries relevant to the entire work. We define the weak notions of pseudorandomness that we study, and describe the standard representation scheme of $GF(2^n)$ and its properties which we use.

In Chapter 3 we consider constructions of weak pseudorandom generators based on $GF(2^n)$-polynomials. We start with a bound on the stretch of any $GF(2^n)$-polynomial of bounded degree that is a small bias generator (which is the weakest of the notions we consider). We then consider the Nisan generator that fools space-bounded distinguishers, and possible variants, with respect to the aforementioned bound (and while referring to the implementation of its hash functions by affine $GF(2^n)$-functions). We continue by presenting the "geometric generator", a simple $GF(2^n)$-based small-bias generator, and present a variant that almost matches the bound on the stretch that we showed. We end the chapter with a variant of the geometric generator (with exponential stretch and bias) that separates small-bias from fooling small-space. The same

technique also shows that the resilience of the Nisan generator to small-space distinguishers is sensitive to the order of its output bits.

In Chapter 4 we turn the table around and consider using $GF(2^n)$-polynomials as distinguishers between distributions. Specifically, we compare their distinguishing power to the power of $GF(2)$-polynomials of the same degree. After a discussion of different ways in which $GF(2^n)$-polynomials can be viewed as distinguishing between distributions, we study linear tests (degree 1 polynomials), and present a full picture of the relationships between different types of $GF(2^n)$-linear tests and $GF(2)$-linear tests. Studying higher degree tests is done by a reduction from the results of Chapter 5, but here we only have partial results. We end the chapter by proving the tightness of a theorem of Viola, which states that the sum of d small-bias generators fools $GF(2)$-polynomials of degree d, by presenting an explicit polynomial of degree $d+1$ that distinguishes an instantiation of this construction. This polynomial is derived using previous results of this chapter.

In Chapter 5 we compare the power of $GF(2^n)$-polynomials to that of $GF(2)$-polynomials for computing and approximating functions, over inputs drawn from the uniform distribution. Focusing mostly on degree 2, we study when a $GF(2)$-polynomial can be either calculated exactly or approximated to various levels by a $GF(2^n)$-polynomial. Most of the results in this chapter refer to bilinear forms, and specifically we present a $GF(2)$-bilinear form that cannot be approximated significantly better than guessing from any $GF(2^n)$-bilinear form. Partial results regarding quadratic forms are also presented.

Prior publications of parts of the thesis. Section 4.3 and a version of Section 4.2 for the linear case $d = 1$, as well as the resulting Section 3.4, were published in [Tzu]. Section 4.5, which was done in joint work with Shachar Lovett, was published in [LT].

1.3 Highlights

We present some highlights of the thesis.

1.3.1 Separating small-bias from fooling small-space

Any generator that fools all nonuniform small-space distinguishers (defined in Definition 3.3) necessarily has small bias (meaning that it fools linear tests, as in Definition 2.1). However, the other direction does not hold: there exist small-bias distributions that can be distinguished from random by a nonuniform constant-space distinguisher. While existing separations had very poor parameters (see [Eve]), we present (in Section 3.5), a construction of an exponential-stretch and exponentially-small-bias generator that can be distinguished from random by a nonuniform constant-space machine.

1.3.2 Order-sensitivity of Nisan's generator

The generator of Nisan ([Nis]) fools all nonuniform distinguishers that use less memory than some given space bound. An interesting corollary to our work is Corollary 3.18 stating that, for a natural instantiation of the Nisan generator, there exists some reordering of its output bits that produces a generator that can be distinguished from random in (nonuniform) constant space. Thus, the resilience of Nisan's generator to space-bounded distinguishers is sensitive to the order of its output bits.

1.3.3 Disqualifying a variant of Nisan's generator

The generator of [Nis], fooling nonuniform space-bounded machines, is based on applying randomly selected hash functions to a randomly selected seed element. Different *subsets* of these functions are applied to the same element several times, in the *same fixed order*, producing the output blocks of the generator. A natural question that arises asks whether using the same functions in *different orders* will still yield a small-space-fooling construction, thus potentially improving the stretch of the generator. In Subsection 3.3.4 we answer this question in the negative, disqualifying a natural extension of the Nisan generator.

1.3.4 Using $GF(2^n)$ to construct small-bias generators

We present a methodology for constructing small-bias generators (defined in Definition 2.1), showing (in Corollary 4.6) that it suffices that a generator fool all $GF(2^n)$-linear tests (Definition 4.1). This methodology is used in Section 3.4, where a simple construction of a small-bias generator is presented. This construction, called "the geometric generator", outputs a random $GF(2^n)$-geometric series, that is, the ℓ blocks $\tilde{a} \cdot \tilde{b}^i$, for $i = 0, \ldots, \ell - 1$ and $\tilde{a}, \tilde{b}$ uniformly selected from $GF(2^n)$. A variant of this generator, presented in Subsection 3.4.3, uses more input variables, to obtain a seed shorter than the constructions of [AGHP], when the desired bias is not too small compared to the output length.

1.3.5 An explicit lower bound for fooling polynomials by the sum of small-bias generators

The result of Viola ([Vio]) shows that the sum of d independent instances of a small-bias generator (fooling all linear tests, as in Definition 2.1), fools all polynomials of degree at most d (see [Vio] for definition of fooling). In Section 4.5 we show that this result is tight with respect to the number of small-bias generators summed, presenting an explicit degree-$d+1$ polynomial that distinguishes from random the sum of d instances of a small-bias generator of our choice.

1.3.6 Approximating $GF(2)$-bilinear forms from $GF(2^n)$-bilinear forms

In Section 5.3 we study which $GF(2)$-bilinear forms (described by $M(x, y) = x^T My$ for some $n \times n$ matrix M) can be approximated to various rates from the (only nondegenerate) $GF(2^n)$-bilinear form; that is, the approximator is allowed to be an arbitrary function of the multiple of the two elements $\tilde{x}$ and $\tilde{y}$ represented by the two vectors $x, y \in \{0,1\}^n$. We provide a full characterization of the $GF(2)$-bilinear form according to possible approximation rates ranging from "close to 1" to "very close to $\frac{1}{2}$".

Chapter 2

Preliminaries

2.1 Weak notions of pseudorandomness

We define here the notions of pseudorandomness that are used throughout the entire work, and refer to the relevant section for notions that are chapter-specific.

The weakest notion of pseudorandomness that we consider is the notion of *small bias*, introduced by [NN]:

Definition 2.1 (Small-biased distribution)**.** *For $\ell \in \mathbb{N}$, $\varepsilon > 0$, a distribution D over $\{0,1\}^\ell$ is called ε-*biased *if for every nonzero $\alpha \in \{0,1\}^\ell$:*

$$\left| \Pr_{x \sim D}[\langle \alpha, x \rangle = 0] - \frac{1}{2} \right| \leq \varepsilon,$$

where $\langle \alpha, x \rangle$ denotes the inner product $\sum_i \alpha_i x_i$ (over $GF(2)$).

We prefer to view distributions as the outputs of pseudorandom generators:

Definition 2.2 (Small-bias generator)**.** *For $k, \ell \in \mathbb{N}$, $\varepsilon > 0$, a mapping $G : \{0,1\}^k \to \{0,1\}^\ell$ is called an ε-*bias generator *of* stretch *$\ell(k)$, if the distribution induced by $G(s)$ for s selected uniformly in $\{0,1\}^k$ is ε-biased.*

Remark. When discussing pseudorandom generators, it is common to also consider the complexity of the generator itself (as opposed to the complexity of potential distinguishers). We do not include this in the definition of a small-bias generator, since in Chapter 3 we will consider a specific class of constructions

anyway, whereas in Chapter 4 the complexity of the generator is not our focus.

A basic notion for comparing distributions is the notion of *statistical distance*. It will be especially useful to us when we work with distributions over sets larger than $\{0,1\}$, and specifically the elements of $GF(2^n)$.

Definition 2.3 (Statistical distance)**.** *For $\varepsilon > 0$, two distributions X, Y are said to be ε-*close (in statistical distance) *if for every event E,*

$$\left|\Pr_X[E] - \Pr_Y[E]\right| \leq \varepsilon.$$

*Conversely, if there exists an event such that $|\Pr_X[E] - \Pr_Y[E]| \geq \varepsilon$, then X and Y are said to be ε-*far (in statistical distance)*.*

Chapter-specific notions of pseudorandomness. The generalization of the notion of small-bias to that of resisting higher degree polynomials (over bits) was initiated by [LVW] (in fact, they considered the larger class of depth-2 boolean circuits), and further studied in [Bog] (although there, only super-constant sized fields were considered). It is studied in Chapter 4, with the relevant definitions in Section 4.2.

Section 4.2 also defines the various notions of $GF(2^n)$-tests (as opposed to the bit-based tests above), where Section 4.3 discusses the case of linear tests.

Fooling space-bounded distinguishers is discussed in Chapter 3, and defined in Subsection 3.3.1, among with the generalized notion of *block-automata*.

2.2 Representation of $GF(2^n)$

2.2.1 Basic Notations

We present our notations and conventions regarding the representation scheme of the finite field of order 2^n, called $GF(2^n)$.

Notation 2.4. *For a vector $a \in \{0,1\}^n$, we will denote by $\tilde{a}$ the $GF(2^n)$ element represented by a. When writing an expression in $GF(2^n)$ elements (denoted by a tilde), the arithmetic will usually be that of $GF(2^n)$; otherwise (when elements are without a tilde), we treat them as vectors in $\{0,1\}^n$ and use the arithmetic of the vector space (over $GF(2)$).*

Notation 2.5 (The polynomials $c(x)$ and $p_a(x)$)**.** *We will use the standard representation of $GF(2^n)$ as the quotient $GF(2)[x]/\langle c(x)\rangle$, fixing an irreducible polynomial $c(x) \in GF(2)[x]$ of degree n (where $\langle c(x)\rangle$ is the ideal generated by $c(x)$ in the ring $GF(2)[x]$). An element $\tilde{a} \in GF(2^n)$, represented by the bit string $a = a_0 a_1 ... a_{n-1}$, corresponds to $p_a(x) = \sum_{i=0}^{n-1} a_i x^i \in GF(2)[x]/\langle c(x)\rangle$. For details on this representation, see any standard algebra textbook (e.g., [BM]).*

Notation 2.6 (The matrix C)**.** *Denote by C the companion matrix of $c(x)$, with ones under the diagonal and the coefficients $c_0, ..., c_{n-1}$ in the right column:*

$$C = \begin{pmatrix} 0 & 0 & \dots & c_0 \\ 1 & 0 & \dots & c_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & c_{n-1} \end{pmatrix}$$

Note that for an element $\tilde{b} \in GF(2^n)$ represented by $b \in \{0,1\}^n$, the vector $C \cdot b$ corresponds to multiplying p_b by the fixed polynomial x (represented by the bit-string $e_1 = 010...0$) and reducing the result modulo $c(x)$, that is, $C \cdot b$ represents the multiplication $\tilde{e}_1 \cdot \tilde{b}$.

Notation 2.7 (The matrix M_a)**.** *For $a \in \{0,1\}^n$, let $M_a = p_a(C)$. This is the linear operator that performs multiplication by $\tilde{a}$ on elements viewed as n-dimensional vectors over $GF(2)$. That is, for every $\tilde{b} \in GF(2^n)$, represented by the vector $b \in \{0,1\}^n$, the binary representation of the element $\tilde{a} \cdot \tilde{b}$ is $M_a \cdot b$. To see this, write $p_a(C) \cdot b = \sum_i a_i C^i b$, and note that this vector represents the reduction of $\sum_i a_i p_b(x) \cdot x^i = \sum_{i,j} a_i b_j x^{i+j}$ modulo $c(x)$, which indeed corresponds to multiplying $\tilde{b}$ by $\tilde{a}$ in the field $GF(2^n)$.*

An alternate formulation of the multiplication-by-$\tilde{a}$ matrix M_a, used and studied in Subsection 3.5.2, is the multiplication $M_a = R \cdot L_a$ of the two matrices defined in the next two paragraphs.

Notation 2.8 (The matrix L_a)**.** *The $2n-1 \times n$ matrix L_a corresponds to multiplying a polynomial in $GF(2)[x]$ by $p_a(x)$, without reducing modulo the irreducible polynomial $c(x)$ used to represent the field. The matrix L_a is often*

referred to as the convolution matrix. Explicitly,

$$L_a = \begin{pmatrix} a_0 & 0 & 0 & \dots & 0 \\ a_1 & a_0 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ a_{n-2} & \dots & a_1 & a_0 & 0 \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \\ 0 & a_{n-1} & \dots & a_2 & a_1 \\ \vdots & & \ddots & & \\ 0 & \dots & 0 & a_{n-1} & a_{n-2} \\ 0 & \dots & 0 & 0 & a_{n-1} \end{pmatrix},$$

i.e., $[L_a]_{ij} = a_{i-j}$ *if* $j \leq i < j + n$ *and 0 otherwise.*

Notation 2.9 (The matrix R)**.** *The* $n \times 2n - 1$ *matrix* R *reduces* $GF(2)$*-polynomials (of degree at most* $2n - 1$*) modulo* $c(x)$*. The* j*-th column of* R *is the coefficients vector of the polynomial* x^j *reduced modulo* $c(x)$*.*

2.2.2 Basic Properties

The distributivity of $GF(2^n)$ implies that the matrix M_a should be linear in the vector a. This indeed follows from each of the above definitions of M_a.

Proposition 2.10. *Every* $a, b \in \{0,1\}^n$ *satisfy* $M_a + M_b = M_{a+b}$*.*

The following lemma says that the i-th bit of the representation of $\tilde{a} \cdot \tilde{b}$ can be written as the inner product of $Q \cdot a$ and b, where Q is a fixed matrix over $GF(2)$. The statement generalizes to any fixed linear combination in the representation of $\tilde{a} \cdot \tilde{b}$, denoted γ (where the aforementioned case corresponds to $\gamma = e_i$).

Lemma 2.11 (The matrix Q_γ)**.** *For every fixed linear combination* $\gamma \in \{0,1\}^n$*, there exists a matrix* $Q_\gamma \in \{0,1\}^{n \times n}$ *such that for every two vectors* $u, v \in \{0,1\}^n$*:*

$$\langle \gamma, M_u \cdot v \rangle = \langle Q_\gamma \cdot u, v \rangle,$$

Moreover, Q_γ *is invertible whenever* γ *is nonzero.*

Remark. By commutativity of multiplication in $GF(2^n)$, it always holds that $M_u v = M_v u$, and thus $\langle u, Q_\gamma \cdot v\rangle = \langle v, Q_\gamma \cdot u\rangle$. We will sometimes refer to Lemma 2.11 as asserting $\langle \gamma, M_u \cdot v\rangle = \langle u, Q_\gamma \cdot v\rangle$. Using $u^T Q_\gamma v = \langle u, Q_\gamma v\rangle = \langle v, Q_\gamma u\rangle = v^T Q_\gamma u = u^T Q_\gamma^T v$ for every $u, v \in \{0,1\}^n$, we conclude that the matrix Q_γ is symmetric.

We present a simple nonconstructive proof for Lemma 2.11; a constructive proof giving an explicit expression for Q_γ is given in Subsection 4.3.3.

Proof (nonconstructive). Fixing γ, the value $\langle \gamma, M_u \cdot v\rangle$ is a bilinear form in the bits of u and v since the bits of M_a are linear in the bits of a (Proposition 2.10). Thus, $\langle \gamma, M_u \cdot v\rangle$ can be represented as $v^T Q_\gamma u$ for some $n \times n$ matrix Q_γ. For the moreover part, let d satisfy $\langle \gamma, d\rangle = 1$ (e.g., if the i-th bit of γ is 1, set $d = e_i$). Then, for every nonzero $u \in \{0,1\}^n$, setting v_u to represent $\tilde{u}^{-1}\tilde{d} \in GF(2^n)$ (i.e., $v_u = M_u^{-1} d$), we get $\langle Q_\gamma \cdot u, v_u\rangle = \langle \gamma, M_u \cdot v_u\rangle = \langle \gamma, d\rangle = 1$ and so $Q_\gamma \cdot u$ cannot be the zero vector. This implies that the kernel of Q_γ is trivial. □

We also show that the matrix Q_γ is linear in the vector γ (as is the case for the matrix M_a in the vector a):

Proposition 2.12. *Every $\gamma_1, \gamma_2 \in \{0,1\}^n$ satisfy $Q_{\gamma_1} + Q_{\gamma_2} = Q_{\gamma_1+\gamma_2}$.*

Proof. For every two vectors $u, v \in \{0,1\}^n$:

$$\begin{aligned}
\langle u, (Q_{\gamma_1} + Q_{\gamma_2})v\rangle &= \langle u, Q_{\gamma_1} v\rangle + \langle u, Q_{\gamma_2} v\rangle \\
&= \langle \gamma_1, M_u v\rangle + \langle \gamma_2, M_u v\rangle \\
&= \langle \gamma_1 + \gamma_2, M_u v\rangle \\
&= \langle u, Q_{\gamma_1+\gamma_2} v\rangle\,,
\end{aligned}$$

where the second and last equality are due to Lemma 2.11. Letting u and v range over all basis vectors e_i and e_j, we get that the matrices $Q_{\gamma_1} + Q_{\gamma_2}$ and $Q_{\gamma_1+\gamma_2}$ are equal. □

The squaring operation over $GF(2^n)$, that is, $\tilde{a} \mapsto \tilde{a}^2$, is a $\{0,1\}^n$-linear operation in the bits of the vector a representing $\tilde{a}$, since $(\tilde{a} + \tilde{b})^2 = \tilde{a}^2 + \tilde{b}^2$. (In fact, this mapping of $GF(2^n)$ is called the Frobenius automorphism.) The transformation over $\{0,1\}^n$ can thus be described by a matrix S. We write S explicitly:

Notation 2.13 (The matrix S)**.** *The matrix S, describing the squaring transformation over the representation of $GF(2^n)$ elements, can be written as $R \cdot E$ where E is the $2n-1 \times n$ matrix with 1 in the $(2i+1, i)$-th entry for $i = 0, \ldots, n-1$, and zeroes in the other entries, and R is the matrix defined in Notation 2.9. Note that for a vector a representing the $GF(2^n)$-element corresponding to the polynomial $p_a(x) = \sum_i a_i x^i$, the vector Ea corresponds to the degree $\leq 2n-1$ polynomial $\sum_i a_i x^{2i+1}$, which is exactly $p_a(x)^2$, so the vector $Sa = R \cdot Ea$ corresponds to the polynomial $p_a(x)^2$ reduced modulo $c(x)$, that is, Sa indeed represents the $GF(2^n)$-element $\tilde{a}^2$.*

2.3 Block-generators and $GF(2^n)$-generators

We will often consider generators that work over $GF(2^n)$. Specifically, in Chapter 3 we consider such $GF(2^n)$-generators as $GF(2^n)$-polynomials maps:

Definition 2.14. *We say that a map $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$ is $GF(2^n)$-*polynomial of individual degree *d if each of the ℓ output elements can be written as a polynomial over $GF(2^n)$ of individual degree at most d in each of the k input elements.*

Having fixed a representation of $GF(2^n)$ by $\{0,1\}^n$, we will often work with the binary generators that represent $GF(2^n)$-generators:

Definition 2.15 (Binary version)**.** *Fix $n, k, \ell \in \mathbb{N}$. For a mapping $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$, its* binary version *is the block-generator $G : \{0,1\}^{n \cdot k} \to \{0,1\}^{n \cdot \ell}$, parsing its input to k blocks $x_1, \ldots, x_k \in \{0,1\}^n$, and outputting the concatenation of the ℓ vectors representing the ℓ output elements of $\tilde{G}$ applied to the k elements $\tilde{x}_1, \ldots, \tilde{x}_k$.*

Throughout this work, we will sometimes omit the explicit term *binary version*, and just refer to G as the binary version of a previously defined generator $\tilde{G}$, or vice versa.

Abusing the term in Definition 2.14, for a $GF(2^n)$-polynomial map $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$ we sometimes refer to its binary version, $G : \{0,1\}^{k \cdot n} \to \{0,1\}^{\ell \cdot n}$ as a $GF(2^n)$-polynomial map as well.

The general concept of a (binary) generator that parses its input to n-bit blocks, works on logical units in $\{0,1\}^n$, and outputs the concatenation of n-bit output blocks, is called a *block-generator*:

Definition 2.16 (Block-generator)**.** *For $n, k, \ell \in \mathbb{N}$, a generator $G : \{0,1\}^{n \cdot k} \to \{0,1\}^{n \cdot \ell}$ is called a* block-generator *with* block-length *n and* block-stretch *ℓ.*

2.4 Using a larger prime field

Many of our results hold also when using a prime field other than $GF(2)$. When of specific interest, we will sometimes write this explicitly in the text. We present here some generalizations of the above definitions, over the prime field $GF(q)$ for q a prime number.

The $GF(q)$ analogue of Definition 2.1 is (see, e.g., [Eve] or [GW]):

Definition 2.17. *For $\ell \in \mathbb{N}$, $\varepsilon > 0$ and a prime q, a distribution X over $GF(q)^\ell$ is called ε-*biased *if for every nonzero $\alpha \in GF(q)^\ell$:*

$$\left\| \mathbb{E}_{x \sim X}[e^{\langle x, \alpha \rangle \cdot 2\pi i/q}] \right\| \leq \varepsilon,$$

where here $\langle \alpha, x \rangle$ denotes the inner product $\sum_i \alpha_i \cdot x_i$ over $GF(q)$, and the multiplication by $2\pi i/q$ is then done over the complex field $\mathbb{C}$.

This notion can be related to the notion of statistical distance (from the uniform distribution over $GF(q)$). Standard arguments (e.g. [Gol95] or Appendix B in [BV]) give that an ε-biased (over $GF(q)$) distribution is $\sqrt{q-1} \cdot \varepsilon/2$-close to the uniform distribution (in statistical distance).

The notion of *bias* can also be defined over other (nonprime) fields; we refer the interested reader to [Eve].

We remark that the matrix S defined in Notation 2.13 is $GF(2)$-specific; in $GF(q)$ the mapping $\tilde{a} \mapsto \tilde{a}^p$ is $GF(q)^n$-linear only for p a multiple of q (see any algebra textbook, e.g. [BM]).

Chapter 3

$GF(2^n)$-Polynomials as Weak Pseudorandom Generators

3.1 Introduction

3.1.1 Motivation

As opposed to the case of pseudorandomness against all efficient distinguishers, explicit and unconditional constructions are known for many weak notions of pseudorandomness. Among other such notions, pseudorandomness against space-bounded distinguishers is achieved by the generator of [Nis]; and small-biased distributions were constructed in [NN], and then more simply in [AGHP].

While the basic definitions refer to distributions over bits, many of the constructions benefit from parsing their input to blocks, and provide guarantees regarding block-based distinguishers. Specifically, considering each seed-block of n bits as the representation of an element in $GF(2^n)$, and outputting the representation of some $GF(2^n)$-based function of the seed, has found use in several of the above constructions. It is often beneficial to write these functions as $GF(2^n)$-polynomials, either because they have a simple form as a polynomial, of because they have bounded degree. We present two such examples, which will be studied more extensively within this chapter.

The first example is the generator of [Nis], which when instantiated with the standard construction of a universal hash function as an affine $GF(2^n)$-linear function, has each output block representing a $GF(2^n)$-multilinear polynomial of the (elements represented by the) seed blocks (see section 3.3 for details). The second example is the powering construction of [AGHP], which is based on the $GF(2^n)$-linear test resistance of a random geometric series $(\tilde{a} \cdot \tilde{b}^i)_i$ over $GF(2^n)$, although there $\tilde{a}$ is set to 1, and instead each output bit is the inner product of the representation of $\tilde{b}^i$ with some fixed random vector in $\{0,1\}^n$. Actually, one contribution of this chapter is showing that the "pure" geometric generator has small-bias.

Given the apparent usefulness of $GF(2^n)$-polynomials in this context, we undertake in this chapter a study of various aspects of using $GF(2^n)$-polynomials to construct weak pseudorandom generators.

3.1.2 Overview

After defining the generators and distinguishers relevant to this chapter in Subsection 3.3.1, we begin by presenting in Section 3.2 an upper bound on the stretch of any small-bias generator that, when parsing its input and output to blocks, can have its output blocks presented as $GF(2^n)$-polynomials of bounded degree in its input blocks. We mention that this section uses the results of Section 4.3, which relate indistinguishablility by $GF(2^n)$-linear tests to small bias.

Indeed, this bound also applies to a stronger notion of pseudorandomness that fools nonuniform space-bounded distinguishers. In Section 3.3 we show that a natural instantiation of the well known Nisan generator ([Nis]) that does fool nonuniform space-bounded distinguishers can be described as a member of the aforementioned class of generators, and more specifically the class of $GF(2^n)$-multilinear polynomials, and has essentially optimal stretch within this class. Specifically, in Subsection 3.3.4 we rule out a seemingly natural improvement of the Nisan generator, which is based on reusing the same hash functions in different orders.

Since the bound presented in Section 3.2 increases with the degree of the polynomials, a natural suggestion for a small-bias generator consists of using high degree polynomials. In Section 3.4, we show that indeed a random geometric series over $GF(2^n)$ has small bias, using again the connection estab-

lished in Section 4.3. On the other hand, this generator is distinguishable by a strengthened version of a space-bounded distinguisher, namely a nonuniform block-automaton, which the Nisan generator does fool. This provides a very natural separation between these two classes of distinguishers.

We also derive in Section 3.5 a variant of the geometric generator that still has small bias and large stretch, but is distinguishable by a (standard) nonuniform constant-space machine. Indeed, this provides another natural separation, which is finer. Using the same technique, we also show that the Nisan generator's resilience to (nonuniform) space-bounded distinguishers is sensitive to the order of the output bits.

3.2 A bound on the stretch of $GF(2^n)$-polynomial small-bias generators

In this section we prove a bound $\ell \leq (d+1)^k$ on small-bias generators (as in Definition 2.1, i.e., fooling bit-linear tests) that are $GF(2^n)$-polynomial maps of individual degree d (as in Defintion 2.14) with block-stretch $k \mapsto \ell$.

To prove a bound on the stretch of $GF(2^n)$-polynomial maps that are small-bias generators, we present a generator-specific nontrivial $GF(2)$-linear relation that must be satisfied by the output bits of (the binary version of) this generator if its stretch is too big. We will do this by first finding a $GF(2^n)$-linear relation that must be satisfied by the generator, and then use the results of Section 4.3 to derive a $GF(2)$-linear relation in the output bits of the binary version.

We note that in Section 7 of [MST], the case $n = 1$ is studied and a bound $\ell \leq O(k^d)$ is proven for small-bias generators whose output bits are polynomials of (total) degree at most d, using an argument similar to the one in our Theorem 3.1. In contrast, our focus here is on the general case of $n \geq 1$.[1] (We mention that the rest of [MST] studies a more restricted class of small-bias generators, in which each output bit depends on at most d input bits.)

[1]Note that since each bit of the binary version of a $GF(2^n)$-polynomial in k variables of individual degree at most d is a $GF(2)$-polynomial in kn variables of total degree at most kd (by the simple Lemma 4.4), we can derive for any n an upper bound $\ell \cdot n = O((kn)^{kd})$. This bound is inferior to that of Theorem 3.1.

3.2.1 A linear dependency over $GF(2^n)$

Theorem 3.1. *Let $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$ be a $GF(2^n)$-polynomial map of individual degree d. If $\ell > (d+1)^k$, then there is a nontrivial $GF(2^n)$-linear combination that takes value zero on every output of $\tilde{G}$.*

Since a nontrivial $GF(2^n)$-linear combination is distributed uniformly over $GF(2^n)$ when applied to uniformly distributed elements, this linear tests distinguishes $\tilde{G}$ from random with gap $1 - 2^{-n}$, and thus $\tilde{G}$ does not α-fool $GF(2^n)$-linear tests for any $\alpha < 1 - 2^{-n}$ (see Definition 4.1).

Proof. We treat each of the ℓ polynomials describing the output blocks of $\tilde{G}$ as a $GF(2^n)$-linear combination of monomials. The generator $\tilde{G}$ can be described by a $GF(2^n)$-matrix with ℓ rows and $(d+1)^k$ columns, where the (i,j)-th entry is the coefficient of the j-th monomial (among the $(d+1)^k$ possible monomials of individual degree at most d in each of the k input elements[2]) in the polynomial describing the i-th output element. Since the hypothesis assigns this matrix more rows (representing output blocks) than columns (representing possible monomials), a linear dependency between its rows is implied. Thus, we have obtained a nontrivial $GF(2^n)$-linear combination of the output elements of $\tilde{G}$ that always sums to zero. □

Remark. In Subsection 3.4.3 we show, for any d, k and n, a $GF(2^n)$-polynomial generator of individual degree d with stretch $\ell = (d+1)^{k-1}$, that $\frac{\ell}{2^n}$-fools $GF(2^n)$-linear tests. This leaves a multiplicative gap of $d+1$ between the lower and upper bounds.

3.2.2 A linear dependency over bits

In Section 4.3 we relate linear dependencies over $GF(2^n)$ and linear dependencies of the bits representing the $GF(2^n)$-elements. Specifically, by combining Theorem 3.1 with Lemma 4.4 (which is the "easy direction" of that relation), we obtain:

[2]We note that this bound can be decreased by one: since a (nontrivial) $GF(2^n)$-linear combination that always sums to a constant element of $GF(2^n)$ (not necessarily zero) also leads to a distinguisher with the same gap $1 - 2^{-n}$, the constant monomial should not be included in the matrix, reducing the bound to $(d+1)^k - 1$.

Corollary 3.2. *Consider any $GF(2^n)$-polynomial map $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$ of individual degree d, and let $G : \{0,1\}^{kn} \to \{0,1\}^{\ell n}$ be its binary version. If $\ell > (d+1)^k$, then there is a nontrivial $GF(2)$-linear combination that takes value zero on the output of G. Thus, G is not an ε-bias generator for any $\varepsilon < \frac{1}{2}$.*

Remark. The bound of Corollary 3.2 can be (insignificantly) improved to $\ell \leq (d+1)^k - (\lfloor d/2 \rfloor + 1)^k$, using the linear dependence between the representation of a $GF(2^n)$-element $\tilde{x}$ and its square $\tilde{x}^2$. This is shown in Appendix A.1.

3.3 The Nisan generator

The generator presented in [Nis], to which we refer as the Nisan generator, is shown to fool all nonuniform machines that use at most cn space, for some universal constant $c > 0$, and more generally, all block automata (see the following Subsection 3.3.1 for the definitions of the distinguisher classes). An output sequence is produced by choosing an element $x \in \{0,1\}^n$ and k' functions $h_1, \ldots, h_{k'}$ from a 2-universal family of hash functions over $\{0,1\}^n$, and outputting the image of x under all possible compositions of any of the $2^{k'}$ subsets of the hash functions, in a fixed order (for a definition of universal hash functions, see, e.g., [Nis] or Section D.2 of [Gol08]). Specifically, if we identify the $2^{k'}$ output blocks with bit-strings σ in $\{0,1\}^{k'}$, then the σ-th output block will be $h_1^{\sigma_1} \circ h_2^{\sigma_2} \circ \ldots \circ h_{k'}^{\sigma_{k'}}(x)$.

We will use a standard implementation of universal hashing, described in Subsection 3.3.3, to get an instantiation of the Nisan generator that can be seen as a $GF(2^n)$-polynomial generator of individual degree 1. Moreover, the resulting generator will be a member of a smaller class of generators, which we describe in Subsection 3.3.5 as the "seed-and-hash" class, and have essentially optimal stretch within this class (for individual degree 1). We also show, in Subsection 3.3.4, that a natural extension of the Nisan generator, in which the hash functions $h_1, \ldots, h_{k'}$ are applied to x in different orders, does not have small bias (and thus does not fool space-bounded distinguishers).

3.3.1 Definitions of the distinguisher classes

The generator of [Nis] is designed to fool nonuniform space bounded machines. These are modeled as nonuniform Turing Machines (with space-complexity guar-

antees) with unidirectional access to their *random* tape (thus having to record every past coin toss, if needed for future use).

Since we are interested in the distinguishing capabilities of such machines with respect to their *random* tape, we model our distinguishers (that take the pseudorandom bits in their *input* tape) as machines that have unidirectional access to their input:

Definition 3.3 (Space-bounded distinguishers)**.** *A* nonuniform space-$s(\ell)$ distinguisher *is a nonuniform Turing Machine that on input of length ℓ uses at most $s(\ell)$ bits of storage and runs in time at most $2^{s(\ell)}$, with unidirectional access to its input tape (that is, the reading head can only move right).*

In fact, the Nisan generator is shown to fool the following stronger type of distinguishers:

Definition 3.4. *A* block automaton with block-length n *is an automaton over the alphabet $\Sigma = \{0,1\}^n$, with at most 2^{cn} states for some universal constant c.*

If we set $n = 1$ in Definition 3.4, we get a model equivalent to that of Definition 3.3, with $s(\ell) = c$. Since a block automaton with block-length n is also a block-automaton with block-length kn for every $k \in \mathbb{N}$, every nonuniform space-cn distinguisher is also a block-automaton with any block-length n, so Definition 3.3 is indeed stronger than Definition 3.4 (and thus fooling all block-automata implies fooling all nonuniform space-cn distinguishers).

3.3.2 Overview of the analysis of Nisan's generator

Although not of direct use to us, we describe briefly the ideas of the proof that the Nisan generator fools block-automata. This proof motivates our interest in this class of distinguishers.

The proof of resilience to block-automata is based on the idea that for almost every hash function h, the distribution of the pair $(x, h(x))$ for x selected uniformly from $\{0,1\}^n$ behaves similarly to a uniformly selected pair $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, with respect to hitting arbitrary subsets of $\{0,1\}^n$. Considering for each state of the automaton the subsets of $\{0,1\}^n$ that bring the automaton to this state, the automaton cannot distinguish the pair $(x, h(x))$ from (x, y). The application rules of the hash functions can be seen as a recursive use of this idea, where in each step an output consisting of two independently

and identically distributed parts $z_1, \ldots, z_t$ and $z'_1, \ldots, z'_t$ is replaced by the output $z_1, \ldots, z_t, h(z_1), \ldots, h(z_t)$ for a newly selected hash function h. Each step reduces the randomness required by a factor of 2, with an additive cost of selecting one hash function, eventually giving an exponential improvement of the seed length.

3.3.3 A $GF(2^n)$-polynomial generator

The Nisan generator uses as a primitive a collection of universal hash functions over $\{0,1\}^n$. A standard implementation of such a collection is the collection of all affine functions over $GF(2^n)$: a function h is specified by two elements $\tilde{a}, \tilde{b} \in GF(2^n)$, and sends a vector $x \in \{0,1\}^n$, representing the element $\tilde{x} \in GF(2^n)$, to the representation of the element $\tilde{a} \cdot \tilde{x} + \tilde{b}$ (in the arithmetic of $GF(2^n)$). For details, see, e.g., Section D.2 of [Gol08]. The usage of this standard implementation instantiates the Nisan generator as a $GF(2^n)$-polynomial map; moreover, the individual degree in each input element is 1 (i.e., it is $GF(2^n)$-multilinear):

Claim 3.5. *For every integer $r \geq 0$ and index-vector $\bar{j} \in [k']^r$,*

$$h_{j_1} h_{j_2} ... h_{j_r}(\tilde{x}) = \tilde{x} \cdot \prod_{i=1}^{r} \tilde{a}_{j_i} + \sum_{s=1}^{r} \tilde{b}_{j_s} \cdot \prod_{i=1}^{s-1} \tilde{a}_{j_i},$$

where $h_j(\tilde{x}) = \tilde{a}_j \cdot \tilde{x} + \tilde{b}_j$.

(Recall that for every function $T(i)$, for $p > q$, the empty product $\prod_{i=p}^{q} T(i)$ is defined as 1, while the empty sum $\sum_{i=p}^{q} T(i)$ is defined as 0.)

Proof. We prove this by induction on r. For $r = 0$ the claim is trivial. Now consider any $r \geq 1$. Using the induction hypothesis on $h_{j_2}, \ldots, h_{j_r}(\tilde{x})$, we get:

$$\begin{aligned}
\tilde{x} \cdot \prod_{i=1}^{r} \tilde{a}_{j_i} + \sum_{s=1}^{r} \tilde{b}_{j_s} \cdot \prod_{i=1}^{s-1} \tilde{a}_{j_i} &= \tilde{x} \cdot \prod_{i=1}^{r} \tilde{a}_{j_i} + \sum_{s=2}^{r} \tilde{b}_{j_s} \cdot \prod_{i=1}^{s-1} \tilde{a}_{j_i} + \tilde{b}_{j_1} \cdot \prod_{i=1}^{0} \tilde{a}_{j_i} \\
&= \tilde{a}_{j_1} \cdot \left(\tilde{x} \cdot \prod_{i=2}^{r} \tilde{a}_{j_i} + \sum_{s=2}^{r} \tilde{b}_{j_s} \cdot \prod_{i=2}^{s-1} \tilde{a}_{j_i} \right) + \tilde{b}_{j_1} \cdot 1 \\
&= \tilde{a}_{j_1} \cdot h_{j_2} ... h_{j_r}(\tilde{x}) + \tilde{b}_{j_1} \\
&= h_{j_1} h_{j_2} ... h_{j_r}(\tilde{x}).
\end{aligned}$$

□

By Theorem 3.1, the best block-stretch that a small-bias $GF(2^n)$-polynomial generator of individual degree 1 in k output blocks (with block size n) can have is $\ell = 2^k$, whereas the above instantiation of the Nisan generator (fooling nonuniform space-bounded machines, which is a stronger notion than having small-bias) has block-stretch $\ell = 2^{k'} = 2^{\frac{k-1}{2}}$, establishing:

Proposition 3.6. *There exists a $GF(2^n)$-polynomial map of individual degree 1 with block-stretch $\ell(k) = 2^{\frac{k-1}{2}}$ that fools block-automata (with block-size n).*

3.3.4 The permutations variant

We consider a natural idea for modifying the Nisan generator in order to improve its stretch: after choosing a seed element x, and k' pairwise-independent hash functions, instead of producing output blocks by applying different subsets of the hash functions in one fixed order (i.e., for every $\sigma \in \{0,1\}^{k'}$, output $h_1^{\sigma_1} \circ h_2^{\sigma_2} \circ \ldots \circ h_{k'}^{\sigma_{k'}}(x)$), we produce output by applying all the hash functions in different orders (i.e., for every permutation $\pi : [k'] \to [k']$, output $h_{\pi(1)} \circ h_{\pi(2)} \circ \ldots \circ h_{\pi(k')}(x)$). This potentially suggests a substantially better block-stretch: $k'!$ instead of $2^{k'}$.

However, instantiating the generator as above with the standard implementation of pairwise independent hash functions as $GF(2^n)$-affine functions, this generator is again a $GF(2^n)$-multilinear map (by Claim 3.5) and is thus ruled out immediately by the stretch bound of Theorem 3.1, because its block stretch is $k'! = (\frac{k-1}{2})! = \omega(2^k)$. This establishes nonconstructively:

Corollary 3.7. *For large enough k', the permutations variant of the Nisan generator with k' hash functions is not a small bias generator (and certainly does not fool nonuniform small-space machines).*

That is, we get nonconstructively that for large enough k' (in fact, for $k' \geq 20$) there exists a $GF(2^n)$-linear combination of the output elements of the permutations variant that always sums to zero. We will now improve the result by showing an *explicit* linear dependency, for any $k' \geq 4$. If $y_1 = h_1h_2(z), y_2 = h_2h_1(z)$ and $y_3 = h_1h_2(z'), y_4 = h_2h_1(z')$ are four output blocks of the generator[3], for two arbitrary expressions z, z' (e.g. $z = h_3h_4h_5...h_{k'}(x), z' =$

[3]Or, in fact, of any seed-and-hash generator; see Subsection 3.3.5.

$h_4 h_3 h_5 ... h_{k'}(x))$, then

$$\begin{aligned}(\tilde{y}_1 - \tilde{y}_2) - (\tilde{y}_3 - \tilde{y}_4) &= (\tilde{a}_1(\tilde{a}_2\tilde{z} + \tilde{b}_2) + \tilde{b}_1 - \tilde{a}_2(\tilde{a}_1\tilde{z} + \tilde{b}_1) - \tilde{b}_2) \\ &\quad -(\tilde{a}_1(\tilde{a}_2\tilde{z}' + \tilde{b}_2) + \tilde{b}_1 - \tilde{a}_2(\tilde{a}_1\tilde{z}' + \tilde{b}_1) - \tilde{b}_2) \\ &= ((\tilde{a}_1 - 1)\tilde{b}_2 - (\tilde{a}_2 - 1)\tilde{b}_1) - ((\tilde{a}_1 - 1)\tilde{b}_2 - (\tilde{a}_2 - 1)\tilde{b}_1) \\ &= 0\end{aligned}$$

(recall that the hash functions are implemented as $h_i(\tilde{x}) = \tilde{a}_i \cdot \tilde{x} + \tilde{b}_i$).

In fact, if for some fixed permutation $\pi : [m] \to [m]$ we have four output blocks $y_1 = h_{j_1} h_{j_2} ... h_{j_m}(z), y_2 = h_{j_{\pi(1)}} h_{j_{\pi(2)}} ... h_{j_{\pi(m)}}(z)$ and $y_3 = h_{j_1} h_{j_2} ... h_{j_m}(z'), y_4 = h_{j_{\pi(1)}} h_{j_{\pi(2)}} ... h_{j_{\pi(m)}}(z')$, then both differences $y_1 - y_2$ and $y_3 - y_4$ are independent of z (or z')[4] and must be the same, giving a linear combination $(y_1 - y_2) - (y_3 - y_4) = 0$.

3.3.5 Seed-and-hash variants

When presenting the Nisan generator as a $GF(2^n)$-polynomial map (of individual degree 1), as well as the permutations variant, we used the structure of choosing one seed element $x \in \{0,1\}^n$ and k' functions from a collection of universal hash functions, and outputting various applications of the different hash functions to x, in some arbitrary order; this is a $GF(2^n)$-polynomial map if we use the standard implementation of a universal hash function as a $GF(2^n)$-affine function.

While the permutations variant turned out to have too long a stretch, other variants of the Nisan generator, using different strategies of applying the hash functions to the seed element x, can still potentially improve the block-stretch from $\ell = 2^{\frac{k-1}{2}}$ up to the bound $\ell = 2^k$ of Theorem 3.1.[5] We consider this class of variants, based on choosing a seed element x and k' hash functions $h_1, \ldots, h_{k'}$, and outputting the results of applying the functions $h_1, \ldots, h_{k'}$ to x in some strategy. That is, each output index in $[\ell]$ is mapped to a (potentially long) tuple of indices in $[k']$, where the output block whose index is mapped to the index-vector $\bar{i} = (i_1, \ldots, i_t) \in [k']^t$ is produced by $h_{i_1} \circ \ldots \circ h_{i_t}(x)$. We call this class the "seed-and-hash" class. Indeed, for the standard implementation of

[4] By claim 3.5, the only term dependent on z (or z') is $z \cdot \prod_{i=1}^{k'} \tilde{a}_{\pi(i)}$, which is independent of the ordering π by commutativity. Hence the difference is independent of z (or z').

[5] Or almost to $(d+1)^k$; see Appendix A.1 for a slightly improved bound.

universal hashing that we used, these generators yield simply-described $GF(2^n)$-polynomial maps. While so far we allowed each hash function to be applied only once in the expression of each output block, resulting in a $GF(2^n)$-multilinear polynomial, if we now bound by d the number of times each hash function can be used for a single output block, we get a $GF(2^n)$-polynomial map of individual degree d (by Claim 3.5).

Inspecting the proof of Theorem 3.1, its bound can be improved on the seed-and-hash class, since the number of possible monomials is now smaller. By Claim 3.5, all monomials are a multiplication of some of the $\tilde{a}_i$'s (each raised to a power at most d) times either $\tilde{x}$ or some $\tilde{b}_j$, and thus the total number of possible monomials is $(d+1)^{k'} \cdot (k'+1)$. For $d = 1$, we get that the best block-stretch possible is $2^{k'} \cdot (k'+1) = 2^{k'(1+o(1))}$, giving that the Nisan generator (with block-stretch $2^{k'}$) is essentially optimal within this class.

3.4 The geometric generator

The bound of Theorem 3.1 suggests that a small-bias generator with good stretch might be obtained by letting the output blocks be described by polynomials of potentially high degree. As a first attempt, one may consider the sequence $\tilde{a}, \tilde{a}^2, \tilde{a}^3, \ldots$ for a randomly chosen $\tilde{a} \in GF(2^n)$; however, recalling that the bits representing $\tilde{a}^2$ and the bits representing $\tilde{a}$ are linearly dependent (see Notation 2.13, where the squaring matrix S is defined), this cannot be a small-bias generator. We thus consider a generator that on seed two elements $\tilde{a}, \tilde{b} \in GF(2^n)$, outputs the geometric series $(\tilde{a} \cdot \tilde{b}^i)_{i=0}^{\ell}$. This generator is similar, though not identical, to a known generator from [AGHP], called the powering generator.

3.4.1 Small bias

Not violating the bound of Theorem 3.1, this generator is not disqualified from having small-bias. We show that this is indeed the case, using the definitions and results from Section 4.3. We note that this generator was considered in [ASS], where it was implicitly proven to have small bias (see further discussion in Section 4.3).

Proposition 3.8 (The geometric generator)**.** *For $n, \ell \in \mathbb{N}$, the generator $\tilde{G}$:*

$GF(2^n)^2 \to GF(2^n)^{\ell+1}$ *defined by* $\tilde{g}_i(\tilde{a},\tilde{b}) = \tilde{a}\cdot\tilde{b}^i$ *for* $i = 0,\ldots,\ell$*, called the* geometric generator*,* $\frac{\ell}{2^n}$*-fools* $GF(2^n)$*-linear tests (as in Definition 4.1).*

Proof. Fix any nontrivial $GF(2^n)$-linear combination $\bar{c} = (\tilde{c}_0,\ldots,\tilde{c}_\ell) \in GF(2^n)^{\ell+1}$, and consider the expression

$$\left\langle \bar{c}, \tilde{G}(\tilde{a},\tilde{b}) \right\rangle = \sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{a} \cdot \tilde{b}^i = \tilde{a} \cdot \sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{b}^i$$

(where here $\langle \bar{x}, \bar{y} \rangle$ denotes the $GF(2^n)$-inner product $\sum_i \tilde{x}_i \tilde{y}_i$). When $\sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{b}^i$ is nonzero, $\tilde{a} \cdot \sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{b}^i$ is uniformly distributed in $GF(2^n)$. Thus, for any fixed $\bar{c}$ the statistical distance between the distribution induced by the above expression over uniformly selected $\tilde{a},\tilde{b} \in GF(2^n)$, and the uniform distribution over $GF(2^n)$, is at most $\Pr_{\tilde{b}}\left[\sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{b}^i = 0\right]$, which is bounded by $\frac{\ell}{2^n}$, since the nonzero $GF(2^n)$-polynomial $\sum_{i=0}^{\ell} \tilde{c}_i \cdot \tilde{x}^i$ of degree at most ℓ can have at most ℓ roots (in $GF(2^n)$). □

Using Corollary 4.6, we immediately get:

Corollary 3.9. *For* $n, \ell \in \mathbb{N}$*, the generator* $G : \{0,1\}^{2n} \to \{0,1\}^{(\ell+1)\cdot n}$ *defined as the binary representation of* $\tilde{G}$*, is an* $\frac{\ell}{2^n}$*-bias generator.*

3.4.2 Distinguishable by a nonuniform block-automaton

Since the geometric generator passes the small-bias test, and given its similarity to the above instantiation of Nisan's generator, one may hope that the ideas of the proof of Nisan's generator would apply here (and establish that it fools space-bounded distinguishers as well). We show that this is not the case: the strengthened notion of a block-automaton, which Nisan's generator is shown to fool, can distinguish the geometric generator from random, for many values of n. Specifically, whenever $2^n - 1$ has a small (constant) factor, we will describe a block-automaton that distinguishes the geometric series from a random sequence of elements in $GF(2^n)$ with a large gap. Note that this happens for infinitely many n's: specifically for all even n's, it holds that 3 divides $2^n - 1$.[6] We thereby assume that $2^n - 1 = q \cdot p$ for some constant q.[7]

[6] This is because $2^n - 1 = (2^{n/2} - 1)(2^{n/2} + 1)$, and 3 must divide one of them since it does not divide $2^{n/2}$.

[7] This would give an automaton with a finite number of states. In fact, to comply with Definition 3.4, we can take any subexponential $q = 2^{o(n)}$.

Let α be a generator of the cyclic multiplicative group $GF(2^n)^* = GF(2^n) \setminus \{0\}$. For an element $x \in GF(2^n)^*$, define its discrete logarithm, $\log_\alpha(x)$ to be the unique integer $j \in \{0, \ldots, 2^n - 2\}$ such that $x = \alpha^i$. Note that since $\alpha^{2^n-1} = 1$, for every two elements $x, y \in GF(2^n)^*$ it holds that

$$\log_\alpha(x \cdot y) \equiv \log_\alpha(x) + \log_\alpha(y) \pmod{2^n - 1}. \tag{3.1}$$

The distinguisher is based on the fact that the group $GF(2^n)^*$ contains a large multiplicative subgroup (when $2^n - 1$ has a small nontrivial factor):

Proposition 3.10. *The set $G_0 = \{x \in GF(2^n)^* : \log_\alpha(x) \equiv 0 \pmod{q}\}$ is a multiplicative subgroup of $GF(2^n)^*$ of size $\frac{2^n-1}{q} = p$.*

Proof. The claim follows from Equation (3.1) since q divides $2^n - 1$. The size of G_0 is the number of integers in $\{0, \ldots, 2^n - 2\}$ divisible by q, which is exactly $\frac{2^n-1}{q} = p$. □

We now observe that if both seed elements $\tilde{a}$ and $\tilde{b}$ happen to fall in G_0, an event that occurs with probability $(\frac{|G_0|}{|GF(2^n)|})^2 = (\frac{2^n-1}{q \cdot 2^n})^2 > \frac{1}{q^2} - 2^{-n}$, then all the output elements of the geometric generator must fall in G_0, an event that happens with probability $\left(\frac{|G_0|}{|GF(2^n)|}\right)^\ell < q^{-\ell}$ for a random sequence of ℓ elements.

Recognizing whether a block represents an element in G_0 may require large space, but as a block automaton, allowed to compute arbitrary functions when processing a block, our distinguisher can do it. We therefore define the distinguisher to first check whether the first two blocks, supposedly $\tilde{a}$ and $\tilde{a} \cdot \tilde{b}$, are in G_0. If not, the distinguisher flips a coin and answers randomly; otherwise, the distinguisher checks the next output block and answers "pseudorandom" if and only if it is also in G_0. By the discussion above, the distinguishing gap obtained is at least

$$\begin{aligned}
\Pr_{y_0, y_1}[y_0, y_1 \in G_0] \cdot (1 - \Pr_{y_2}[y_2 \in G_0]) &> (\frac{1}{q^2} - 2^{-n}) \cdot (1 - \frac{1}{q} + 2^{-n}) \\
&\geq \frac{q-1}{q^3} - 2^{-n} \\
&\geq \Omega(1).
\end{aligned}$$

Remark. If we check that *all* the remaining $\ell - 2$ output blocks are in G_0, we get the better (but still just constant) distinguishing gap

$$(\frac{1}{q^2} - 2^{-n}) \cdot (1 - q^{-\ell+2}) \geq \frac{1}{q^2} - (2^{-n} + q^{-\ell}) \geq \Omega(1).$$

We have thus established:

Theorem 3.11. *For $\ell \geq 3$ and infinitely many n's, the geometric generator over $GF(2^n)$ with block-stretch ℓ can be distinguished with constant gap by a block-automaton (with block-size n).*

We have obtained a separation between the notion of a small-bias generator and the notion of fooling block-automata. Our separation generator has exponential stretch and bias, showing that small-bias generators that have good parameters may be distinguished by a simple block-automaton with a good gap. Separating small-bias from fooling nonuniform space-space machines, which are weaker than block-automata, is considered in Section 3.5.

Large distinguishing gap. We improve the above distinguishing gap by noting that the subgroup G_0 is just a specific case of a more general phenomenon, namely that multiplying two elements is equivalent to adding their discrete logs. For $i \in \{0, \ldots, q-1\}$, define the set G_i as $\{x \in GF(2^n)^* : \log_a(x) \equiv i \pmod q\}$. Then, each G_i is of size $p = \frac{2^n-1}{q}$, and by Equation (3.1) we get:

Proposition 3.12. *For $i_1, i_2 \in \{0, \ldots, q-1\}$, let $i_3 = i_1 + i_2 \bmod q$. Then for every $x \in G_{i_1}$ and $y \in G_{i_3}$, it holds that $x \cdot y \in G_{i_3}$.*

If we first determine the sets G_{i_1} and G_{i_2} that contain the first two output elements (supposedly $\tilde{a}$ and $\tilde{a} \cdot \tilde{b}$), which can indeed be done and remembered by a block-automaton since there are only $q = O(1)$ sets G_i, we can predict the discreet log of the j-th output element, using Equation (3.1):

$$\log_\alpha(\tilde{a} \cdot \tilde{b}^j) \equiv \log_\alpha(\tilde{a}) + j \cdot \log_\alpha(\tilde{b}) \equiv i_1 + j \cdot (i_2 - i_1) \pmod q. \tag{3.2}$$

Accordingly, our distinguisher will first check if any of the first two blocks is zero. If so (an event that has probability at most $2 \cdot 2^{-n}$ for both a random output of the generator and a truly random sequence of elements), it will just check that all other elements are zero, which is guaranteed for all outputs of the generator but happens with probability $2^{-n \cdot (\ell-2)}$ on a random sequence.

Otherwise, the distinguisher will remember $i_1 = \log_\alpha(\tilde{g}_0)$ and $i_2 = \log_\alpha(\tilde{g}_1)$ and verify, for $j = 2 \dots \ell$ that the j-th output block is nonzero and that its discreet log agrees with Equation (3.2). Since only $p = \frac{2^n-1}{q}$ elements in $GF(2^n)$ are accepted, this happens for each block with probability at most $\frac{2^n-1}{q \cdot 2^n} < \frac{1}{q}$ in a random sequence (whereas every output of the geometric generator satisfies Equation (3.2)). Thus a random sequence (that does not start with zeroes) is accepted with probability at most $q^{-\ell+2}$, and the total distinguishing gap we obtain is at least

$$\begin{aligned} 2 \cdot 2^{-n} \cdot (1 - 2^{-n \cdot (\ell-2)}) + (1 - 2 \cdot 2^{-n}) \cdot (1 - q^{-(\ell-2)}) &\geq 1 - q^{-(\ell-2)} - 2^{-n+1} \\ &= 1 - \exp(-\Omega(n)) - \exp(-\Omega(\ell)). \end{aligned}$$

3.4.3 Using more variables

We describe a generalization of the geometric generator (as in Proposition 3.8). Consider the generator that on input k elements $\tilde{a}, \tilde{b}_1, \dots, \tilde{b}_{k-1} \in GF(2^n)$ outputs all elements $\tilde{a} \cdot M(\tilde{b}_1, \dots, \tilde{b}_{k-1})$ for M ranging over all monomials of total degree at most t. Note that $GF(2^n)$-linear tests are still fooled (with probability $1 - \frac{t}{2^n}$), as a multivariate polynomial of total degree at most t can still have at most t roots. We stretch $n \cdot k$ bits to $n \cdot \binom{t+k-1}{k-1}$ bits,[8] which is larger than $n \cdot (\frac{t+k-1}{k-1})^{k-1} \geq 2^{(k-1)(\log t - \log(k-1))}$ bits. Thus, for a bias of $\varepsilon = \frac{t}{2^n}$, to get $m = 2^{(k-1)(\log t - \log(k-1))} = 2^{(k-1)(n - \log(1/\varepsilon) - \log(k-1))}$ output bits, we need a seed of length at most $\frac{k}{k-1} \log m + k \log(1/\varepsilon) + k \log(k-1)$. (For comparison, recall that the original generator corresponds to the special case $k = 2$, which has seed length $2 \log m + 2 \log(1/\varepsilon)$.)

Theorem 3.13. *For every $\varepsilon \in (0,1)$ and integers $m > 0$ and $k \geq 2$, the k-variate geometric generator with ε-bias and m output bits requires a seed of length at most $\frac{k}{k-1} \log m + k \log(1/\varepsilon) + k \log(k-1)$ bits.*

Tightness of Theorem 3.1. If we bound the individual degree of each seed element (rather than the total degree) by d, we obtain a block-stretch of $k \mapsto (d+1)^{k-1}$. As a member of the class of $GF(2^n)$-polynomial generators, this almost matches the bound $(d+1)^k$ of Theorem 3.1.

[8] The number of monomials in $k-1$ variables $b_1, \dots, b_{k-1}$ of total degree exactly t can be thought of as the number of ways to choose t elements to multiply from the $k-1$ different elements $b_1, \dots, b_{k-1}$, ignoring order, which is $\binom{t+k-2}{k-2}$. If we want all monomials of total degree *at most* t, we add the constant 1 as a k-th variable, to get $\binom{t+k-1}{k-1}$.

Optimizing the seed length. The expression $\frac{k}{k-1}\log m + k\log(1/\varepsilon)$ is minimized when $k = \sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 1$. Clearly, if $\log m \leq \log(1/\varepsilon)$, then the minimal $k = 2$ (which is the original geometric generator) yields the shortest seed. However, when $\log m$ is significantly greater than $\log(1/\varepsilon)$, it is clear that a larger k would give a shorter seed (since the expression $\frac{k}{k-1}$ decreases as k increases). Since k must be an integer (and at least 2), we set $k = \left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 1 \right\rceil$, and get a seed of length at most

$$\frac{\left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 1 \right\rceil}{\left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} \right\rceil} \cdot \log m + \left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 1 \right\rceil \cdot \log(1/\varepsilon) + \left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 1 \right\rceil \log \left\lceil \sqrt{\frac{\log m}{\log(1/\varepsilon)}} \right\rceil,$$

bounded by

$$\left(1 + \frac{1}{\sqrt{\frac{\log m}{\log(1/\varepsilon)}}}\right) \log m + \left(\sqrt{\frac{\log m}{\log(1/\varepsilon)}} + 2\right) \log(1/\varepsilon) + \tilde{O}(\sqrt{\log m}),$$

which can be simplified to

$$\log m + 2\sqrt{\log m \cdot \log(1/\varepsilon)} + 2\log(1/\varepsilon) + \tilde{O}(\sqrt{\log m}).$$

We have established

Corollary 3.14. *For every $\varepsilon \in (0,1)$ and $m \in \mathbb{N}$ there exists an explicit ε-bias generator that generates m output bits with a seed of length at most $\log m + 2\sqrt{\log m \cdot \log(1/\varepsilon)} + 2\log(1/\varepsilon) + \tilde{O}(\sqrt{\log m})$. Specifically, if $\varepsilon \geq 2^{-\operatorname{poly}\log\log m} \geq 2^{-\tilde{O}(\sqrt{\log m})}$, this is $\log m + \tilde{O}(\sqrt{\log m})$.*

Comparison to other generators. The standard explicit constructions of [AGHP] use a seed of length $2\log m + 2\log(1/\varepsilon)$, which is longer than the above if m is significantly greater than $1/\varepsilon$ (explicitly, if $\sqrt{\log m} > 2\sqrt{\log(1/\varepsilon)} + \operatorname{poly}\log\log m$, i.e. $m^{1-o(1)} > \varepsilon^{-4}$). We note that a construction of [NN] achieves a shorter seed when m is significantly greater than $1/\varepsilon$: they obtain $\log m + O(\log(1/\varepsilon))$; however, our construction is simpler and more natural (as are the constructions of [AGHP]).

Note that if the output length m is exponential in a "security parameter" n (for example, but not necessarily, the field size), then $\varepsilon \geq 2^{-\operatorname{poly}\log\log m}$, means $\varepsilon \geq 2^{-\operatorname{poly}\log n}$. For instance, to get 2^n bits with bias $1/\operatorname{poly}(n)$, we only need $n + \tilde{O}(\sqrt{n})$ bits of seed, as opposed to $2n$ bits in the original construction.

3.4.4 Using a larger prime field

The construction of the geometric generator and its proof work similarly if we replace the prime field $GF(2)$ by the larger $GF(q)$, for some prime q. The distribution of any linear combination in the $\ell+1$ output blocks of the geometric generator is $\frac{\ell}{q^n}$-close to uniform (over $GF(q)$) in statistical distance. The results of Section 4.3, including Corollary 4.6, also extend to $GF(q)$, giving a generator that fools $GF(q)$-linear tests. To get m elements of $GF(q)$ that ε-fool $GF(q)$-linear tests, we need a seed of about $2\log_q m + 2\log_q(1/\varepsilon) = \frac{2}{\log q}(\log m + \log(1/\varepsilon))$ elements of $GF(q)$. These can be described by $2\log m + 2\log(1/\varepsilon)$ bits. For more about using a larger prime field, see Subsection 2.4.

3.5 A small-bias generator of large stretch that is $O(1)$-space distinguishable

While psedurandomness against nonuniform machines of small space implies fooling $GF(2)$-linear tests (for any positive space bound), it is interesting to note that this is indeed a strictly stronger notion. A known separation result, having small-bias but distinguishable in constant space, is described in [Eve]: the uniform distribution over the set of bit sequences with number of ones that is divisible by 3, has small bias - but can clearly be identified by an $O(1)$-space machine that counts ones. However, this yields a pseudorandom generator of very poor stretch (even when ignoring the complexity of sampling the distribution): to generate m bits, $m - \log_2(3)$ bits of seed are required (to choose one of about $2^m/3$ bit strings).

In Subsection 3.4.2 we have shown that the geometric generator (see Proposition 3.8), which has small bias and exponential stretch, can be distinguished by a stronger notion of a distinguisher, namely a block-automaton. In this section we show a construction with similar bias and stretch, and present a constant-space nonuniform machine that distinguishes it from truly random bits.

The construction is based on a generic transformation that amounts to reordering the output bits of a generator, such that, under a suitable assumption (regarding the irreducible polynomial used to represent the field $GF(2^n)$), allows to verify simple $GF(2^n)$-algebraic dependencies of the output elements of the original generator, in constant space. Clearly, such a reordering preserves bias

and stretch. Note that performing standard arithmetic operations in $GF(2^n)$, such as multiplication, requires $\Omega(n)$ space, so in general even simple algebraic relations cannot be verified in small space.

The aforementioned transformation can be used on any $GF(2^n)$-based generator whose output blocks satisfy such algebraic relations, and in specific, as we will show, on both the geometric generator (from Section 3.4) and the Nisan generator ([Nis], described in Section 3.3). We also note that since the transformation is based on reordering the output bits, its application to the Nisan generator shows that its resilience to nonuniform space-bounded distinguishers depends on the output-bits order.

3.5.1 Bilinear dependencies

Small-bias generators can have no linear dependencies that are satisfied by all (or most) of their outputs. This applies both over $GF(2)$, and over $GF(2^n)$ by Corollary 4.6. However, dependencies of degree 2, and specifically bilinear dependencies, often arise from the construction. To obtain a separation between small-bias and fooling small-space machines, we will develop a method to be able to verify (some) $GF(2)$-bilinear dependencies in small space. To achieve this, we will reorder the bits representing $GF(2^n)$-elements that participate in a $GF(2^n)$-bilinear dependency, which implies (by Lemma 4.4) a $GF(2)$-bilinear dependency, so that we can verify this dependency in small (in fact, constant) space, under a reasonable assumption regarding the representation of $GF(2^n)$.

Decomposing a bilinear dependency. Assume that we want to verify a $GF(2)$-bilinear dependency of two n-bit output blocks a and b. That is, for some bit-matrix T, we want calculate $\langle a, Tb\rangle$. Expanding the expression as $\langle a, Tb\rangle = \sum_{i,j} T_{ij} a_i b_j$, we can group the n^2 terms to at most n sums of disjoint paris $a_i b_j$. That is, $\langle a, Tb\rangle$ can be written as a sum of at most n "simple inner products of a and b", where a simple inner product of a and b is some $GF(2)$-bilinear combination in the bits of the vectors a and b, in which each bit a_i or b_i only appears once. For example, $\langle a, b\rangle$, or $\langle a, Pb\rangle$ for any permutation matrix P, are simple inner products.

Being allowed to only read the bits of a and b once, from left to right, it is easy to see that we cannot even compute one simple inner product without remembering all the bits of a that participate in this inner product.

Reordering the bits. Despite the impossibility of computing an inner product when going over the bits of a and then the bits of b, if we were to go over the bits in any order we choose, still reading every bit only once, we can compute one single simple inner product by going over the bits in the order they appear in the expression. As a motivating example, consider the simple inner product $\langle a, b\rangle$. While it cannot be computed when going over all the bits of a first, it can be computed in constant space if the bits are interleaved, as in $a_0 b_0 a_2 b_2 \dots a_{n-1} b_{n-1}$. Any other simple inner product can be reordered in a similar manner, to allow computation in small space.

Calculating more than one simple inner product (where in general we need to calculate the sum of n simple inner products) in one pass is trickier; however, for only two simple inner products we can still find an ordering of the bits that will allow us to calculate them both, in constant space. Given an inner product $\langle a, Tb\rangle$ that can be written as the sum of two simple inner products, we define a bipartite graph G, with n nodes in each side: the bits of a on one side and the bits of b on the other. We let G have an edge between a_i and b_j if the term $a_i \cdot b_j$ appears in the sum we want to calculate. This is, in fact, the undirected graph described by T as an adjacency matrix (assuming without loss of generality that T is upper-triangular). Since $\langle a, Tb\rangle$ can be written as the sum of only two simple inner products, each containing each variable at most once, the degree of each node in G is at most 2.

We want to calculate the sum, over all edges (a_i, b_j) in the edge set of G, of the multiplication $a_i \cdot b_j$. We partition the graph to connected components, and note that, since the graph has maximal degree 2, each component is either a path or a cycle. The sum of the edges on every path can be calculated by going over the path adding $a_i \cdot b_j$ in each step, whereas calculating the sum over a cycle requires, in addition, remembering the value of the first node (chosen arbitrarily from the cycle). We can thus calculate in small space the sum over each of the connected components of G (in some arbitrary order) and sum the resulting bits to get the final result. As an example, the sum $(a_1 b_3 + a_2 b_2 + a_3 b_1) + (a_1 b_2 + a_2 b_1)$ can be computed by going over the bits in the order $a_3, b_1, a_2, b_2, a_1, b_3$.

Thus, any inner product that can be written as the sum of two simple inner products, which are $GF(2)$-bilinear expressions in which each bit participates in at most one term, has some reordering of the bits that allows to calculate it in constant space.

3.5.2 The first bit of $M_a b$

We will now show how the $GF(2)$-bilinear dependencies that are implied by $GF(2^n)$-bilinear dependencies (which are common in $GF(2^n)$-polynomial generators that are only designed to resist linear tests), can have the above desired property, that they can be written as the sum of only two simple inner products (and can thus be verified in small space by a reordering of the bits, as described above).

We will consider the basic task of calculating (some bit of the representation of) a multiplication of two $GF(2^n)$-elements. Suppose that, for two output blocks $a, b \in \{0,1\}^n$, we wish to calculate the first bit of the vector $M_a b$ representing the $GF(2^n)$-element $\tilde{a} \cdot \tilde{b}$ (see Notation 2.7), in small space (say, $o(n)$). If we just write the expression for $\langle e_0, M_a b \rangle = \langle a, Q_{e_0} b \rangle$ (where Q_{e_0} is defined in Lemma 2.11), we get a sum of (potentially) n simple inner products between the bits of a and b.

We thus take a closer look on the multiplication-by-$\tilde{a}$ matrix M_a, and show how, under an assumption regarding the irreducible polynomial used to represent the field $GF(2^n)$, this inner product $\langle a, Q_{e_0} b \rangle$ can be written as the sum of only two simple inner products (thus allowing calculation in constant space), and a few more terms that can also be calculated in small space.

As described in Subsection 2.2, the matrix M_a can be written as $R \cdot L_a$, where L_a (defined in Notation 2.8) multiplies (representations of) polynomials in $GF(2)[x]$ by the polynomial p_a, and R (Notation 2.9) reduces such representations modulo the polynomial $c(x)$, which is the irreducible polynomial fixed in Notation 2.5. Explicitly, $[L_a]_{ij} = a_{i-j}$ if $j \leq i < j + n$ and 0 otherwise, so the i-th entry of $L_a \cdot b$ is

$$\sum_{j=0}^{n-1} [L_a]_{ij} b_j = \sum_{j=\max(0,i-n+1)}^{\min(n-1,i)} a_{i-j} b_j,$$

which is a simple inner product. To have the final expression for $M_a b$ written as the sum of only two simple inner products, we want to only sum two entries of $L_a \cdot b$ when calculating the first bit of $R \cdot L_a \cdot b$. Equivalently, we want the first row of R to only have two bits set.

Furthermore, observe that for $i < O(1)$ or $i > 2n - O(1)$, the i-th bit of $L_a \cdot b$ only depends on a constant number of bits of a and b; those bits can be

stored during a pass on the bits to allow calculation of this i-th bit.[9] Hence, to be able to calculate the first bit of $M_a b = RL_a b$, it suffices that the first row of R have at most two bits set that at a superconstant distance from either of the edges 1 and $2n-1$. Then, we will reorder the bits so these two simple inner products (resulting from the two middle-bits that are set) can be calculated in one pass, in which we will also store the $O(1)$ bits required to calculate the $O(1)$ inner products that depend on $O(1)$ bits, and finally sum everything to get our wanted result.

Having defined our goal - to have the first row of R only contain two bits set that are more than a constant away from either of the matrix edges - we will now present a (reasonable) assumption regarding the irreducible polynomial $c(x)$ used to represent the field $GF(2^n)$, that will guarantee this. This will follow from an application of the following Lemma:

Lemma 3.15. *Let $c(x)$ be the irreducible polynomial used to represent $GF(2^n)$, and let m and k be the second and third highest degrees of $c(x)$. That is, for $k < m < n$, write $c(x) = x^n + x^m + x^k + p(x)$ for some polynomial $p(x)$ of degree smaller than k. Then for exactly two values of $1 \leq j < \min\{2n-k, 3n-2m\}$, the j-th bit of the first row of R is set.*

Proof. The j-th column of R is defined as the representation of the polynomial x^j reduced modulo $c(x)$ (see Notation 2.9). The entry at the first row corresponds to the coefficient of the free term of that polynomial. We will go over the $2n-1$ columns, considering which bits can be set. We will partition the index range $\{1, \ldots, \min\{2n-k, 3n-2m\}\}$ to parts, showing that only the n-th and $2n-m$-th bits are set.

For $0 < j < n$, the first bit of the j-th column of R is never set, since the polynomial $x^j = x^j \bmod c(x)$ has no free coefficient. The n-th column is exactly the representation of $c(x)$ without the coefficient of x^n, which always has the first bit set (since $c(x)$ is irreducible and thus not divisible by x). By the hypothesis of the lemma, this polynomial $c(x) - x^n$ is $x^m + x^k + p(x)$, which has degree m. Since this is the reduction of x^n modulo $c(x)$, all polynomials $x^{n+t} \bmod c(x)$ for $1 \leq t < n-m$ can be written as $x^n \cdot x^t \bmod c(x) = (x^m + x^k + p(x))x^t \bmod c(x)$. Since $(x^m + x^k + p(x))x^t$ has degree $m + t < n$, this polynomial is unaffected by the reduction modulo $c(x)$. Since this polynomial is divisible by x^t (with

[9] In fact, even $i < o(n)$ or $i > 2n - o(n)$ would do, if we are interested in an $o(n)$-space distinguisher.

$t > 0$), its free coefficient is zero. Thus, for $n < j < 2n - m$, the first bit in the representation of x^j, which is the j-th entry of the first row of R, is not set. When $j = 2n - m$ we get

$$\begin{aligned} x^j &= x^n \cdot x^{n-m} \\ &\equiv x^n + x^{k+n-m} + p(x) \cdot x^{n-m} \pmod{c(x)} \\ &\equiv x^m + x^{k+n-m} + x^k + p(x)(x^{n-m} + 1) \pmod{c(x)}, \end{aligned}$$

which has degree at most $d = \max\{m, k + n - m\}$. This polynomial necessarily has the free coefficient set. However, for $1 \leq t < n - d$, the polynomial $x^{2n-m+t} \bmod c(x)$ can again be written as the above expression times x^t, which is a polynomial of degree smaller than n and thus needs not be reduced modulo $c(x)$. This means that the free coefficient in of the reduction of x^j modulo $c(x)$ is 0 for $2n - m < j < 2n - m + n - d = \min\{3n - 2m, 2n - k\}$.

In summary, we got that for $1 < j < \min\{3n - 2m, 2n - k\}$, the first bit of the j-th column of R is set only for $j = n$ and $j = 2n - m$. □

Lemma 3.15 gives us that if $c(x) = x^n + x^m + x^k + p(x)$ as above, for $k = O(1)$ and $m < \frac{n}{2} + O(1)$, then there are only two bits set in the first row of R that are more than $O(1)$-far from either edge of the matrix (indices 0 and $2n - 1$). Specifically, all trinomials $c(x) = x^n + x^m + 1$ for $m < n/2 + O(1)$ satisfy.[10] We note that infinitely many such irreducible trinomials are known to exist (see, e.g., Theorem 1.1.28 of [Lin]), so this assumption is indeed reasonable. We have thus obtained:

Theorem 3.16. *For any irreducible trinomial $c(x) = x^n + x^m + 1$ with $m < n/2 + O(1)$ used to represent $GF(2^n)$, the expression $\langle e_0, M_a b \rangle$, for $a, b \in \{0,1\}^n$, can be written as the sum f two simple inner products in the bits of a and b, plus $O(1)$ terms that depend on $O(1)$ bits of a and b.*

3.5.3 Bilinear dependencies in known generators

Theorem 3.16 implies that if some generator has three output blocks a, b, c that always satisfy $\tilde{a} \cdot \tilde{b} = \tilde{c}$, then a generator resulting from some permutation of its output bits can be distinguished from random with gap 1/2 by calculating

[10] In fact, since the reciprocal of an irreducible trinomial $x^n + x^m + 1$, namely $x^n + x^{n-m} + 1$, is known to also be irreducible, we can assume without loss of generality that any irreducible trinomial has $m \leq \frac{n}{2}$. For details, see any standard algebra textbook, e.g. [BM].

the first bit of $M_a b$, and comparing to the first bit of c. The same methods can be used more generally, if a, b and c are not written directly in the output of the generator, but can be deduced in small space from its output, such that the sets of output bits required to compute each bit a_j, b_j or c_0 are all disjoint. We demonstrate this on the two constructions discussed in this chapter.

The geometric generator. Let g_0, g_1, g_2 denote the first three output blocks of the geometric generator (in fact, we can take any three consecutive blocks). By definition, it must hold that $\tilde{g}_0 \cdot \tilde{g}_2 = \tilde{g}_1{}^2$ (over $GF(2^n)$). Over $\{0,1\}^n$, this can be written as $M_{g_0} g_2 = S g_1$, where S is the squaring matrix defined in Notation 2.13. If the representation of $GF(2^n)$ satisfies the hypothesis of Theorem 3.16, then by reordering the bits of g_0 and g_2 as in Subsection 3.5.2, we can calculate the first bit of $M_{g_0} g_2$. Separately, we would just go over the bits of g_1 and calculate the linear combination $\langle e_0, S g_1 \rangle$, finally comparing the two (which on random input, are equal only with probability $\frac{1}{2}$).

Since reordering the output bits preserves the bias (and stretch) of the generator, we have obtained a generator with the same parameters as the geometric generator, which means exponentially small bias and exponential stretch, that is (nonuniformly) distinguishable from random in constant space.

We have thus established:

Theorem 3.17. *For infinitely many $n \in \mathbb{N}$, for every $\ell \in \mathbb{N}$, there exists an $\frac{\ell}{2^n}$-bias generator $G : \{0,1\}^{2n} \to \{0,1\}^{\ell \cdot n}$ that can be distinguished from random with constant gap by a nonuniform machine that uses $O(1)$ space.*

The Nisan generator. Even the Nisan generator itself, fooling all nonuniform distinguishers that use $o(n)$ space, has some reordering of its output bits that makes it $O(1)$-space distinguishable from random. This shows that pseudorandomness against space-bounded distinguishers is sensitive to the ordering of the bits (which indeed seems intuitive, as demonstrated previously by the simple example of calculating the inner product $\langle a, b \rangle$).

To show how to apply the above methods to the Nisan generator, we recall from Section 3.3 that in every iteration of the construction, we introduce a new pairwise-independent hash function h, and replace every two consecutive blocks x, y by $x, h(x)$. For our distinguisher, we will require at least eight output blocks. First, assume that this single iteration is applied to four output

blocks (two pairs of blocks), so the final output of the generator has the four blocks $(y_0, y_1, y_2, y_3) = (x_0, h(x_0), x_1, h(x_1))$. Assume as usual that we are using the standard construction of a universal collection of hash functions, namely the collection of all $GF(2^n)$-affine functions. Then for some $\tilde{a}, \tilde{b} \in GF(2^n)$, we have that $(\tilde{y}_0, \tilde{y}_1, \tilde{y}_2, \tilde{y}_3) = (\tilde{x}_0, \tilde{a} \cdot \tilde{x}_0 + \tilde{b}, \tilde{x}_1, \tilde{a} \cdot \tilde{x}_1 + \tilde{b})$. This gives that $\tilde{y}_1 + \tilde{y}_3 = \tilde{a} \cdot (\tilde{y}_0 + \tilde{y}_2)$. Now assume the same function h is applied to the next four output blocks, $(y_4, y_5, y_6, y_7) = (x_2, h(x_2), x_3, h(x_3))$. Similarly to above, $\tilde{y}_5 + \tilde{y}_7 = \tilde{a} \cdot (\tilde{y}_4 + \tilde{y}_6)$. Finally, this gives the following $GF(2^n)$-bilinear dependency in the output blocks $(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7)$:

$$(\tilde{y}_1 + \tilde{y}_3) \cdot (\tilde{y}_4 + \tilde{y}_6) = (\tilde{y}_0 + \tilde{y}_2) \cdot (\tilde{y}_5 + \tilde{y}_7).$$

If we interleave the bits of y_1 and y_3, it is easy to calculate each bit of $y_1 + y_3$. Similarly, we can treat the above dependency as $\tilde{a} \cdot \tilde{b} = \tilde{c} \cdot \tilde{d}$ for four vectors a, b, c, d whose bits can be computed from disjoint sets of output bits. Thus, if the irreducible polynomial $c(x)$ used to represent $GF(2^n)$ satisfies the hypothesis of Theorem 3.16, we can calculate (in constant space) the first bit of $M_a b$ and the first bit of $M_c d$ separately, for some ordering of the output bits (as explained in Subsection 3.5.2). Finally, we compare the resulting bits to check if they are equal. The first bit of the expression $(\tilde{y}_1 + \tilde{y}_3) \cdot (\tilde{y}_4 + \tilde{y}_6) - (\tilde{y}_0 + \tilde{y}_2) \cdot (\tilde{y}_5 + \tilde{y}_7)$, while always zero for an output of the Nisan generator, has probability at most $\frac{1}{2} + \exp(-\Omega(n))$ to be zero when evaluated on uniformly chosen $(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7)$, giving a distinguishing gap of $\frac{1}{2} - neg(n)$.

We have obtained:

Corollary 3.18. *For infinitely many $n \in \mathbb{N}$, and for every $\ell \geq 8$, there exists a permutation π of the index set $\{1, \ldots, n\ell\}$ such that the generator resulting from applying π to an output of the Nisan generator with block-length n and block-stretch ℓ, using the standard implementation of a collection of hash functions as the collection of $GF(2^n)$-affine functions, is distinguishable from random by a nonuniform machine that uses constant space.*

Chapter 4

$GF(2^n)$-Polynomials vs. $GF(2)$-Polynomials: as Distinguishers

4.1 Introduction

We consider polynomials over $GF(2^n)$ of bounded degree as distinguishers for pseudorandom distributions, and compare their distinguishing capabilities to those of polynomials over $GF(2)$ of the same degree.

4.1.1 Motivation

In addition to the inherent interest in comparing different classes of distinguishers, we mention that such a comparison is intructive towards the following methodology: when aiming at various types of pseudorandomness against bit-based tests, it is often useful to first prove pseudorandomness against a corresponding class of $GF(2^n)$-based tests, and then use a reduction of distinguishing capabilities to obtain the desired pseudorandomness guarantees. As an example, Section 4.3 shows that fooling bit-linear tests reduces to fooling $GF(2^n)$-linear tests. Indeed, this reduction is used in Section 3.4 to show that the geometric generator has small bias.

The same idea is used in the other direction in Section 4.5. There, a con-

struction by Viola, fooling $GF(2)$-polynomials of degree at most d by summing d small-bias generators, is shown to be tight with respect to the number of small-bias generators. Indeed, for the sum of d copies of a small-bias generator of our choice, we obtain a $GF(2)$-polynomial distinguisher of degree $d+1$. Interestingly, this is done by presenting a $GF(2^n)$-polynomial distinguisher of degree $d+1$, and then observing its behaviour over the bits representing the $GF(2^n)$-elements.

4.1.2 Overview

We begin by introducing in Section 4.2 the various classes of distinguishers that we consider and compare: unlike in the case of $GF(2)$-polynomials, when using polynomials over $GF(2^n)$ to identify the distribution from which a sequence was chosen, we need a way to convert the resulting field element to a binary verdict. We consider three such ways: allowing arbitrary functions, allowing a specific function, and allowing all linear functions (in the bits representing the final $GF(2^n)$-element).

In Section 4.3 we consider the case of degree 1, that is, linear tests. Comparing $GF(2)$-linear tests to $GF(2^n)$-linear tests, we present constructively a generic transformation of the former to the latter. Moreover, in the resulting $GF(2^n)$-linear test, the final verdict function is linear in the bits representing the resulting $GF(2^n)$-element, thus showing that $GF(2)$-linear tests are, in fact, equivalent to a strict subclass of the $GF(2^n)$-linear tests (in the most generic notion).

Higher degrees are discussed in Section 4.4, where separating the distinguishing powers of classes is reduced to separating the approximating power of the classes of functions, over the uniform distribution. A corollary is then derived from the results of Chapter 5, giving that unlike the case of linear distinguishers, the class of $GF(2^n)$-bilinear distinguishers is not as powerful as the class of $GF(2)$-bilinear distinguishers.

Finally, in Section 4.5 we consider a construction of Viola that fools degree-d polynomials (over bits) by summing d copies of a small-bias distinguisher. We present an explicit polynomial of degree $d+1$ that distinguishes the sum of d copies of a small-bias generator of our choice. This establishes that Viola's result is tight with respect to the number of copies of a small-bias generator needed to fool polynomials of degree d.

4.2 The general picture

We consider polynomials over $GF(2^n)$ as distinguishers for pseudorandom distributions, and compare their distinguishing capabilities to that of polynomials over $GF(2)$. When applying a $GF(2^n)$-polynomial to a sequence of bits, we divide the bits to blocks of length n, treat each block as representing an element of $GF(2^n)$ in some fixed representation scheme, and evaluate the polynomial. The output of the polynomial is an element of $GF(2^n)$; we finally need some function $f : GF(2^n) \to \{0,1\}$ to decide whether we accept or reject it (i.e., whether we think this sequence of bits was selected from the pseudorandom distribution or uniformly). The function f can be alternatively viewed as a set $B = f^{-1}(1)$.

When trying to fool $GF(2^n)$-polynomials, we also need to specify which final functions f we allow. Letting f be arbitrary means that any $GF(2^n)$ polynomial should be distributed similarly on the pseudorandom distribution and on the uniform distribution (i.e., the two distributions are statistically close). However, limiting f to be, say, a linear function (in the bits representing the $GF(2^n)$ element), or even specifying a specific set B in advance, restricts the distinguishers and potentially eases the work of fooling them.

When studying the relation between $GF(2^n)$-polynomials (with arbitrary final function f) and $GF(2)$-polynomials, we will also consider the above restricted classes of distinguishers.

4.2.1 Formal definitions

We define the above informal notions of the various classes of polynomial tests. These definitions are motivated (for the linear case $d = 1$) in Subsection 4.3.1. We define the notions of fooling for $GF(2^n)$-generators, but will abuse the terms and refer to the binary version as also fooling $GF(2^n)$ polynomials.

Definition 4.1 (ε-fooling $GF(2^n)$-polynomials)**.** *For $\varepsilon > 0$ and $n, k, \ell, d \in \mathbb{N}$, a generator $\tilde{G} : GF(2^n)^k \to GF(2^n)^{\ell+1}$ is said to ε-*fool $GF(2^n)$-polynomials of degree *d if for every polynomial $\tilde{p} \in GF(2^n)[x_0, \ldots, x_\ell]$ of degree at most d, the distributions $\tilde{p}(\tilde{G}(\tilde{U}_{k \cdot n}))$ and $\tilde{p}(\tilde{U}_{(\ell+1) \cdot n})$ are ε-close in statistical distance, where $\tilde{U}_{m \cdot n}$ is the uniform distribution over $GF(2^n)^m$. That is, for every set $B \subseteq GF(2^n)$, it holds that*

$$\left| \Pr_{\tilde{s} \in GF(2^n)^k} \left[\tilde{p}(\tilde{G}(s)) \in B \right] - \Pr_{\tilde{x} \in GF(2^n)^{\ell+1}} \left[\tilde{p}(\tilde{x}) \in B \right] \right| \leq \varepsilon. \tag{4.1}$$

A weaker definition that only considers a specific set B is:

Definition 4.2 ((ε, B)-fooling $GF(2^n)$-polynomials)**.** *For $\varepsilon > 0$, $n, k, \ell, d \in \mathbb{N}$ and $B \subseteq GF(2^n)$, a generator $\tilde{G} : GF(2^n)^k \to GF(2^n)^{\ell+1}$ is said to* (ε, B)-fool $GF(2^n)$-polynomials of degree d *if for every $GF(2^n)$-polynomial $\tilde{p} \in GF(2^n)[x_0, \ldots, x_\ell]$ of degree at most d, Equation (4.1) holds.*

Indeed, a generator ε-fools $GF(2^n)$-polynomials of degree d if and only if, for any $B \subseteq GF(2^n)$, the generator (ε, B)-fools $GF(2^n)$-polynomials of degree d. If we consider only all sets B that are linear subspaces of co-dimension 1, i.e. sets of the form $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\}$ for some nonzero vector $\gamma \in \{0,1\}^n$, we actually require the n bits representing the resulting field element to fool $GF(2)$-linear tests. This case is referred to as *ε-linear-fooling*:

Definition 4.3 (ε-linear-fooling $GF(2^n)$-polynomials)**.** *For $\varepsilon > 0$, $n, k, \ell, d \in \mathbb{N}$, a generator $\tilde{G} : GF(2^n)^k \to GF(2^n)^{\ell+1}$ is said to* ε-linear-fool $GF(2^n)$-polynomials of degree d *if for every nonzero vector $\gamma \in \{0,1\}^n$, it holds that G does (ε, Γ)-fool $GF(2^n)$-polynomials of degree d, where $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\} \subseteq GF(2^n)$.*

Remark. Definition 4.3 means, in specific, that for polynomials $\tilde{p}$ whose output is distributed uniformly on uniform input (which is always the case, for example, if $d = 1$ and $\tilde{p}$ is nonconstant), the (binary version) distribution $p(G(s))$, for uniformly chosen seed $s \in \{0,1\}^{kn}$, is ε-biased. For $d = 1$, Definition 4.3 is the formal generalization of ε-bias to $GF(2^n)$ (see [Eve]).

Clearly, for $n = 1$ the above three definitions (with the only nontrivial $B = \{1\}$ in Definition 4.2) coincide with the notion of ε-bias (Definition 2.2).

4.2.2 Relations between the different notions of $GF(2^n)$-polynomial tests, and their relation to $GF(2)$-polynomials

By definition, it is easier to ε-linear fool $GF(2^n)$-polynomials of degree d than to ε-fool $GF(2^n)$-polynomials of degree d. This implication is clearly strict, for example, for the case $\ell = 1$ (that is, considering only one block, thus reducing the set of tests to its set of allowed final verdict functions).

Also, the linearity of the representation of $GF(2^n)$ (as in Lemma 2.11) gives that ε-linear fooling $GF(2^n)$-polynomials of degree d is also easier than ε-fooling $GF(2)$-polynomials of degree d. This is captured by the following Lemma:

Lemma 4.4. *Fix an ℓ-variate polynomial $\tilde{p} : GF(2^n)^\ell \to GF(2^n)$ of degree d, (letting its binary version $p : \{0,1\}^{\ell \cdot n} \to \{0,1\}^n$ treat its input as the representation of ℓ elements $\tilde{x}_1, \ldots, \tilde{x}_\ell \in GF(2^n)$, and output the vector representing $\tilde{p}(\tilde{x}_1, \ldots, \tilde{x}_\ell)$). Then, each of the n output bits of p is a polynomial of degree at most d in its $\ell \cdot n$ input bits.*

Proof. We will show the claim for a polynomial consisting of a single monomial; the general claim follows from the fact that addition in the field $GF(2^n)$ is exactly bitwise addition in the vector space $\{0,1\}^n$. We proceed by induction on the degree d. For $d = 0$ the claim is immediate since $\tilde{p}$ is constant. Now fix $d > 0$, and assume without loss of generality that $\tilde{p}(\tilde{x}_1, \ldots, \tilde{x}_\ell) = \tilde{x}_1 \cdot \ldots \cdot \tilde{x}_d$ (recall that over bits, each variable cannot have degree higher than 1). By Lemma 2.11, the representation of $\tilde{p}(\tilde{x}_1, \ldots, \tilde{x}_\ell)$ is a bilinear expression in the bits of the two vectors x_1 and y_1, where $\tilde{y}_1 = \tilde{x}_2 \cdot \ldots \cdot \tilde{x}_d$. By the induction hypothesis, every bit of y_1 is a polynomial of degree at most $d-1$ in the bits of $x_2, \ldots x_d$, so each bit of a bilinear form in $\tilde{x}_1$ and $\tilde{y}_1$ is a polynomial of degree at most d in the bits of $x_1, \ldots, x_d$. □

In this chapter we investigate when ε-linear-fooling $GF(2^n)$-polynomials is strictly weaker than ε-fooling $GF(2)$-polynomials, and also study the relation between the (generally incomparable) notions of ε-fooling $GF(2^n)$-polynomials and ε-fooling $GF(2)$-polynomials.

4.3 Degree 1: linear tests

We refer to *$GF(2^n)$-polynomials of degree 1* as *$GF(2^n)$-linear tests*, and to *ε-fooling $GF(2)$-linear tests* as being *ε-biased*.

In this section we will show that for $d = 1$, the converse of Lemma 4.4 holds as well:

Theorem 4.5. *For $\varepsilon > 0$, $n \in \mathbb{N}$, and for any nonzero vector $\gamma \in \{0,1\}^n$, every G that (ε, Γ)-fools $GF(2^n)$-linear tests, where $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\}$, is an ε-bias generator.*

This gives that for a distribution over ℓ bits, being ε-biased and ε-linear-fooling $GF(2^n)$-linear tests is equivalent, for every n that divides ℓ.

Since ε-fooling $GF(2^n)$-linear tests implies ε-linear-fooling $GF(2^n)$-linear tests, Theorem 4.5 also yields:

Corollary 4.6. *For $\varepsilon > 0$, $n \in \mathbb{N}$, a generator G that ε-fools $GF(2^n)$-linear tests is also an ε-bias generator.*

We note that the converse of Corollary 4.6 does not hold, but it is known that being ε-biased implies $(2^{n/2} \cdot \varepsilon)$-fooling $GF(2^n)$-linear tests (see, e.g., [Gol95]).

4.3.1 Motivating the definitions

We use the linear case $d = 1$ to demonstrate the motivation to our interest in our definitions (presented in Subsection 4.2.1) and the relations between them.

One concrete motivation to Definitions 4.1 and 4.3, for $d = 1$, is their role in the following two-step methodology for constructing natural small-bias generators based on $GF(2^n)$-sequences: first show that a generator fools $GF(2^n)$-linear tests (resp., linear-fools $GF(2^n)$-linear tests), and next use Corollary 4.6 (resp., Theorem 4.5) to conclude that it has small bias. This is demonstrated in Subsection 3.4.1, where the geometric generator is shown to have small bias.

As further motivation for our definitions, we note that [ASS] uses the construction of the geometric generator for obtaining a graph with normalized second eigenvalue $\frac{\ell}{2^n}$. Their argument implicitly shows that any generator that ε-linear-fools $GF(2^n)$-linear tests[1] yields a Cayley graph with normalized second eigenvalue of 2ε (The case of $n = 1$ was previously shown in [AR]). Indeed, in [ASS] this is done directly (and not by using a reduction similar to our Theorem 4.5). However, Theorem 4.5 can be (non-constructively) derived by combining the above claim (i.e., ε-fooling $GF(2^n)$-linear tests implies (normalized) second eigenvalue 2ε) with its converse for the case of $n = 1$. We mention that the converse for $n = 1$ was known before, and can be derived for any n by reversing the argument of [ASS].

[1] Or even, in fact, (ε, Γ)-fools $GF(2^n)$-linear tests for any nontrivial $\Gamma \subseteq GF(2^n)$ which is a linear subspace over $\{0,1\}$ of co-dimension 1 (i.e., $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\}$ for some nonzero $\gamma \in \{0,1\}^n$).

4.3.2 Proof of Theorem 4.5

Theorem 4.5 says that for every $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\}$, where $\gamma \in \{0,1\}^n$ is nonzero, (ε, Γ)-fooling $GF(2^n)$-linear tests is as strong as ε-fooling $GF(2)$-linear tests. This will be shown by transforming every $GF(2)$-linear test to a $GF(2^n)$-linear test, followed by the final linear function $\langle \gamma, \cdot \rangle$.

Proof of Theorem 4.5. Fix a nonzero $\gamma \in \{0,1\}^n$, and let $G : \{0,1\}^{k \cdot n} \to \{0,1\}^{(\ell+1) \cdot n}$ be a generator that (ε, Γ)-fools $GF(2^n)$-linear tests, where $\Gamma = \{\tilde{a} : \langle \gamma, a \rangle = 0\}$. Fix an arbitrary linear combination $\bar{\alpha} \in \{0,1\}^{(\ell+1)n}$ of the bits of G, and parse it to $\ell + 1$ vectors $\alpha_0, ..., \alpha_\ell \in \{0,1\}^n$. Define a series of $GF(2^n)$ elements $\tilde{b}_0, ..., \tilde{b}_\ell \in GF(2^n)$, represented by the vectors $b_i = Q_\gamma^{-1} \alpha_i$ for $i = 0, ..., \ell$, where Q_γ is the matrix guaranteed by Lemma 2.11. We get that for any output of the generator G, denoted $(g_0, ..., g_\ell) \in \{0,1\}^{(\ell+1)n}$:

$$\sum_{i=0}^{\ell} \langle \alpha_i, g_i \rangle \underset{\text{Def. of } b_i}{\overset{\uparrow}{=}} \sum_{i=0}^{\ell} \langle Q_\gamma b_i, g_i \rangle \underset{\text{Lemma 2.11}}{\overset{\uparrow}{=}} \sum_{i=0}^{\ell} \langle \gamma, M_{b_i} g_i \rangle = \left\langle \gamma, \sum_{i=0}^{\ell} M_{b_i} g_i \right\rangle .$$

Recalling the definitions of Γ and M_{b_i} (Notation 2.7), we get that

$$\begin{aligned} \Pr_s \left[\langle \bar{\alpha}, G(s) \rangle = 0 \right] &= \Pr_s \left[\sum_{i=0}^{\ell} \langle \alpha_i, g_i(s) \rangle = 0 \right] \\ &= \Pr_s \left[\left\langle \gamma, \sum_{i=0}^{\ell} M_{b_i} g_i(s) \right\rangle = 0 \right] \\ &= \Pr_s \left[\sum_{i=0}^{\ell} \tilde{b}_i \cdot \tilde{g}_i(s) \in \Gamma \right] . \end{aligned} \tag{4.2}$$

Assuming $\bar{\alpha} \neq 0^{(\ell+1) \cdot n}$, there exists an i such that $\alpha_i \neq 0^n$ and so $\tilde{b}_i$, represented by the vector $Q_\gamma^{-1} \alpha_i$, is nonzero. The right hand side of Equation (4.2) is bounded by ε since G (ε, Γ)-fools $GF(2^n)$-linear tests, giving the same bound on the left hand side. This completes the proof. □

An interesting corollary to Lemma 2.11 is that any linear combination in the bits of any vector g can be computed as a prefixed linear combination in the bits of the representation of $\tilde{b} \cdot \tilde{g}$ for a suitable choice of $\tilde{b} \in GF(2^n)$:

Corollary 4.7. *For every nonzero $\alpha, \gamma \in \{0,1\}^n$ there exists $b \in \{0,1\}^n$ such that for every $g \in \{0,1\}^n$, it holds that $\langle \alpha, g \rangle = \langle \gamma, M_b \cdot g \rangle$.*

Proof. Set $b = Q_\gamma^{-1}\alpha$. By Lemma 2.11, $\langle \gamma, M_b \cdot g \rangle = \langle Q_\gamma \cdot b, g \rangle = \langle \alpha, g \rangle$. □

Corollary 4.7 can be seen as an interpretation for the proof of Theorem 4.5: every inner product $\langle \alpha_i, g_i \rangle$ is calculated as $\langle \gamma, M_{b_i} \cdot g_i \rangle$ for the adequate b_i that depends on α_i and γ; linearity of the inner products is then used to give $\sum_{i=0}^{\ell} \langle \alpha_i, g_i \rangle = \left\langle \gamma, \sum_{i=0}^{\ell} M_{b_i} \cdot g_i \right\rangle$.

4.3.3 A constructive proof of Lemma 2.11

In the proof of Theorem 4.5, we used Lemma 2.11 to transform a $GF(2)$-linear test to a $GF(2^n)$-linear test. This gives an existential result, while often we are interested in knowing exactly how the resulting $GF(2^n)$-linear test looks. In this subsection we will analyze the representation of $GF(2^n)$ more deeply, and show exactly how the matrices Q_γ are obtained. This will give a fully constructive proof of Theorem 4.5.

Mathematical preliminaries

We begin with some mathematical facts and notations that we need.

As noted earlier, we use vectors and matrices over $\{0,1\}$, and use a tilde when we want to refer to the $GF(2^n)$-elements represented. However, in this subsection we sometimes use the larger vector space $GF(2^n)^n$, and work with matrices and vectors over $GF(2^n)$. In such cases, we will note this explicitly.

Fact 4.8. *The eigenvalues of the matrix C defined in Notation 2.6 (over $GF(2^n)$) are exactly the roots of the polynomial $c(x)$ of notation 2.5. Moreover, if $c(x)$ has n* distinct *roots $\lambda_0, ..., \lambda_{n-1} \in GF(2^n)$, it is diagonalizable as $C = V^{-1} \cdot \mathrm{diag}(\lambda_0, ..., \lambda_{n-1}) \cdot V$, with $\mathrm{diag}(\lambda_0, ..., \lambda_{n-1})$ denoting the diagonal matrix with λ_i in the (i,i)-th entry, and V being the Vandermonde matrix defined as $[V]_{ij} = \lambda_i^j$, for $i, j = 0, ..., n-1$.*

(Note: the entries of V defined above are in $GF(2^n)$. Although the matrices in the expression for C have entries in $GF(2^n)$, the matrix C is over $GF(2)$.)

Fact 4.8 is a direct corollary of the transposed version of Theorem 6.13 in [Dym], applying the same arguments to $GF(2^n)$ rather than to the complex field $\mathbb{C}$.

Fact 4.9. *Every irreducible polynomial over a finite field has no multiplied roots.*

Fact 4.9 appears as a note in Section XV.6 of [BM], at the end of page 413.

We get a convenient diagonalization of the multiplication matrix M_a, defined in Notation 2.7:

Corollary 4.10. *For any $\tilde{a} \in GF(2^n)$, it holds that*

$$M_a = V^{-1} \cdot \mathrm{diag}(p_a(\lambda_0), ..., p_a(\lambda_{n-1})) \cdot V.$$

(Note: although the matrices in the expression have entries in $GF(2^n)$, the matrix M_a is over $GF(2)$.)

Proof. By Fact 4.9, $c(x)$ has distinct roots. We thus write C using Fact 4.8 as $V^{-1} \cdot \mathrm{diag}(\lambda_0, ..., \lambda_{n-1}) \cdot V$. Observe that $C^i = \left(V^{-1} \cdot \mathrm{diag}(\lambda_0, ..., \lambda_{n-1}) \cdot V\right)^i = V^{-1} \cdot \mathrm{diag}(\lambda_0^i, ..., \lambda_{n-1}^i) \cdot V$, and for $\tilde{a} \in GF(2^n)$ we have, recalling the Notations 2.7 and 2.5:

$$\begin{aligned}
M_a &= p_a(C) \\
&= \sum_{i=0}^{n-1} a_i C^i \\
&= \sum_{i=0}^{n-1} a_i \cdot V^{-1} \cdot \mathrm{diag}(\lambda_0^i, ..., \lambda_{n-1}^i) \cdot V \\
&= V^{-1} \left(\sum_{i=0}^{n-1} a_i \cdot \mathrm{diag}(\lambda_0^i, ..., \lambda_{n-1}^i) \right) V \\
&= V^{-1} \cdot \mathrm{diag}(p_a(\lambda_0), ..., p_a(\lambda_{n-1})) \cdot V.
\end{aligned}$$

□

Fact 4.11. *Every symmetrical multinomial $p(x_1, ..., x_n)$ over a field $\mathbb{F}$, evaluated on the roots $\lambda_0, ..., \lambda_{n-1}$ of any polynomial $q(x)$ over $\mathbb{F}$ (the roots possibly in a larger algebraic extension of $\mathbb{F}$), takes value in $\mathbb{F}$.*

Fact 4.11 is Theorem 10 in Section XV.4 of [BM].

Using Fact 4.11, it follows that while the entries of V are in $GF(2^n)$, the entries of $V^T V$ (and its inverse) are in $GF(2)$:[2]

[2]Actually, Corollary 4.12 does not even require that the polynomial $c(x)$ be irreducible: any Vandermonde matrix V of the roots of a degree n polynomial would have the entries of $V^T V$ in the base field.

Corollary 4.12. *The entries of $V^T V$ are all in $GF(2)$.*

Proof. The (i,j)-th entry of $V^T V$ is $\sum_{k=0}^{n-1} V_{ik}^T V_{kj} = \sum_{k=0}^{n-1} \lambda_k^{i+j}$. For any fixed i,j this is a symmetric polynomial over $GF(2)$, evaluated on the roots of $c(x)$. So by Fact 4.11, it takes value in the base field $GF(2)$. □

Define $U = V^{-1} \cdot {V^{-1}}^T$. By Corollary 4.12, the entries of U^{-1} and U are in $GF(2)$. These matrices allow to replace M_a^T by M_a as follows:

Claim 4.13. *For every $a \in \{0,1\}^n$, it holds that $M_a^T = U^{-1} M_a U$.*

Proof. Using the diagonalization from Corollary 4.10, we have $M_a = V^{-1} \operatorname{diag}(p_a(\lambda_1), ..., p_a(\lambda_n)) V$ and so

$$\begin{aligned} U^{-1} M_a U &= V^T V \cdot V^{-1} \operatorname{diag}(p_a(\lambda_0), ..., p_a(\lambda_{n-1})) V \cdot V^{-1} {V^{-1}}^T \\ &= V^T \operatorname{diag}(p_a(\lambda_0), ..., p_a(\lambda_{n-1}))^T {V^{-1}}^T \\ &= M_a^T. \end{aligned}$$

□

A constructive proof of Lemma 2.11

Given the above analysis of the multiplication-by-$\tilde{a}$ matrix M_a, we can write every Q_γ explicitly:

Proof of Lemma 2.11 (constructive). For U as defined above, given $\gamma \in \{0,1\}^n$, we set $Q_\gamma = U^{-1} M_{U\gamma}$. We get for every $u, v \in \{0,1\}^n$:

$$\begin{aligned} \langle \gamma, M_u v \rangle &= \langle M_u^T \gamma, v \rangle \\ \text{(by Claim 4.13)} \qquad &= \langle U^{-1} M_u U \gamma, v \rangle \\ \text{(by commutativity of } GF(2^n)) \qquad &= \langle U^{-1} M_{U\gamma} \cdot u, v \rangle \\ &= \langle Q_\gamma \cdot u, v \rangle . \end{aligned}$$

The moreover part follows from the invertibility of U and $M_{U\gamma}$ when $\gamma \neq 0$. □

4.4 Higher degrees: a reduction to approximators

Having fully understood the relations between the different classes of tests for $d = 1$ (Definition 4.3 coincides with ε-bias, which is strictly weaker than Def-

inition 4.1), we study higher degree polynomials. The relations here are more complex, and for different ranges of parameters the different classes have different distinguishing capabilities. To separate the distinguishing capabilities of two classes of functions, one needs to present two distributions μ, ν and some function p in one class, and show that every function from the second class cannot imitate the distinguishing capabilities of p between the distributions μ, ν (this shows that the first class is not weaker than the second class). Instead of working with arbitrary generic distributions, we find it more convenient to compare the behaviour of functions over the uniform distribution. This is also interesting on its own, and is the focus of Chapter 5. We use the results obtained there to derive a separation between classes of distinguishers, using a generic reduction of inapproximability results over the uniform distribution, to separation results as above, that works for unbiased (on uniform input) functions.

4.4.1 Inapproximability on the uniform distribution suffices for unbiased tests

Let $\mathcal{C}$ be a class of tests over a universe Ω. Let U denote the uniform distribution over Ω. For a test $p : \Omega \to \{0,1\}$ and two distributions μ, ν over Ω, define $\Delta_p(\mu,\nu) = \Pr_{x\leftarrow\mu}[p(x)=1] - \Pr_{x\leftarrow\nu}[p(x)=1] = \mathbb{E}_\mu[p] - \mathbb{E}_\nu[p]$.

Lemma 4.14. *Let p be a test over Ω such that $\Pr_{x\leftarrow U}[p(x)=1] = \frac{1}{2}+\delta$ (with $\delta \geq 0$). If p is such that for every $q \in \mathcal{C}$:*

$$\Pr_{x\leftarrow U}[p(x)=q(x)] < \frac{1}{2}+\varepsilon,$$

then there exist distributions μ, ν over Ω such that for every $q \in \mathcal{C}$:

$$\Delta_p(\mu,\nu) - \Delta_q(\mu,\nu) > 1 - 2(\varepsilon+\delta).$$

Proof. Set μ to be the uniform distribution over $\{x \in \Omega : p(x)=1\}$, and ν to be the uniform distribution over the complement set $\{x \in \Omega : p(x)=0\}$. Then $\Delta_p(\mu,\nu) = 1$, but

$$\begin{aligned}
\frac{1}{2}+\varepsilon &> \Pr_{x\leftarrow U}[p(x)=q(x)] \\
&= \left(\frac{1}{2}+\delta\right)\Pr_{x\leftarrow\mu}[q(x)=1] + \left(\frac{1}{2}-\delta\right)\Pr_{x\leftarrow\nu}[q(x)=0] \\
&= \frac{1}{2}\cdot\left(\Pr_{x\leftarrow\mu}[q(x)=1] - \Pr_{x\leftarrow\nu}[q(x)=1]\right) + \left(\frac{1}{2}-\delta\right) + \delta\cdot\left(\Pr_{x\leftarrow\mu}[q(x)=1] + \Pr_{x\leftarrow\nu}[q(x)=1]\right)
\end{aligned}$$

$$\geq \quad \frac{1}{2} \cdot \Delta_q(\mu, \nu) + \frac{1}{2} - \delta,$$

giving that $\Delta_q(\mu, \nu) \leq 2(\varepsilon + \delta)$, thus proving the claim. □

The lemma means that if an unbiased p cannot be approximately computed by a class $\mathcal{C}$ (over the uniform distribution), its distinguishing capabilities of distributions over Ω cannot be emulated by any test from $\mathcal{C}$, i.e., the following does *not* hold: for every two distributions μ, ν there exists a test $q \in \mathcal{C}$ such that $\Delta_p(\mu, \nu) \approx \Delta_q(\mu, \nu)$.

4.4.2 Inapproximability of bilinear tests

Using Lemma 4.14, we transform the inapproximability result of bilinear functions from Subsection 5.3.3 to a result about bilinear tests. Let T be a matrix of rank $\frac{n}{2}$. By Proposition 5.14, it cannot be approximated better than $\frac{1}{2} + 2^{-n/6+1/3}$ from $GF(2^n)$-bilinear forms (where the input variables x and y are chosen uniformly). That is, for every $f : \{0,1\}^n \to \{0,1\}$:

$$\Pr_{x,y}[\langle x, Ty \rangle = f(M_x y)] \leq \frac{1}{2} + 2^{-n/6+1/3}.$$

Since $\Pr_{x,y}[\langle x, Ty \rangle = 0] = \frac{1}{2} + 2^{-n/2-1}$ by Fact 5.6, we get from Lemma 4.14, by setting $p(x, y) = \langle x, Ty \rangle + 1$ and $\mathcal{C}$ the class of tests resulting from an arbitrary function composed with $B = (x, y) \mapsto M_x y$ (where M_x is defined in Notation 2.7):

Corollary 4.15. *There exist two distributions μ, ν over $\{0,1\}^n \times \{0,1\}^n$ such that for every $f : \{0,1\}^n \to \{0,1\}$:*

$$\Delta_p(\mu, \nu) - \Delta_{f \circ B}(\mu, \nu) \geq 1 - 2 \cdot (2^{-n/6+1/3} + 2^{-n/2-1}) > 1 - 2^{-n/6+7/3}.$$

Namely, μ will be the uniform distribution over $\ker T$, and ν the uniform distribution over its complement.

4.5 A sum of d small-bias generators that is distinguishable by a degree-$d+1$ polynomial

4.5.1 Background

Viola has shown in [Vio] that the sum of d independent instances of a small-bias generator fools polynomials of degree d (for sub-logarithmic d), improving over

[BV] and [Lov]. This result has been shown in [BV] to be essentially tight with respect to the number of copies needed: using a counting argument, they show that for fixed bias, any generator with output length ℓ that fools all degree $d+1$ polynomials must have seed length $(d+1)\cdot\log\ell - O(1)$. Thus, for every generator with shorter seed, there *exists* a polynomial expression of degree at most $d+1$ that distinguishes a random output of the generator from truly random bits. For a suitable choice of $\varepsilon = o(1)$, the length of d separate seeds for a standard construction of an ε-bias generator is still small enough, giving that in general the sum of d small-bias generators does not necessarily fool polynomials of degree $d+1$. This result has two disadvantages: it is nonexplicit, and the bias used cannot be too small (they require $\varepsilon \geq 1/\text{poly}(\ell)$).

4.5.2 Our result

In this section we give a tightness result that overcomes these two disadvantages: using a simple small-bias generator of our choice (namely, the geometric generator from Section 3.4), we show that the sum of d copies of this generator does not fool an *explicit* polynomial expression of degree $d+1$ in its output bits. Indeed, for a suitable choice of parameters, this generator can have bias that is exponentially small in ℓ.

While the result of [Vio] refers to polynomials over bits, we will first consider the geometric generator over $GF(2^n)$, and show a degree $d+1$ polynomial over $GF(2^n)$ that distinguishes d copies of it from random. We will then use the linearity of the representation of $GF(2^n)$ to derive a polynomial over bits, that has degree $d+1$ and distinguishes from random the sum of d copies of the (binary version of the) geometric generator.

Let $\tilde{G} : GF(2^n)^{2d} \to GF(2^n)^{\ell+1}$ be the sum of d copies of the geometric generator. Namely, for seed $(\tilde{a}_1, \tilde{b}_1, \ldots, \tilde{a}_d, \tilde{b}_d)$, the i-th output element of $\tilde{G}$ is $\tilde{g}_i(\tilde{a}_1, \tilde{b}_1, \ldots, \tilde{a}_d, \tilde{b}_d) = \sum_{j=1}^{d} \tilde{a}_j \cdot \tilde{b}_j{}^i$ (using the arithmetic of $GF(2^n)$). For our purposes, any $\ell \geq 2d+1$ would suffice. To get a final bias of ε over bits, we choose $n = \log\ell + \log(1/\varepsilon)$ (and note that ε can indeed be $2^{-\Omega(\ell)}$ if $n = \Omega(\ell)$).

4.5.3 Distinguishing over $GF(2^n)$

We present a polynomial $\tilde{D}$ over $GF(2^n)$ of the first $2d+1$ output elements of $\tilde{G}$, denoted $\tilde{g}_0, ..., \tilde{g}_{2d}$, that has degree $d+1$, and show that while the output of

this polynomial is close to uniform on uniform input, it always takes the value zero when applied to an output of $\tilde{G}$.

The polynomial $\tilde{D}(\tilde{g}_0, \ldots, \tilde{g}_{2d})$ will be defined as the determinant of the following $(d+1) \times (d+1)$ Hankel matrix (with entries in $GF(2^n)$):

$$A_g^{(d)} = \begin{pmatrix} \tilde{g}_0 & \tilde{g}_1 & \cdots & \tilde{g}_d \\ \tilde{g}_1 & \tilde{g}_2 & \cdots & \tilde{g}_{d+1} \\ \vdots & \vdots & & \vdots \\ \tilde{g}_d & \tilde{g}_{d+1} & \cdots & \tilde{g}_{2d} \end{pmatrix}$$

(that is, the (i,k)-th entry of $A_g^{(d)}$ is $\tilde{g}_{i+k}$).

Indeed, this is a polynomial over $GF(2^n)$ of degree $d+1$ in the output blocks of $\tilde{G}$. We first claim that it is close to uniform when applied to uniform input:

Lemma 4.16. *Let M be a random $m \times m$ Hankel matrix over a finite field $\mathbb{F}$ (i.e., the Hankel matrix defined by $M_{ik} = y_{i+k}$ for $y_0, \ldots, y_{2m-2}$ chosen uniformly at random from $\mathbb{F}$). Then, the distribution of the determinant of M is $\frac{m-1}{|\mathbb{F}|}$-close to uniform (in statistical distance).*

Proof. We proceed by induction on m. For $m = 1$, $\det(M)$ is exactly the only element of M, chosen uniformly from $\mathbb{F}$. Now fix $m > 1$, and let $x = y_0$ be the first (top-left) element of M, and $\bar{y} = (y_1, \ldots, y_{2m-2})$ denote the rest of the elements (on the top row and rightmost column). Denote the submatrix resulting from removing the first row and column by M', and note that it only contains the elements $y_2, \ldots, y_{2m-2}$. We develop the determinant of M by the first row, and write $\det(M) = x \cdot \det(M') + f(\bar{y})$, for some function f of $\bar{y}$. By the induction hypothesis, $\det(M')$ is distributed $\frac{m-2}{|\mathbb{F}|}$-close to uniform, so $\Pr[\det(M') = 0] \leq \frac{1}{|\mathbb{F}|} + \frac{m-2}{|\mathbb{F}|} = \frac{m-1}{|\mathbb{F}|}$. For any fixed nonzero value of $\det(M') \neq 0$ and for any fixed value of $\bar{y}$, the function $\det(M)$ is a (nonconstant) affine function of the uniformly chosen x, implying that, conditioned on $\det(M') \neq 0$, the determinant of M is distributed uniformly in $\mathbb{F}$. The lemma follows. □

Corollary 4.17. *For $\tilde{g}_0, \ldots, \tilde{g}_{2d}$ chosen uniformly at random from $GF(2^n)$, the distribution of $\tilde{D}(\tilde{g}_0, \ldots, \tilde{g}_{2d})$ is $\frac{d}{2^n}$-close to uniform (in statistical distance).*

On the other hand, the polynomial $\tilde{D}$ is always zero on an output of $\tilde{G}$:

Proposition 4.18. *For every seed $\bar{s} = (\tilde{a}_1, \tilde{b}_1, \ldots, \tilde{a}_d, \tilde{b}_d) \in GF(2^n)^{2d}$, the expression $\tilde{D}(\tilde{G}(\bar{s}))$ evaluates to zero.*

Proof. We will show that the matrix $A_g^{(d)}$ is singular for every seed, and thus the polynomial $\tilde{D}(\tilde{g}_0, \ldots, \tilde{g}_{2d}) = \det(A_g^{(d)})$, will always take the value zero when evaluated on an output of $\tilde{G}$. To show that $A_g^{(d)}$ is singular for any seed, we show that its columns are always linearly dependent. More specifically, we show that for any $\tilde{b}_1, \ldots, \tilde{b}_d \in GF(2^n)$ there exist $\tilde{\lambda}_0, \ldots, \tilde{\lambda}_d \in GF(2^n)$, not all zero, such that for all $0 \leq k \leq d$, it holds that $\sum_{i=0}^{d} \tilde{\lambda}_i \tilde{g}_{i+k} = 0$. Letting $\bar{c}_i = (\tilde{g}_i, \ldots, \tilde{g}_{i+d})^T$ denote the i-th column of $A_g^{(d)}$, this means that $\sum_{i=0}^{d} \tilde{\lambda}_i \bar{c}_i$ is the zero vector of $GF(2^n)^{d+1}$.

Consider the polynomial $\tilde{\Lambda}(x) = \prod_{j=1}^{d}(x - \tilde{b}_j)$, the degree d polynomial with roots $\tilde{b}_1, \ldots, \tilde{b}_d$, and set each $\tilde{\lambda}_i$ to be the coefficient of x^i in $\tilde{\Lambda}(x)$. Note that always $\tilde{\lambda}_d = 1$. Then, using the definition of $\tilde{G}$ (i.e., $\tilde{g}_i = \sum_j \tilde{a}_j \tilde{b}_j{}^i$), we get for every $0 \leq k \leq \ell - d$:

$$\begin{aligned}
\sum_{i=0}^{d} \tilde{\lambda}_i \tilde{g}_{i+k} &= \sum_{i=0}^{d} \tilde{\lambda}_i \sum_{j=1}^{d} \tilde{a}_j \tilde{b}_j{}^{i+k} \\
&= \sum_{j=1}^{d} \tilde{a}_j \tilde{b}_j{}^{k} \cdot \sum_{i=0}^{d} \tilde{\lambda}_i \tilde{b}_j{}^{i} \\
&= \sum_{j=1}^{d} \tilde{a}_j \tilde{b}_j{}^{k} \cdot \tilde{\Lambda}(\tilde{b}_j),
\end{aligned}$$

which is 0 as the $\tilde{b}_j$'s are all roots of $\tilde{\Lambda}(x)$. □

We have thus obtained, using the event $\tilde{D} = 0$ in Definition 2.3:

Theorem 4.19 ($\tilde{D}$ distinguishes $\tilde{G}$ from random)**.** *For $\tilde{D}$ the determinant of $A_g^{(d)}$, the distributions $\tilde{D}(\tilde{U}_{\ell+1})$ and $\tilde{D}(\tilde{G}(\tilde{U}_{2d}))$ are $(1 - \frac{d+1}{2^n})$-far (in statistical distance), where $\tilde{U}_k$ denotes the uniform distribution over $GF(2^n)^k$.*

4.5.4 Distinguishing over bits

Recall that for G the binary version of $\tilde{G}$, Viola's result [Vio] guarantees fooling $GF(2)$-polynomials of degree d. We will show an explicit polynomial of degree $d + 1$ that distinguishes a random output of G from a random element of $\{0,1\}^{(\ell+1)\cdot n}$. By combining Theorem 4.19 with Lemma 4.4, we obtain that D_1, the first bit (say) of D, the binary version of the $GF(2^n)$-polynomial $\tilde{D}$, distinguishes G from random:

Theorem 4.20 (D_1 distinguishes G from random)**.** *The polynomial* $D_1 : \{0,1\}^{(\ell+1)\cdot n} \to \{0,1\}$ *has degree at most* $d+1$ *(over* $GF(2)$*) and satisfies:*

$$\left| \Pr_{s\in\{0,1\}^{2d\cdot n}}[D_1(G(s)) = 0] - \Pr_{x\in\{0,1\}^{(\ell+1)\cdot n}}[D_1(x) = 0] \right| \geq \frac{1}{2} - \frac{d}{2^n}.$$

Proof. We first note that by Lemma 4.4, the polynomial D_1 indeed has degree at most $d+1$.

By Corollary 4.17, the distribution of $\tilde{D}(\tilde{x})$ is $\frac{d}{2^n}$-close to uniform over $GF(2^n)$ when $\tilde{x}$ is chosen uniformly from $GF(2^n)^{\ell+1}$, and thus (by definition) the distribution of $D(x)$ is $\frac{d}{2^n}$-close to uniform over $\{0,1\}^n$, where x is chosen uniformly from $\{0,1\}^{(\ell+1)\cdot n}$. Specifically, considering the event "first bit is zero" in Definition 2.3, we have

$$\Pr_{x\in\{0,1\}^{(\ell+1)\cdot n}}[D_1(x) = 0] \leq \frac{1}{2} + \frac{d}{2^n}.$$

On the other hand, by Proposition 4.18, $\tilde{D}(\tilde{G}(\bar{s})) = 0$ for every $\bar{s} \in GF(2^n)^{2d}$, giving that $D(G(s)) = 0^n$ for every $s \in \{0,1\}^{2d\cdot n}$, and specifically

$$\Pr_{s\in\{0,1\}^{2d\cdot n}}[D_1(G(s)) = 0] = 1.$$

The theorem follows. □

Remark. While the polynomial $\tilde{D}$ is explicit as a polynomial over $GF(2^n)$, the polynomial D_1 can be described using Lemma 2.11, which is proved non-constructively in Section 2.2. A fully-explicit D_1 can be obtained from the constructive proof of this lemma, presented in Section 4.3.3.

4.5.5 Using larger prime fields

All the above results (including the geometric generator; see Subsection 3.4.4) generalize naturally to larger prime fields, as does the result of [Vio]. See Section 2.4 for the generalization of the definitions.

Generalizing D_1 of Theorem 4.20 to ${D_1}^{(q)}$, we obtain:

Theorem 4.21. *For every prime* q*, the polynomial* ${D_1}^{(q)} : GF(q)^{(\ell+1)\cdot n} \to GF(q)$ *has degree at most* $d+1$ *and satisfies:*

$$\left| \Pr_{s\in GF(q)^{2d\cdot n}}\left[{D_1}^{(q)}(G^{(q)}(s)) = 0\right] - \Pr_{x\in GF(q)^{(\ell+1)\cdot n}}\left[{D_1}^{(q)}(x) = 0\right] \right| \geq 1 - \frac{1}{q} - \frac{d}{q^n}.$$

Chapter 5

$GF(2^n)$-Polynomials vs. $GF(2)$-Polynomials: as Approximators

5.1 Introduction

5.1.1 Motivation

We strive to compare the approximating powers of $GF(2^n)$-polynomials of bounded degree to that of $GF(2)$-polynomials of the same degree. Apart from the inherent interest in studying the relations between the computing power of various classes of functions, a concrete motivation is given in Section 4.4: the different approximation powers imply different distinguishing capabilities.

In this chapter we mainly focus on the bilinear case, studying which $GF(2)$-bilinear forms can be calculated, or approximated, from $GF(2^n)$-bilinear forms.

5.1.2 The general picture

In this chapter we study the relations between the various forms of polynomials as classes of functions, analogously to the picture presented in Section 4.2:

- The class of ℓ-variate $GF(2^n)$-polynomials of degree at most d, composed with an arbitrary boolean function $f : GF(2^n) \to \{0,1\}$ (referred to as

the final function). We denote this class by $\mathcal{P}_d^{(\ell)}$.

- The class of ℓ-variate $GF(2^n)$-polynomials of degree at most d, composed with a linear (final) function in the bits representing the resulting $GF(2^n)$-element. We denote it by $\mathcal{L}_d^{(\ell)}$.
- The class of ℓn-variate $GF(2)$-polynomials of degree at most d. We denote it by $\mathcal{B}_d^{(\ell)}$.

Note that n is implicit in all these notions. The classes $\mathcal{P}_d^{(\ell)}, \mathcal{L}_d^{(\ell)}$ and $\mathcal{B}_d^{(\ell)}$ are exactly the classes of distinguishers fooled according to Definitions 4.1, 4.3 and 2.1, respectively.

For every d, n and ℓ, we know that $\mathcal{L}_d^{(\ell)} \subset \mathcal{P}_d^{(\ell)}$ by definition, and $\mathcal{L}_d^{(\ell)} \subseteq B_d^{(\ell)}$ by Lemma 4.4. While Theorem 4.5 shows that the classes $\mathcal{L}_1^{(\ell)}$ and $\mathcal{B}_1^{(\ell)}$ coincide for every n and ℓ, we study their relation for larger d. We believe that then, this is no longer the case: that is, when $d \geq 2$, $\mathcal{L}_d^{(\ell)}$ is strictly weaker than $\mathcal{B}_d^{(\ell)}$. We also strive to separate the classes $\mathcal{P}_d^{(\ell)}$ from $\mathcal{B}_d^{(\ell)}$, by showing that each has a function that cannot be emulated (either exactly or approximately) by any function from the other class, for a suitable choice of ℓ. The case $\ell = 1$ immediately has $\mathcal{P}_d^{(1)}$ strictly stronger than $\mathcal{B}_d^{(1)}$; we will mostly study when $\mathcal{B}_d^{(\ell)}$ can be shown to be not weaker than $\mathcal{P}_d^{(\ell)}$, that is, some function in $\mathcal{B}_d^{(\ell)}$ cannot be computed, or approximated, by any function from $\mathcal{P}_d^{(\ell)}$.

Our main focus in this chapter is the case $d = 2$, and even more specifically, the case of bilinear forms (that is, multilinear polynomials of total degree at most 2). We define the bilinear versions of the above classes: the classes $\mathcal{P}_{1,1}^{(\ell)}$ and $\mathcal{L}_{1,1}^{(\ell)}$ contain $GF(2^n)$-quadratic polynomials in which squaring an element is not allowed (with either arbitrary or linear final functions allowed, respectively). The class $\mathcal{B}_{1,1}^{(\ell)}$ consists of $GF(2)$-quadratic polynomials in which two bits in the same n-bit block are never multiplied. We will also restrict ourselves to the case of only two blocks, $\ell = 2$, for most of the chapter.

5.1.3 Overview

Striving to separate the class $\mathcal{B}_d^{(\ell)}$ from $\mathcal{P}_d^{(\ell)}$, In Section 5.2 we first consider perfect emulation of $\mathcal{B}_d^{(\ell)}$ by $\mathcal{P}_d^{(\ell)}$. We begin with counting arguments to show nonconstructively that $\mathcal{P}_d^{(\ell)}$ and $\mathcal{B}_d^{(\ell)}$ are separated (for any $d \geq 2$ and for large ℓ). Then, we focus on the case $d = 2, \ell = 2$ and specifically on bilinear forms. We find exactly which functions in $\mathcal{B}_{1,1}^{(2)}$ can be computed by functions in $\mathcal{P}_{1,1}^{(2)}$.

Still focusing on bilinear forms, In Section 5.3 (which contains the main results of this chapter) we consider *approximating* functions from $\mathcal{B}^{(2)}_{1,1}$ by functions from $\mathcal{P}^{(2)}_{1,1}$ or $\mathcal{L}^{(2)}_{1,1}$. We characterize the bilinear functions that can be approximated to various rates, and present explicit forms (in $\mathcal{B}^{(2)}_{1,1}$) that cannot be reasonably approximated at all (by bilinear functions from $\mathcal{P}^{(2)}_{1,1}$). We extend the argument to give partial results for quadratic forms (that is, functions in $\mathcal{P}^{(2)}_{2}$).

Section 5.4 ends the chapter with a discussion of a power functions in $\mathcal{B}^{(\ell)}_{2}$ have over those in $\mathcal{P}^{(\ell)}_{2}$: the ability to multiply bits within the same block (that is, the class $\mathcal{P}^{(\ell)}_{2}$ is replaced by a larger class, calculating n functions in $\mathcal{B}^{(\ell)}_{1,1}$ at the same time and taking verdict according to their n outcomes). We show that for some natural class of polynomials, this does not give additional strength.

5.2 Computing exactly

5.2.1 Counting arguments

For $\ell = 1$, the class $\mathcal{P}^{(1)}_{d}$ contains all functions $GF(2^n) \to \{0,1\}$ while the classes $\mathcal{L}^{(1)}_{d}$ and $\mathcal{B}^{(1)}_{d}$ only contain some of those functions. On the other hand, we will show that for very large ℓ, the class $\mathcal{P}^{(\ell)}_{d}$ contains too few functions to cover the number of different functions in $\mathcal{B}^{(\ell)}_{d}$.

The number of ℓ-variate $GF(2^n)$-polynomials of degree at most d is $(2^n)^{\binom{\ell+d}{d}}$ (see footnote 8 on page 31), which is at most $2^{n(\ell+d)^d}$. The number of final functions $f : GF(2^n) \to \{0,1\}$ is 2^{2^n}.[1] So for every n, d and ℓ, we get $|\mathcal{P}^{(\ell)}_{d}| \leq 2^{n(\ell+d)^d + 2^n}$.

On the other hand, the number of $n\ell$-variate $GF(2)$-polynomials of degree at most d is $2^{\binom{n\ell}{d}} \geq 2^{(\frac{n\ell}{d})^d}$. It can be verified that when ℓ is large enough, namely, $\ell > 2\max\{2^{n/d}, 2d/n^{1-1/d}\}$, indeed $|\mathcal{P}^{(\ell)}_{d}| < |\mathcal{B}^{(\ell)}_{d}|$.

This gives that for every n and $d > 1$, for a suitable choice of ℓ, there are functions in $\mathcal{B}^{(\ell)}_{d}$ that cannot be *exactly* computed by functions from $\mathcal{P}^{(\ell)}_{d}$.

However, this method requires very large ℓ and produces a nonexplicit separating function. In addition, it does not rule out the possibility of a reasonable approximation of $\mathcal{B}^{(\ell)}_{d}$ by $\mathcal{P}^{(\ell)}_{d}$.

[1] Note that we have counted some functions many times. E.g., we can change the value of the final function f on inputs outside the range of the polynomial chosen, or ignore the leading coefficient and fix it in f. However, we are interested in an upper bound.

5.2.2 Bilinear forms

We consider the question of which $GF(2)$-bilinear functions $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ (functions in $\mathcal{B}^{(2)}_{1,1}$) can be computed from a $GF(2^n)$-bilinear function $GF(2^n) \times GF(2^n) \to GF(2^n)$ (composed with the final function, these are functions in $\mathcal{P}^{(2)}_{1,1}$). Specifically, we consider the only nontrivial $GF(2^n)$-bilinear function $\tilde{B}(\tilde{x}, \tilde{y}) = \tilde{x} \cdot \tilde{y}$, and ask which $GF(2)$-bilinear functions $(x, y) \mapsto \langle x, Ty \rangle$ can be computed from it (we interchangeably talk about $GF(2)$-bilinear forms and the $n \times n$ matrices describing them). We will show that the set of matrices $\{Q_\gamma\}_{\gamma \in \{0,1\}^n}$ defined by Lemma 2.11 and specified explicitly in Subsection 4.3.3, is exactly the set of $GF(2)$-bilinear forms that can be computed from the multiplication $B(x, y) = M_x y$.

Definition 5.1. *We say that a bilinear form T* can be computed from *B if there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that for all $x, y \in \{0,1\}^n$ it holds that $f(M_x y) = \langle x, Ty \rangle$.*

Proposition 5.2. *For all $\gamma \in \{0,1\}^n$, the form Q_γ can be computed from B.*

Proof. Use as f the linear function $f(v) = \langle \gamma, v \rangle$. Then by definition of Q_γ (Lemma 2.11), for all x, y it holds that $f(M_x y) = \langle \gamma, M_x y \rangle = \langle x, Q_\gamma y \rangle$. □

Proposition 5.3. *If a form T can be computed from B, then $T = Q_\gamma$ where γ is the first row of T, namely $\gamma = T^T \cdot e_0$. Moreover, the corresponding final function f is the linear function $f(v) = \langle \gamma, v \rangle$.*

Proof. Using the hypothesis, fix f such that $f(M_x y) = \langle x, Ty \rangle$ for all $x, y \in \{0,1\}^n$. Note that, specifically, for $x = e_0$, we have for every y:

$$f(y) = f(M_{e_0} y) = \langle e_0, Ty \rangle = \left\langle T^T e_0, y \right\rangle = \langle \gamma, y \rangle \,,$$

using $M_{e_0} = I$ which follows from e_0 representing the unit of $GF(2^n)$.

In addition to having established the moreover part of the claim, we can now get that, for all $x, y \in \{0,1\}^n$:

$$\langle x, Ty \rangle = f(M_x y) = \langle \gamma, M_x y \rangle = \langle x, Q_\gamma y \rangle \,,$$

implying that T and Q_γ describe the same bilinear form (and thus they are equal as matrices). □

Remark. The above two Propositions 5.2 and 5.3 yield $\mathcal{L}^{(2)}_{1,1} = \mathcal{B}^{(2)}_{1,1} \cap \mathcal{P}^{(2)}_{1,1}$.

5.3 Approximating bilinear forms

We now consider the possibility of approximating a $GF(2)$-bilinear form in x and y from their $GF(2^n)$-multiplication $B(x,y) = M_x y$. We will show that while the forms Q_γ can be computed exactly (as shown in Subsection 5.2.2), every form that can be reasonably approximated from B differs from some Q_γ by a matrix of small rank. This is shown by reducing any approximating function to a linear approximating function. By the contrapositive we derive the existence of matrices that cannot be approximated from B noticeably better than guessing. On the other hand, somewhat reasonable approximations are possible for some matrices other than the Q_γ's.

Definition 5.4. *For a function $f : \{0,1\}^n \to \{0,1\}$ and $\delta \in [0,1]$, we say that a bilinear form T is δ-*approximated by *$f \circ B$ if*

$$\Pr_{x,y}[f(M_x y) = \langle x, Ty\rangle] \geq \delta.$$

5.3.1 Very good approximations

We first show that only the forms Q_γ can be approximated really well (i.e., better than $\frac{7}{8}$):

Theorem 5.5. *If a bilinear form T can be δ-approximated by some $f \circ B$ for any $\delta > \frac{7}{8} + 2^{-n}$, then there exists $\gamma \in \{0,1\}^n$ such that $T = Q_\gamma$.*

Proof. First, we modify f so that $f(0) = 0$, possibly only improving the approximation rate (since $M_x y = 0$ implies $\langle x, Ty\rangle = 0$). Since by the hypothesis

$$\delta \leq \Pr_{x,y}[f(M_x y) = \langle x, Ty\rangle] \leq \Pr_{x \neq 0, y}[f(M_x y) = \langle x, Ty\rangle] + \Pr_x[x = 0],$$

we know that $\Pr_{x\neq 0,y}[f(M_x y) = \langle x, Ty\rangle] \geq \delta - 2^{-n}$, so there exists some fixed $z \neq 0$ such that $\Pr_y[f(M_z y) = \langle z, Ty\rangle] \geq \delta - 2^{-n}$. Since $z \neq 0$, the matrix M_z is invertible and we set $\gamma = \gamma_z = {M_z^{-1}}^T T^T z$. Using the definition of Q_γ (Lemma 2.11) and $f(0) = 0$,

$$\begin{aligned}
\Pr_{x,y}[f(M_x y) = \langle x, Q_\gamma y\rangle] &= \Pr_{x,y}[f(M_z \cdot M_z^{-1} M_x y) = \langle \gamma, M_x y\rangle] \\
&= \Pr_{\substack{x\neq 0, y,\\ v = M_z^{-1} M_x \cdot y}}[f(M_z v) = \langle \gamma, M_z v\rangle] \cdot \Pr_x[x \neq 0] + 1 \cdot \Pr_x[x = 0] \\
&= \Pr_{v \in \{0,1\}^n}\left[f(M_z v) = \left\langle {M_z^{-1}}^T T^T z, M_z v\right\rangle\right](1 - 2^{-n}) + 2^{-n}
\end{aligned}$$

$$
\begin{aligned}
&\geq \Pr_v\left[f(M_z v) = \left\langle z, \left(M_z^{-1^T} T^T\right)^T M_z v\right\rangle\right] - 2^{-n} + 2^{-n} \\
&= \Pr_v[f(M_z v) = \langle z, Tv\rangle] \\
&\geq \delta - 2^{-n}.
\end{aligned}
$$

We get:

$$
\begin{aligned}
\Pr_{x,y}[\langle x, Ty\rangle = \langle x, Q_\gamma y\rangle] &\geq \Pr_{x,y}[\langle x, Ty\rangle = f(M_x y) = \langle x, Q_\gamma y\rangle] \\
&= 1 - \Pr[\langle x, Ty\rangle \neq f(M_x y) \vee f(M_x y) \neq \langle x, Q_\gamma y\rangle] \\
&\geq 1 - \Pr_{x,y}[\langle x, Ty\rangle \neq f(M_x y)] - \Pr_{x,y}[f(M_x y) \neq \langle x, Q_\gamma y\rangle] \\
&\geq 1 - (1-\delta) - (1 - \delta + 2^{-n}) \\
&= 2\delta - 1 - 2^{-n} \\
&> \frac{3}{4}.
\end{aligned}
$$

Since the bilinear form $T - Q_\gamma$ is zero on more than $\frac{3}{4}$ of its inputs, it must be identically zero by the following folklore Fact 5.6, implying $T = Q_\gamma$. □

Fact 5.6 (folklore)**.** *For all T, it holds that* $\Pr_{x,y}[\langle x, Ty\rangle = 0] = \frac{1}{2} + 2^{-\operatorname{rank}(T)-1}$.

Proof. Since $\Pr_y[y \in \ker T] = 2^{\dim \ker T}/2^n = 2^{-\operatorname{rank}(T)}$,

$$
\begin{aligned}
\Pr_{x,y}[\langle x, Ty\rangle = 0] &= \Pr_y[y \in \ker T] \cdot 1 + \Pr_y[y \notin \ker T] \cdot \Pr_{x,y}[\langle x, Ty\rangle = 0 | Ty \neq 0] \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr_y[y \in \ker T] \\
&= \frac{1}{2} + 2^{-\operatorname{rank}(T)-1}.
\end{aligned}
$$

□

Remark. By combining Theorem 5.5 with Proposition 5.2, we get that if a bilinear form can be approximated to a rate significantly higher than $\frac{7}{8}$, it can be computed exactly.

5.3.2 Somewhat good approximations

Despite having disqualified approximations better than $\frac{7}{8}$, there are matrices other than the Q_γ's that can be somewhat (i.e., $\frac{3}{4}$) approximated:

Proposition 5.7. *There exists a bilinear form $T \notin \{Q_\gamma\}_\gamma$ that can be $\frac{3}{4}$-approximated by some $f \circ B$. Moreover, f is linear, that is, there exists $\gamma \in \{0,1\}^n$ such that $f(M_x y) = \langle \gamma, M_x y\rangle$.*

Proof. Take any $\gamma \in \{0,1\}^n$, and any rank-1 matrix E. Set $T = Q_\gamma + E$. Then using $\langle x, Ty\rangle = \langle x, Q_\gamma y\rangle + \langle x, Ey\rangle$, we get

$$\Pr_{x,y}[\langle \gamma, M_x y\rangle = \langle x, Ty\rangle] = \Pr_{x,y}[\langle x, Q_\gamma y\rangle = \langle x, Ty\rangle] = \Pr_{x,y}[\langle x, Ey\rangle = 0] = \frac{3}{4}$$

by Fact 5.6, so $f(v) = \langle \gamma, v\rangle$ composed with B does $\frac{3}{4}$-approximate T. □

However it is clear that such matrices can be approximated from B simply because they are approximated by a matrix Q_γ, which can be computed exactly from B. We will next show that in fact this is the only way a bilinear form can be approximated from B, to any approximation rate.

5.3.3 Arbitrary Approximations

We show that the "reason" a $GF(2)$-bilinear form can be approximated from B is always the fact that it is close enough to a bilinear form that can be computed exactly. First, we show that this is straightforward for *linear* approximating functions (that is, approximating a function in $\mathcal{B}^{(2)}_{1,1}$ using a function from $\mathcal{L}^{(2)}_{1,1}$):

Proposition 5.8. *If a matrix T is $(\frac{1}{2} + \frac{\varepsilon}{2})$-approximated by $f \circ B$ for a linear function f, then $T = Q_\gamma + E$ for some γ and some matrix E, such that* $\operatorname{rank}(E) \leq \log(1/\varepsilon)$.

Proof. Suppose T is $(\frac{1}{2} + \frac{\varepsilon}{2})$-approximated by $f(v) = \langle \gamma, v\rangle$. Thus, using $f(M_x y) = \langle \gamma, M_x y\rangle = \langle x, Q_\gamma y\rangle$, we get:

$$\frac{1}{2} + \frac{\varepsilon}{2} \leq \Pr_{x,y}[\langle x, Ty\rangle = f(M_x y)] = \Pr_{x,y}[\langle x, Ty\rangle = \langle x, Q_\gamma y\rangle].$$

Setting $E = T - Q_\gamma$, we get that $\frac{1}{2} + \frac{\varepsilon}{2} \leq \Pr_{x,y}[\langle x, Ey\rangle = 0] = \frac{1}{2} + \frac{1}{2} \cdot 2^{-\operatorname{rank}(E)}$ (where the equality follows from Fact 5.6), so indeed $\operatorname{rank}(E) \leq \log(1/\varepsilon)$. □

To show a similar claim for an arbitrary final function f, we will prove a generic reduction to the case of a linear final function. This shows that with respect to approximating functions from $\mathcal{B}^{(2)}_{1,1}$, the class $\mathcal{L}^{(2)}_{1,1}$ is essentially as powerful as the class $\mathcal{P}^{(2)}_{1,1}$.

Theorem 5.9. *If a bilinear form T can be $(\frac{1}{2} + \varepsilon)$-approximated by some function $f \circ B$, then it can be $(\frac{1}{2} + \frac{\delta}{2})$-approximated by a* linear *function for $\delta \geq \frac{\varepsilon^3}{1-\varepsilon} - 2^{-n-2}$. That is, there exists $\gamma \in \{0,1\}^n$ such that T is $(\frac{1}{2} + \frac{\delta}{2})$-approximated by $\ell \circ B$ for $\ell(v) = \langle \gamma, v\rangle$.*

Remark. More careful analysis can yield $\delta \geq \frac{\varepsilon^2}{4(\log(1/\varepsilon)+2)} - 2^{-n}$. See Section A.2 for details.

Proof. Suppose that for some function f, it holds that $\Pr_{x,y}[f(M_x y) = \langle x, Ty\rangle] \geq \frac{1}{2} + \varepsilon$. Define the advantage of this function for fixed x by

$$A(x) = \Pr_y[f(M_x y) = \langle x, Ty\rangle] - \frac{1}{2}.$$

Then, $\mathbb{E}_x[A(x)] \geq \varepsilon$, while for all x it holds that $A(x) \leq \frac{1}{2}$. Using an averaging argument, we first lower bound $p = \Pr_x[A(x) \geq \frac{\varepsilon}{2}]$:

$$\begin{aligned} \varepsilon &\leq \mathbb{E}_x[A(x)] \\ &\leq \Pr_x\left[A(x) < \frac{\varepsilon}{2}\right] \cdot \frac{\varepsilon}{2} + \Pr_x\left[A(x) \geq \frac{\varepsilon}{2}\right] \cdot \frac{1}{2} \\ &= (1-p) \cdot \frac{\varepsilon}{2} + p \cdot \frac{1}{2}, \end{aligned}$$

giving $p \geq \frac{\varepsilon}{1-\varepsilon}$.

We have shown that $p \geq \frac{\varepsilon}{1-\varepsilon}$ of the x's satisfy $\Pr_y[f(M_x y) = \langle x, Ty\rangle] \geq \frac{1}{2} + \frac{\varepsilon}{2}$. For each $x \neq 0$ denote the vector ${M_x^{-1}}^T T^T x$ by γ_x, and the function $v \mapsto \langle \gamma_x, v\rangle$ by ℓ_x. Then, for all $x \neq 0$:

$$\begin{aligned} \Pr_y\left[f(M_x y) = \langle x, Ty\rangle\right] &= \Pr_{v = M_x \cdot y}\left[f(v) = \left\langle x, T \cdot M_x^{-1} v\right\rangle\right] \\ &= \Pr_{v \in \{0,1\}^n}\left[f(v) = \left\langle {M_x^{-1}}^T T^T x, v\right\rangle\right] \\ &= \Pr_v[f(v) = \ell_x(v)]. \end{aligned}$$

Thus, for at least $p \cdot 2^n - 1$ nonzero x's, the function f agrees with the linear function ℓ_x on at least a $\frac{1}{2} + \frac{\varepsilon}{2}$ fraction of their domain.

The following well known fact can be deduced, e.g., from Theorem 15 (part 2) of [GRS]. For self containment, we also prove it in Appendix B.

Fact 5.10. *Any function $(\frac{1}{2} + \frac{\varepsilon}{2})$-agrees with at most $\frac{1}{\varepsilon^2}$ affine functions.*

This gives us that the $p \cdot 2^n - 1$ different nonzero x's are matched to at most $\frac{1}{\varepsilon^2}$ distinct linear functions ℓ_x, i.e., distinct vectors γ_x. By an averaging argument, there exists $\gamma \in \{0,1\}^n$ that corresponds to at least $\frac{p \cdot 2^n - 1}{\varepsilon^{-2}} = \varepsilon^2(p \cdot 2^n - 1)$ different nonzero x's, so

$$\delta = \Pr_{x \neq 0}[\gamma_x = \gamma] \geq \frac{\varepsilon^2(p \cdot 2^n - 1)}{2^n - 1} \geq \varepsilon^2(p - 2^{-n}) \geq \frac{\varepsilon^3}{1-\varepsilon} - 2^{-n-2}$$

(where the last inequality used $p \geq \frac{\varepsilon}{1-\varepsilon}$ and $\varepsilon \leq \frac{1}{2}$).

Given this γ, we now define the function $\ell(v) = \langle \gamma, v \rangle$. We finally show that $\ell \circ B$ does $(\frac{1}{2} + \frac{\delta}{2})$-approximate T:

$$\begin{aligned}
\Pr_{x,y}[\ell(M_x y) = \langle x, Ty \rangle] &= \Pr[x \neq 0] \cdot \Pr_{\substack{x \neq 0 \\ v = M_x \cdot y}} \left[\ell(v) = \left\langle x, T \cdot M_x^{-1} v \right\rangle\right] + \Pr[x = 0] \cdot 1 \\
&= (1 - 2^{-n}) \cdot \Pr_{x \neq 0, v} [\langle \gamma, v \rangle = \langle \gamma_x, v \rangle] + 2^{-n} \\
&\geq \Pr_{x \neq 0, v} [\langle \gamma - \gamma_x, v \rangle = 0] \\
&= \Pr_{x \neq 0}[\gamma_x = \gamma] \cdot \Pr_{x \neq 0, v} [\langle 0, v \rangle = 0] + \Pr_{x \neq 0}[\gamma_x \neq \gamma] \cdot \Pr_{x \neq 0, v} \left[\langle \gamma - \gamma_x, v \rangle = 0 \middle| \gamma \neq \gamma_x\right] \\
&= \delta \cdot 1 + (1 - \delta) \cdot \tfrac{1}{2} \\
&= \frac{1}{2} + \frac{\delta}{2}. \qquad (5.1)
\end{aligned}$$

□

As a corollary we get:

Corollary 5.11. *If a matrix T can be $(\frac{1}{2} + \varepsilon)$-approximated (by any function $f \circ B$) for any $\varepsilon \geq 2^{-\frac{n+1}{3}}$, then $T = Q_\gamma + E$ for some γ and some matrix E with* $\operatorname{rank}(E) \leq 3\log(1/\varepsilon) + 1$.

Proof. By Theorem 5.9, T can be $(\frac{1}{2} + \frac{\delta}{2})$-approximated by a linear function for $\delta \geq \frac{\varepsilon^3}{1-\varepsilon} - 2^{-n-2} > \varepsilon^3 - \varepsilon^3/2 = \varepsilon^3/2$. The corollary now follows directly from Proposition 5.8. □

Using Corollary 5.11, we can find $GF(2)$-bilinear forms that cannot be approximated from the $GF(2^n)$-bilinear form $\tilde{B}$, thus separating $\mathcal{B}_{1,1}^{(2)}$ from $\mathcal{P}_{1,1}^{(2)}$:

Proposition 5.12. *There exists a matrix T that can only be $(\frac{1}{2} + 2^{-\Omega(n)})$-approximated from B.*

Proof. By Corollary 5.11, we need only show a matrix T such that for all γ, the matrix $E = Q_\gamma - T$ has rank $\Omega(n)$. First, we use a very rough upper bound on the number of matrices with bounded rank:[2]

[2]The number of subspaces of dimension at most k is at most the number of k spanning vectors, $(2^n)^k = 2^{nk}$. Each subspace of dimension at most k is the range of at most $(2^k)^n = 2^{nk}$ different matrices, since each column of the matrix must be a vector of the subspace. Since each matrix of rank at most k has a subspace of dimension at most k as its range, the number of matrices of rank at most k is at most $2^{nk} \cdot 2^{nk} = 2^{2nk}$.

Fact 5.13. *The number of $n \times n$ matrices of rank at most k is at most 2^{2nk}.*

Now we can upper bound (for arbitrary $k < n$) the number of matrices T such that there exist $\gamma \in \{0,1\}^n$ and a matrix E such that $T = Q_\gamma + E$ and $\mathrm{rank}(E) \leq k$, by 2^{2nk} (number of matrices E) times 2^n (number of matrices Q_γ). This number fails to account for the number of all $n \times n$ matrices as long as $2^{(2k+1)n} < 2^{n^2}$, that is, $k < \frac{n-1}{2}$. The proposition now follows for any choice of $\Omega(n) \leq k < \frac{n-1}{2}$, e.g., $k = \lfloor \frac{n}{3} \rfloor$. □

We can also prove a constructive version of Proposition 5.12:

Proposition 5.14. *For any constant $0 < c \leq \frac{1}{2}$ and every $\gamma \in \{0,1\}^n$ and matrix E such that $c \cdot n \leq \mathrm{rank}(E) \leq (1-c) \cdot n$, the matrix $T = Q_\gamma + E$ cannot be approximated from B better than $\frac{1}{2} + 2^{-cn/3+1/3}$. As a specific case, any matrix of rank $\frac{n}{2}$ cannot be approximated from B better than $\frac{1}{2} + 2^{-n/6+1/3}$.*

Proof. Fix γ and E as above. While we are given that T differs from Q_γ by a matrix of rank at least cn, for every other $\gamma' \neq \gamma$,

$$\begin{aligned} \mathrm{rank}(T - Q_{\gamma'}) &\geq \mathrm{rank}(Q_\gamma - Q_{\gamma'}) - \mathrm{rank}(Q_\gamma - T) \\ &= \mathrm{rank}(Q_{\gamma-\gamma'}) - \mathrm{rank}(-E) \\ &= n - \mathrm{rank}(E) \\ &\geq cn, \end{aligned}$$

Using the generic inequality $\mathrm{rank}(A) \geq \mathrm{rank}(A+B) - \mathrm{rank}(B)$, the linearity of the matrix Q_γ in γ (Proposition 2.12) and the fact that $\mathrm{rank}(Q_{\gamma-\gamma'}) = n$ when $\gamma - \gamma'$ is not zero. Thus T differs from *every* matrix $Q_{\gamma'}$ by a matrix of rank at least cn. Again by Corollary 5.11, the form T cannot be $(\frac{1}{2} + \varepsilon)$-approximated from B unless $\varepsilon < 2^{-cn/3+1/3}$. □

5.3.4 Quadratic forms

The arguments of Subsection 5.3.3 can be somewhat extended to quadratic forms. $GF(2)$-bilinear forms are a special case of $GF(2)$-quadratic forms, but to show that one form cannot be approximated we now need to consider $GF(2^n)$ quadratic forms other than $\tilde{B}(\tilde{x}, \tilde{y}) = \tilde{x} \cdot \tilde{y}$. The generic $GF(2^n)$-quadratic form can be written as

$$\tilde{p}(\tilde{x}, \tilde{y}) = \tilde{x} \cdot \tilde{y} + \tilde{a} \cdot \tilde{x} + \tilde{b} \cdot \tilde{y} + \tilde{c} \cdot \tilde{x}^2 + \tilde{d} \cdot \tilde{y}^2$$

for four constants $\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d} \in GF(2^n)$, where without loss of generality we removed the leading and free coefficients as they can be accounted for by the final function f (the case of coefficient 0 to $\tilde{x} \cdot \tilde{y}$ is the bilinear case handled previously).

Denote as usual by $p : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ the respective polynomial over the bits of the representing vectors: $p(x,y) = M_x y + M_a x + M_b y + M_c S x + M_d S y$ (where S is the matrix corresponding to the squaring operation, defined in Notation 2.13).

Definition 5.15. *For a matrix $T \in \{0,1\}^{n \times n}$ and two vectors $r, s \in \{0,1\}^n$, we say that the affine bilinear form, $q(x,y) = \langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle$* can be computed from *$p$ if there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that for all $x, y \in \{0,1\}^n$: $f(p(x,y)) = q(x,y)$.*

Definition 5.16. *For a matrix $T \in \{0,1\}^{n \times n}$ and two vectors $r, s \in \{0,1\}^n$, we say that the affine bilinear form, $q(x,y) = \langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle$* is *$(\frac{1}{2} + \varepsilon)$*-approximated by *$f \circ p$ for $f : \{0,1\}^n \to \{0,1\}$ if*

$$\Pr_{x,y}[f(p(x,y)) = q(x,y)] \geq \frac{1}{2} + \varepsilon.$$

Claim 5.17. *For every three vectors $\gamma, r, s \in \{0,1\}^n$ with nonzero γ,[3] there exists a quadratic polynomial p as above such that the affine bilinear form $q(x,y) = \langle x, Q_\gamma y \rangle + \langle r, x \rangle + \langle s, y \rangle$ can be computed from p.*

Proof. Take $a = Q_\gamma^{-1} r$, $b = Q_\gamma^{-1} s$, and set $p(x,y) = M_x y + M_a x + M_b y$. Use the function $f(v) = \langle \gamma, v \rangle$. Then for all $x, y \in \{0,1\}^n$, using the defitinion of the matrix Q_γ (Lemma 2.11):

$$f(p(x,y)) = \langle \gamma, M_x y + M_a x + M_b y \rangle = \langle x, Q_\gamma y \rangle + \langle x, Q_\gamma a \rangle + \langle y, Q_\gamma b \rangle = q(x,y).$$

□

Before extending our negative results, we will first show for convenience that the constants $\tilde{a}, \tilde{b}$ in $\tilde{p}$ can be set to zero without limiting the generality. We substitute the variables $\tilde{x}$ and $\tilde{y}$ by the expressions $\tilde{x}' = \tilde{x} + \tilde{b}$ and $\tilde{y}' = \tilde{y} + \tilde{a}$, noting that they are distributed uniformly over $GF(2^n)$ when $\tilde{x}$ and $\tilde{y}$ are. This gives the quadratic form

$$(\tilde{x}' + \tilde{b})(\tilde{y}' + \tilde{a}) + \tilde{a}(\tilde{x}' + \tilde{b}) + \tilde{b}(\tilde{y}' + \tilde{a}) + \tilde{c}(\tilde{x}' + \tilde{b})^2 + \tilde{d}(\tilde{y} + \tilde{a})^2,$$

[3] When γ is the zero vector, the form can be computed by a $GF(2^n)$-linear function. See Theorem 4.5.

which opens up to $\tilde{p}'(\tilde{x}', \tilde{y}') = \tilde{x}' \cdot \tilde{y}' + \tilde{c} \cdot \tilde{x}'^2 + \tilde{d} \cdot \tilde{y}'^2 + (\tilde{a}\tilde{b} + \tilde{c}\tilde{b}^2 + \tilde{d}\tilde{a}^2)$. The free term $\tilde{a}\tilde{b} + \tilde{c}\tilde{b}^2 + \tilde{d}\tilde{a}^2$ can (as before) be accounted for by the final function f. This gives that a form $\langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle$ can be computed (respectively, approximated) from the polynomial $p(x, y)$ using a function f if and only if the form $\langle x', Ty' \rangle + \langle Ta + r, x' \rangle + \langle T^T b + s, y' \rangle$ can be computed (respectively, approximated) from the polynomial $p'(x', y')$ using a function f' defined as $f'(v) = f(v+k)+t$, where $k = M_a b + M_c S b + M_d S a$ and $t = \langle b, Ta \rangle + \langle r, b \rangle + \langle s, a \rangle$. Note also that f' is an affine function if and only if f is an affine function.

We now note that Fact 5.6 extends to an inequality for affine bilinear forms:

Fact 5.18. *For all matrices T and vectors $r, s \in \{0,1\}^n$:*

$$\Pr_{x,y}[\langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle = 0] \leq \frac{1}{2} + 2^{-\operatorname{rank}(T)-1}.$$

Proof. Since for every r it holds that $\Pr_y[Ty = r] \leq 2^{-\operatorname{rank}(T)}$,

$$\begin{aligned}
\Pr_{x,y}[\langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle = 0] &= \Pr_y[Ty = r] \cdot \Pr_{y:Ty=r}[\langle s, y \rangle = 0] \\
&\quad + \Pr_y[Ty \neq r] \cdot \mathop{\mathbb{E}}_{y:Ty \neq r} \left[\Pr_x[\langle x, Ty + r \rangle = \langle s, y \rangle] \right] \\
&\leq \Pr_y[Ty = r] \cdot 1 + (1 - \Pr_y[Ty = r]) \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr_y[Ty = r] \\
&\leq \frac{1}{2} + 2^{-\operatorname{rank}(T)-1}.
\end{aligned}$$

□

Thus, Proposition 5.8 can be extended to $GF(2^n)$-quadratic forms:

Proposition 5.19. *If an affine bilinear form $q(x, y) = \langle x, Ty \rangle + \langle r, x \rangle + \langle s, y \rangle$ is $(\frac{1}{2} + \frac{\varepsilon}{2})$-approximated by $f \circ p$ for a linear function f and any quadratic form $p(x, y) = M_x y + M_c S x + M_d S y$, then $T = Q_\gamma + E$ for some γ and some matrix E, such that* $\operatorname{rank}(E) \leq \log(1/\varepsilon)$.

Proof. Suppose that for some $f(v) = \langle \gamma, v \rangle$,

$$\Pr_{x,y}[f(p(x, y)) = q(x, y)] \geq \frac{1}{2} + \frac{\varepsilon}{2}.$$

Then using $\langle \gamma, M_x y \rangle = \langle x, Q_\gamma y \rangle$ we get:

$$\begin{aligned}\frac{1}{2}+\frac{\varepsilon}{2} &\leq \Pr_{x,y}[\langle \gamma, M_x y + M_c S x + M_d S y\rangle = \langle x, Ty\rangle + \langle r, x\rangle + \langle s, y\rangle] \\ &= \Pr_{x,y}[\langle x, Q_\gamma y\rangle + \left\langle S^T M_c^T \gamma, x\right\rangle + \left\langle S^T M_d^T \gamma, y\right\rangle = \langle x, Ty\rangle + \langle r, x\rangle + \langle s, y\rangle] \\ &= \Pr_{x,y}[\langle x, (Q_\gamma - T)y\rangle + \left\langle S^T M_c^T \gamma - r, x\right\rangle + \left\langle S^T M_d^T \gamma - s, y\right\rangle = 0].\end{aligned}$$

Setting $E = T - Q_\gamma, r' = S^T M_c^T \gamma - r$ and $s' = S^T M_d^T \gamma - s$, we get that

$$\frac{1}{2}+\frac{\varepsilon}{2} \leq \Pr_{x,y}[\langle x, Ey\rangle + \langle r', x\rangle + \langle s', y\rangle = 0] \leq \frac{1}{2} + \frac{1}{2}\cdot 2^{-\operatorname{rank}(E)}$$

(by Fact 5.18), so indeed $\operatorname{rank}(E) \leq \log(1/\varepsilon)$. □

However, Theorem 5.9 does not extend to quadratic forms just as easily, since its proof uses the fact that for every fixed nonzero value of $\tilde{x}$, the range of the restriction[4] $\tilde{p}|_{\tilde{x}}$ is the whole field $GF(2^n)$ (there, $\tilde{p}$ was called $\tilde{B}$). We can, however, characterize all polynomials $\tilde{p}$ that do satisfy this requirement:

Claim 5.20. *Let $\tilde{p}(\tilde{x}, \tilde{y}) = \tilde{x}\cdot\tilde{y} + \tilde{c}\cdot\tilde{x}^2 + \tilde{d}\cdot\tilde{y}^2$. The restriction $\tilde{p}|_{\tilde{x}}$ is invertible for every fixed $\tilde{x} \neq 0$, if and only if $\tilde{d} = 0$.*

Proof. First assume that indeed $\tilde{d} = 0$; then for every fixed nonzero $x \in GF(2^n)$, the restriction $\tilde{p}|_{\tilde{x}}$ is the affine nonconstant map $\tilde{p}|_{\tilde{x}}(\tilde{y}) = \tilde{x}\cdot\tilde{y} + \tilde{c}\cdot\tilde{x}^2$, which is invertible.

Now assume that $\tilde{d} \neq 0$. Then for every nonzero $\tilde{x}$, the restriction $\tilde{p}|_{\tilde{x}}$ is not invertible since the two vectors $\tilde{y}_1 = 0^n$ and $\tilde{y}_2 = \tilde{d}^{-1}\cdot\tilde{x}$ are mapped to the same image:

$$\tilde{p}|_{\tilde{x}}(\tilde{y}_1) = \tilde{x}\cdot 0 + \tilde{c}\cdot\tilde{x}^2 + \tilde{d}\cdot 0^2 = \tilde{c}\cdot\tilde{x}^2 = \tilde{x}\cdot(\tilde{d}^{-1}\cdot\tilde{x}) + \tilde{c}\cdot\tilde{x}^2 + \tilde{d}\cdot(\tilde{d}^{-1}\cdot\tilde{x})^2 = \tilde{p}|_{\tilde{x}}(\tilde{y}_2).$$

□

Instead of extending Theorem 5.9 to quadratic forms on these polynomials, and then deriving an analogue of Corollary 5.11 to get inapproximability results, we will prove directly that a form that can be approximated from a quadratic

[4]For fixed $\tilde{x} \in GF(2^n)$, the restriction $\tilde{p}|_{\tilde{x}}$ is defined as $\tilde{y} \mapsto \tilde{p}(\tilde{x}, \tilde{y})$. We also define the restriction $p|_x$ of the binary version of p analogously.

polynomial p satisfying the condition of Claim 5.20 (with any final function f) differs from some Q_γ by a matrix of low rank:[5]

Theorem 5.21. *For every $n \times n$ matrix T and two vectors $r, s \in \{0,1\}^n$, for every constant vector $c \in \{0,1\}^n$, and for every $\varepsilon \geq 2^{-\frac{n+1}{3}}$, if the form $q(x,y) = \langle x, Ty\rangle + \langle r, x\rangle + \langle s, y\rangle$ can be $(\frac{1}{2}+\varepsilon)$-approximated by $f \circ p$ for $p(x,y) = M_x y + M_c S x$ and some function $f : \{0,1\}^n \to \{0,1\}$, then $T = Q_\gamma + E$ for some $\gamma \in \{0,1\}^n$ and matrix E such that* $\mathrm{rank}(E) \leq 3\log(1/\varepsilon) + 1$.

Clearly, by symmetry the same would hold for any $p(x,y) = M_x y + M_d S y$.

Proof. We proceed similarly to the proof of Theorem 5.9, and begin by using a standard argument to get that if $\Pr_{x,y}[f(p(x,y)) = q(x,y)] \geq \frac{1}{2} + \varepsilon$, then for $\eta \geq \frac{\varepsilon}{1-\varepsilon} > \varepsilon$ of the x's, $\Pr_y[f(p|_x(y)) = q|_x(y)] \geq \frac{1}{2} + \frac{\varepsilon}{2}$. For every nonzero x, Claim 5.20 has that $p|_x$ is invertible; explicitly we can write $p|_x^{-1}(v) = M_x^{-1}(v - M_c S x)$. For every nonzero x we define $\gamma_x = {M_x^{-1}}^T(T^T x + s)$ and $k_x = \left\langle (M_c S)^T \gamma_x + r, x \right\rangle$; then

$$\begin{aligned}
\Pr_y[f(p|_x(y)) = q|_x(y)] &= \Pr_{v=p|_x(y)}[f(v) = q|_x(p|_x^{-1}(v))] \\
&= \Pr_{v\in\{0,1\}^n}\left[f(v) = \left\langle x, T\cdot p|_x^{-1}(v)\right\rangle + \langle r, x\rangle + \left\langle s, p|_x^{-1}(v)\right\rangle\right] \\
&= \Pr_v\left[f(v) = \left\langle x, T\cdot M_x^{-1}(v - M_c S x)\right\rangle + \langle r, x\rangle + \left\langle s, M_x^{-1}(v - M_c S x)\right\rangle\right] \\
&= \Pr_v\left[f(v) = \left\langle {M_x^{-1}}^T(T^T x + s), v\right\rangle + \left\langle (M_c S)^T \cdot {M_x^{-1}}^T(T^T x + s) + r, x\right\rangle\right] \\
&= \Pr_v[f(v) = \langle \gamma_x, v\rangle + k_x].
\end{aligned}$$

So, for at least $\eta \cdot 2^n - 1$ nonzero x's, f agrees with $\ell_x = v \mapsto \langle \gamma_x, v\rangle + k_x$ on at least $\frac{1}{2} + \frac{\varepsilon}{2}$ of the space $\{0,1\}^n$. Using Fact 5.10, an averaging argument gives as before the existence of some (affine) function $\ell = v \mapsto \langle \gamma, v\rangle + k$ such that $\Pr_{x\neq 0}[\ell_x = \ell] \geq \frac{\varepsilon^3}{1-\varepsilon} - 2^{-n-2} \geq \varepsilon^3 - \varepsilon^3/2 = \varepsilon^3/2$, using $\varepsilon \geq 2^{-\frac{n+1}{3}}$. Certainly, $\Pr_{x\neq 0}[\gamma_x = \gamma] \geq \Pr_{x\neq 0}[\ell_x = \ell] \geq \varepsilon^3/2$. We analyse the definition of γ_x, and observe that for any nonzero x, the condition $\gamma = \gamma_x = {M_x^{-1}}^T(T^T x + s)$ is equivalent to $T^T x + s = M_x^T \gamma$. Using the notations of Subsection 4.3.3 and the explicit definition of Q_γ given there, and using Claim 4.13, this condition is

[5] The reason for this is that extending the proof of Theorem 5.9, as started in the proof of Theorem 5.21, gives and *affine*, rather than *linear*, function ℓ such that $\ell_x = \ell$ with high probability. Thus the Equation (5.1) in the proof of Theorem 5.9 does not hold (as two different affine functions ℓ and ℓ_x can disagree on all their domain), giving that ℓ does not necessarily approximate T from p.

equivalent to $T^T x + s = U^{-1} M_x U\gamma = U^{-1} M_{U\gamma} x = Q_\gamma x$. Thus, $\gamma_x = \gamma$ if and only if $(T^T - Q_\gamma)x = s$. Let $E^T = T^T - Q_\gamma$; since $\Pr_x[E^T x = s] \leq 2^{-\operatorname{rank}(E^T)}$ holds for every s and matrix E^T, and since $\Pr_{x \neq 0}[\gamma_x = \gamma] \geq \varepsilon^3/2$, we get that $2^{-\operatorname{rank}(E^T)} \geq \varepsilon^3/2$ and thus $\operatorname{rank}(E^T) \leq 3\log(1/\varepsilon) + 1$. Setting $E = E^{T^T}$, we get by definition that $T = (Q_\gamma + E^T)^T = Q_\gamma^T + E = Q_\gamma + E$ (by symmetry of the matrix Q_γ, see remark after Lemma 2.11), as wanted. □

Reflection. The vector γ_x was defined (both in the proof of Theorem 5.9 and in that of Theorem 5.21) so that if $f(p(x,y)) = q(x,y)$ with high probability (over the choice of y) then f has high agreement with $\ell_x = \langle \gamma_x, \cdot \rangle$ (plus k_x in Theorem 5.21). The argument in both cases is based on analyzing when many γ_x's "collide", where in the proof of Theorem 5.9 we conclude that a linear function $\ell = \langle \gamma, \cdot \rangle$ approximates T, whereas in Theorem 5.21 we show directly that $T = Q_\gamma + E$ for some E of low rank. We get that the approximability of a form using the matrix T depends on the size of collisions of the function $\gamma(x) = M_x^{-1^T} T^T x$: a good approximation implies an image γ with many preimages x.

By Theorem 5.21 we get that there are functions in $\mathcal{B}_2^{(2)}$ (with T differing from every Q_γ by a matrix of high rank, e.g., the matrices T considered in Propositions 5.12 and 5.14) for which if a function from $\mathcal{P}_2^{(2)}$ approximates it with a noticeable advantage over $\frac{1}{2}$, then it must use some polynomial $\tilde{p}(\tilde{x}, \tilde{y}) = \tilde{x} \cdot \tilde{y} + \tilde{a} \cdot \tilde{x} + \tilde{b} \cdot \tilde{y} + \tilde{c} \cdot \tilde{x}^2 + \tilde{d} \cdot \tilde{y}^2$ in which both $\tilde{c}$ and $\tilde{d}$ are nonzero.

5.4 Multi-$GF(2)$-bilinear functions vs. $GF(2)$-quadratic functions

When trying to show that $GF(2)$-quadratic functions are stronger than $GF(2^n)$-quadratic functions, we have so far only used $GF(2)$-bilinear functions. That is, when dividing the input to ℓ blocks of n bits (now also considering $\ell > 2$), we never multiplied two bits from the same block. However, it seems plausible that this limitation causes a substantial loss of power against $GF(2^n)$-quadratic polynomials, since when looking at the binary version of a $GF(2^n)$-quadratic polynomial, each bit is a $GF(2)$-bilinear function, i.e. has no multiplication of two

bits from the same block.[6] It thus seems that one can find a $GF(2)$-quadratic polynomial that cannot be calculated/approximated by $GF(2^n)$-quadratic polynomials by using the extra power of multiplying bits within the same block. That is, perhaps the class $\mathcal{B}_2^{(\ell)}$ is stronger that the class $\mathcal{P}_2^{(\ell)}$ simply because it is stronger than the larger class of functions resulting from n separate $GF(2)$-bilinear functions (i.e., functions in $\mathcal{B}_{1,1}^{(\ell)}$), finally mapped to $\{0,1\}$ using some arbitrary function.

In this section we give a partial negative answer: we show that some polynomials, namely the polynomials that are symmetric in the blocks (or can be transformed to such), *can* be somewhat approximated, and sometimes even computed exactly, by $GF(2)$-bilinear functions with a final function.

5.4.1 Definition of the model

We define the above class of functions formally:

Definition 5.22. *We call a function* $q : \{0,1\}^{\ell \cdot n} \to \{0,1\}$ multi-$GF(2)$-bilinear *if there exist* n *functions* $p_1, \ldots, p_n \in \mathcal{B}_{1,1}^{(\ell)}$ *and an arbitrary function* $f : \{0,1\}^n \to \{0,1\}$ *such that* $q = f(p_1, \ldots, p_n)$. *We denote the class of multi-*$GF(2)$*-bilinear functions by* $\mathcal{M}_{1,1}^{(\ell)}$.

As noted above, $\mathcal{P}_2^{(\ell)} \subseteq \mathcal{M}_{1,1}^{(\ell)}$ follows from Lemma 4.4 and from the squaring operation over $GF(2^n)$ being a linear function in the bits of the representation (see the definition of the matrix S in Notation 2.13).

We will also use a convenient decomposition of functions in $\mathcal{B}_2^{(\ell)}$, since we are interested in the difference between multiplications of bits within the same block and multiplications of bits from different blocks: we parse the $n \cdot \ell$ input bits to blocks $x^{(1)}, \ldots, x^{(\ell)}$, and write each $q \in \mathcal{B}_2^{(\ell)}$ as

$$q(x^{(1)}, \ldots, x^{(\ell)}) = q^*(x^{(1)}, \ldots, x^{(\ell)}) + \sum_{j=1}^{\ell} q_j(x^{(j)}), \tag{5.2}$$

where $q^* \in \mathcal{B}_{1,1}^{(\ell)}$ (i.e., does not multiply bits from the same block) and every q_j is a quadratic function over $\{0,1\}^n$. We also write explicitly $q_j(x) = \langle x, E_j x \rangle$.

[6] Recall that squaring in $GF(2^n)$ is a linear operation over the bits of the representation (see Notation 2.13).

Definition 5.23. *We say that a $GF(2)$-quadratic polynomial q is* block-symmetric *if, when decomposing q as in Equation (5.2), all the functions q_j are equal.*

Note that this is a weaker requirement than the polynomial q being really invariant to permuting the blocks, since we do not require that the $GF(2)$-bilinear part q^* be symmetric as well.

5.4.2 Computing block-symmetric $GF(2)$-quadratic functions from $n+1$ bilinear functions

While Definition 5.22 only allows n separate $GF(2)$-bilinear functions, we will first show how to compute block-symmetric $GF(2)$-quadratic functions from $n+1$ separate $GF(2)$-bilinear functions.

Proposition 5.24. *For every block-symmetric $GF(2)$-quadratic function $q : \{0,1\}^{\ell \cdot n} \to \{0,1\}$, there exist $n+1$ $GF(2)$-bilinear functions $p_1, \ldots, p_\ell, p^* \in \mathcal{B}_{1,1}^{(\ell)}$ and a function $f : \{0,1\}^{n+1} \to \{0,1\}$ such that $q = f(p_1, \ldots, p_n, p^*)$.*

Proof. Decompose q as in Equation (5.2), and let E be the matrix describing the common quadratic function (that is, $q_j(x) = \langle x, Ex \rangle$ for every $j \in \{1, \ldots, \ell\}$). For $i = 1, \ldots, n$, we set $p_i(x^{(1)}, \ldots, x^{(\ell)}) = \sum_{j=1}^{\ell} x_i^{(j)}$, which is the sum of the bits in the i-th channel. We also set

$$p^*(x^{(1)}, \ldots, x^{(\ell)}) = q^*(x^{(1)}, \ldots, x^{(\ell)}) + \sum_{j_1 \neq j_2} \left\langle x^{(j_1)}, E \cdot x^{(j_2)} \right\rangle .$$

Note that while all p_i's are in fact linear, the last function p^* is $GF(2)$-bilinear since q^* is $GF(2)$-bilinear and in the sum we are only taking inner products between different blocks $j_1 \neq j_2$. Finally, we define f as follows: on input $n+1$ bits $b_1, \ldots, b_n, b^*$, define the vector $\bar{b} = (b_1, \ldots, b_n)$ and output $b^* + \langle \bar{b}, E \cdot \bar{b} \rangle = b^* + \sum_{i,k} E_{i,k} \cdot b_i \cdot b_k$. It is left to verify that for all $\bar{x} = (x^{(1)}, \ldots, x^{(\ell)}) \in \{0,1\}^{\ell \cdot n}$:

$$\begin{aligned}
f(p_1(\bar{x}), \ldots, p_n(\bar{x}), p^*(\bar{x})) &= p^*(\bar{x}) + \sum_{i,k} E_{ik} \cdot p_i(\bar{x}) \cdot p_k(\bar{x}) \\
&= q^*(\bar{x}) + \sum_{j_1 \neq j_2} \left\langle x^{(j_1)}, E \cdot x^{(j_2)} \right\rangle + \sum_{i,k} E_{ik} \cdot p_i(\bar{x}) \cdot p_k(\bar{x}) \\
&= q^*(\bar{x}) + \sum_{j_1 \neq j_2} \sum_{i,k} E_{ik} \cdot x_i^{(j_1)} \cdot x_k^{(j_2)} \\
&\quad + \sum_{i,k} E_{ik} \cdot \left(\sum_{j_1} x_i^{(j_1)} \right) \cdot \left(\sum_{j_2} x_k^{(j_2)} \right)
\end{aligned}$$

$$
\begin{aligned}
&= q^*(\bar{x}) + \sum_j \sum_{i,k} E_{ik} \cdot x_i^{(j)} \cdot x_k^{(j)} \\
&= q^*(\bar{x}) + \sum_j \left\langle x^{(j)}, E \cdot x^{(j)} \right\rangle \\
&= q(\bar{x}).
\end{aligned}
$$

□

5.4.3 Losing one p_i

Since we can only afford n functions to comply with Definition 5.22, we need to get rid of one function p_i to accommodate the extra polynomial p^*. We first note that if the function $\langle x, Ex\rangle$ does not depend on all the bits of x, e.g. it is invariant to the i-th bit, then the polynomial q can be computed exactly by a multi-$GF(2)$-bilinear polynomial (by just not taking p_i).

Otherwise, we take some i such that the variable x_i appears in a nonlinear term in $\langle x, Ex\rangle$, i.e., some i such that some off-diagonal entry in the i-th row or column of E is set.[7] We assume without loss of generality that $i = 1$, and decompose $\langle x, Ex\rangle$ to $x_1 \cdot \ell(x_2, \ldots, x_\ell) + g(x_2, \ldots, x_\ell)$ where ℓ is a nonconstant affine function and g is some quadratic function. We then set E' to describe the quadratic function g (that is, fix the first bit to zero, or as a matrix, zero the first row and column in E). This yields:

$$
\begin{aligned}
\Pr_x[\langle x, Ex\rangle \neq \langle x, E'x\rangle] &= \Pr_{x_1}[x_1 = 0] \cdot \Pr_{x_2,\ldots,x_\ell}[0 \cdot \ell(x_2, \ldots, x_\ell) + g(x_2, \ldots, x_\ell) \neq g(x_2, \ldots, x_\ell)] \\
&\quad + \Pr_{x_1}[x_1 = 1] \Pr_{x_2,\ldots,x_\ell}[1 \cdot \ell(x_2, \ldots, x_\ell) + g(x_2, \ldots, x_\ell) \neq g(x_2, \ldots, x_\ell)] \\
&= \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{4}
\end{aligned}
$$

(since the nonconstant multivariate affine function ℓ takes value 1 with probability exactly $\frac{1}{2}$). Since $\langle x, Ex\rangle$ can be $\frac{3}{4}$-approximated by $\langle x, E'x\rangle$ that does not depend on its first bit, we can calculate only $p_2(\bar{x}), \ldots, p_n(\bar{x})$ and $p^*(\bar{x})$, and set our new $f' : \{0,1\}^n \to \{0,1\}$ to fix b_1 to zero (that is, on input $b_2, \ldots, b_n, b^*$,

[7] if no such i exists, $\langle x, Ex\rangle$ is linear in x, and q does not multiply bits from the same block.

output $b^* + \langle \bar{b}, E' \cdot \bar{b} \rangle = b^* + \sum_{i=2}^{n} \sum_{k=2}^{n} E_{ik} \cdot b_i \cdot b_k$). Then

$$\begin{aligned}
\Pr_{\bar{x}}[q(\bar{x}) \neq f'(p_2(\bar{x}), \quad \ldots, \quad & p_n(\bar{x}), p^*(\bar{x}))] \\
&= \Pr_{\bar{x}}[f(p_1(\bar{x}), \ldots, p_n(\bar{x}), p^*(\bar{x})) \neq f'(p_2(\bar{x}), \ldots, p_n(\bar{x}), p^*(\bar{x}))] \\
&= \Pr_{\bar{b}=(p_1(\bar{x}),\ldots,p_n(\bar{x}))}[\langle \bar{b}, E \cdot \bar{b}) \rangle \neq \langle \bar{b}, E' \cdot \bar{b} \rangle] \\
&= \frac{1}{4},
\end{aligned}$$

since the p_i's are all linear and thus $\bar{b}$ is uniformly distributed when $\bar{x}$ is uniformly distributed.

We have obtained:

Corollary 5.25. *Every block-symmetric $q \in \mathcal{B}_2^{(\ell)}$ can be $\frac{3}{4}$-approximated by a function in $\mathcal{M}_{1,1}^{(\ell)}$.*

5.4.4 Different linear combinations of the blocks

Suppose that, in contrast to the case of Definition 5.22, the q_j's are not all equal, but can nevertheless be described as the same quadratic form applied to different linear combinations in the bits of each block. That is, there exists a matrix E and ℓ matrices P_j such that for all j, the from $q_j(x)$ (from Equation (5.2)) equals $\langle P_j x, E \cdot P_j x \rangle$. Then, the same method above still works: we define each $p_i(\bar{x})$ as $\sum_j e_{i-1}^T P_j x^{(j)}$ and the extra polynomial $p^*(\bar{x})$ as $q^*(\bar{x}) + \sum_{j_1 \neq j_2} \langle P_{j_1} x^{(j_1)}, E \cdot P_{j_2} x^{(j_2)} \rangle$. The same function f will then give $f(p_1(\bar{x}), \ldots, p_n(\bar{x}), p^*(\bar{x})) = q(\bar{x})$. We can thus still compute q from $n+1$ bilinear functions, which means as before that n functions allow us to either $\frac{3}{4}$-approximate q, or compute it exactly if E does not depend on one of the bits.

This generalization covers, in specific, the case that the different E_j's describe isomorphic graphs (only differ in the order of the rows and columns). This corresponds to each P_j being a permutation matrix.

Bibliography

[AGHP] N. Alon, O. Goldreich, J. Hastad and R. Peralta, "Simple Constructions of Almost k-wise Independent Random Variables", *Random Structures and Algorithms*, vol. 3, pp. 289–304, 1992.

[AR] N. Alon and Y. Roichman, "Random Cayley Graphs and Expanders", *Random Structures and Algorithms*, vol. 5, pp. 271–284, 1994.

[ASS] N. Alon, O. Schwartz and A. Shapira, "An Elementary Construction of Constant-Degree Expanders", In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[BM] G. Birkhoff and S. MacLane, *A Survey of Modern Algebra*, third edition, MacMillan, New York, 1965.

[Bog] A. Bogdanov, "Pseudorandom generators for low degree polynomials", In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pp. 21–30, 2005.

[BSSVW] 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 35th Annual ACM Symposium on Theory of Computing*, pp. 612-621, New York, 2003.

[BV] A. Bogdanov and E. Viola, "Pseudorandom bits for polynomials", In *48th Annual Symposium on Foundations of Computer Science*, pp. 41–51, 2007.

[Dym] H. Dym, *Linear Algebra in Action*, AMS Bookstore, 2007.

[Eve] G. Even, "Construction of Small Probabilistic Spaces for Deterministic Simulation", M.Sc. (in computer science) thesis, submitted to the Senate of the Technion (Israel Intitute of Technology) in Aug. 1991.

[Gol95] O. Goldreich, "Three XOR-Lemmas - an Exposition", *Electronic Colloquium on Computational Complexity*, TR 95-050, 1995.

[Gol08] O. Goldreich, *Computational Complexity: A Conceptural Perspective*, Cambridge University Press, 2008.

[GRS] O. Goldreich, R. Rubinfeld and M. Sudan, "Learning polynomials with queries: The highly noisy case", In *36th Annual Symposium on Foundations of Computer Science*, pp. 294, 1995.

[GW] O. Goldreich and A. Wigderson, "Tiny Families of Functions with Random Properties: A Quality–Size Trade–off for Hashing", *Journal of Random structures and Algorithms*, vol. 11, No. 4, pp. 315–343, 1997.

[Lin] J. H. Van Lint, *Introduction to Coding Theory*, Springer-Verlag New York, Inc., Secaucus, NJ, 1998

[Lov] S. Lovett, "Unconditional pseudorandom generators for low degree polynomials", In *40th Annual Symposium on the Theory of Computing*, pp. 557–562, 2008.

[LT] S. Lovett and Y. Tzur, "Explicit lower bound for fooling polynomials by the sum of small-bias generators", *Electronic Colloquium on Computational Complexity*, TR 09-088, 2009.

[LVW] M. Luby, B. Velickovic and A. Wigderson, "Deterministic approximate counting of depth-2 circuits", In *Proceedings of the 2nd Israeli Symposium on Theoretical Computer Science*, pp. 18–24, 1993.

[MST] E. Mossel, A. Shpilka and L. Trevisan, "On ε-Biased Generators in NC^0", In *44th Annual Symposium on Foundations of Computer Science*, pp. 136, 2003.

[NN] J. Naor and M. Naor, "Small-Bias Probability Spaces: Efficient Constructions and Applications", *SIAM Journal on Computing*, vol. 22, pp. 838–856, 1993.

[Nis] N. Nisan, "Pseudorandom generators for space-bounded computations", In *Proceedings of the 22nd annual ACM Symposium on Theory of Computing*, pp. 204–212, 1990.

[Tzu] Y. Tzur, "$GF(2^n)$-Linear Tests versus $GF(2)$-Linear Tests", *Electronic Colloquium on Computational Complexity*, TR 09-018, 2009.

[Vio] E. Viola, "The sum of d small-bias generators fools polynomials of degree d", In *23rd IEEE Conference on Computational Complexity*, 2008.

Appendix A

Small parameter improvements

A.1 A better stretch bound for Section 3.2

The bound of Corollary 3.2 can be slightly improved due to the linear dependence between the bits representing an element $\tilde{x}$ and the bits representing $\tilde{x}^2$, presented in Notation 2.13. The proof of Theorem 3.1 bounds the number ℓ of output blocks by the number $(d+1)^k$ of possible monomials of individual degree d in k variables, since the rows of the $\ell \times (d+1)^k$ matrix of coefficients (in $GF(2^n)$) describing the output blocks, should not be linearly dependent.

When trying to avoid linear dependencies between bits, as in Corollary 3.2, we can further restrict the matrix. As a motivating example, consider the case $d = 2, k = 1$. The bound $(d+1)^k - 1$ of Corollary 3.2 disregarding the constant monomial (see footnote 2 in page 21), has $\ell \leq 2$. However, the two possible nonconstant monomials of degree 2 in one variable $\tilde{x}$, namely, $\tilde{x}$ and $\tilde{x}^2$, have a linear dependency between their representing bits, described by the matrix S defined in Notation 2.13.

In general, the $(\lfloor d/2 \rfloor + 1)^k$ monomials $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$ of individual degree at most $\lfloor d/2 \rfloor$ all have their square $\tilde{m}^2(\tilde{x}_1, \ldots, \tilde{x}_k)$ of individual degree at most d, and thus should be removed. Our general improved bound is:

Theorem A.1. *Consider any $GF(2^n)$-polynomial map $\tilde{G} : GF(2^n)^k \to GF(2^n)^\ell$ of individual degree d, and let $G : \{0,1\}^{kn} \to \{0,1\}^{\ell n}$ be its binary version.*

If $\ell > (d+1)^k - (\lfloor d/2 \rfloor + 1)^k$, then there is a nontrivial $GF(2)$-linear combination that takes value zero on every output of G. Thus, G is not an ε-bias generator for any $\varepsilon < \frac{1}{2}$.

Proof. Similarly to the proof of Theorem 3.1, we will describe the generator G by a matrix, with each row describing an output block as a linear combination of the monomials in the polynomial expression defining it. However, this time we will not work with the $\ell \times ((d+1)^k - 1)$ matrix[1] of $GF(2^n)$-coefficients describing $\tilde{G}$, but rather with the $n\ell \times n((d+1)^k - 1)$ matrix of bits, describing the binary version G directly. The latter matrix can be obtained from from the former matrix by replacing each $GF(2^n)$-element $\tilde{a}$ with with the $n \times n$ matrix M_a (defined in Notation 2.7). Indeed, the $(jn+i)$-th row of this matrix describes the $(jn+i)$-th output bit of G, which is the i-th bit of the representation of the j-th output block of $\tilde{G}$. This output bit is described as a $GF(2)$-linear combination of the $n(d+1)^k$ variables, which are the n bits representing each of the $(d+1)^k - 1$ possible nonconstant monomials of individual degree at most d (in the k input variables). In general, a mapping is described by such a bit matrix in the sense that if the $n \times n$ matrix C is the intersection of the n rows corresponding to the j-th output block with the n columns corresponding to the n bits of the monomial $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$, then the $(nj+i)$-th output bit of the generator will be $\langle e_i, C \cdot m(x_1, \ldots, x_k) \rangle$.

Using this bit-matrix, we will produce a matrix that describes an equivalent generator (that is, describes the same mapping G), but has only $n \cdot \left((d+1)^k - (\lfloor d/2 \rfloor + 1)^k\right)$ columns. Thus, if $\ell > (d+1)^k - (\lfloor d/2 \rfloor + 1)^k$, this matrix has more rows than columns and must have linear dependent rows, giving a fixed linear dependence satisfied by the bits of every output of G.

For every nonconstant monomial $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$ of individual degree $\lfloor d/2 \rfloor$, the monomial $\tilde{m}^2(\tilde{x}_1, \ldots, \tilde{x}_k)$ has individual degree at most d (but strictly greater than the individual degree of $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$). We go over these $(\lfloor d/2 \rfloor + 1)^k - 1$ monomials, in order of increasing (maximal individual) degree, and for every such monomial $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$ we emulate the n columns corresponding to the bits of $\tilde{m}(\tilde{x}_1, \ldots, \tilde{x}_k)$ by an adequate addition to the columns corresponding to the bits of $\tilde{m}^2(\tilde{x}_1, \ldots, \tilde{x}_k)$: for every n rows corresponding to each output block of G, intersecting with the above n columns by the $n \times n$ matrix B (which

[1] We have already incorporated the improvement of removing the constant coefficient, presented by footnote 2 in page 21.

indeed describes the matrix-coefficient of the monomial $m(x_1,\ldots,x_k)$ in the expression for this output block), we add the matrix $B \cdot S^{-1}$ to the matrix in the intersection of these n rows with the n columns corresponding to the bits of the monomial $\tilde{m}^2(\tilde{x}_1,\ldots,\tilde{x}_k)$, with S being the squaring matrix defined in Notation 2.13. After we are done with the monomial $\tilde{m}(\tilde{x}_1,\ldots,\tilde{x}_k)$, we remove the n columns representing its bits from the matrix. Every such step preserves the mapping described by the matrix, as by definition of the matrix S, the i-th output bit of G will be, for every two $n \times n$ matrices B and C:

$$\langle e_i, B \cdot m(x_1,\ldots,x_k)\rangle + \langle e_i, C \cdot m^2(x_1,\ldots,x_k)\rangle = \left\langle e_i, (BS^{-1}+C) \cdot m^2(x_1,\ldots,x_k)\right\rangle.$$

Since there are $(\lfloor d/2 \rfloor + 1)^k - 1$ nonconstant monomials of individual degree at most $\lfloor d/2 \rfloor$, each leading to the removal of n columns, and since we begin with a matrix with $n \cdot ((d+1)^k - 1)$ columns - we end up with a matrix with $n \cdot \left((d+1)^k - (\lfloor d/2 \rfloor + 1)^k\right)$ columns, as wanted. Thus, a stretch ℓ satisfying the hypothesis of the theorem necessarily implies linear dependent rows for the describing matrix, giving a nontrivial linear dependence between the output bits of the generator. □

A.2 A better approximation rate for Subsection 5.3.3

The reduction of Theorem 5.9 can be made (asymptotically) more efficient:

Theorem A.2. *If a bilinear form T can be $(\frac{1}{2}+\varepsilon)$-approximated by some function $f \circ B$, then it can be $(\frac{1}{2}+\frac{\delta}{2})$-approximated by a* linear *function for $\delta \geq \frac{\varepsilon^2}{4(\log(1/\varepsilon)+2)} - 2^{-n}$.*

Proof. Define as before $A(x) = \Pr_y[f(M_x y) = \langle x, Ty\rangle] - \frac{1}{2}$, so that $\mathbb{E}_x[A(x)] \geq \varepsilon$. Set $\ell = \lceil \log(1/\varepsilon) \rceil$ and partition the x's to $\ell + 2$ bins, defining

$$B_i = \{x \in \{0,1\}^n : 2^{-i-1} < A(x) \leq 2^{-i}\}$$

for $i = 0,\ldots,\ell$, and the remaining x's, $C = \{x : A(x) \leq 2^{-\ell-1}\}$, and note that

$2^{-\ell-1} \leq \frac{\varepsilon}{2}$. Define $p_i = \sum_{x \in B_i} A(x)$, and write

$$\begin{aligned}
\varepsilon \cdot 2^n &\leq \mathbb{E}_x[A(x)] \cdot 2^n \\
&= \sum_{x \in \{0,1\}^n} A(x) \\
&= \sum_{i=0}^{\ell} \sum_{x \in B_i} A(x) + \sum_{x \in C} A(x) \\
&\leq \sum_{i=0}^{\ell} p_i + 2^n \cdot \frac{\varepsilon}{2}.
\end{aligned}$$

We get $\sum_{i=0}^{\ell} p_i \geq \frac{\varepsilon}{2} \cdot 2^n$, so there must exist some i for which $p_i \geq \frac{\varepsilon}{2(\ell+1)} \cdot 2^n$. Since for this i, all x's in B_i have $A(x) \leq 2^{-i}$, and since $\sum_{x \in B_i} A(x) \geq \frac{\varepsilon}{2(\ell+1)} \cdot 2^n$, the set B_i must contain at least $b = \frac{\varepsilon}{2(\ell+1)} \cdot 2^{n+i}$ elements. Possibly excluding 0, we have that for at least $b-1$ nonzero x's, the functions f and ℓ_x have agreement at least $\frac{1}{2} + 2^{-i-1}$. Using Fact 5.10, these x's match at most $\frac{1}{(2^{-i})^2} = 2^{2i}$ distinct linear functions. By an averaging argument, there exists some linear function that is matched by at least $\frac{b-1}{2^{2i}}$ different nonzero x's, i.e. there exists $\gamma \in \{0,1\}^n$ such that

$$\delta = \Pr_{x \neq 0}[\gamma_x = \gamma] \geq \frac{b-1}{2^{2i}} \cdot \frac{1}{2^n - 1} \geq \frac{\varepsilon \cdot 2^i}{2(\ell+1) \cdot 2^{2i}} - 2^{2i-n} \geq \frac{\varepsilon^2}{4(\log(1/\varepsilon)+2)} - 2^{-n},$$

where the last inequality used $0 \leq i \leq \ell < \log(1/\varepsilon) + 1$. The rest of the proof proceeds as before. □

This, in turn, improves Corollary 5.11 to

Corollary A.3. *If a matrix T can be $(\frac{1}{2} + \varepsilon)$-approximated (by any function $f \circ B$) for any $\varepsilon \geq 2^{-\frac{n+5}{3}}$, then $T = Q_\gamma + E$ for some γ and some matrix E with* $\operatorname{rank}(E) \leq 2\log(1/\varepsilon) + 3 + \log(\log(1/\varepsilon) + 2)$ *(which is $(2 + o(1))\log(1/\varepsilon)$ if ε is subconstant).*

Proof. By Theorem A.2, T can be $(\frac{1}{2} + \frac{\delta}{2})$-approximated by a linear function for $\delta \geq \frac{\varepsilon^2}{4(\log(1/\varepsilon)+2)} - 2^{-n}$. Since $\varepsilon \geq 2^{-\frac{n+5}{3}}$, we have $2^{-n} \leq \frac{1}{32}\varepsilon^3 = \frac{\varepsilon^2}{8 \cdot (4/\varepsilon)} \leq \frac{1}{2} \cdot \frac{\varepsilon^2}{4(\log(1/\varepsilon)+2)}$, so $\delta \geq \frac{\varepsilon^2}{8(\log(1/\varepsilon)+2)}$. By Proposition 5.8, $T = Q_\gamma + E$ with $\operatorname{rank}(E) \leq \log(1/\delta) \leq 2\log(1/\varepsilon) + 3 + \log(\log(1/\varepsilon) + 2)$.

□

Appendix B

Agreement with affine functions

We prove that for $0 < \varepsilon \leq 1$, every function $f : \{0,1\}^n \to \{0,1\}$ can $(\frac{1}{2} + \frac{\varepsilon}{2})$-agree with at most $\frac{1}{\varepsilon^2}$ affine functions.

We identify functions from $\{0,1\}^n$ to $\{0,1\}$ with vectors in $\{1,-1\}^{2^n} \subseteq \mathbb{R}^{2^n}$, by writing their full truth table, $[v_f]_i = -1^{f(b(i))}$ where $b(i) \in \{0,1\}^n$ is the binary representation of i. We define the inner product between two functions f, g as

$$\langle v_f, v_g \rangle = \mathbb{E}_{b \in \{0,1\}^n}[-1^{f(b)+g(b)}] = 2 \cdot \Pr_{b \in \{0,1\}^n}[f(b) = g(b)] - 1.$$

Then, for every function f, we have $\langle v_f, v_f \rangle = 1$.

The set $\mathcal{L} = \{u_\gamma\}_{\gamma \in \{0,1\}^n}$ of linear functions ($u_\gamma(x) = \langle \gamma, x \rangle$) can be shown to be an orthonormal basis of $\mathbb{R}^{2^n}$ with respect to this inner product. We can thus write for every f that $v_f = \sum_{\gamma \in \{0,1\}^n} \langle v_f, u_\gamma \rangle \cdot u_\gamma$.

This way we get:

$$\begin{aligned} 1 = \langle v_f, v_f \rangle &= \left\langle \sum_{\gamma \in \{0,1\}^n} \langle v_f, u_\gamma \rangle \cdot u_\gamma, \sum_{\gamma' \in \{0,1\}^n} \langle v_f, u_{\gamma'} \rangle \cdot u_{\gamma'} \right\rangle \\ &= \sum_{\gamma, \gamma' \in \{0,1\}^n} \langle v_f, u_\gamma \rangle \cdot \langle v_f, u_{\gamma'} \rangle \cdot \langle u_\gamma, u_{\gamma'} \rangle \\ &= \sum_{\gamma \in \{0,1\}^n} \langle v_f, u_\gamma \rangle^2, \end{aligned} \tag{B.1}$$

where the last equality follows from the orthonormality of $\mathcal{L}$.

Finally, we note that if f has agreement $\frac{1}{2}+\frac{\varepsilon}{2}$ with k different affine functions, then for k different γ's, either $\Pr_{b\in\{0,1\}^n}[f(b) = \langle\gamma, b\rangle] \geq \frac{1}{2}+\frac{\varepsilon}{2}$ or $\Pr_{b\in\{0,1\}^n}[f(b) = \langle\gamma, b\rangle] \leq \frac{1}{2}-\frac{\varepsilon}{2}$, implying that $\langle v_f, u_\gamma\rangle^2 \geq \varepsilon^2$. By Equation (B.1), we get $k \cdot \varepsilon^2 \leq 1$, giving $k \leq 1/\varepsilon^2$ as wanted.