Sum-of-Squares Meets Program Obfuscation, Revisited

Boaz Barak*, Samuel B. Hopkins**, Aayush Jain***, Pravesh Kothari†, and Amit Sahai‡.

Abstract We develop attacks on the security of variants of pseudorandom generators computed by quadratic polynomials. In particular we give a general condition for breaking the one-way property of mappings where every output is a quadratic polynomial (over the reals) of the input. As a corollary, we break the degree-2 candidates for security assumptions recently proposed for constructing indistinguishability obfuscation by Ananth, Jain and Sahai (ePrint 2018) and Agrawal (ePrint 2018). We present conjectures that would imply our attacks extend to a wider variety of instances, and in particular offer experimental evidence that they break assumption of Lin-Matt (ePrint 2018).

Our algorithms use semidefinite programming, and in particular, results on low-rank recovery (Recht, Fazel, Parrilo 2007) and matrix completion (Gross 2009).

* Harvard University. b@boazbarak.org. Supported by NSF awards CCF 1565264 and CNS 1618026 and a Simons Investigator Fellowship.

** University of California, Berkeley. hopkins@berkeley.edu. Supported by a Miller Postdoctoral Fellowship and NSF award CCF 1408673.

*** University of California, Los Angeles. aayushjain1728@gmail.com. Research supported from grants listed under Amit Sahai and a Google PhD fellowship award in Privacy and Security.

† Princeton University and the Institute for Advanced Study. kothari@cs.princeton.edu. Partially supported by Ma fellowship from the Schmidt Foundation and Avi Wigderson's NSF award CCF-1412958.

‡ University of California, Los Angeles. sahai@cs.ucla.edu. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, and NSF grant 1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, the U.S. Government or Google.

1 Introduction

In this work, we initiate the algorithmic study of cryptographic hardness that may exist in general expanding families of low-degree polynomials over $\mathbb{R}$. As a result, we obtain strong attacks on certain pseudorandom generators whose output is a "simple" function of the input. Such "simple" pseudorandom generators are interesting in their own right, but have recently become particularly important because of their role in candidate constructions for *Indistinguishabily Obfuscators*.

The question of whether Indistinguishabily Obfuscators (iO) exist is one of the most consequential open questions in cryptography. On one hand, a sequence of works [14,29] has shown that iO, if it exists, would imply a huge variety of cryptographic objects, several of which we know of no other way to achieve. On the other hand, the current candidate constructions for iO's are not based on well-studied standard assumptions, and there have been several attacks on several iO constructions as well as underlying primitives.

A promising line of works [19,23,3,20,22] has aimed at basing iOs on more standard assumptions, and in particular Lin and Tessaro [22] reduced constructing iO to the combination following three assumptions:

1. The *learning with errors (LWE)* assumption.
2. Existence of *three local* pseudorandom generators with sufficiently large super linear stretch. These are pseudorandom generators $G : \{0,1\}^n \to \{0,1\}^{n^{1+\varepsilon}}$ (for arbitrarily small $\varepsilon > 0$) such that if we think of the input as split into n/k blocks of length k each (for some $k = n^{o(1)}$) then every output of G depends on at most three blocks of the input.
3. Existence of *trilinear maps* satisfying certain strengthening of the Decisional-Diffie-Hellman assumption.

Of the three assumptions, the learning with errors assumption is well studied and widely believed. The existence of local pseudorandom generators has also been recently extensively studied; it also relates to questions on random constraint satisfaction problems that have been looked at by various communities. Based on our current knowledge, it is reasonable to assume that such three-local generators exist with stretch, say, $n^{1.1}$ which would be sufficient for the Lin-Tessaro construction.

The most problematic assumption is the existence of trilinear maps. Since the seminal work of Garg, Gentry and Halevi [13], there have been some candidate constructions for (noisy) k-linear maps for $k > 2$, but these are not based on any standard assumption, and in fact there have been several *attacks* [8,6,10,18,7,17,9,26,25] showing that these construction fail to satisfy natural analogs of the Decisional Diffie Hellman assumption. This is in contrast to the

$k = 2$ or *bilinear* case, where we have had constructions for almost 20 years that are believed to be secure (with respect to classical polynomial-time algorithms) based on elliptic curve groups that admit certain *pairing* operations [5]. Thus a main open question has been whether one can achieve iO based only on cryptographic *bilinear maps* as well as local (or otherwise "simple") pseudorandom generators that can be reasonably conjectured to be secure.

1.1 Basing iO on bilinear maps and our results

In the first version of their manuscript, Lin and Tessaro [22] gave a construction of iO based on *two local* generators with a certain stretch, and a candidate construction for the latter object based on a random two-local map with a certain nonlinear predicate. Alas, Barak, Brakerski, Komargodski and Kothari, [4], as well as Lombardi and Vaikuntanathan [24] showed that the Lin-Tessaro candidate construction, as well as *any* generator with their required parameters, can be broken using semidefinite programming, and specifically the degree two *sum of squares* program [4].

Very recently, the work of Ananth, Jain, and Sahai [2], followed shortly by the independent works of Agrawal [1] and Lin and Matt [21], proposed a way around that hurdle, obtaining constructions for iO where the role of the trilinear map is replaced with objects that:

1. Satisfy security notions that are weaker than being a full fledged pseudorandom generators.
2. Satisfy structural properties that are weaker than being two-local, and in particular requiring the outputs only to be a *degree two*[1] *polynomial* of the input.

As such, these objects do not automatically fall under the attacks described by [4,24]. However, in this work we show that:

- The specific candidate objects in all these three works (based on random polynomials) can be broken using a distinguisher built on the same sum-of-squares semidefinite program.
- Moreover, this results extends to other families of constructions, including ones that are not based on random polynomials. In fact, we do not know of *any* degree-2 construction that does not fall prey to a variant of the same attack.

[1] The work of Ananth, Jain, and Sahai [2] also considered degree-3 polynomials. We do not have attacks on such degree-3 polynomials; we discuss this further below.

The Ananth-Jain-Sahai "Cubic Assumption". The work of [2] also obtained a construction of iO based on a (variant of a) pseudorandom generator where every output is a *cubic* polynomial of the input, but where some information about the input is "leaked" in a way "masked" using instances of LWE[2]. Our attacks in their current form are not applicable to this new construction. The question of whether secure degree-3 ΔRGs exist, or whether an extended form of the sum of squares algorithm can be applied to it, is one that deserves further study. More generally, understanding the structure of hard distributions for expanding families of constant-degree polynomials over the integers, is a fascinating and important area of study, which is strongly motivated by the problem of securely constructing iO. Taking inspiration from SoS lower bounds [15,30], we also suggest a candidate for the same. Our candidate is inspired by the hardness of refuting random satisfiable 3SAT instances. For further details, see Section 7.

1.2 Our Results

We consider the following general hypothesis that, if true, would rule out not just the three proposed approaches based on quadratic polynomials for obtaining iO, but also a great many potential generalizations of them. Below we say that an n-variate polynomial q is Λ-*bounded* if all of q's coefficients are integers in the interval $[-\Lambda, +\Lambda]$. We say that a distribution $\mathcal{X}$ over $\mathbb{Z}^n$ is Λ *bounded* if it is supported over $[-\Lambda, +\Lambda]^n$.

Hypothesis 1 (No expanding weak quadratic pseudorandom generators). *For every $\varepsilon > 0$, polynomial $\Lambda(n)$, sufficiently large $n \in \mathbb{N}$, if:*

- *$q_1, \ldots, q_m : \mathbb{R}^n \to \mathbb{R}$ are quadratic $\Lambda(n)$-bounded polynomials for $m \geqslant n^{1+\varepsilon}$*
- *$\mathcal{X}$ is a $\Lambda(n)$-bounded distribution over $\mathbb{Z}^n$*
- *For every i, Δ_i is a $\Lambda(n)$ bounded distribution over $\mathbb{Z}$ such that $\mathbb{P}[\Delta_i = z] < 0.9$ for every $z \in \mathbb{Z}$.*

then there exists an algorithm $\mathcal{A}$ that can distinguish between the following distributions with $\Omega(1)$ bias:

- *$(q_1, \ldots, q_m, q_1(x), \ldots, q_m(x))$ for $x \sim \mathcal{X}$.*
- *$(q_1, \ldots, q_m, q_1(x) + \delta_1, \ldots, q_m(x) + \delta_m)$ where for every i, δ_i is drawn independently from Δ_i.*

Note that this hypothesis would be violated by the existence of a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{n^{1+\varepsilon}}$ whose outputs are degree two polynomials. It would also be violated if the distribution $G(x)$ is indistinguishable from

[2] This cubic version of their assumption was made explicit in an update to [2].

the distribution $(G(x)+\delta'_1,\ldots,G(x)+\delta'_m)$ where $\delta'_1,\ldots,\delta'_m$ are drawn independently from some distribution Δ over integers that satisfies $\mathbb{P}[\Delta=0]\leqslant 0.9$.

An efficient algorithm to recover x from $q_1(x),\ldots,q_m(x)$ would allow to distinguish between the two distributions. However, generally speaking, it need not even be information theoretically possible to recover x from this information. Even if it is information-theoretically possible this can be computationally intractable, as recovering x from $q_1(x),\ldots,q_m(x)$ is an instance of the NP hard problem of *Quadratic Equations*.

In Hypothesis 1, the polynomials are arbitrary. However, the candidate constructions of pseudorandom generators considered so far used $q_1,\ldots,q_m$ that are sampled *independently* from some distribution $\mathcal{Q}$. This is natural, as intuitively if we want $q_1(x),\ldots,q_m(x)$ to look like a product distribution, then the more randomness in the choice of the q_i's the better.

However, in this work we give general attacks on candidates that have this form. As these are some of the most natural approaches to refute Hypothesis 1, our work can be seen as providing some (partial) evidence to its veracity. To state our result, we need the following definition of "nice" distributions.

Definition 1 (Nice distributions). *Let $\mathcal{Q}$ be a distribution over n-variate quadratic polynomials with integer coefficients. We say that $\mathcal{Q}$ is* nice *if it satisfies that:*

- *There is a constant $C=C(\mathcal{Q})=O(1)$ such that $\mathcal{Q}$ is supported on* homogeneous *(i.e. having no linear term) degree-2 polynomials q with $\|q\|_2^2\leqslant C\,\mathbb{E}\,\|q\|_2^2$, where $\|q\|$ is the ℓ_2-norm of the vector of coefficients of q.*
- *$Var(Q_{i,j})=1$ where $Q_{i,j}$ denotes the coefficient of x_ix_j in a polynomial Q sampled from $\mathcal{Q}$.*
- *If $\{i,j\}\neq\{k,\ell\}$ then the random variables $Q_{i,j}$ and $Q_{k,\ell}$ are* independent

Roughly speaking, a distribution over quadratic polynomials is nice if it satisfies certain normalization properties as well as *pairwise independence* of the vectors of coefficients. Many natural distributions on polynomials are nice, and in particular random dense as well as random sparse polynomials are nice.

The following theorem shows that it is always possible to recover x from a superlinear number of quadratic observations, if the latter are chosen from a nice distribution.

Theorem 2 (Recover from random quadratic observations). *There is a polynomial-time algorithm $\mathcal{A}$ (based on the sum of squares algorithm) with the following guarantees. For every nice distribution $\mathcal{Q}$ and every $t\leqslant n^{O(1)}$, for large-enough n, with probability at least $1-n^{-\log(n)}$ over $x\sim\{-t,-t+1,\ldots,0,\ldots,t-1,t\}^n$ and $q_1,\ldots,q_m\sim\mathcal{Q}$, if $m\geqslant n(\log n)^{O(1)}$ then $\mathcal{A}(q_1,\ldots,q_m,q_1(x),\ldots,q_m(x))=x$.*

On "niceness". The definition of "nice" distributions above is fairly natural, and captures examples such as when the polynomials are chosen with all coefficients as independent Gaussian or Bernoulli variables. In particular as a corollary of Theorem 2 we break the candidate pseudorandom generator of Ananth et al (and even its Δ-RG property). Moreover, we obtain such results even in the *sparse* case where most of the coefficients of the polynomials $q_1, \ldots, q_m$ are zero.

At the moment however our theoretical analysis does not extend to the "blockwise random" polynomials that were used by Lin and Matt which can be thought of as a sum of random dense polynomial and a random sparse polynomial. While this combination creates theoretical difficulty in the analysis, we believe that it can be overcome and that it is possible to recover in this case as well. In particular, we also have *experimental results* showing that we can break the Lin-Matt generator as well.

Finally, we note that by Markov's inequality for any $\mathcal{Q}$ we have $\mathbb{P}(\|q\|^2 \geqslant C\,\mathbb{E}\,\|q\|^2) \leqslant 1/C$. Our niceness assumption just has the effect of restricting $\mathcal{Q}$ to this relatively high-probability event. If $\mathcal{Q}$ is not pathological – that is, it is not dominated by events with probability $\ll 1/C$ for a large constant C – then this kind of truncation will result in a nice distribution.[3]

On the distribution of x. For concreteness, we phrase Theorem 2 so that the distribution of x is uniform over $\{-t, \ldots, t\}^n$. However, the proof of the theorem allows x to be a more general $\mathbb{R}^n$-valued random variable. In particular, x may be any n-dimensional real-valued random vector which has $\mathbb{E}\,x = 0$ and is $O(\mathbb{E}\,\|x\|^2/n)$-sub-Gaussian. The coordinates of x need not even be independent: for instance, x may be drawn from the uniform distribution on the unit sphere.

Experiments. We implement the sum-of-squares attack and verify that indeed it efficiently breaks random dense quadratic polynomials. Furthermore, we implement a variant of the attack that efficiently breaks the Lin-Matt candidate:

[3] Along the same lines, we note that if $\mathcal{Q}$ is nice and $\mathbb{E}_{q\sim\mathcal{Q}}\, q = 0$ (as we observe later, the latter can be enforced without loss of generality) then $\mathcal{Q}$ is also $\Lambda(n)$-bounded for $\Lambda(n) \leqslant O(n)$. The reason is that if $\mathcal{Q}$ is nice and has $\mathbb{E}\,q = 0$ then

$$\mathbb{E}\,\|q\|^2 = \sum_{i,j\leqslant n} \mathbb{E}\,Q_{ij}^2 = \sum_{i,j\leqslant n} Var(Q_{ij}) = n^2\,.$$

For every i, j and every q in the support of $\mathcal{Q}$, we have by niceness that $|q_{ij}| \leqslant \|q\|_2 \leqslant Cn$. Hence $\mathcal{Q}$ is $O(n)$-bounded.

One implication is that $\mathcal{Q}$ cannot be a distribution on where the all-zero polynomial appears with probability, say, $1 - 1/n$, as otherwise its support would also have to contain polynomials with coefficients $\gg n$. Our main theorem could not apply to such a distribution, since clearly at least $\Omega(n^2)$ independent samples would be needed to get enough information to recover x from $\{q_i, q_i(x)\}$, while we assume $m \leqslant n(\log n)^{O(1)} \ll n^2$.

the Lin-Matt candidate is, roughly speaking, a sum of two independent polynomials, where one is dense and one is sparse. Since the planted solution must be composed of polynomially-bounded integers, we observe that it is possible to efficiently guess the squared L2 norm of the portion of the planted solution that corresponds to the sparse part of the polynomial. Given this guess, we can introduce a new constraint into the semidefinite program that fixes the trace of the portion of the semidefinite matrix that corresponds to the sparse matrix. We show experimentally that this attack breaks the Lin-Matt candidate for moderate values of n. In particular, in Figure 1 we plot the correlation between the recovered solution with the planted solution, where the x-axis is labeled by the ratio m/n showing the expansion needed for the attack to work, for $n = 60$ total variables. More details can be found in Section 6.

In particular, we are not aware of any candidate construction of weak pseudorandom generator computed by quadratic polynomials that is not broken experimentally by our algorithms.

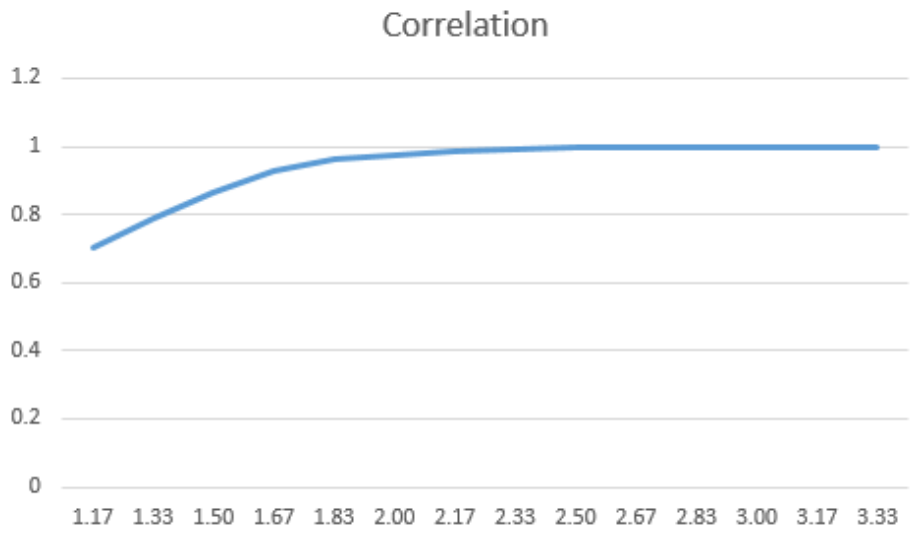

Figure 1: Experimentally breaking Lin-Matt candidate. Graph shows quality of recovered solution vs. planted solution, for various values of m/n shown in the x-axis. Let $\mathbf{v}$ be the eigen vector with largest eigen value of the optimum matrix returned by the SDP. Let $\mathbf{x}$ be the planted solution. Quality of solution is defined as $\frac{\langle \mathbf{v},\mathbf{x} \rangle}{\langle \mathbf{v},\mathbf{v} \rangle^{\frac{1}{2}} \langle \mathbf{x},\mathbf{x} \rangle^{\frac{1}{2}}}$

2 Our techniques

Our algorithms use essentially the same semidefinite program constraints that were used in the work of [4], namely the *sum of squares* program. However, we use a different, simpler, objective function, and moreover we crucially use a different *analysis* (which also inspired some tweaks to the algorithm that seem to help in experiments). Specifically, consider the task of recovering an unknown vector $x \in \mathbb{R}^n$ from the values $(q_1(x), \ldots, q_m(x))$ where $q_1, \ldots, q_m$ are quadratic polynomials. We focus on the case that the q_i's are *homogenous* polynomials, which means that (thinking of x as a column vector), $q_i(x) = x^\top Q_i x$ for some $n \times n$ matrix Q_i. Another way to write this is that $q_i(x) = \langle Q_i, X \rangle$ where X is the rank one matrix $xx^\top$.

In the above notation, our problem becomes the task of recovering a rank one matrix X from the observations

$$\langle Q_1, X \rangle, \ldots, \langle Q_m, X \rangle \tag{2.1}$$

for some known $n \times n$ matrices $Q_1, \ldots, Q_m$ where $m \geqslant n^{1+\varepsilon}$. Luckily, this task has been studied in the literature and is known as the *low rank recovery problem* [28]. This can be thought of as a matrix version of the well known problem of *sparse recovery* (a.k.a. *compressed sensing*) of recovering an k-sparse sparse vector $x \in \mathbb{R}^n$ (for $n \gg k$) from linear observations of the form $A_1 x, \ldots, A_{k'} x$ where k' is not much bigger than k.

While the low rank recovery problem is NP hard in the worst-case, for many inputs of interest it can be solved by a semidefinite program minimizing the *nuclear norm* of a matrix. This semidefinite program can be thought of as the matrix analog of the L_1 minimization linear program used to solve the sparse recovery problem. In particular, it was shown by Recht, Fazel and Parrilo [28] that if the Q_i's are *random* (with each entry independently chosen from, say, a random Gaussian or Bernoulli distribution), then they would satisfy a condition known as *matrix restricted isoperimetry property (matrix RIP)* that ensures that the semidefinite matrix recovers X in our regime of $m \geqslant n^{1+\varepsilon}$.

This already rules out certain candidates, but more general candidates have been considered. In particular, the results of Recht et al are not applicable when the Q_i's are *sparse* random matrices, which have been used in some of the iO constructions such Lin-Matt's. Luckily, this problem has been studied by the optimization community as well. The extremely sparse case, where each of the Q_i's has just a single nonzero coordinate, is particularly well studied. In this case, recovering X from (2.1) corresponds to *completing* X using m observations of its entries, and is known as the *matrix completion* problem.

Specifically, a beautiful paper of Gross [16] gave quite general bounds that in some sense interpolate between these two extremes. Specifically, Gross

showed that it is possible to recover X from (2.1) as long as these observations $Q_1, \ldots, Q_m$ are sampled independently from a collection $\{Q^1, \ldots, Q^N\}$ that satisfies certain "isotropy" and "incoherence with respect to X" properties. We show that under the "niceness" conditions of Theorem 2, we can "massage" our input so that it is of the form where Gross's theorem applies. Once we do so we can appeal to this theorem to obtain our result. A key property that we use in our proof is that in the cryptographic setting, we do not need to recover $X = xx^\top$ for *every* $x \in \mathbb{Z}^n$ but rather only for *most* x's. This allows us to achieve the incoherence property even in settings where it would not hold for a worst-case choice of a vector.

3 Preliminaries

For a matrix X, we write $\|X\|$ for its operator norm: $\sup_{v:\|v\|_2=1} |\langle v, Xv\rangle$. We use the standard inner product on the Hilbert space of $n \times n$ matrices: $\langle A, B\rangle = \mathrm{tr}(AB)$. The *nuclear* norm of a matrix X is defined by $\|X\|_* = \sup_{A:\|A\|\leqslant 1}\langle A, X\rangle$. For a positive semidefinite matrix X, $\|X\|_* = \mathrm{tr}(X)$.

For any matrix $Q \in \mathbb{R}^{n\times n}$, $\mathsf{vec}(Q)$ denotes "vectorization" of the matrix Q as a n^2 dimensional vector.

For a matrix $M \in \mathbb{R}^{n\times n}$, we define the operator norm (also called the spectral norm) of M as $\max_{x\in\mathbb{R}^n} \|Mx\|/\|x\|$. The Frobenius norm of M is $\|M\|_F = \sqrt{\sum_{ij\leqslant n} M_{ij}^2}$.

For a matrix M, we write $M \in (1 \pm \varepsilon)\,\mathrm{Id}$ if $\|M - \mathrm{Id}\| \leqslant \varepsilon$, where $\|\cdot\|$ is the operator norm.

3.1 ΔRGs (Ananth-Jain-Sahai)

Ananth-Jain-Sahai proposed a variant of (integer valued) PRG that such that it is hard to distinguish between the output of a PRG and a small perturbation of it. Specifically, the following definition describes the object they proposed.

Definition 2 (**(n, λ, B, χ)-ΔRG**). *Let $f : \chi^n \to \mathbb{Z}^m$ be an integer valued function with the ith output described by $f_i : \chi^n \to \mathbb{Z}$ and at any $x \in \chi^n$, $f_i(x) = q_i(x)$ for quadratic polynomials q_i for $1 \leqslant i \leqslant m$.*

f is said to be a ΔRG, if for distributions D_1, D_2 on $\mathbb{Z}^m$ defined below and for any circuit $\mathcal{A}$ of size 2^λ,

$$|\mathop{\mathbb{P}}_{z\sim D_1}[\mathcal{A}(z) = 1] - \mathop{\mathbb{P}}_{z\sim D_2}[\mathcal{A}(z) = 1]| < 1 - 2/\lambda$$

Distribution D_1
Sample $x \leftarrow \chi$. Output $\{q_i, q_i(x)\}_{i\in[m]}$

Distribution D_2
Sample $x \leftarrow \chi$. Output $\{q_i, q_i(x) + \delta_i\}_{i\in[m]}$
Here $\delta_i \in \mathbb{Z}$ are arbitrary perturbations such that $|\delta_i| < B$ for all $i \in [m]$.

Concurrently and independently, [21] proposed Pseudo-Flawed Smudging Generators which have similar security guarantees.

4 Candidates for Quadratic PRGs

In this section we formally describe the candidate polynomial and input distributions proposed by [2,21,1] to realize corresponding notions of pseudo-random generators of $\mathbb{Z}$.

Note that any algorithm that given the polynomials $q_1, \ldots, q_m$ and measurements $q_1(x), \ldots q_m(x)$ when x, $q_1, .., q_m$ are sampled from required distributions of the pseudorandom generator, successfully recovers x, also breaks the corresponding candidate for the pseudorandom generator.

To be precise, we describe the candidate polynomials and input distributions proposed by:

- Ananth et al. [2] to instantiate ΔRGs.
- Lin-Matt [21] to instantiate Pseudo Flawed-Smudging Generators.
- Agarwal [1] to instantiate Non-boolean PRGs.

Along with assumptions on cryptographic bilinear maps, learning with error assumption and PRGs with constant block locality, either of these three assumptions imply iO.

4.1 Candidate for ΔRG

Ananth-Jain-Sahai proposed the following candidate construction for a ΔRG. Let χ be the uniform distribution in $[-B_1, B_1]$. Choose $m = n^{1+\varepsilon}$ for some small enough constant $\varepsilon > 0$. Let C be some constant positive integer and B_1 be a polynomial in λ, the security parameter.
Distribution Q: Sample each polynomial as follows. Let $q(x_1, ..., x_n) = \Sigma_{i\neq j} c_{i,j} \cdot x_i \cdot x_j$, where each coefficient $c_{i,j}$ is chosen uniformly from $[-C, C]$.

Distribution X: Inputs are sampled as follows. Sample x_i for $i \in [n]$ uniformly from $[-B_1, B_1]$. Output $x = (x_1, ..., x_n)$. Implicitly, [1] also considered these polynomials for their notion of a non-boolean PRG.

4.2 Candidate for Pseudo Flawed-Smudging Generators (Lin-Matt)

Lin and Matt [21] proposed a variant of pseudorandom generators with security properties closely related to the notion of ΔRGs above. Here, we recall their candidate polynomials.

Distribution Q: For each $j \in [m]$,

$$q_j(x_1, ..., x_n, x'_1, ..., x'_{n'}) = S_j(x_1, ..., x_n) + MQ_j(x'_1, ..., x'_{n'})$$

Here we write more about polynomials S_j, MQ_j.

1. MQ_j **Polynomials:** MQ_j are random quadratic polynomials over $(x'_1, ..., x'_{n'})$, where the coefficients of each degree two monomial $x'_i x'_k$ and degree one monomial x'_i are integers chosen independently at random from $[-C, C]$.
2. S_j **Polynomials:** S_j are random quadratic polynomials over $(x_1, ..., x_n)$ of the form:

   $$S_j(x_1, ..., x_n) = \Sigma_{i=1}^{n/2} \alpha_i x_{\sigma_j(2\cdot i)} x_{\sigma_j(2\cdot i-1)} + \Sigma_{i=1}^{n} \beta_i x_i + \gamma$$

   Here each coefficient α_i, β_i and γ are random integers chosen independently from $[-C, C]$. Here, σ_j is a random permutation from $[n]$ to $[n]$.

Distribution X:

1. Each x_i for $i \in [n]$ is chosen as a random integer sampled independently from the distribution χ_{B_1,B_2}. χ_{B_1,B_2} samples a random integer from $[-B_1, B_1]$ with probability 0.5 and from $[-B_2, B_2]$ with probability 0.5.
2. **Distribution of inputs** $x'_1, ..., x'_{n'}$**:** Each x'_i is chosen as a random integer sampled independently from the distribution $\chi_{B'}$. $\chi_{B'}$ samples a random integer from $[-B', B']$.

Parameters: Set B_1, B_2, B', n, n' as follows:

- Set $n = n'$ and $m = n^{1+\varepsilon}$, for some $\varepsilon > 0$.
- B_1, B' and C are set arbitrarily.
- Set $B_2 = \Omega(nB^2 + nBB_1)$.

Here B is some polynomial in the security parameter.

All the pseudorandom generators we consider are maps from $\mathbb{Z}^n$ into $\mathbb{Z}^m$ where each of the m output is computed by a degree 2 polynomial with integer coefficients in the input. Since any degree two polynomial in $\mathbb{R}^n$ can be seen as a linear map on $\mathbb{R}^{n\times n}$[4], one can equivalently think of such PRGs as linearly mapping symmetric rank 1 matrices into $\mathbb{R}^m$.

[4] For any $q(x) = \sum_{i,j} q_{i\leqslant j} x_i x_j$, we define $Q : \mathbb{R}^{n\times n} \to \mathbb{R}$ by $Q_{i,j} = Q_{j,i} = q_{i,j}/2$. Then, $Q(X) = \mathrm{tr}(QX) = \langle Q, X\rangle$ is a linear map on $\mathbb{R}^{n\times n}$.

5 Inverting Linear Matrix Maps

In this section, we describe the main technical tool that we rely on in this work - an algorithm based on semidefinite programming for inverting *linear matrix maps*.

Definition 3 (Linear Matrix Maps). *A linear matrix map $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ described by a collection of $n \times n$ matrices $Q_1, Q_2, \ldots, Q_m$ is a linear map that maps any matrix $X \in \mathbb{R}^{n\times n}$ to the vector $\mathcal{A}(X) \in \mathbb{R}^m$ such that $\mathcal{A}(X)_i = \langle Q_i, X\rangle$.*

We will use calligraphic letters such as $\mathcal{A}$ and $\mathcal{B}$ to denote such maps.

We are interested in the algorithmic problem of *inverting* such maps, that is, finding X given $\mathcal{A}(X)$. If Q_is are linearly independent and $m \gg n^2$, then this can be done by linear equation solvers. Our interest is in inverting such maps for low rank matrices X with the "number of measurements" $m \ll n^2$. Indeed, our results will show that for various classes of linear maps $\mathcal{A}$, we can efficiently find a low-rank solution to $\mathcal{A}(X) = z$, whenever it exists, for $m = \tilde{O}(n)$.

Such problems have been well-studied in the literature and rely on a primitive based on semidefinite programming called "nuclear norm minimization". We will use this algorithm and rely on various known results about the success of this algorithm in our analysis.

Algorithm 3 (Trace Norm Minimization).

Given: $-$ $\mathcal{A}$ *described by* $Q_1, Q_2, \ldots, Q_m \in \mathbb{R}^{n\times n}$.
$-$ $z \in \mathbb{R}^m$.

Operation: *Output* $X = \arg\min_{\substack{X\succeq 0 \\ \mathcal{A}(X)=z}} \operatorname{tr}(X)$.

In what follows, we will give an analysis of this algorithm for a class of linear matrix maps.

5.1 Incoherent Linear Measurements

In this section we describe a remarkably general result due to Gross on a class of instances $x, Q_1, \ldots, Q_m$ for which trace norm minimization recovers x [16]. These instances are called *incoherent*. Gross's result is the main tool in the proof of our main theorem, which will ultimately show that "nice" distributions $\mathcal{Q}$ produce incoherent instances of trace norm minimization.

We note that many other sufficient conditions for the success of trace norm minimization have been discussed in the literature. One prominent condition is matrix-RIP (Restricted Isometry Property), analyzed in [27]. The restricted isometry property is not known to apply in many natural settings for which

we would like to apply our main theorem – for example, if $Q_1, \ldots, Q_m$ have independent entries with on average 1 nonzero entry per row.

Definition 4 (Incoherent Overcomplete Basis). *Let $\mathcal{B} = \{B_1, B_2, \ldots, B_N\}$ be a collection of matrices in $\mathbb{R}^{n\times n}$. For any rank 1 matrix $X \in \mathbb{R}^{n\times n}$, $\mathcal{B}$ is said to be ν-incoherent basis for X if the following holds:*

1. $(1-o(1))1/n^2 I_{n^2\times n^2} \preceq 1/N \sum_{i=1}^{N} \mathsf{vec}(B_i)\mathsf{vec}(B_i)^\top \preceq (1+o(1))1/n^2 I_{n^2\times n^2}$.
2. *For each $i \leqslant N$, $|\langle X, B_i\rangle| \leqslant \nu/n \cdot \|X\|_F$.*

We can now define a ν-incoherent measurement.

Definition 5 (Incoherent Measurement). *Let $\mathcal{B}$ be a ν-incoherent overcomplete basis for an $n \times n$ rank 1 matrix X, and suppose $\mathcal{B}$ has size $N = \mathrm{poly}(n)$. Let $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ be a map obtained by choosing Q_i for each $i \leqslant m$ to be a uniformly random and independently chosen element of $\mathcal{B}$. Then, $\mathcal{A}$ is said to be a ν-incoherent measurement of X.*

The following result follows directly from the Proof of Theorem 3 in the work of Gross [16]. While that work focuses on $\mathcal{B}$ being orthonormal - the proof extends to approximately orthonormal basis (i.e., part 1 in the above definition) in a straightforward way.

Theorem 4. *Let $\mathcal{B}$ be a ν-incoherent basis for a rank 1 matrix X of size $N = \mathrm{poly}(n)$. Let $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ be a map obtained by choosing Q_i for each $i \leqslant m$ to be a uniformly random and independently chosen element of $\mathcal{B}$. Then, for large enough $m = \Theta(\nu n \,\mathrm{poly}\log n)$, Algorithm 3, when given input $\mathcal{A}$ and $\mathcal{A}(X)$ recovers X, with probability at least $1 - n^{-10\log(n)}$ over the choice of $\mathcal{A}$.*

5.2 Invertible Linear Matrix Maps

In this section we prove Theorem 2 on solving random quadratic systems.

Proof (Proof of Theorem 2). Fix $t \leqslant n^{O(1)}$ and a nice distribution $\mathcal{Q}$.

Centering We may assume that $\mathbb{E}_{\mathcal{Q}} Q = 0$. Otherwise, we can replace $\mathcal{Q}$ with $\mathcal{Q}'$ where $Q' = \frac{1}{\sqrt{2}}(Q_0 - Q_1)$ for independent draws $Q_0, Q_1 \sim \mathcal{Q}$. This is because $\mathcal{Q}'$ remains nice if $\mathcal{Q}$ is, clearly $\mathbb{E}\, Q' = 0$, and given $q_1, \ldots, q_m, q_1(x), \ldots, q_m(x)$ our algorithm can pair i to $i+1$ (for even i) and instead consider $m/2$ samples of the form $(1/\sqrt{2})(q_i + q_{i+1}), (1/\sqrt{2})(q_i(x) + q_{i+1}(x))$. Thus for the remainder of the proof we assume $\mathbb{E}\, Q = 0$.

Our goal is to establish that there is $N \leqslant n^{O(1)}$ such that if $Q_1, \ldots, Q_N$ are i.i.d. draws from $\mathcal{Q}$, then $(1/n)Q_1, \ldots, (1/n)Q_N$ are ν-incoherent with respect to most $x \in [-t, t]^n$.

Incoherence part one: orthogonal basis First observe that since $\mathbb{E}\, Q = 0$ and $\mathbb{E}\, Q_{ij}^2 = 1$ and our pairwise independence assumption, we have

$$\mathbb{E}\, \mathsf{vec}(Q)\mathsf{vec}(Q)^\top = \mathrm{Id}_{n^2 \times n^2} \ .$$

Also, by niceness, every $|Q_{ij}| \leqslant O(n)$ with probability 1, for every i, j. Fix $i, j, k, \ell \leqslant n$. By the Bernstein inequality, given N independent draws $Q^{(1)}, \dots, Q^{(N)}$, for any $s \geqslant 0$,

$$\mathbb{P}\left\{ \left| \frac{1}{N} \sum_{a \leqslant N} Q_{ij}^{(a)} Q_{k\ell}^{(a)} - \mathbb{E}\, Q_{ij} Q_{k\ell} \right| > s \right\} \leqslant \exp\left(\frac{-CNs^2}{n^4 + sn^2} \right)$$

for some universal constant C. Take $s = 1/n^4$ and $N = n^{10}$, this probability is at most $\exp(-O(n^2))$. Taking a union bound over $i, j, k, \ell \in [n]$, we find that with probability at least $1 - \exp(-O(n^2))$,

$$(1 - o(1))\, \mathrm{Id}_{n^2 \times n^2} \preceq \frac{1}{N} \sum_{a \leqslant N} \mathsf{vec}(Q^{(a)})\mathsf{vec}(Q^{(a)})^\top \preceq (1 + o(1))\, \mathrm{Id}_{n^2 \times n^2} \ .$$

Incoherence part two: small inner products Next we establish the other part of incoherence: that $\frac{1}{n^2}\langle x, Q^{(a)} x \rangle \leqslant \nu/n$ for all $a \leqslant N$. The coordinates of the vector x are independent, and each is bounded by t. Thus x is sub-Gaussian, with variance proxy $O(t^2)$. Since the coordinates of x have $\mathbb{E}\, x_i^2 \geqslant \Omega(t^2)$, the random vector y with coordinates $x_i / \sqrt{\mathbb{E}\, x_i^2}$ has sub-Gaussian norm $O(1)$.

Consider a fixed matrix $M \in \mathbb{R}^{n \times n}$, where M has Frobenius norm $\|M\|_F$ and spectral norm $\|M\|$. By the Hansen-Wright inequality, for any $s \geqslant 0$,

$$\mathop{\mathbb{P}}_{y} \left\{ |y^\top M y - \mathbb{E}\, y^\top M y| > s \right\} \leqslant \exp\left(-Cs^2 / (\|M\|_F^2 + s\|M\|) \right)$$

for some constant C.

If Q is any matrix in the support of $\mathcal{Q}$, then $\|Q/n\|_F \leqslant O(1)$ by niceness, and $\|Q\| \leqslant \|Q\|_F$. So for any such Q,

$$\mathop{\mathbb{P}}_{y} \left\{ |y^\top (Q/n) y - \mathbb{E}\, y^\top (Q/n) y| > s \right\} \leqslant \exp\left(-Cs^2 / (1 + O(s)) \right) \ .$$

Taking $s = (\log n)^4$, this probability is at most $n^{-(\log n)^2}$ for large-enough n. Taking a union bound over $N \leqslant n^{O(1)}$ samples $Q^{(a)}$, with probability at least $1 - n^{-(\log n)^{1.5}}$ over y (for large enough n), every $Q^{(a)}$ has

$$\left| x^\top \cdot \frac{Q^{(a)}}{n} \cdot x \right| \leqslant \frac{(\log n)^{O(1)}}{n} \cdot \|x x^\top\|_F \, .$$

Putting it together, for $N = n^{O(1)}$, with probability at least $1 - n^{-(\log n)^{1.4}}$ for big-enough n, if $x \sim \{-t, \dots, t\}^n$ then $Q^{(1)}/n, \dots, Q^{(N)}/n$ are a $(\log n)^{O(1)}$-incoherent basis for x. Thus with probability at least $1 - n^{-10 \log n}$ over

$Q^{(1)}, \ldots, Q^{(N)}$, we have $\mathbb{P}_x(Q^{(1)}/n, \ldots, Q^{(N)}/n$ is ν-incoherent for $x) \geqslant 1 - n^{-10 \log n}$ (again for large enough n).

From incoherence to recovery We can simulate the procedure of sampling $Q_1, \ldots, Q_m$ as in the theorem statement by first sampling $Q_1, \ldots, Q_N$, then randomly subsampling m of the Q's. If $Q_1, \ldots, Q_N$ are $(\log n)^{O(1)}$-incoherent for $xx^\top$, then Theorem 4 shows that with probability $1 - n^{-\log n}$ over the second sampling step, trace norm minimization recovers x, so long as the number of samples m is at least $n(\log n)^{O(1)}$. This finishes the proof.

6 Experiments

In this section, we describe the experiments that we performed on various classes of polynomials and how well do they perform in practice. All the codes were run and analysed on a MacBook Air (2013) laptop with 4GB 1600 Mhz DDR3 RAM and an intel i5 processor with clock speed of 1.3 Ghz. We used Julia as our programming language and the package "Mosek" for the implementation of an SDP solver.

6.1 Experimental Cryptanalysis of Dense or Sparse Polynomials

First, we describe the setting of multivariate quadratic polynomials over the integers where the coefficients of each monomial is chosen independently at random from some distribution $\mathcal{D}$. Such dense polynomials were considered in [2,1]. We denote such polynomials by MQ.

The function `genmatrixDMQ` takes as input number of variables n and a coefficient bound C, and does the following:

1. For every monomial $x_i x_j$ where $i, j \in [n]$ and $j \geqslant i$, it samples a coefficient as a uniformly random integer in $[-C, C]$.
2. This coefficient is stored as $V[i][j]$ inside the matrix V.
3. The entire coefficient matrix is then made symmetric by just computing sum of itself with its transpose. Note that this quadratic form is the same as the one obtained in step 2.

The code can be found in Section A

Having described how to sample a polynomial, now we turn to the procedure to sample the input.

The function `genxMQ` on input number of variables n and a bound B, and does the following:

1. It samples an input vector $(x[1], ..., x[n])$ where each $x[i]$ is a sampled independently as a random integer between $[-B, B]$

The code of this function can be found in Section A.

Once we know how to sample polynomials and inputs we generate observations.

The function `genobsMQ` takes as input the number of input variables n, number of random polynomials m, coefficient bound C and bound on the planted input B. The function does the following:

1. It generates m polynomials randomly as per the distribution given by function `genmatrixDMQ` and stores them inside the vector L.
2. Then, it samples a planted input vector $\mathbf{x} = (x_1, ..., x_n)$ given by the distribution `genxMQ`.
3. Finally, it creates m observations of the form $obs[i] = \mathbf{x}^T L[i]\mathbf{x}$ for $i \in [m]$ where $\mathbf{x}^T$ is the transpose of vector $\mathbf{x}$.
4. It outputs polynomials, input and the observations.

This code can also be found in Section A

Once we have the observation we compute the function `recoverMQ` which implements the attack.

This function `recoverMQ` takes as m input observations as a vector obs along with the polynomial vector L. Then it finds a semi-definite matrix X constrained to the linear constraints that $Tr(L[i] * X) = obs[i]$ for $i \in [m]$, with the objective to minimize $Tr(X)$. Clearly, such an SDP is feasible as $X = \mathbf{x} \cdot \mathbf{x}^T$ (product of input vector with its transpose) satisfies the constraints.

Our experiments support the theorems given earlier in this paper. Indeed, for $m > 3n$, the SDP successfully recovers x for MQ polynomials. We similarly conducted experiments for sparse polynomials, where again the SDP successfully recovers x for $m > 3n$ in all experiments. We omit details of the sparse case to avoid redundancy.

6.2 Attacking [Lin-Matt18] Candidate Polynomials

In this section, we mount an attack on systems of quadratic polynomials with special structure. In particular, we consider the quadratic polynomials conjectured to provide security by [21]. Recall that the polynomials described in [21] are of the following structure. For each $j \in [m]$,

$$q_j(x_1, ..., x_n, x'_1, ..., x'_{n'}) = S_j(x_1, ..., x_n) + MQ_j(x'_1, ..., x'_{n'})$$

Here we write more about polynomials S_j, MQ_j as well as the input vector $(x_1, ..., x_n, x'_1, ..., x'_{n'})$.

1. MQ_j **Polynomials:** MQ_j are random quadratic polynomials over $(x'_1, ..., x'_{n'})$, where the coefficients of each degree two monomial $x'_i x'_k$ and degree one monomial x'_i are integers chosen independently at random from $[-C, C]$.

2. S_j **Polynomials:** S_j are random quadratic polynomials over $(x_1, ..., x_n)$ of the form:

$$S_j(x_1, ..., x_n) = \Sigma_{i=1}^{n/2} \alpha_i x_{\sigma_j(2*i)} x_{\sigma_j(2*i-1)} + \Sigma_{i=1}^{n} \beta_i x_i + \gamma$$

   Here each coefficient α_i, β_i and γ are random integers chosen independently from $[-C, C]$. Here, σ_j is a random permutation from $[n]$ to $[n]$.
3. **Distribution of inputs** $x_1, ..., x_n$**:** Each x_i is chosen as a random integer sampled independently from the distribution χ_{B_1,B_2}. χ_{B_1,B_2} samples a random integer from $[-B_1, B_1]$ with probability 0.5 and from $[-B_2, B_2]$ with probability 0.5.
4. **Distribution of inputs** $x'_1, ..., x'_{n'}$**:** Each x'_i is chosen as a random integer sampled independently from the distribution $\chi_{B'}$. $\chi_{B'}$ samples a random integer from $[-B', B']$.
5. Set B_1, B_2, B', n, n' as follows:
   - Set $n = n'$ and $m = n^{1+\varepsilon}$, for some $\varepsilon > 0$.
   - B_1, B' and C are set arbitrarily.
   - Set $B_2 = \Omega(nB^2 + nBB_1)$.

   Here B is some polynomial in the security parameter.

The function `genmatrixsmq` generates polynomials of the form $S + MQ$. Then, we sample inputs using the function `genxdiscsmq`. Note that, this function samples input of length $n+n'+2$. Two special variables $x[1]$ and $x[n+2]$ are set to 1 to achieve linear terms in the polynomials (as such $\mathbf{x}^T V \mathbf{x}$ is a homogeneous degree two polynomial in $\mathbf{x}$). Now we generate observation using the function genobssmq, which is implemented similarly.

Changing the SDP. To attack these special polynomials, we modify the SDP to introduce new constraints that help capture the structure of the polynomial. Specifically, because we know that the values $x_1, \ldots, x_n$ take small polynomially bounded values, we can enumerate over all possible "guesses" for $\Sigma_{i\in[n]} x_i^2$, and be sure that one of these will be correct. Let `val1` be this guess. As such, we can add a constraint that $\Sigma_{i\in[n]} X[i,i] =$ `val1` to the SDP, where X is the SDP matrix variable of size $n + n'$ by $n + n'$, and then solve. The code is formally described in Section A.

The Figure 2 shows the plot of the ratio m/n versus the correlation of the recovered solution with the planted solution, for $n+n' = 60$ total variables, where $n = n' = 30$. Larger values of n that were still experimentally feasible, such as $n + n' = 120$ yielded similar graphs. We also remark that similar experimental observations can be made if we replace polynomials S with randomly generated sparse polynomials with $O(n)$ monomials. As before, for $m > 3(n + n')$, in our experiments, we always recover the correct solution.

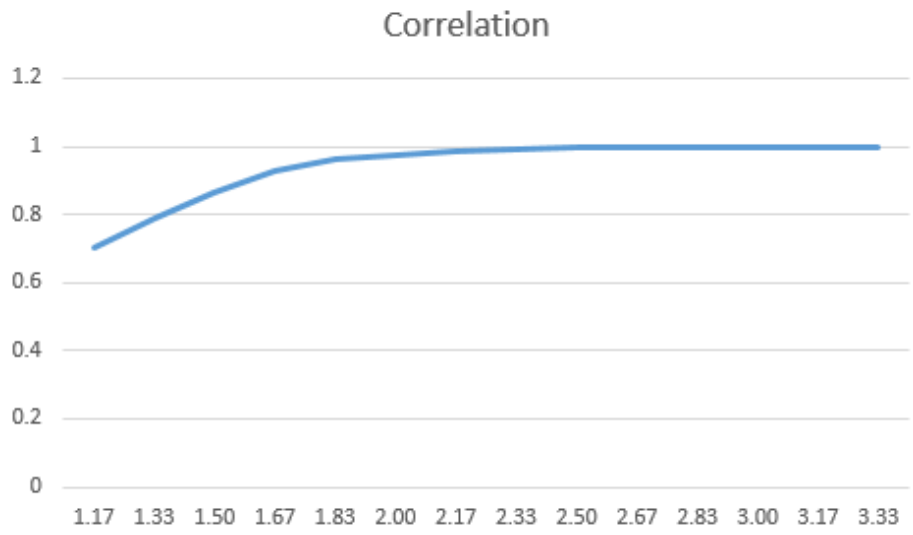

Figure 2: Experimentally breaking Lin-Matt candidate. Graph shows quality of recovered solution vs. planted solution, for various values of m/n shown in the x-axis. Let $\mathbf{v}$ be the eigen vector with largest eigen value of the optimum matrix returned by the SDP. Let $\mathbf{x}$ be the planted solution. Quality of solution is defined as $\frac{\langle \mathbf{v},\mathbf{x} \rangle}{\langle \mathbf{v},\mathbf{v} \rangle^{\frac{1}{2}} \langle \mathbf{x},\mathbf{x} \rangle^{\frac{1}{2}}}$

6.3 Attacking polynomials of the form $S + S + MQ$

Now we consider attacking a more general form of systems where each polynomial $q_j(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is of the following form:

- q_j takes as input three input vectors $\mathbf{x}_\ell = (x_{\ell,1}, \ldots, x_{\ell,n})$ for $\ell \in [3]$.
- Then, $q_j = S_{j,1}(\mathbf{x}_1) + S_{j,2}(\mathbf{x}_2) + MQ_j(\mathbf{x}_3)$

Inputs $\mathbf{x}_\ell$ for $\ell \in [3]$ are chosen as in the previous section. We observe that when we constrain the sum $\Sigma_{\ell \in [2], j \in [n]} x^2_{\ell,n}$, then SDP successfully recovers the planted solution using about same number of samples (for the same size of input) as for the previous case. This code of the recovery function `recoverspecialssm` is given in Section A. Note that in the code, the sum `val1` + `val2` is used to constrain this sum. This seems to generalise. If we consider a family where the polynomials q are of the form $S_1 + \ldots + S_k + MQ_1 + \ldots + MQ_k$, for values of k we could experimentally try (specifically $k \in \{1, 2, 3\}$) constraining the sum of trace corresponding to inputs of S polynomials leads to a break with probability 1.

7 Cubic Assumption

In this section, we discuss the cubic assumption proposed by [2]. Let us first recall the cubic version of the $\Delta\mathsf{RG}$ assumption considered by [2]. First, we define a notion of a polynomial sampler Q

Definition 6. *(Polynomial Sampler Q) A polynomial sampler Q is a probabilistic polynomial time algorithm that takes as input $n, B, \in \mathbb{N}$ along with a constant $1 > \varepsilon > 0$ and outputs:*

- *Polynomials $(q_1, ..., q_{\lfloor n^{1+\varepsilon} \rfloor})$.*
- *Each polynomial $q_j(e_1, ..., e_n, y_1, ..., y_n, z_1, ..., z_n) = \Sigma_{i_1,i_2,i_3 \in [n]} c_{i_1,i_2,i_3} e_{i_1} y_{i_2} z_{i_3}$. Here, each coefficient c_{i_1,i_2,i_3} are integers bounded in absolute value by a polynomial in n and $e_1, ..., e_n, y_1, .., y_n, z_1, ..., z_n$ are the variables of the polynomials.*

Cubic $\Delta\mathsf{RG}$ Assumption. There exists a polynomial sampler Q and a constant $\varepsilon > 0$, such that for every large enough $\lambda \in \mathbb{N}$, and every polynomial bound $B = B(\lambda)$ there exist large enough polynomial $n_B = \lambda^c$ such that for every positive integer $n > n_B$ there exists an efficiently samplable bounded distribution χ that is bounded by some polynomial in λ, n such that for every collection of integers $\{\delta_i\}_{i \in [\lfloor n^{1+\varepsilon} \rfloor]}$ with $|\delta_i| \leqslant B$, the following holds for the two distributions defined below:

Distribution dist_1:

- Fix a prime modulus $p = O(2^\lambda)$.
- (**Sample Polynomials.**) Run $Q(n, B, \varepsilon) = (q_1, ..., q_{\lfloor n^{1+\varepsilon} \rfloor})$.
- (**Sample Secret.**) Sample a secret $\mathbf{s} \leftarrow \mathbb{Z}_p^\lambda$
- Sample $a_i \leftarrow \mathbb{Z}_p^\lambda$ for $i \in [n]$.
- (**Sample LWE Errors.**) For every $i \in [n]$, sample $e_i, y_i, z_i \leftarrow \chi$. χ is a bounded distribution with a bound $poly(n)$ such that LWE assumption holds with error distribution χ, modulus p and dimension λ.
- Output $\{a_i, \langle a_i, s \rangle + e_i \mod p\}_{i \in [n]}, \{q_j, q_j(e_1, .., e_n, y_1, ..., y_n, z_1, ..., z_n)\}_{j \in [\lfloor n^{1+\varepsilon} \rfloor]}$

Distribution dist_2

- Fix a prime modulus $p = O(2^\lambda)$.
- (**Sample Polynomials.**) Run $Q(n, B, \varepsilon) = (q_1, ..., q_{\lfloor n^{1+\varepsilon} \rfloor})$.
- (**Sample Secret.**) Sample a secret $\mathbf{s} \leftarrow \mathbb{Z}_p^\lambda$
- Sample $a_i \leftarrow \mathbb{Z}_p^\lambda$ for $i \in [n]$
- (**Sample LWE Errors**). For every $i \in [n]$, sample $e_i, y_i, z_i \leftarrow \chi$. χ is a bounded distribution with a bound $poly(\lambda)$ such that LWE assumption holds with error distribution χ, modulus p and dimension λ.

– Output $\{a_i, \langle a_i, s\rangle + e_i \mod p\}_{i\in[n]}, \{q_j, q_j(e_1, .., e_n, y_1, ..., y_n, z_1, ..., z_n) + \delta_j\}_{j\in[\lfloor n^{1+\varepsilon}\rfloor]}$

The assumption states that there exists a constant $\varepsilon_{adv} > 0$ such that for any adversary $\mathcal{A}$ of size $2^{\lambda^{\varepsilon_{adv}}}$, the following holds:

$$|\,\mathbb{P}[\mathcal{A}(\text{dist}_1) = 1] - \mathbb{P}[\mathcal{A}(\text{dist}_2) = 1]| < 1 - 1/\lambda$$

Linearization Attack for n^2 stretch. The assumption above is only required to hold for stretch $n^{1+\varepsilon}$ for any small constant ε. However, we observe that the assumption described above suffers from an attack if the stretch is $O(\lambda \cdot n^2)$. The attack is simple and is described below.

Theorem 5. *The cubic ΔRG assumption does not hold with $m = O(\lambda \cdot n^2)$ polynomials $q_1, \ldots q_m$.*

Proof. Here is the breaking algorithm. For notational convenience, we only consider homogenous degree-3 polynomials. In this case, we can set $m = n^2(\lambda + 1)$. However, the algorithm trivially generalizes to all degree-3 polynomials with $m = n^2(\lambda + 3) + 2n + \lambda$.

1. Consider a polynomial $q_\ell(e_1, ..., e_n, y_1, ..., y_n, z_1, ..., z_n) = \Sigma_{i,j,k} c_{i,j,k,\ell} e_i y_j z_k$.
2. Rewrite $q_\ell(e_1, ..., e_n, y_1, ..., y_n, z_1, ..., z_n) = \Sigma_{i,j,k} c_{i,j,k,\ell}(\langle a_i, s\rangle + e_i - \langle a_i, s\rangle) y_j z_k$. Now note that a_i and $b_i = \langle a_i, s\rangle + e_i$ is given. Set $y_j z_k = w_{j,k}$ and $s_{\mathsf{ind}} y_j z_k = w_{\mathsf{ind},j,k}$ for $\mathsf{ind} \in [\lambda], j \in [n], k \in [n]$.
3. Note that since $q_\ell(e_1, ..., e_n, y_1, ..., y_n, z_1, ..., z_n) = \Sigma_{i,j,k} c_{i,j,k,\ell}(b_i - \langle a_i, s\rangle) y_j z_k$ in $\mathbb{Z}_p$, the entire system of $m = n^{1+\varepsilon}$ samples can be written as a system of linear equations over $\mathbb{Z}_p$ in $(\lambda + 1)n^2$ variables $w_{\mathsf{ind},j,k}$ and $w_{j,k}$. A simple gaussian elimination then recovers the solution.

On the existence of hard degree-3 polynomials. Feige [12] conjectured that its hard to distinguish a satisfiable random 3-SAT instance from a random 3-SAT instance with $C \cdot n$ clauses. Each disjunction $x_1 \vee x_2 \vee x_3$ corresponds to the polynomial $1 - (1 - x_1)(1 - x_2)(1 - x_3)$. This intuition gives rise to a set of candidate polynomials $q_{i,j}$, which depends on three randomly chosen variables and maps $\{0, 1\}^n$ to $\{0, 1\}$. Each $q_{i,j}$ has at most 8 monomials. Intuitively speaking, to choose clauses, instead of chosing clauses at random – something that is known to lead to weak RANDOM 3SAT instances – we first choose a planted *boolean* solution $x^* \in \{0, 1\}^n$, and always choose clauses such that exactly one or all three literals in the clause evaluate to true. This has the property that each clause individually induces a uniform constraint on any pair of variables x_i and x_j. In the boolean case, this distribution of clauses is believed to give rise

to hard distributions, which suggests that the expanding polynomial systems corresponding to these clauses should be hard to solve in general.

To construct obfuscation, we need the stretch to be at least $n^{1+\varepsilon}$ for any constant $\varepsilon > 0$. All known algorithms take exponential time as long as $n^{1.5-\varepsilon}$ clauses are given out. This leads to a conjecture, which is also related to the work of [11]. As a result, we conjecture that the following candidate expanding family of degree-3 polynomials is hard to solve.

3SAT Based Candidate.. Let $t = B^2\lambda$. Here, $B(\lambda)$ is the magnitude of the perturbations allowed. Sample each polynomial q'_i for $i \in [\eta]$ as follows. $q'_i(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{z}_1, \ldots, \mathbf{z}_t) = \Sigma_{j\in[t]} q'_{i,j}(\mathbf{x_j}, \mathbf{y_j}, \mathbf{z_j})$. Here $\mathbf{x}_j \in \chi^{d\times n}$ and $\mathbf{y}_j, \mathbf{z}_j \in \chi^n$ for $j \in [t]$. In other words, q'_i is a sum of t polynomials $q'_{i,j}$ over t disjoint set of variables. Let χ denote a discrete gaussian random variable with mean 0 and standard deviation n. Now we describe how to sample $q'_{i,j}$ for $j \in [\eta]$.

1. Sample randomly inputs $\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^* \in \{0,1\}^n$.
2. To sample $q'_{i,j}$ do the following. Sample three indices randomly and independently $i_1, i_2, i_3 \leftarrow [n]$. Sample three signs $b_{1,i,j}, b_{2,i,j}, b_{3,i,j} \in \{0,1\}$ uniformly such that $b_{1,i,j} \oplus b_{2,i,j} \oplus b_{3,i,j} \oplus \mathbf{x}^*[i_1] \oplus \mathbf{y}^*[i_2] \oplus \mathbf{z}^*[i_3] = 1$.
3. Set $q'_{i,j}(\mathbf{x_j}, \mathbf{y_j}, \mathbf{z_j}) = 1 - (b_{1,i,j} \cdot \mathbf{x_j}[i_1] + (1 - b_{1,i,j}) \cdot (1 - \mathbf{x_j}[i_1])) \cdot (b_{2,i,j} \cdot \mathbf{y_j}[i_2] + (1 - b_{2,i,j}) \cdot (1 - \mathbf{y_j}[i_2])) \cdot (b_{3,i,j} \cdot \mathbf{z_j}[i_3] + (1 - b_{3,i,j}) \cdot (1 - \mathbf{z_j}[i_3]))$

References

1. Agrawal, S.: New methods for indistinguishability obfuscation: Bootstrapping and instantiation. IACR Cryptology ePrint Archive **2018**, 633 (2018), https://eprint.iacr.org/2018/633 2, 9, 14
2. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. IACR Cryptology ePrint Archive **2018**, 615 (2018), https://eprint.iacr.org/2018/615 2, 3, 9, 14, 18
3. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. EUROCRYPT (2017) 1
4. Barak, B., Brakerski, Z., Komargodski, I., Kothari, P.: Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). Electronic Colloquium on Computational Complexity (ECCC) **24**, 60 (2017) 2, 7
5. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography **324** (11 2002) 2
6. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. IACR Cryptology ePrint Archive **2014**, 930 (2014), http://eprint.iacr.org/2014/930 1
7. Brakerski, Z., Gentry, C., Halevi, S., Lepoint, T., Sahai, A., Tibouchi, M.: Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845 (2015), http://eprint.iacr.org/ 1

8. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: EUROCRYPT (2015) 1
9. Cheon, J.H., Lee, C., Ryu, H.: Cryptanalysis of the new clt multilinear maps. Cryptology ePrint Archive, Report 2015/934 (2015), http://eprint.iacr.org/ 1
10. Coron, J., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In: CRYPTO (2015) 1
11. Daniely, A., Linial, N., Shalev-Shwartz, S.: From average case complexity to improper learning complexity. In: STOC. pp. 441–448. ACM (2014) 20
12. Feige, U.: Relations between average case complexity and approximation complexity. In: STOC. pp. 534–543. ACM (2002) 19
13. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings (2013) 1
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA. pp. 40–49 (2013) 1
15. Grigoriev, D.: Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. Theor. Comput. Sci. **259**(1-2), 613–622 (2001) 3
16. Gross, D.: Recovering low-rank matrices from few coefficients in any basis. IEEE Trans. Inform. Theory **57**(3), 1548–1566 (2011). https://doi.org/10.1109/TIT.2011.2104999, http://dx.doi.org/10.1109/TIT.2011.2104999 7, 11, 12
17. Halevi, S.: Graded encoding, variations on a scheme. IACR Cryptology ePrint Archive **2015**, 866 (2015) 1
18. Hu, Y., Jia, H.: Cryptanalysis of GGH map. IACR Cryptology ePrint Archive **2015**, 301 (2015) 1
19. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. pp. 28–57 (2016) 1
20. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 prgs. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. pp. 599–629 (2017) 1
21. Lin, H., Matt, C.: Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. IACR Cryptology ePrint Archive **2018**, 646 (2018), https://eprint.iacr.org/2018/646 2, 9, 10, 15
22. Lin, H., Tessaro, S.: Indistinguishability obfuscation from bilinear maps and block-wise local prgs. Cryptology ePrint Archive, Report 2017/250 (2017), http://eprint.iacr.org/2017/250 1, 2
23. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In: IEEE 57th Annual Sympo-

sium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA. pp. 11–20 (2016) 1
24. Lombardi, A., Vaikuntanathan, V.: On the non-existence of blockwise 2-local prgs with applications to indistinguishability obfuscation. IACR Cryptology ePrint Archive **2017**, 301 (2017), http://eprint.iacr.org/2017/301 2
25. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In: Advances in Cryptology - CRYPTO (2016) 1
26. Minaud, B., Fouque, P.A.: Cryptanalysis of the new multilinear map over the integers. Cryptology ePrint Archive, Report 2015/941 (2015), http://eprint.iacr.org/ 1
27. Recht, B.: A simpler approach to matrix completion. Journal of Machine Learning Research **12**, 3413–3430 (2011) 11
28. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. **52**(3), 471–501 (2010). https://doi.org/10.1137/070697835, http://dx.doi.org/10.1137/070697835 7
29. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014. pp. 475–484 (2014) 1
30. Schoenebeck, G.: Linear level lasserre lower bounds for certain k-csps. In: 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA. pp. 593–602 (2008) 3

A Julia Code

```
function genmatrixDMQ(n, C)
    V = randn(n,n)
    for i in 1:n
        for j in 1:n
            V[i,j] = 0
        end
    end

    for i in 1:n
        for j in i:n
            V[i,j] = nr.randint(-C, high=C+1)
        end
    end

    (V'+V)/2
end
```

```
function genxMQ(n ,B)
    x = randn(n,1)
    for i in 1:n
        x[i] = nr.randint(-B, high=B+1)
    end
    x
end

function genobsMQ(n,m,C,B)
    L = [genmatrixDMQ(n,C) for i in 1:m]
    x = genxMQ(n,B)
    obs = [x'*L[i]*x for i in 1:m]
    L,obs,x
end

function recoverMQ(L,obs)

    n = size(L[1])[1]
    m = length(L)

    model = Model(solver = MosekSolver())
    @variable(model, X[1:n,1:n], SDP)

    # let's maximize the trace
    @objective(model, Min, trace(X))

    # this makes the constraints
    for i in 1:m
        @constraint(model, trace(L[i]*X).==obs[i])
    end

    # this solves the problem
    solve(model)
    getvalue(X)

end


function genmatrixsmq(n, n2, nprime, C)
    V = randn(n+nprime+2,n+nprime+2)
    Z = randn(n+nprime+2,n+nprime+2)

    for i in 1:n+nprime+2
```

```
    for j in 1:n+nprime+2
        V[i,j] = 0
    end
end

a=randperm(n)

#sparse monomials

for i in 1:n2
    V[ a[2*i-1]+1,a[2*i]+1] = rand(-C:C)
end



#MQ monomials
for i in n+3:n+nprime+2
    for j in n+3:n+nprime+2
        V[i,j] = rand(-C:C)
    end
end

#Linear terms in S


for j in 2:n+1
    V[1,j]=rand(-C:C)

end

#Linear terms in MQ

for j in n+3:n+nprime+2
    V[n+2,j]=rand(-C:C)
end



Z=V'+V
```

```
    Z

end

function genxdiscsmq(n,n2,nprime,B1,B2,Bprime)
    x = randn(n+nprime+2,1)
    x[1]=1
    x[n+2]=1
    for i in n+3:n+nprime+2
        x[i] = rand(-Bprime:Bprime)
    end

    for i in 2:n+1
        temp1=rand(-B1:B1)
        temp2=rand(-B2:B2)
        temp3=rand(0:1)
        x[i] = (temp3*temp1+(1-temp3)*temp2 )
    end

    x
end

function genobssmq(n,n2,nprime,m,C,B1,B2,Bprime)
    L = [genmatrixsmq(n,n2,nprime,C) for i in 1:m]
    x = genxdiscsmq(n,n2, nprime,B1,B2,Bprime)
    obs = [x'*L[i]*x for i in 1:m]
    L,obs,x
end


function recoverspecialsmq(L,obs,n,n2,nprime,m,val1)




    model = Model(solver = MosekSolver())
    @variable(model, X[1:nprime+n+2,1:nprime+n+2], SDP)

    # let's maximize the trace
    @objective(model, Min, trace(X))

    # this makes the constraints
```

```
    for i in 1:m
        @constraint(model, trace(L[i]*X).==obs[i])
    end

    @constraint(model, X[1,1]==1)
    @constraint(model, X[n+2,n+2]==1)


    @constraint(model, trace(X[1:n+1,1:n+1])=val1[1])


    # this solves the problem
    solve(model)
    getvalue(X)

end

function recoverspecialssm(L,obs,n,n2,nprime,m,val1, val2,val3)
    model = Model(solver = MosekSolver())
    @variable(model, X[1:nprime+2*n+3,1:nprime+2*n+3], SDP)

    # let's maximize the trace
    @objective(model, Min, trace(X))

    # this makes the constraints
    for i in 1:m
        @constraint(model, trace(L[i]*X).==obs[i])
    end


    @constraint(model, X[1,1]==1)
    @constraint(model, X[n+2,n+2]==1)
    @constraint(model, X[n*2+3,2*n+3]==1)



    @constraint(model, trace(X[1:n+1,1:n+1]) + trace(X[n+2:2*n+2,n+2:2*n+2])
    >= val1[1] + val2[1])
     # this solves the problem
    solve(model)
    getvalue(X)
```

```
end
```