Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 66 (2010)

New Algorithms for Learning in Presence of Errors*

Sanjeev Arora
arora@cs.princeton.edu

Rong Ge
rongge@cs.princeton.edu

Department of Computer Science and
Center for Computational Intractability,
Princeton University

Abstract

We give new algorithms for a variety of randomly-generated instances of computational problems using a *linearization* technique that reduces to solving a system of linear equations.

These algorithms are derived in the context of learning with *structured* noise, a notion introduced in this paper. This notion is best illustrated with the *learning parities with noise* (LPN) problem —well-studied in learning theory and cryptography. In the standard version, we have access to an oracle that, each time we press a button, returns a random vector $a \in \mathrm{GF}(2)^n$ together with a bit $b \in \mathrm{GF}(2)$ that was computed as $a \cdot u + \eta$, where $u \in \mathrm{GF}(2)^n$ is a *secret* vector, and $\eta \in \mathrm{GF}(2)$ is a *noise* bit that is 1 with some probability p. Say $p = 1/3$. The goal is to recover u. This task is conjectured to be intractable.

In the structured noise setting we introduce a slight (?) variation of the model: upon pressing a button, we receive (say) 10 random vectors $a_1, a_2, \ldots, a_{10} \in \mathrm{GF}(2)^n$, and corresponding bits $b_1, b_2, \ldots, b_{10}$, of which at most 3 are noisy. The oracle may arbitrarily decide which of the 10 bits to make noisy. We exhibit a polynomial-time algorithm to recover the secret vector u given such an oracle. We think this structured noise model may be of independent interest in machine learning.

We discuss generalizations of our result, including learning with more general noise patterns. We also give the first nontrivial algorithms for two problems, which we show fit in our structured noise framework.

We give a slightly subexponential algorithm for the well-known *learning with errors* (LWE) problem over $\mathrm{GF}(q)$ introduced by Regev for cryptographic uses. Our algorithm works for the case when the gaussian noise is small; which was an open problem.

We also give polynomial-time algorithms for learning the MAJORITY OF PARITIES function of Applebaum et al. for certain parameter values. This function is a special case of Goldreich's pseudorandom generator.

*Research supported by NSF Grants CCF-0832797, 0830673, and 0528414

ISSN 1433-8092

1 Introduction

Sometimes, generating difficult instances of computational problems seems too easy. Of course this can be a boon for cryptography, but on the other hand can frustrate machine learning theorists, since it implies that hard-to-learn instances of learning problems can arise easily.

Consider for instance Goldreich's simple random number generator: it takes a random string u of length n and a fixed predicate f on $d = O(1)$ bits that is balanced (i.e., $|f^{-1}(0)| = |f^{-1}|(1))$, and applies f on m random (or pseudorandom) subsets of u. Can the resulting string of m bits be distinguished from a random m-bit string? If not, we have a pseudorandom generator, and in particular can conclude that the task of *learning* u from the resulting string is also hard.

Or to take another example, consider the *Learning parities with noise* (LPN) problem: we are given a set of data points $(a_1, b_1), (a_2, b_2) \ldots$, where $a_i \in \mathrm{GF}(2)^n$ and $b_i \in \mathrm{GF}(2)$. Each a_i was chosen randomly, and b_i was computed as $a_i \cdot u + \eta_i$, where $u \in \mathrm{GF}(2)^n$ is a *secret* vector, and $\eta_i \in \mathrm{GF}(2)$ is a *noise* bit that is 1 with some probability p. (All a_i's and η_i's are iid.) The goal is to distinguish these m bits from a truly random string. It can be shown that such a distinguisher also allows us to recover u with high probability. All evidence suggests that this recovery problem is hard, and this conjectured hardness is the basis of some cryptosystems [5, 7, 12]. The best algorithm to recover u runs in $2^{O(n/\log n)}$ time (Blum et al. [8]), only very slightly better than the trivial $2^{O(n)}$. The hardness of the LPN problem has important implications for machine learning. A standard technique to make learning algorithms noise-tolerant is using Kearns [14]'s *statistical query* (SQ) model, but parity is perhaps the simplest concept class for which the approach fails (and must fail because parity has exponential SQ dimension). This is frustrating, because the algorithm for learning parities in the noise-free setting is so simple and canonical: gaussian elimination (i.e., solving linear equations). If even this algorithm cannot be made noise-tolerant then the prospects for more complicated learning algorithms seem bleak.

Clearly, we need more algorithmic ideas for this type of problems. Even if they do not always work, they may help us understand parameter ranges where the problem transitions from hard to easy. Sometimes the success/failure of specific algorithms can lead to interesting new research questions, as happened with phase transitions for Random3SAT. If the problem is being used in cryptography, then having several candidate algorithms —and the parameter values for which they fail—is useful for setting parameter values in practice. Regev's *Learning with Errors* (LWE) problem, a generalization of LPN, provides a good example. This problem has become ubiquitous in new cryptographic research in the last few years, since it shares similar hardness properties as lattice-based cryptosystems a la Ajtai [1], Ajtai-Dwork [3] and others, while having more algebraic structure which enables new uses (e.g., oblivious transfer protocol by Peikert et al. [17], and a leakage-resistant crytosystem by Akavia et al. [4]; see Micciancio and Regev [15] for a survey). There are no nontrivial algorithms for this problem, and it is known to become as hard as (worst-case) approximate lattice vector problems when the *noise* parameter is more than $\sqrt{n}$. Does it have any nontrivial algorithms for noise below $\sqrt{n}$ (or for that matter, any other noise rate)? This is not an idle question since the problem structure could be quite different for different noise rates; for example in case of LPN the known construction for public-key cryptosystems from LPN [5] need to assume the hardness of LPN for noise rate below $1/\sqrt{n}$.

In this paper we use an algorithmic technique called *linearization* to make progress on some of these questions. The main idea is an obvious one and similar to the linearization technique used in practical cryptanalysis [6]: given a nonlinear equation that holds among some variables, introduce new variables for the monomials and obtain a linear system in the new variables. (Similar linearization ideas underlie Sherali-Adams lift and project methods in linear programming.) Our

contribution is to fruitfully apply this idea to the above settings and analyze it. We hope this will inspire new results.

To illustrate our idea in the simplest way we introduce a variant on LPN that does not appear to have been considered before and that, surprisingly, turns out to be tractable using our methods. We call it the *LPN problem with structured noise.* In the standard noise model, pushing an oracle button gives us a random $a \in \mathrm{GF}(2)^n$ and a corresponding noisy value $b \in \mathrm{GF}(2)$ that is wrong with probability p. In our model, pushing the oracle button returns m random vectors $a_1, a_2, \ldots, a_m \in \mathrm{GF}(2)^n$ and m bits $b_1, b_2, \ldots, b_m$, of which at most p fraction are noisy (in other words, for at least $(1-p)m$ of the indices i, we are guaranteed $a_i \cdot u = b_i$). This may be viewed as a guarantee that the data errors, though still happening p fraction of the time, are not arbitrarily bursty. Every sample returned by the oracle contains untainted data. Perhaps this and other structured noise models should be considered in machine learning since the standard noise model has often led to intractable problems. Our algorithm for LPN with structured noise runs in time that is polynomial in n^m, which is polynomial if m is a constant. Interestingly, it works even if the oracle is allowed to look at a_i's while deciding where to add noise. It also works for more general "noise patterns" as described later.

The idea in our algorithm is fairly simple: the above structured noise model implies that the hidden secret u is the solution to a system of degree-m constraints. These degree-m constraints are *linearized* by replacing monomials $\prod_{i \in S} u_i$ with a new variable y_S, thus obtaining a linear constraint in $\binom{n}{m}$ variables. Then we prove that the resulting system has a unique solution. The proof requires some work; see Section 2. We also give an exact characterization of noise-models for which our technique works.

We also apply the linearization technique to give a subexponential algorithm for the LWE problem with noise rate below $\sqrt{n}$, which clarifies why existing hardness results (which show that LWE is as hard as lattice approximation [18, 16]) did not extend to this case.

Then we apply the linearization technique to an important subcase of Goldreich's generator, whereby f = MAJORITY of three XORs. This was proposed by Applebaum, Barak and Wigderson [2] as a suitable choice of f following some earlier attacks on Goldreich's generator for "structured" f's (Bogdanov and Qiao [9]). We show that if $m = \tilde{\Omega}(n^2)$ then the secret vector u can be efficiently learned from the output of the generator. (Note that Applebaum et al. had proposed $m = n^{1.1}$ as a safe parameter choice, which is not contradicted by our results. However, thus far there was no result indicating that even $m = n^{d/3}$ is an unsafe choice.)

An explanation for the ubiquity of linearization Of course, our linearization idea fits in a long line of mathematical results that involve reducing a complicated mathematical problem to linear algebra —specifically, solving linear equations via gaussian elimination. Thus it is natural to think of "reduction to gaussian elimination" as a simple (and possibly restricted) computational model and to consider its strengths and limitations. In our algorithm, the reduction to gaussian elimination actually involves equations whose coefficients are low-degree polynomials in the data values. We can call this technique "reduction to gaussian elimination with polynomial coefficients." While trying to prove limitations of this restricted model for problems such as LPN and LWE, we ended up with an observation (see Section C) that we have not seen earlier: every algebraic algorithm (i.e., an algebraic circuit) for a decision problem can be converted into one that "reduces to gaussian elimination with polynomial coefficients." This gives some justification for the naturalness of this algorithm design technique and also shows that lowerbounds for this technique will be difficult until we make progress on circuit lower bounds.

2 Learning Parities with Structured Noise

This section introduces our linearization technique in context of the LPN problem. The following is the most general kind of oracle our algorithm can work with. The oracle has an associated polynomial P in m variables over GF(2). Any $\eta \in \mathrm{GF}(2)^m$ such that $P(\eta) = 0$ is an *acceptable noise pattern*. Each time a button is pressed the oracle picks m random $a_1, a_2, \ldots, a_m \in \mathrm{GF}(2)^n$, and then picks an arbitrary acceptable noise pattern η. Then it outputs $a_1, a_2, \ldots, a_m$ together with m bits $b_1, b_2, \ldots, b_m$ defined as $b_i = a_i \cdot u + \eta_i$. For example, the polynomial such that $P(\eta) = 0$ iff η is a vector of hamming weight at most $m/3$ corresponds to the intuitive notion of "noise rate 1/3." This polynomial also satisfies the hypothesis of the main theorem below.

Note that this noise model allows the oracle to look at the a_i's in choosing the noise pattern (this is analogous to the *agnostic-learning* or *worst-case* noise model of the standard LPN, which Khot et al. [10] have shown to be equivalent to the *white noise* case.)

Our algorithm requires $P(\cdot)$ to be such that there is at least one $\rho \in \mathrm{GF}(2)^m$ such that $\rho \neq \eta + \eta'$ for all η, η' satisfying $P(\eta) = P(\eta') = 0$. For example, we could have $P(\eta) = 0$ iff the number of 1's in η is fewer than $m/2 - 1$, corresponding to a noise rate of less than 1/2. But one can conceive of more exotic noise polynomials.

Theorem 2.1 (Main) *Suppose $P(\cdot)$ is such that there is at least one $\rho \in GF(2)^m$ such that $\rho \neq \eta + \eta'$ for all η, η' satisfying $P(\eta) = P(\eta') = 0$. Then there is an algorithm that learns the secret u in time* $\mathrm{poly}(n^m)$ *time.*

There is an obvious generalization of this theorem (see Section B in the appendix) to learning degree d polynomials (where parity is the subcase $d = 1$). The running time is $\mathrm{poly}(n^{dm})$. This also leads to a learning algorithm for low-depth decision trees in our structured noise model, since depth-d decision trees can be represented by degree d polynomials.

In this section we do a simpler version of Theorem 2.1 where the oracle's noise pattern η is not allowed to depend upon the points $a_1, a_2, \ldots, a_m \in \mathrm{GF}(2)^n$. We think of this as *white noise*. The proof of the general result appears in the Appendix.

The learning algorithm has access to an oracle $Q(u, m, P, \mu)$, where $u \in \mathrm{GF}(2)^n$ is the secret vector. Let m be an integer and $P(\eta_1, \eta_2, ..., \eta_m)$ be a multilinear polynomial over the vector $\eta \in \mathrm{GF}(2)^m$ of degree d (P cannot be identically 0). Let μ be a distribution over the values of η that satisfies $P(\eta) = 0$. The algorithm knows m and P, and tries to compute the secret u by querying the oracle. (The algorithm does not know the distribution μ).

When queried by the algorithm, the oracle with secret $u \in \mathrm{GF}(2)^n$ returns m random vectors $a_1, a_2, \ldots, a_m \in \mathrm{GF}(2)^n$ together with m bits $b_1, b_2, \ldots, b_m \in \mathrm{GF}(2)$. The b values are determined by first choosing a random vector η that satisfies $P(\eta) = 0$ using the distribution μ, then computing $b_i = a_i \cdot u + \eta_i$.

Many structures can be expressed in this framework, for example, "at least one equation $a_i \cdot u = b_i$ is satisfied" can be expressed by $P(\eta) = \prod_{i=1}^{m} \eta_i$. In fact, any "structure" within the group of size m that excludes at least one value of η can be expressed by a non-zero polynomial.

Let z be an n dimensional vector, z_i's are considered to be variables. The algorithm uses the sample returned by the oracle to write a polynomial constraint for the variables z_i's. Ideally the solution to the equations will be $z = u$.

First write the formula for η that is known to be satisfied given the oracle Q:

$$P(\eta_1, \eta_2, ..., \eta_m) = 0. \tag{1}$$

Then substitute $\eta_i = a_i \cdot z + b_i$ to obtain

$$P(a_1 \cdot z + b_1, a_2 \cdot z + b_2, \ldots, a_m \cdot z + b_m) = 0, \tag{2}$$

which is a degree d polynomial constraint in the variable vector z that is always satisfied by the samples obtained from the oracle when $z = u$. Since $z_i^2 = z_i$ in GF(2), without loss of generality such a polynomial is multilinear.

Now convert this polynomial constraint to a linear one by the obvious linearization: For each $S \subseteq [n]$ and $|S| \leq d$ replace the monomial $\prod_{i \in S} z_i$ by the new variable y_S. Thus (2) turns into a linear constraint in a vector y of $N = \sum_{i=1}^{d} \binom{n}{i}$ new variables. Ideally the solution should have y being the "tensor" of u, that is, $y_S = \prod_{i \in S} u_i$, and this is what we will show.

Properties of this linearization process are important in our proof so we define it more formally.

Definition 2.2 (linearization) *Let $p(z) = \sum_{S \subseteq [n], |S| \leq d} c_S \prod_{i \in S} z_i$, be a multilinear polynomial of degree d in n variables where the coefficients c_S's are in GF(2). The* linearization *of p, denoted $L(p)$, is a linear function over the variables y_S, where S ranges over subsets of $[n]$ of size at most d:*

$$L(p) = \sum_{S \subseteq [n], |S| \leq d} c_S y_S.$$

We will assume there is a variable $y_\emptyset$ that is always 1, so the number of new variables is $N + 1 = \sum_{i=0}^{d} \binom{n}{i}$

In this section z will denote a vector of variables, and y is the vector of all variables used in linearizing degree d polynomials in z. Thus y has dimension $N + 1$, and is indexed by $S \subseteq [n]$ and $|S| \leq d$.

Our learning algorithm will take $\text{poly}(N, 2^{m+d})$ samples from the oracle, thus obtaining a linear system in N variables and $\text{poly}(N, 2^{m+d})$ constraints. Each constraint is the linearization of equation (2):

$$L(P(a_1 \cdot z + b_1, a_2 \cdot z + b_2, \ldots, a_m \cdot z + b_m)) = 0. \tag{3}$$

Note that the set of constraints always has at least one solution, namely, the vector y obtained from the hidden vector u. We will show that whp all solutions will reveal the hidden vector u.

Theorem 2.3 *The linear system obtained by $10N2^{m+d}$ samples always has at least one solution, and with high probability (over the oracle's random choices) all solutions to the system satisfy $y_{\{i\}} = u_i$.*

We will need the following version of Schwartz-Zippel Lemma, which is essentially the fact that degree d Reed-Muller Codes over GF(2) have distance 2^{-d}. We call it "Schwartz-Zippel" because that is the name given to a similar lemma over GF(q).

Lemma 2.4 (Schwartz-Zippel) *If p is a nonzero multilinear polynomial over GF(2) of degree d, then*

$$\mathbf{Pr}_{x \in U}[p(x) \neq 0] \geq 2^{-d}$$

Recall that for any n-dimensional vector z the tensor product $z^{\otimes k}$ is the vector in n^k dimensions whose component corresponding to $(i_1, i_2, i_3, ..., i_k) \in [n]^k$ is $z_{i_1} z_{i_2} \cdots z_{i_k}$. In the proof we will be often interested in linearization of tensors of the vector $(z + u)$, where u is the secret of the oracle and z is a vector of variables. The components u_i are considered to be constants. Let $Z^k = (z + u)^{\otimes k}$. Each component of Z^k is a multilinear polynomial over z of degree at most k. Let us linearize each component separately, and obtain vector Y^k. Notice that components of Y^k form a subset of components in Y^{k+1}, because for any index $(i_1, i_2, ..., i_k)$, $Z^k_{i_1 i_2 \cdots i_k} = \prod_{j=1}^{k} (z_{i_j} + u_{i_j}) =$

$(z_{i_k} + u_{i_k}) \prod_{j=1}^{k} (z_{i_j} + u_{i_j}) = Z^{k+1}_{i_1 i_2 \cdots i_k i_k}$, so it is equivalent to the component corresponding to the vector $(i_1, ..., i_k, i_k)$ (the original index with last component repeated) in Z^{k+1}.

Each component of Y^d is a linear combination of the coordinates of vector y. We represent this linear transformation using a matrix M_u, in other words $Y^d = M_u y$. This linear transformation maps y to a higher dimensional vector, in the analysis we consider Y^d as a new set of variables. Sometimes we will also use Y^k where $k < d$; these are replaced by variables in Y^d that have the same value.

In the next lemma, the $\otimes$ notation denotes tensor product. Note that every degree d polynomial in variables z can be represented as $\mathbf{c} \cdot z^{\otimes d}$ where $\mathbf{c}$ is the coefficient vector.

Lemma 2.5 *Let a_1, a_2, ..., a_m be vectors of n variables each. Consider a linearized polynomial constraint of the form*

$$\sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S (\otimes_{i \in S} a_i) \cdot Y^{|S|} = 0, \tag{4}$$

where c_S are coefficients in GF(2). This is a homogeneous linear constraint in terms of Y^d (here "homogeneous" means there's no constant term, so the equation is satisfied when $Y^d = \mathbf{0}$) and a degree d polynomial constraint in terms of the variables in a_i's.

Let an assignment to Y^d be such that equation (4) is satisfied for all possible values of a_i's. If there is a subset S of size k such that $c_S = 1$, then $Y^k = \mathbf{0}$.

Proof: Assume towards contradiction that $Y^k \neq \mathbf{0}$. Let $I = \{i_1, i_2, ..., i_k\}$ be the set of size k with $c_I = 1$, and let $(j_1, j_2, ..., j_k)$ be an index where Y^k is nonzero. We use $a_{i,j}$ to denote the j-th component of the vector a_i. Consider the polynomial over $\{a_{i,j}\}$

$$\sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S (\otimes_{i \in S} a_i) \cdot Y^{|S|}.$$

The monomial $\prod_{t=1}^{k} a_{i_t, j_t}$ appears in the sum only when $S = I$, and it is the monomial in the component of $(\otimes_{i \in S} a_i)$ whose index is $(j_1, j_2, ..., j_k)$. If the polynomial is represented as the sum of monomials, $\prod_{t=1}^{k} a_{i_t, j_t}$ has coefficient $c_I Y^k_{j_1, j_2, ..., j_k} = 1$. Therefore for this assignment of Y^d, the left hand side of Equation (4) is a polynomial over $\{a_{i,j}\}$ that is not identically 0, it cannot be 0 for all possible values of a_i's. This contradicts with the hypothesis, so we must have $Y^k = \mathbf{0}$.

■

Now we are ready to prove Theorem 2.3:

Proof: The proof consists of inverting the viewpoint about what is a variable and what is a constant. Constraint (2) is properly viewed as a polynomial constraint in the z's as well as a_i's and η_i's.

It's easy to see that if we set $y_S = \prod_{i \in S} u_i$ then Equation (2) becomes $P(\eta) = 0$ and is always satisfied. Now we show that every wrong vector y_S results in a nonzero polynomial over the a_i's, which is not satisfied for random choices of a_i's.

Substituting $b_i = a_i \cdot u + \eta_i$, Equation (2) becomes

$$P(\eta_1 + a_1 \cdot (z + u), \eta_2 + a_2 \cdot (z + u), ..., \eta_m + a_m \cdot (z + u)) = 0. \tag{5}$$

We observe there must be a value of η that get picked by distribution μ with probability at least $1/2^m$. Let $\eta^* \in \mathrm{GF}(2)^m$ be this value, we have $\mathbf{Pr}[\eta = \eta^*] \geq 1/2^m$ and $P(\eta^*) = 0$. We show

that, incorrect y's will have difficulty even restricting η to η^*. Assume the event $\eta = \eta^*$ happens, Equation (2) becomes

$$P(a_1 \cdot (z+u) + \eta_1^*, ..., a_m \cdot (z+u) + \eta_m^*) = 0. \tag{6}$$

The polynomial $P(a_1 \cdot (z+u) + \eta_1^*, ..., a_m \cdot (z+u) + \eta_m^*)$ is then expanded into a similar form as Equation (4). In the expansion, $a_i \cdot (z+u)$ are considered variables, η_i^* are constants, and when multiplying variables in a monomial we use tensor of a_i's:

$$P(a_1 \cdot (z+u) + \eta_1^*, ..., a_m \cdot (z+u) + \eta_m^*) = P(\eta^*) + \sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S(\otimes_{i \in S} a_i) \cdot Z^{|S|}. \tag{7}$$

Here the constant term is exactly $P(\eta^*)$ (which is equal to 0), because if we express P as the sum of monomials, each monomial of the form $\prod_{i \in S}(a_i \cdot (z+u) + \eta_i^*)$ has constant term $\prod_{i \in S} \eta_i^*$.

Now we linearize (7) by replacing Z^k with Y^k, and use the fact $P(\eta^*) = 0$ to get

$$\sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S(\otimes_{i \in S} a_i) \cdot Y^{|S|} = 0. \tag{8}$$

We can view this as a linear constraint over the variables Y^d, and because (3) and (8) are the same constraint (they are linearizations of the same polynomial), if y satisfies (3) then $Y^d = M_u y$ satisfies (8). The left hand side of Constraint (8) cannot be identically 0 because Constraint (3) is not trivial. Let S be a subset such that $c_S = 1$, and let its size be k. We claim that with high probability all solutions will have $Y^k = \mathbf{0}$.

For any nontrivial $Y^d = M_u y$ such that $Y^k \neq \mathbf{0}$, by Lemma 2.5, the left hand side of (8) is a nonzero polynomial over the a_i's. This polynomial has degree at most d, by Schwartz-Zippel Lemma it must be 1 with probability at least 2^{-d}. The right hand side of Equation (8) is always 0. Hence when $\eta = \eta^*$ a solution with $Y^k \neq \mathbf{0}$ will violate the constraint with probability at least 2^{-d}. The probability that such a solution violates a random constraint is at least 2^{-m-d} because $\mathbf{Pr}[\eta = \eta^*] \geq 2^{-m}$. Since constraints are independent the probability that this assignment Y^d satisfies all $10N2^{d+m}$ constraints is at most $(1 - 2^{-d-m})^{10N2^{d+m}} \leq e^{-10N}$.

For any solution y such that $Y^d = M_u y$ and $Y^k \neq \mathbf{0}$, we know the probability that it satisfies all constraints is at most e^{-10N}. By union bound the probability that there exists a y' such that the corresponding $Y^k \neq \mathbf{0}$ and it satisfies all constraints is at most $e^{-10N}2^N \ll 1$. Therefore with high probability all solutions to the system of linear equations satisfy $Y^k = \mathbf{0}$. When $Y^k = \mathbf{0}$, consider the component $(i, i, ..., i)$ of Y^k, by definition it is equal to $y_{\{i\}} + u_i$, and we know it is 0, hence $y_{\{i\}} = u_i$.

Therefore with high probability, all solutions will have $y_{\{i\}} = u_i$. ∎

3 Learning With Errors

The learning with errors problem is a generalization of LPN to a larger field. There is an unknown vector $u \in \mathrm{GF}(q)^n$ where q is a prime number that is usually bounded by a polynomial of n. The algorithm is given noisy samples of the type $a \cdot u + \eta$ where the noise η is generated according to the *Discrete Gaussian* distribution Ψ_α with standard deviation αq, which consists of picking a random real number r from the (usual) Gaussian distribution with standard deviation $\sigma = \alpha q$ (i.e., with density function $D_\sigma(r) = 1/\sigma \exp(-(\pi r/\sigma)^2)$), then rounding it up to the nearest integer $\lfloor r \rceil$ and outputting $\lfloor r \rceil (\mathrm{mod}\ q)$. The algorithm is given access to an oracle $Q(u, \Psi_\alpha)$. which picks

$a \in \mathrm{GF}(q)^n$ uniformly at random, then picks η independent of a from distribution Ψ_α, and returns a, b where $b = a \cdot u + \eta$. It is conjectured that no polynomial time algorithm can learn the secret u for some specific parameters, and the best known algorithm works in exponential time. We give the first nontrivial algorithm for LWE:

Theorem 3.1 *When $\alpha q = n^\varepsilon$ where ε is a constant strictly smaller than $1/2$, and $q \gg (\alpha q \log n)^2$, there is a $2^{\tilde{O}(n^{2\varepsilon})}$ time algorithm algorithm that learns u when given access to the oracle $Q(u, \Psi_\alpha)$.*

The theorem is derived by showing that the low error case fits in our "structured noise" setting, for which we show a subexponential algorithm. In the structured noise model the algorithm has access to a different oracle $Q(u, \Psi_\alpha, d)$. The new parameter d is an integer that is considered to be the "bound" of the error η and satisfies $2d + 1 < q$. The oracle ensures that the noise η is picked uniformly from Ψ_α conditional on it being at most d in magnitude. (We always interpret η as an integer between $-(q-1)/2$ and $(q-1)/2$.)

We will rely on two well known facts about the gaussian distribution:

Lemma 3.2 *For $\eta \in [-(q-1)/2, (q-1)/2]$, we have*

1. $\mathbf{Pr}_{\eta \sim \Psi_\alpha}[|\eta| > k\alpha q] \leq e^{-O(k^2)}$.
2. $\mathbf{Pr}_{\eta \sim \Psi_\alpha}[\eta = 0] = \Omega(1/\alpha q)$.

Part 1 implies (we omit the simple calculation) that if we draw fewer than $e^{o(k^2)}$ samples from the standard oracle then these are statistically very close to samples from the structured noise oracle $Q(u, \Psi_\alpha, d)$ for $d = k\alpha q$. Thus from now on we will assume that samples are drawn from the latter. Part 2 of the lemma will be useful in the midst of the proof.

The algorithm works similarly as the algorithm for LPN, with an additional twist needed because the field is not GF(2), so the underlying polynomials are non-multilinear. We first write a univariate degree $2d + 1$ polynomial P such that $P(\eta) = 0$ whenever η is drawn from oracle $Q(u, \Psi_\alpha, d)$.

$$P(\eta) = \eta \prod_{i=1}^{d} (\eta + i)(\eta - i). \tag{9}$$

From now on we use similar notation as in the LPN section. We use z (an n-dimensional vector) as variables, and try to write a system of equations which whp have solutions that allow us to recover u. To do this we substitute $\eta = a \cdot z + b$ in the polynomial $P(\eta)$ to obtain a degree $2d + 1$ polynomial over the variables z_i: $(a \cdot z + b) \prod_{i=1}^{d} (a \cdot z + b + i)(a \cdot z + b - i) = 0$.

This constraint is always satisfied if $z = u$. Let $D = 2d + 1$ denote the degree of the polynomial $P(\eta)$. Finally we linearize this equation using variable vector y that is indexed by vectors $v \in \mathbf{Z}^n$ such that $1 \leq \sum_{i=1}^{n} v_i \leq D$. The variable y_v corresponds to the monomial $\prod_{i=1}^{n} z_i^{v_i}$. We denote the degree of this monomial namely $\sum_{i=1}^{n} v_i$ as $deg(v)$ or $deg(y_v)$. For simplicity we add one component $y_{\mathbf{0}}$, which always has the value 1. The number of variables is $N = \binom{n+D}{n}$. We define y^k to be the vector of all the variables with degree k. Thus $y = (1, y^1, y^2, ..., y^D)$.

The new linearization operator L replaces each monomial in the polynomial with the corresponding y variable. The linearized equation will be a linear constraint on the y variables,

$$L\left(P(a \cdot z + b)\right) = 0. \tag{10}$$

The algorithm queries the oracle $O(N\alpha q^2 \log q)$ times, generating a system of linear equations over the variables y's.

Theorem 3.3 *With high probability, all solutions to the system of linear equations generated as above satisfy $y_{e_i} = u_i$ where e_i is the vector that has 1 in the i-th coordinate and 0 elsewhere.*

Proof: Clearly the system always has a solution: the one where $y_v = \prod_{i=1}^{n} u_i^{v_i}$. With this choice of y_v's constraint (10) is equivalent to $P(\eta) = 0$, which holds. Now we prove that whp the system has no solution when some $y_{e_i} \neq u_i$.

The proof is similar to that for Theorem 2.3. We consider a new set of variables $\tilde{y}$ indexed the same way as the vector y, and the coordinate of $\tilde{y}$ indexed by vector v corresponds to the linearization of $\prod_{i=1}^{n}(z_i + u_i)^{v_i}$. Let $\tilde{y}^k$ denote the entries corresponding to the "degree k" terms in $\tilde{y}$. Thus each component of $\tilde{y}$ is a linear combination of coordinates of y. We denote the linear transformation from y to $\tilde{y}$ by a matrix M_u, that is, $\tilde{y} = M_u y$. It's easy to see that M_u defines a bijection, because the linearization of $\prod_{i=1}^{n}(z_i + u_i)^{v_i}$ will only contain variables with equal or smaller degree, and there's exactly one variable with the same degree as v which is y_v. Hence when the order of y vector is $(1, y^1, y^2, ..., y^D)$ and the order of $\tilde{y}$ vector is $(1, \tilde{y}^1, \tilde{y}^2, ..., \tilde{y}^D)$, the matrix M_u is a lower triangular matrix with 1's in the diagonal. Therefore M_u is invertible and defines a bijection between y and $\tilde{y}$, and $y_{e_i} \neq u_i$ iff $\tilde{y}^1_{e_i} \neq 0$.

Thus to finish the proof it suffices to show that no solution has $\tilde{y}^1 \neq 0$ whp. This follows from the following Claim, since it implies that the probability that an incorrect $\tilde{y}$ satisfies all $CN\alpha q^2 \log q$ constraints is at most q^{-2N} when the constant C is sufficiently. A union bound completes the proof since the number of possible incorrect solutions is at most q^N.

Claim: *When the linear constraints are generated as above, then each candidate solution with $\tilde{y}^1 \neq 0$ violates a randomly generated constraint with probability at least $\Omega(1/\alpha q^2)$.*

Recall that Lemma 3.2 part 2, the noise $\eta =$ with probability at least αq (independent of the choice of the a_i's). If $\eta = 0$ then the constraint that the algorithm writes, is (by substituting $b = a \cdot u$ into constraint (10) simply $L(P(a \cdot (z + u))) = 0$ where L is the linearization operator and P is the polynomial in (9).

The corresponding linear constraint is obtained by thinking of $(z + u)$ as the variables, and doing linearization by replacing $(z + u)^{\otimes k}$ with $\tilde{Y}^k$ ($\tilde{Y}^k$ is similar to Y^k in Section 2 and is defined later), yielding

$$\sum_{i=1}^{D} c_i (a^{\otimes i} \cdot \tilde{Y}^i) = 0, \tag{11}$$

where c_i's are the coefficients of the univariate polynomial $P(\eta)$. Notice that $c_1 = \prod_{i=1}^{d} i(-i) \neq 0$, Lemma 3.4 below implies when $\tilde{y}^1 \neq \mathbf{0}$ the above multivariate polynomial over the a's is not identically zero. The standard version of Schwartz-Zippel Lemma says that when the a's are chosen randomly the polynomial in (11) can be 0 with probability at most D/q. Since $D < q$ the left hand side is nonzero with probability at least $1/q$. Therefore a nonzero $\tilde{y}$ violates a random constraint with probability at least $\Omega(1/\alpha q)1/q$, and this finishes the proof of the Claim and of the theorem. ∎

Now we prove the statement alluded to above. Let $\tilde{Y}^k$ be a vector of dimension n^k that is indexed by $w \in [n]^k$. For any $w \in [n]^k$, define the corresponding vector $v(w) \in \mathbf{Z}^n$, where the i-th component of $v(w)$ is the number of coordinates in w that are equal to i (for example if $w = (1, 2, 3, 2, 2, 3)$ then $v = (1, 3, 2)$). Each component of $\tilde{Y}^k$ is a variable in $\tilde{y}^k$, and $\tilde{Y}^k_w = \tilde{y}_{v(w)}$. Now suppose we have a polynomial over a_i's as the LHS of Equation (11), where $\tilde{Y}^i$ and c_i's are considered to be constants.

We have the following analog of Lemma 2.5 whose proof is in Section D.

Lemma 3.4 *The LHS Polynomial of Equation (11) over the a_i's is identically 0 if and only if for all $i \in [D]$ such that $c_i \neq 0$, we have $\tilde{y}^i = \mathbf{0}$.*

4 Learning the Majority of Parities Function

Applebaum et al. [2] proposed the"DSF assumption" and showed how to use it (with other assumptions) to build public-key cryptosystems. Let $\mathcal{M}_{m,n,d}$ be a random bipartite graph with m vertices on top, n vertices at the bottom, and d edges from each top vertex. If G is such a graph, f is a function $f : \{0,1\}^d \to \{0,1\}$, define the function $G_f : \{0,1\}^n \to \{0,1\}^m$ to be the function obtained by mapping every $u \in \{0,1\}^n$ to $(f(u_{\Gamma(1)}), \ldots, f(u_{\Gamma(m)}))$, where $\Gamma(i)$ denotes the neighbors of the i-th "top" vertex. The DSF assumption asserts the existence of a function f such that $(G, G_f(U_n))$ is ε-indistinguishable from the distribution (G, U_m) when $G \in_R \mathcal{M}_{m,n,d}$. Here d is a large constant. When $m = n$ this is conjectured by Goldreich [11], but in [2] m is required to be super-linear.

To avoid known attacks by Bogdanov and Qiao [9], Applebaum et al. suggested the "majority of three parities" (that is, f is the majority of three parities on $d/3$ bits each) as a candidate function for f. Indeed, they showed when $m = O(n^{1.1})$, this function has nice properties such as looking pseudorandom for AC^0 circuits (and the proofs can be generalized when $m = o(n^2)$). However, using our algorithm we can show when $m = \Omega(n^2 \log n)$, the function G_f fails to be a pseudorandom generator in a really severe way: not only is the output no longer indistinguishable from uniform distribution, but the function G_f is also invertible with high probability.

Our algorithm is designed by noting that the "majority of parities" function can actually be viewed as an answer given by a learning parities with structured noise oracle. Given $Maj(a_1 \cdot u, a_2 \cdot u, a_3 \cdot u) = b$, where $u \in \{0,1\}^n$ is the input, and a_1, a_2, a_3 are vectors with exactly $d/3$ 1's that are obtained when applying G_f, we can write a group of linear equations: $a_1 \cdot u = b$, $a_2 \cdot u = b$, $a_3 \cdot u = b$. Since b is the *m*ajority of these three parities, we know that at least two out of the three equations are satisfied. This is exactly the kind of structure our LPN algorithm can work with. We can represent this structure by $P(\eta) = \eta_1\eta_2 + \eta_1\eta_3 + \eta_2\eta_3$ where $\eta_i = a_i \cdot u + b$. We could try to use our earlier LPN algorithm, except the earlier analysis does not apply because the a vectors are not uniformly random —each of a_1, a_2, a_3 is randomly chosen from all vectors that have $d/3$ 1's, and have disjoint support. Also, once we fix the input u, the error η is dependent on the a vectors, which we will need to deal with it differently than in Section A. We will see that all our calculations become easier when $m = n^4$ instead of $m = n^2 \log n$.

The algorithm will be analogous to our LPN algorithm. We write out the Equation (8) for this structure explicitly by expanding $P(\eta_1, \eta_2, \eta_3)$:

$$(a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2) \cdot Y^2 + ((\eta_1 + \eta_2)a_3 + (\eta_2 + \eta_3)a_1 + (\eta_1 + \eta_3)a_2) \cdot Y^1 = 0, \qquad (12)$$

where Y^2, Y^1 are linearizations of $(z+u) \otimes (z+u)$ and $(z+u)$ respectively. Unlike in case of LPN, having enough equations of this form *does not* guarantee a unique solution. Nevertheless we can show that the solutions to the system of linear equations allow us to learn the secret u.

A key observation is $\eta_1 + \eta_2 = (a_1 + a_2) \cdot u$. Indeed, when $(a_1 + a_2) \cdot u = 0$, it means $a_1 \cdot u = a_2 \cdot u$, so $\eta_1 = \eta_2 = 0$; when $(a_1 + a_2) \cdot u = 1$, it means $a_1 \cdot u \neq a_2 \cdot u$, so exactly one of the equations is incorrect, and we have $\eta_1 + \eta_2 = 1$. Applying this observation to (12), we get

$$(a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2) \cdot (Y^2 + u \otimes Y^1 + Y^1 \otimes u) = 0. \qquad (13)$$

Let W denote $(Y^2 + u \otimes Y^1 + Y^1 \otimes u)$. Since the diagonal entries of W do not affect the equation we will assume $W_{i,i} = 0$ below. For distinct $p, q, s, t \in [n]$, let $W_{(p,q)\times(s,t)}$ denote the sum $W_{p,s} + W_{p,t} + W_{q,s} + W_{q,t}$.

CLAIM 1: *There is a polynomial-time algorithm that, given any solution Y^2, Y^1 for which $W_{(p,q)\times(s,t)} = 0$ for all p, q, s, t, finds the secret u.*

Proof: Recall that $W_{i,j} = L((z_i + u_i)(z_j + u_j) + u_i(z_j + u_j) + u_j(z_i + u_i)) = y_{\{i,j\}} + u_i u_j$, where L is the linearization operator. Since $W_{(p,q)\times(s,t)} = 0$ for every p, q, s, t, we obtain $y_{\{p,s\}} + y_{\{p,t\}} + y_{\{q,s\}} + y_{\{q,t\}} = u_p u_s + u_p u_t + u_q u_s + u_q u_t$.

Now either $u = \mathbf{0}$ or $u = \mathbf{1}$ (both of which possibilities can be easily checked by substitution) or else there is some p, q such that $u_p = 0$, $u_q = 1$, and assume we have found such p, q since we can exhaustively try all pairs. Then $u_p u_s + u_p u_t + u_q u_s + u_q u_t = u_s + u_t$. Since we know p, q this implies we know $u_s + u_t$ for every s, t, in other words whether or not $u_s = u_t$. That leaves only two possibilities for the vector u and we can easily verify which one is correct. ∎

Now we give the simpler analysis for $m = O(n^4)$.

CLAIM 2: *$O(n^4)$ equations suffice whp to rule out all solutions in which $W_{(p,q)\times(s,t)} = 1$ for some p, q, s, t.*

Proof: We show that if p, q, s, t are such that $W_{(p,q)\times(s,t)} = 1$, then Equation (13) is violated with probability $\Omega(1/n^2)$. Since the number of possible solutions W is 2^{n^2} a simple union bound yields the claim.

Let A_i denote the set whose indicator vector is a_i; thus A_i is a random set of size $d/3$ and A_1, A_2, A_3 are disjoint. Consider the event E that $|A_1 \bigcap \{p, q, s, t\}| = 0$, $|A_2 \bigcap \{p, q\}| = |A_3 \bigcap \{s, t\}| = 1$, $|A_2 \bigcap \{s, t\}| = |A_3 \bigcap \{p, q\}| = 0$. That is, exactly one of p, q appears in A_2 and exactly one of s, t appears in A_3. It's easy to see that this happens with probability $\Omega(1/n^2)$. Now fix A_1, $A_2 \backslash \{p, q\}$, $A_3 \backslash \{s, t\}$ and let the corresponding indicator variables be a_1, a_2^*, a_3^*. Now and consider the four possibilities depending upon which of p, q appears in A_2 and which of s, t appear in A_3: either $a_2 = a_2^* + e_p$ or $a_2 = a_2^* + e_q$, and either $a_3 = a_3^* + e_s$ or $a_3 = a_3^* + e_t$ (e_i is the vector which is 1 only at position i). The sum of the expression $(a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2)$ over these four possibilities evaluates to exactly the matrix $(e_p + e_q) \otimes (e_s + e_t)$, because all other terms appear even number of times.Therefore the sum of LHS of Equation (13) is exactly $W_{(p,q)\times(s,t)}$. Since $W_{(p,q)\times(s,t)}$ is 1, in at least one of the four cases Equation (13) is violated, and this happens with probability 1/4 conditioned on the event E. ∎

We leave the proof that $O(n^2 \log n)$ equations actually suffice for Claim 2 to Section E.

5 Conclusions

Linearization techniques have been applied in various contexts, and in this paper we manage to give provable bounds for this idea in several contexts such as LPN, LWE, and learning the MAJORITY of PARITIES function.

We also introduced a new structured noise model for LPN problems (and by analogy, for related problems such as learning low-depth decision trees) which is a natural modification to the original LPN problem and seems more tractable. We think such structured noise models should be studied more in machine learning since standard models led to intractable problems.

It should be interesting to apply our techniques to other problems and settings, and to investigate the optimality of our parameter choices. Our algorithm for $m = n^2 \log n$ for the MAJORITY of PARITIES function shows that analysis can sometimes be tightened beyond what a first glance suggests.

An obvious open problem is to relate the new structured noise model with the original LPN problem. This seems difficult, though it worked in the special case of LWE with low noise.

Acknowledgements.

We thank several people for useful conversations: Boaz Barak, Avrim Blum, Daniele Micciancio, Oded Regev, David Steurer, Avi Wigderson.

References

[1] M. Ajtai. 1996. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC '96)*. ACM, New York, NY, USA, 99-108.

[2] B. Applebaum, B. Barak, and A. Wigderson. Public Key Cryptography from Different Assumptions . In *Proceedings of STOC*, 2010.

[3] Ajtai, M. and Dwork, C. 1997. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC '97)*. ACM, New York, NY, USA, 284-293.

[4] Akavia, A., Goldwasser, S., and Vaikuntanathan, V. 2009. Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In *Proceedings of the 6th theory of Cryptography Conference on theory of Cryptography* (San Francisco, CA, March 15 - 17, 2009). O. Reingold, Ed. Lecture Notes In Computer Science, vol. 5444. Springer-Verlag, Berlin, Heidelberg, 474-495.

[5] Alekhnovich, M. 2003. More on Average Case vs Approximation Complexity. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* (October 11 - 14, 2003). FOCS. IEEE Computer Society, Washington, DC, 298.

[6] Bard, G. V. Algebraic Cryptanalysis. Springer, 2009.

[7] Blum, A., Furst, M. L., Kearns, M. J., and Lipton, R. J. 1994. Cryptographic Primitives Based on Hard Learning Problems. In *Proceedings of the 13th Annual international Cryptology Conference on Advances in Cryptology* (August 22 - 26, 1993). D. R. Stinson, Ed. Lecture Notes In Computer Science, vol. 773. Springer-Verlag, London, 278-291.

[8] Blum, A., Kalai, A., and Wasserman, H. 2003. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* 50, 4 (Jul. 2003), 506-519. DOI= http://doi.acm.org/10.1145/792538.792543

[9] Bogdanov, A. and Qiao, Y. On the security of goldreich's one-way function. In *APPROX-RANDOM*, pages 392-405, 2009.

[10] Feldman, V., Gopalan, P., Khot, S., and Ponnuswami, A. K. 2006. New Results for Learning Noisy Parities and Halfspaces. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (October 21 - 24, 2006). FOCS. IEEE Computer Society, Washington, DC, 563-574.

[11] Goldreich, O. Candidate one-way functions based on expander graphs. Technical Report TR00-090, Electronic Colloquium on Computational Complexity (ECCC), 2000.

[12] Hopper, N. J. and Blum, M. 2001. Secure Human Identification Protocols. In *Proceedings of the 7th international Conference on the theory and Application of Cryptology and information Security: Advances in Cryptology* (December 09 - 13, 2001). C. Boyd, Ed. Lecture Notes In Computer Science, vol. 2248. Springer-Verlag, London, 52-66.

[13] Hyafil, L. 1978. On the parallel evaluation of multivariate polynomials. In *Proceedings of the Tenth Annual ACM Symposium on theory of Computing* (San Diego, California, United States, May 01 - 03, 1978). STOC '78. ACM, New York, NY, 193-195. DOI= http://doi.acm.org/10.1145/800133.804347

[14] Kearns, M. 1998. Efficient noise-tolerant learning from statistical queries. *J. ACM* 45, 6 (Nov. 1998), 983-1006. DOI= http://doi.acm.org/10.1145/293347.293351

[15] Micciancio, D., Regev, O. 2009. Lattice-Based Cryptography In *Post Quantum Cryptography*, D.J. Bernstein; J. Buchmann; E. Dahmen (eds.), pp. 147-191, Springer (February 2009)

[16] Peikert, C. 2009. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of 41st ACM Symposium on Theory of Computing (STOC)*. ACM, New York, 333342.

[17] Peikert, C., Vaikuntanathan, V., and Waters, B. 2008. A Framework for Efficient and Composable Oblivious Transfer. In *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology* (Santa Barbara, CA, USA, August 17 - 21, 2008). D. Wagner, Ed. Lecture Notes In Computer Science, vol. 5157. Springer-Verlag, Berlin, Heidelberg, 554-571.

[18] Regev, O. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (Sep. 2009), 1-40. DOI= http://doi.acm.org/10.1145/1568318.1568324

[19] Valiant, L. G. Completeness classes in algebra. In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 249–261, 1979.

A Learning With Adversarial Noise

In Section 2, the noise vector η is chosen randomly from a distribution μ, and is independent of the a_i's. Now we consider the more general model where the η is allowed to depend on the a_i's. However, in this model the set of acceptable noise patterns is smaller.

The learning algorithm now has access to a new oracle $Q(u, m, P)$. As before, $u \in \mathrm{GF}(2)^n$ is the secret of the oracle, $P(\eta)$ is a multilinear polynomial such that all acceptable noise patterns η satisfy $P(\eta) = 0$. When queried, the oracle picks m uniformly random vectors a_1, a_2, ..., $a_m \in \mathrm{GF}(2)^n$, then it picks a noise pattern η that satisfies $P(\eta) = 0$. The oracle then returns the vectors $\{a_i\}$ and bits b_1, b_2, ..., b_m such that $b_i = a_i \cdot u + \eta_i$.

As we stated before one major difference here is that the oracle can pick η after looking at the a_i's. Although oracle $Q(u, m, P)$ still has no control over the randomness of $\{a_i\}$, the ability to pick η already gives it power to fool any learning algorithm for some choices of P. For example, the polynomial representing "the noise vector has at most $\lceil m/2 \rceil$ ones" makes it impossible for any algorithm to learn the secret.However, one can hope that if the polynomial P merely represents "At most $m/3$ ones" (in other words, the noise rate is $1/3$) then learning should be possible. The next theorem shows that this is indeed the case. Later in Thm A.3 we will show this theorem is tight in the sense that all other polynomials allow the adversary to fool any algorithm.

Theorem A.1 *Suppose the oracle polynomial P is such that there exists $\eta \in GF(2)^m$, for any α and β satisfying $P(\alpha) = P(\beta) = 0$, $\eta \neq \alpha + \beta$. Then the secret u can be learned in* $\mathrm{poly}(N2^m)$ *time.*

The algorithm works as follows: first construct a multilinear polynomial $R(\eta)$. Let $R(\eta) = 1$ if and only if η cannot be decomposed into $\alpha + \beta$ such that $P(\alpha) = P(\beta) = 0$. In particular, for all α and β that satisfy $P(\alpha) = P(\beta) = 0$, we have $R(\alpha + \beta) = 0$. By assumption R is not identically 0. Let d be the degree of R, and let $N = \sum_{i=1}^{d} \binom{n}{i}$. For each example returned by the oracle, the algorithm adds additional noise vector β such that $P(\beta) = 0$. The effect is that the final noise vector can be represented as $\alpha + \beta$, where α is the noise introduced by the oracle and β is the noise introduced by the algorithm. Although this vector is more "noisy", it nevertheless satisfies $R(\alpha + \beta) = 0$. For all β such that $P(\beta) = 0$, the algorithm defines a constraint

$$R(a_1 \cdot z + b_1 + \beta_1, a_2 \cdot z + b_2 + \beta_2, ..., a_m \cdot z + b_m + \beta_m) = 0. \tag{14}$$

Then the algorithm linearizes it to obtain a linear constraint on the variables y_S

$$L(R(a_1 \cdot z + b_1 + \beta_1, a_2 \cdot z + b_2 + \beta_2, ..., a_m \cdot z + b_m + \beta_m)) = 0. \tag{15}$$

We claim that solving a system of linear equations generated by enough samples will reveal the secret vector.

Theorem A.2 *The linear system obtained by $10N2^d$ samples (which contains at most $10N2^{m+d}$ equations) will always have at least one solution, and with high probability (over the random choices of $\{a_i\}$), all solutions to the system have the following property: the i-th bit of the secret vector is just the value of $y_{\{i\}}$.*

Proof: When $y_S = \prod_{i \in S} u_i$, for any constraint, assume the noise vector that the oracle picked is α, and the noise vector the algorithm picked is β, then the left hand side of Equation (15) is equal to $R(\alpha + \beta)$, where $P(\alpha) = P(\beta) = 0$. By definition of R we have $R(\alpha + \beta) = 0$. Therefore there's always a solution to the system of linear equations, and it satisfies $y_{\{i\}} = u_i$.

Now we show that every incorrect y gets ruled out with high probability, and in fact by a very small subset of the constraints. For any sample returned by the oracle, assume the noise vector that the oracle picked is α, hence $P(\alpha) = 0$. By the construction of linear equations, there will be an equation that corresponds to $\beta = \alpha$. Then $b_i + \beta_i = b_i + \alpha_i = a_i \cdot u$. This equation is

$$L(R(a_1 \cdot (z + u), a_2 \cdot (z + u), ..., a_m \cdot (z + u))) = 0. \tag{16}$$

Equation (16) is equivalent to (3) when $\eta = \mathbf{0}$. If we just consider these constraints, we can view them as generated by an oracle $Q(u, m, R, \mu)$, where the distribution μ gives probability 1 to the $\mathbf{0}$ vector. Clearly $R(\mathbf{0}) = 0$ because for any vector α such that $P(\alpha) = 0$, we have $\alpha + \alpha = \mathbf{0}$. Thus $Q(u, m, R, \mu)$ is a valid oracle, there are $10N2^d$ such constraints, by Theorem 2.3 we know with high probability all solutions to the linear system with these constraints satisfy $y_{\{i\}} = u_i$ (here since $\eta = \mathbf{0}$ with probability 1 we save a 2^m factor in Thm 2.3). Since this set of constraints is only a subset of all the constraints, with high probability all solutions to the entire linear system will also satisfy $y_{\{i\}} = u_i$.

■

Now we show if the algorithm does not work, then no algorithm can learn the secret with probability better than 1/2 for all possible oracles.

Theorem A.3 *If for all $\eta \in GF(2)^m$, there exist α, $\beta \in GF(2)^m$ such that $P(\alpha) = P(\beta) = 0$ and $\eta = \alpha + \beta$, then for any algorithm $A_{m,P}$ there is an oracle $Q(z, m, P)$, such that no matter how many queries A makes, $\mathbf{Pr}[A^Q_{m,P} = z] \leq 1/2$.*

Proof: Assume towards contradiction that there exists an algorithm $A_{m,P}$, for any oracle $Q(z, m, P)$ it satisfies $\mathbf{Pr}[A^Q_{m,P} = z] > 1/2$.

Pick $u, v \in \mathrm{GF}(2)^n$ such that $u \neq v$. We will construct an oracle Q based on u and v. When queried by the algorithm, the oracle Q first generates the vectors $\{a_i\}$ randomly (recall that the oracle has no control over this process). Let M be a matrix in $\mathrm{GF}(2)^{m\times n}$, the i-th row vector of M is equal to a_i. Compute $b^0 = Mu$ and $b^1 = Mv$. Then the oracle constructs a set $S \subseteq \mathrm{GF}(2)^m$, $S = \{b : P(b + b^0) = 0 \text{ and } P(b + b^1) = 0\}$ and returns $\{a_i\}$ and a random $b \in S$. Thus it suffices to show S is not empty. The reason is that by hypothesis the vector $\eta = b^0 + b^1$ can be represented as $\alpha + \beta$, where $P(\alpha) = P(\beta) = 0$. Thus $b^0 + \alpha$ must be in S.

Notice that, u and v are symmetric in the construction. The above oracle Q is a valid oracle for $Q(u, m, P)$, and is also a valid oracle for $Q(v, m, P)$. Let $Q_1(u, m, P) = Q$ and $Q_2(v, m, P) = Q$, by assumption $\mathbf{Pr}[A^{Q_1}_{m,P} = u] > 1/2$ and $\mathbf{Pr}[A^{Q_2}_{m,P} = v] > 1/2$. However Q_1 and Q_2 are actually the same oracle Q, therefore $\mathbf{Pr}[(A^Q_{m,P} = u) \text{ or } (A^Q_{m,P} = v)] > 1$. This is a contradiction. Therefore such an algorithm cannot exist. ∎

A.1 Other noise models

Our linearization can also be useful in situations even when the noise has no explicit "structure". Consider an example of "bursty" noise following a 2-step Markov process: when the last two noise bits are 0 and 1 respectively, the probability of making an error (current noise bit is 1) is $1 - 1/\sqrt{n}$, while in all other cases the probability of making an error is 1/3. Intuitively this means when the oracle makes a first mistake, it's very unlikely for it to recover soon and the oracle will continue to error with high probability. Although this oracle does not satisfy any kind of "structure" by our definition, we can still get a non-trivial algorithm by selecting the right pattern. Consider a group of $C\sqrt{n}$ consecutive answers from the oracle, the probability that the error has pattern $0101\cdots01$ is just $n^{C\sqrt{n}/4}$, the probability is so small that if we define the structure to be "error pattern is not $0101\cdots01$", by union bound we can safely get enough groups of answers that satisfy the structure, and therefore learn the secret vector in time $2^{\tilde{O}(\sqrt{n})}$. It's not clear how to solve such problems without using our algorithm. In fact, since the structure in our algorithm is very general, as long as there are error patterns that occur with smaller than usual probability, our algorithm will be able to utilize that and provide non-trivial improvements over the brute-force solution.

B Learning Low Degree Polynomials

In this section we consider the problem of learning low degree polynomials in the "structured noise" model. Our algorithm will work in both the white noise setting and the adversarial noise setting, but here for simplicity we only show an algorithm in the white noise setting. The algorithm for the adversarial noise setting follows from the same construction as in Section A. The algorithm has access to an oracle $Q(U, m, P, \mu)$, where m, P, μ are analogous to the parameters as in Section 2. The new parameter U is the secret of the oracle. Unlike learning parities with noise, here U is a

degree d' polynomial over GF(2)^n. When queried, the oracle first chooses uniformly random vectors a_1, a_2, ..., a_m from GF(2)^n, then picks η from distribution μ independent of a_i's. The oracle returns a_i's together with $b_i = U(a_i) + \eta_i$.

The algorithm tries to reduce this problem to learning parities with structured noise. We express U in the monomial form:

$$U(a_{i,1}, a_{i,2}, ..., a_{i,n}) = \sum_{W \subseteq [n], |W| \leq d'} u_W \prod_{j \in W} a_{i,j}.$$

Here u_W's are coefficients in GF(2). In the reduction we introduce variables z_W, where W is a subset of $[n]$ with size at most d'. Notice that the set W has a different range from the sets S we used before, and we will always use W for this kind of sets. Let Ω be the set of all W's, that is, $\Omega = \{W : W \subseteq [n] \text{ and } |W| \leq d'\}$. The vector z is also not similar to any of the y vectors we used before. In particular $z_\emptyset$ is also a variable that can be either 0 or 1. Intuitively z_W satisfies all constraints written by the algorithm if $z_W = u_W$.

We define vector A_i based on the vector a_i. The vector A_i is indexed by all sets $W \in \Omega$, and we have $A_{i,W} = \prod_{j \in W} a_{i,j}$. Now we can think of the oracle's secret as the vector u_W, and the oracle returns A_1, A_2, ..., A_m and b. Then it looks like a learning parities with structured noise problem because $b_i = A_i \cdot u_W + \eta_i$. We apply the algorithm in Section 2 to generate $10N2^{m+dd'}$ equations, and claim that with high probability the system of equations will have solutions that reveal the secret u. Here N is the number of variables in the system and $N = O(n^{dd'})$. After linearization, we will have variables y_S, where now $S \subseteq \Omega$ and $|S| \leq d$.

Theorem B.1 *The system of equations with $10N2^{m+dd'}$ constraints always has a solution. With high probability (over the oracle's randomness) all solutions to the system will satisfy $y_{\{W\}} = u_W$, and the secret U is*

$$U(a_{i,1}, a_{i,2}, ..., a_{i,n}) = \sum_{W \subseteq [n], |W| \leq d'} y_{\{W\}} \prod_{j \in W} a_{i,j}.$$

Proof: (sketch) Notice that after reduction the problem is almost the same as learning parities with noise, except that A_i is no longer a uniformly random vector. However, the only places where we use properties of A_i's in the proof for Theorem 2.3 are Lemma 2.5 and Schwartz-Zippel Lemma. We will replace Lemma 2.5 with the following Lemma and claim that the rest of the proof still works.

Lemma B.2 *Consider a polynomial constraint of the form*

$$\sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S (\otimes_{i \in S} A_i) \cdot Y^{|S|} = 0, \tag{17}$$

where c_S are coefficients in GF(2). This is a homogeneous linear constraint in terms of Y^d

If there is a subset S of size k such that $c_S = 1$, and $Y^k \neq \mathbf{0}$, then the left hand side of (17) is a multilinear polynomial over $a_{i,j}$'s of degree at most dd' and is not identically 0.

Proof: If we expand the tensor $(\otimes_{i \in S} A_i)$ and view each component as a monomial over $a_{i,j}$'s, it's clear the left hand side of (17) is a multilinear polynomial over $a_{i,j}$'s of degree at most dd', because in the tensor $A_{i,W}$ cannot be multiplied with $A_{i,W'}$ for any W and W'.

Let the set S of size k and $c_S = 1$ be $\{s_1, s_2, ..., s_k\}$, and let $(W_1, W_2, ..., W_k)$ be an index where Y^k is nonzero(recall that Y^k is now indexed by a vector in Ω^k). The monomial $\prod_{i=1}^{k} \prod_{j \in W_i} a_{s_i,j}$ can only appear in term $c_S(\otimes_{i \in S} A_i) \cdot Y^{|S|}$, its coefficient is $c_S Y^k_{W_1,...,W_k} = 1$. Therefore this polynomial is not identically 0. ∎

Now everything in the proof for Thm 2.3 can go through, except that we now have a polynomial of degree dd', so the probability that a nonzero Y^d violates a random constraint with $\eta = \eta^*$ is at least $2^{-dd'}$.

∎

C Ubiquity of Linearization

Since learning parities with structured noise looks very similar to learning parity with noise, one might ask whether the same kind of technique can be applied to solve the LPN problem. Unfortunately, all reductions from LPN to learning parities with structured noise that we have tried only give $2^{O(n)}$ algorithms, which is no better than the trivial algorithm. Such difficulties might lead one to think that our "reduction to gaussian elimination with polynomial coefficients" is too restricted as a computational model and it is unable to solve the LPN problem. However in this section we will show that any algebraic algorithm that solves the LPN problem can be transformed into solving a system of linear equations with polynomial coefficients, where the size of the linear system is *quasipolynomial* in the size of the algebraic algorithm. Also notice that our proof for Theorem C.3 does not use any special properties of the LPN problem and can be generalized to any decision problems. This shows the difficulty of proving a lowerbound for this model and justifies the naturalness of "reduction to gaussian elimination with polynomial coefficients" technique.

We formulate a decision version of the learning parities with noise problem. Instead of giving the algorithm an oracle Q, we assume the queries to the oracle have already been made, and the result is a matrix A (containing all a_i's as row vectors) and a vector b. Either $b = Au + \eta$ where η_i is 1 with probability strictly smaller than $1/2$ or b is a uniformly random vector. The algorithm will do algebraic computations over GF(2) using elements in A and b, and decide whether there is a u such that $b = Au + \eta$. If the running time of the algorithm is $T(n)$ where T is a non-decreasing function, we always make sure that A has at least $T(n)$ rows so the algorithm never runs out of examples. The decision version of the learning parities with noise problem is defined as follows.

Definition C.1 (Decision Version of Learning Parity With Noise) *Given matrix $A \in GF(2)^{m \times n}$ and vector $b \in GF(2)^m$, learning parities with noise ρ ($DLPN_\rho$) is the problem of distinguishing the following two distributions:*

Distribution D_1: The matrix A and the vector b are uniformly random and independent.

Distribution D_2: The matrix A is uniformly random, b is generated by first choose u uniformly random from $GF(2)^n$, and let $b = Au + \eta$ where η_i's are independent and η_i is 1 with probability ρ and 0 otherwise.

The decision version is closely related to the original problem by the following theorem:

Theorem C.2 *If there's an algorithm M that runs in time $T(n)$ and solves $DLPN_\rho$ with probability at least $1 - \varepsilon/n$, then there exists an algorithm B that makes n calls to M and can recover u in Distribution D_2 with probability at least $1 - \varepsilon - e^{-\Omega(n)}$.*

Proof: Let A_i be the matrix A with i-th column vector removed. For every $i \in [n]$ the algorithm calls M using input (A_i, b). Let $u_i = 1 - M(A_i, b)$ (M outputs 0 for distribution D_1 and 1 for distribution D_2), if $Au + b$ is a vector with at most $(\rho + 1/2)m/2$ 1's, then B claims the the input (A, b) comes from Distribution D_2, and output u; otherwise B will conclude the input comes from Distribution D_1.

If the input really comes from Distribution D_1, then with high probability $(1 - e^{-\Omega(n)})$ there is no vector u such that $Au + b$ has at most $(\rho + 1/2)m/2$ 1's (by Chernoff Bound), so algorithm B is correct with probability $1 - e^{-\Omega(n)}$ in this case.

If the input comes from Distribution D_2, then with probability at least $1 - \varepsilon$, all the n calls to M return the correct answer. Notice that if $u_i = 1$, let $\tilde{a}_i$ be i-th column vector of A, and let x be the vector u with the i-th component removed. Since $b = Au + \eta$, we have $b = A_i x + \tilde{a}_i + \eta$. The vector $\tilde{a}_i$ is uniformly random and independent of A_i. Therefore b is also uniformly random and independent of A_i, and the input (A_i, b) is statistically equivalent to the Distribution D_1. Thus algorithm M will output 0 (recall that we assumed all answers of M are correct). If $u_i = 0$, then $b = A_i x + \eta$, the vector x is uniformly random in $\mathrm{GF}(2)^{n-1}$. Therefore the input (A_i, b) is statistically equivalent to Distribution D_2, and M will output 1. When all calls to M return correct answer, the algorithm can reconstruct u correctly. For the correct value of u , the vector $Au + b$ is a vector with at most $(\rho + 1/2)m/2$ 1's with probability at least $1 - e^{-\Omega(n)}$, thus algorithm B is correct with probability at least $1 - \varepsilon - e^{-\Omega(n)}$ in this case. ∎

Now we try to solve DLPN problem by mapping the problem to a large linear system $Py = q$, as we did in previous sections. The components of the matrix P and the vector q are polynomials over elements in the input (A, b). The vector y is a vector of variables. However, here the variables y may not correspond to any linearization procedure as in our previous algorithms. If the input (A, b) comes from Distribution D_2, then $Py = q$ has a solution with high probability; if (A,b) comes from Distribution D_1, then $Py = q$ has no solution with high probability. The size of the system is the dimension of matrix P, and the degree of the system is the maximum degree of components of P and q when viewed as polynomials over the elements in (A, b). Our algorithm in Section 2 can be viewed as a linear system with size $\mathrm{poly}(n^d, 2^{m+d})$ and degree d.

For an arithmetic circuit C, we use $C(D)$ to denote the probability that C outputs 1 when input is chosen from distribution D. For a linear system $L = (P, q)$, we use $L(D)$ to denote the probability that the equation $Py = q$ has at least one solution when input is chosen from distribution D. The next theorem shows if there's an arithmetic circuit C that distinguishes between D_1 and D_2 ($C(D_1) - C(D_2) > \delta$), then a linear system of larger size will also be able to distinguish the two distributions.

Theorem C.3 *If there's an arithmetic circuit C of size s and degree* $\mathrm{poly}(s)$ *such that $C(D_1) - C(D_2) > \delta$, then there's a linear system $L = (P, q)$ with size $s^{O(\log s)}$ and degree 1 so that $L(D_1) - L(D_2) > \delta$.*

Proof: Hyafil [13] showed that any arithmetic circuit of size s and degree $\mathrm{poly}(s)$ can be transformed into a formula F with depth at most $(\log s)^2$ and size at most $s^{O(\log s)}$. Using a fact proved by Valiant [19], this formula can be computed by the projection of a determinant of size $s^{O(\log s)}$, that is, there is a matrix M of size $s^{O(\log s)}$, whose elements are either 0, 1 or variables in (A, b), such that $\det M = 1$ if and only if the output of circuit C is 1. Let N be the size of M.

Now we construct P as a block diagonal matrix of dimension $2sN \times 2sN$, in its diagonal there are $2s$ blocks, each of which is equal to M. The matrix P looks like

$$\begin{pmatrix} M & O & \dots & O \\ O & M & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & M \end{pmatrix}.$$

We pick q uniformly at random from $\mathrm{GF}(2)^{2sN}$. Clearly, if $\det M = 1$, then $\det P = 1$, the system $Py = q$ always has a solution.

If $\det M = 0$, we break the vector q into $2s$ vectors of length N, and denote them by $q_i (1 \leq i \leq 2s)$. If the equation $Py = q$ has a solution, then because P is a block diagonal matrix, for each block $My' = q_i$ has a solution. However, each q_i is chosen uniformly from random, with probability at least $1/2$ q_i is not a vector in the span of column vectors of M, in which case $My' = q_i$ do not have a solution. Since all vectors q_i's are independent, when $\det M = 0$, $\mathbf{Pr}[Py = q \text{ has a solution}] \leq 1/2^{2s}$. Since the algorithm looks at no more than s components from (A, b), by union bound we have

$$\mathbf{Pr}[\exists (A,b) \quad Py = q \text{ has a solution}] \leq 2^{-2s} \cdot 2^s = 2^{-s}.$$

We fix q so that this event do not happen, then for any input (A, b), the system $Py = q$ has a solution if and only if $\det M = 1$. Thus $L(D) = C(D)$ for any distribution D, and we have $L(D_1) - L(D_2) > \delta$. ∎

D Proof for Lemma 3.4

Proof: Assume that there is a k such that $y^k \neq \mathbf{0}$ and $c_k \neq 0$. Let $v \in \mathbf{Z}^n$ be an index of y^k where $y^k_v \neq 0$. Using extended Binomial Theorem we know that there are exactly $k!/(\prod_{i=1}^n v_i!)$ copies of the monomial $\prod_{i=1}^n a_i^{v_i}$ in the tensor $a^{\otimes k}$. By rule of inner product all these copies are multiplied by y^k_v. Thus the coefficient of this monomial is $y^k_v \cdot k!/(\prod_{i=1}^n v_i!)$ in $\mathrm{GF}(q)$. Here the factorials in $\mathrm{GF}(q)$ are defined similarly as in $\mathbf{Z}$, and division is done by multiplying the inverse. Since $q > D$ and q is a prime, this coefficient is nonzero. Clearly monomials cannot cancel each other so the polynomial over a_i's is not identically 0.

The other direction is trivial, if for each i either $c_i = 0$ or $\tilde{y}^i = \mathbf{0}$ then the polynomial is 0. ∎

E Proof for $O(n^2 \log n)$ equations

This section will prove that only $O(n^2 \log n)$ equations actually suffice for Claim 2 in Section 4. For this we need to carefully analyse the probability that Equation (13) is violated. Intuitively, when the probability is small, W has special structure so we don't need to apply union bound to all 2^{n^2} matrices. Denoting by W_i the i-th row vector of W, we define the *distance* $d_{i,j}$ of W_i and W_j as follows. Let $ham_{i,j}$ be the number of p different from i, j such that $W_{i,p} \neq W_{j,p}$. Then the distance $d_{i,j} = \min\{ham_{i,j}, n - 2 - ham_{i,j}\}$.

Claim: *There exists $\varepsilon > 0$, such that if the probability that Equation (13) is violated is smaller than $\varepsilon t/n^2$, there must be a row i where the sum of distances between i and all other rows of W is at most t.*

Proof: We start with a lemma.

Lemma E.1 *Let $x_1, x_2, \ldots, x_n$ be any variables and $\sum_{i\in S} x_i + c$ be a linear form where $|S| = k$. If d= O(1) of the x_i's are randomly chosen and set to 1 and the rest are set to 0, then the probability that the linear form evaluates to 1 is at least $\Omega(k/n - k^2/n^2)$.*

Proof: We can compute the exact probability. When $c = 0$, the probability is

$$\frac{1}{\binom{n}{k}} \sum_{i=0}^{\lfloor (k-1)/2 \rfloor} \binom{k}{2i+1}\binom{n-k}{d-(2i+1)}.$$

When $c = 1$ the probability is

$$\frac{1}{\binom{n}{k}} \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{2i}\binom{n-k}{d-2i}.$$

The sum of the two is 1, the ratio between the smaller and the larger is at most $O(\min\{k, n-k\})/n$ (by comparing the corresponding terms), therefore the probability that $y = 1$ is at least $\Omega(\min\{k, n-k\})/n^2$, which is $\Omega(k/n - k^2/n^2)$. ∎

Whenever we fix a_1 and a_2, the LHS of Equation (13) becomes a linear form over variables in a_3, and the number of unset variables is $n - 2d/3$. To apply Lemma E.1, we need to know the number of terms in this linear form. Let c_i be the indicator variable that is 1 iff $a_{3,i}$ appears in the linear form after fixing a_1 and a_2, then by Equation (13) we know $c_i = 1$ iff $a_{1,i} + a_{2,i} = 0$ and $(a_1 + a_2) \cdot W_i = 1$ (when $a_{1,i} + a_{2,i} = 1$ then $a_{3,i}$ must be 0 and cannot appear in the linear form). Letting $k = \sum_{i=1}^{n} c_i$, we have by Lemma E.1:

$$\begin{aligned}
\mathbf{Pr}[\text{LHS of Equation (13)} = 1] &= \Omega(\mathbf{E}[\frac{(n-2d/3)k - k^2}{(n-2d/3)^2}]) \\
&= \Omega(\mathbf{E}[\frac{n\sum_{i=1}^{n} c_i - \sum_{1\le i,j\le n} c_i - \frac{2d}{3}\cdot k}{n^2}]) \\
&= \Omega(\mathbf{E}[\frac{\sum_{1\le i<j\le n}(c_i - c_j)^2 - 2dk/3}{n^2}])
\end{aligned}$$

The term $(c_i - c_j)^2$ is closely related to our intuition: when W_i and W_j are roughly the same, c_i and c_j will also be correlated and $(c_i - c_j)^2$ is small. Let $P_{i,j}$ be the probability that $c_i \neq c_j$ conditioned on $a_{1,i} + a_{2,i} = 0$ and $a_{1,j} + a_{2,j} = 0$, then $\mathbf{E}[\sum_{1\le i<j\le n}(c_i - c_j)^2] = \sum_{1\le i<j\le n} P_{i,j} + 2d/3 \cdot \mathbf{E}[k]$, because for any assignment of a_1 and a_2, there are exactly $2d/3 \cdot k$ pairs of (i, j) such that $c_i \neq c_j$ is caused by $a_{1,i} + a_{2,i} = 1$ or $a_{1,j} + a_{2,j} = 1$. Again by Lemma E.1 we know $P_{i,j} = \Omega(d_{i,j}/n)$. Now we have:

$$\begin{aligned}
\mathbf{Pr}[\text{LHS of Equation (13)} = 1] &= \Omega(\frac{\sum_{1\le i<j\le n} P_{i,j}}{n^2}) \\
&= \Omega(\frac{\sum_{1\le i<j\le n} d_{i,j}}{n^3}) \\
&= \Omega(\frac{\min_i \sum_{1\le j\le n, j\neq i} d_{i,j}}{n^2}).
\end{aligned}$$

∎

Now we can try to bound the number of W's such that the probability of violation is small. However there are still more than 2^n possible W's even when $t = 1$, and that will require n^3 equations if we still want to use the union bound.

To solve the problem we observe that some W's are equivalent to each other. Consider the matrix W as the adjacency matrix of an undirected graph. We start by observing when the graph W is a star rooted at vertex k (i.e. $W_{i,j} = 1$ iff $i \neq j$ and $i = k$ or $j = k$), the LHS of Equation (13) is always 0, if the symmetric difference two graphs is a star, the LHS of Equation (13) is the same no matter how we choose a_1, a_2, a_3. More precisely, we have

Lemma E.2 *Let $Star_i$ be the adjacency matrix of a star rooted at vertex i. For two matrices W and W', if $W \oplus W'$ can be expressed as $\oplus_{i\in S} Star_i$ for some set $S \subseteq [n]$, then the two matrices belong to the same equivalence class. Matrices in the same equivalence class will either all be or all not be solutions to the system of linear equations. For any i, W is equivalent to some W' such that the i-th row of W' is $\mathbf{0}$.*

Proof: For any W and W', for any a_1, a_2, a_3,

$$\begin{aligned}(a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2) \cdot W' &= (a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2) \cdot (W + \sum_{i\in S} Star_i) \\ &= (a_1 \otimes a_2 + a_1 \otimes a_3 + a_3 \otimes a_2) \cdot W.\end{aligned}$$

Thus if W is a solution, W' will also satisfy all equations. If W violates one of the equations, W' will violate the same equation. If we let $S = \{j : W_{i,j} = 1\}$, then clearly the i-th row of $W' = W \oplus (\oplus_{i\in S} Star_i)$ is $\mathbf{0}$. ∎

This Lemma allows us to treat matrices in the same equivalence class as a single one when applying union bound, which avoids the loss of a 2^{n-1} factor in the number of matrices. By Lemma E.2 W is always equivalent to some W' where $W'_i = \mathbf{0}$. Since applying symmetric difference with a star does not change $d_{i,j}$, we know $\sum_{1\leq j\leq n, j\neq i} d'_{i,j} \leq t$ where d' is the distance function for W'. To bound the number of such W''s, let V^+ be the set of rows with at most $n/2 - 1$ 1's, V^- contains the rest of rows. It's easy to see that any pair (i, j) such that $i \in V^+$ and $j \in V^-$ will contribute 1 to the sum of distances no matter whether there's an edge between i, j (if there is an edge then it contributes 1 at $W'_{i,j}$ otherwise it contributes one at $W'_{j,i}$). Any pair (i, j) will contribute 2 to the distance if $i, j \in V^+$ and $W'_{i,j} = 1$ or $i, j \in V^-$ and $W'_{i,j} = 0$. For any size of V^+, the number of possible W' is $\binom{n}{|V^+|}\binom{n^2}{t-|V^+||V^-|}2^{|V^+||V^-|} \leq (2n)^{2t}$. Considering different i, $|V^+|$, the total number of such W' is bounded by n^{10t}. Therefore, when we have at least $100n^2 \log n/\varepsilon$ equations, the probability that for some p, q, s, t $W_{(p,q)\times(s,t)} = 1$ is at most

$$\sum_{t=1}^{n^2} (1 - \varepsilon t/n^2)^{100n^2 \log n/\varepsilon} n^{10t} \ll 1.$$

Our algorithm will find the secret vector u with high probability.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de