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

Learning Parities with Structured Noise*

Sanjeev Arora
arora@cs.princeton.edu

Rong Ge
rongge@cs.princeton.edu

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

Abstract

In the *learning parities with noise* problem —well-studied in learning theory and cryptography— 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.

Here 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 discuss generalizations of our result, including learning with more general noise patterns. We can also learn low-depth decision trees in the above structured noise model. We also consider the *learning with errors* problem over $\mathrm{GF}(q)$ and give (a) a $2^{\tilde{O}(\sqrt{n})}$ algorithm in our structured noise setting (b) a slightly subexponential algorithm when the gaussian noise is small.

1 Introduction

In 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 recover u. All evidence suggests that this problem is hard. The best algorithm runs in $2^{O(n/\log n)}$ time (Blum et al. [4]), only very slightly better than the trivial $2^{O(n)}$.

This hardness has fundamental implications for machine learning, where it is a longstanding goal to make learning algorithms resistant to noise in the data. If the noise is adversarial, then learning is hard in many settings, and this is a consequence of work in PCPs. However, in LPN the noise is fairly benign — white noise— and the data is also random. Kearns [7] proposed a generic way to design noise-tolerant algorithms in this white noise setting using his *statistical query* (SQ) model. This work led to noise tolerant algorithms for a variety of concept classes, 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

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

ISSN 1433-8092

(i.e., solving linear equations). If even this algorithm cannot be made noise-tolerant then the prospects for more complicated learning algorithms seem bleak.

The LPN problem has also attracted interest in cryptography. Blum et al. [3] gave a construction of one-way function and pseudorandom generator using the LPN problem. Later Hopper and Blum [5] used the same problem to design efficient identification protocols. Alekhnovich [2] proposed a public-key cryptosystem whose security assumes the hardness of LPN with noise rates lower than $1/\sqrt{n}$. More recently, inspired by lattice-based cryptography, Regev introduced a generalization of the LPN problem to large fields called the *Learning with Errors* problem (LWE) [9]. Here one tries to learn an unknown hyperplane over $\mathrm{GF}(q)$, and the noise is iid (though with "low spread"). Since then the LWE problem has become a source of many innovative ideas in cryptography, such as the oblivious transfer protocol by Peikert et al. [8], and a crytosystem by Akavia et al. [1] that is secure even if almost the entire secret key is leaked.

In this paper we propose a seemingly small change in the statement of the LPN problem which we call the *structured noise* model, and show that the problem can then be solved fairly efficiently. In the standard noise model, pushing an oracle button gives us a random $a \in \mathrm{GF}(2)^n$ and a corresponding noisy value $b \in \mathrm{GF}(2)$. 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. Our algorithm 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. This linearization idea has been used before in practical attacks on cryptosystems, except here we can theoretically prove it works.

We think that this linearization technique may prove useful in other settings. We generalize it to learn low degree polynomials instead of parities, and also to design sub-exponential time algorithms for learning with errors (LWE) problem.

Can something as simple as solving linear equations work for the original LPN problem? Solving linear equations may seem like a very restricted class of algorithms, but we show that it is difficult to rule out the possibility that this class of algorithms can solve LPN. Using standard facts about algebraic computation we show that every algebraic algorithm for the problem (if it exists) can be converted into one that solves linear systems of equations.

1.1 Our results

For LPN, 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 $\mathrm{GF}(2)$. 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 is incomparable with the noise model in standard LPN. On the

one hand it is weaker because not all possible noise patterns are allowed, whereas in standard LPN every noise pattern can occur (albeit with probability as low as p^m). On the other hand it is stronger because the oracle can look at the a_i's in choosing the noise pattern.

Theorem 1.1 (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* $\text{poly}(n^m)$ *time.*

The theorem is proved in Section 3. For simplicity we first describe in Section 2 the case where the oracle is not allowed to look at the a_i's before picking the noise pattern η.

In Section 4 we generalize the algorithm so that it works even if the oracle uses, instead of the linear function $a \to a \cdot u$, some degree d polynomial. The running time increases to n^{md}. This algorithm can be used to learn low-depth decision trees in the structured noise model.

In Section 5 we generalize our results to the learning with errors (LWE) problem, which also concerns learning an unknown linear function in n variables, albeit over $\text{GF}(q)$ for $q = \text{poly}(n)$. The noise model of interest is the *discrete gaussian noise* model of Regev [9]. It is known that this problem with σ parameter $\Omega(\sqrt{n})$ is as hard as approximating worst case lattice problems. We show an algorithm that solves the LWE problem in $2^{\tilde{O}(t^2)}$ time for σ parameter t. This result appears to be new. We also introduce a structured noise variant of this model, and give a $2^{\tilde{O}(\sqrt{n})}$ time algorithm to solve the problem in that model with σ parameter $O(\sqrt{n})$. Unfortunately, algorithms in the structured noise model do not seem to have implications for lattice problems.

In Section 6 we show that the technique of solving linear equations is in principle capable of implementing any algebraic algorithm.

2 LPN: Oracle with white noise

In this section we do a simpler version of the result where the oracle's noise pattern η is not allowed to depend upon the points $a_1, a_2, \ldots, a_m \in \text{GF}(2)^n$. The learning algorithm has access to an oracle $Q(u, m, P, \mu)$, where $u \in \text{GF}(2)^n$ is the secret vector. Let m be an integer and $P(\eta_1, \eta_2, ..., \eta_m)$ be a multilinear polynomial over the vector $\eta \in \text{GF}(2)^m$ of degree d. 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 $u \in \text{GF}(2)^n$ returns m random vectors $a_1, a_2, \ldots, a_m \in \text{GF}(2)^n$ together with m bits $b_1, b_2, \ldots, b_m \in \text{GF}(2)$. 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 2.1 (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 2.2 *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 2.3 (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 2.4 *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 2.2:

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

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

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 2.4, 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 Adversarial Noise

In the previous section, 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 3.3 we will show this theorem is tight in the sense that all other polynomials allow the adversary to fool any algorithm.

Theorem 3.1 *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

$$R(a_1 \cdot z + b_1 + \beta_1, a_2 \cdot z + b_2 + \beta_2, ..., a_m \cdot z + b_m + \beta_m) = 0. \tag{9}$$

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

$$L(R(a_1 \cdot z + b_1 + \beta_1, a_2 \cdot z + b_2 + \beta_2, ..., a_m \cdot z + b_m + \beta_m)) = 0. \tag{10}$$

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

Theorem 3.2 *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 (10) 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

$$L(R(a_1 \cdot (z+u), a_2 \cdot (z+u), ..., a_m \cdot (z+u))) = 0. \tag{11}$$

Equation (11) 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 2.2 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 2.2). 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 3.3 *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. ■

4 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 3. 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 $\mathrm{GF}(2)^n$. When queried, the oracle first chooses uniformly random vectors a_1, a_2, ..., a_m from $\mathrm{GF}(2)^n$, 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 4.1 *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 2.2 are Lemma 2.4 and Schwartz-Zippel Lemma. We will replace Lemma 2.4 with the following Lemma and claim that the rest of the proof still works.

Lemma 4.2 *Consider a polynomial constraint of the form*

$$\sum_{S \subseteq [m], S \neq \emptyset, |S| \leq d} c_S (\otimes_{i \in S} A_i) \cdot Y^{|S|} = 0, \tag{12}$$

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 (12) 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 (12) 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 2.2 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 $2^{-dd'}$.

■

5 Learning With Errors

In the learning with errors problem, all values are in $\mathrm{GF}(q)$, where q is a prime number that is normally bounded by a polynomial of n. The algorithm has access to an oracle $Q(u, \Psi_\alpha)$, where $u \in \mathrm{GF}(q)^n$ is the secret vector, and Ψ_α is a distribution over $\mathrm{GF}(q)$ for the noise vector. The distribution is called the *discrete Gaussians*, it is thought to be a gaussian distribution with standard deviation αq. To sample a point in Ψ_α, first pick a random real number r from Gaussian distribution with standard deviation $\sigma = \alpha q$ (the density function is given by $D_\sigma(r) = 1/\sigma \exp(-(\pi r/\sigma)^2)$), then round it to the nearest integer $\lfloor r \rceil$ and output $\lfloor r \rceil (\mathrm{mod}\ q)$. Here we will rely on the following two well known facts about the distribution:

Lemma 5.1 *For $\eta \in [-(q-1)/2, (q-1)/2]$, we have*

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

The probability that Ψ_α picks 0 is inversely proportional to αq, as stated in the next Lemma.

Lemma 5.2

$$\mathbf{Pr}_{\eta \sim \Psi_\alpha}[\eta = 0] = \Omega(1/\alpha q).$$

The oracle first picks $a \in \mathrm{GF}(q)^n$ uniformly at random. Then it 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. In the "structured noise" setting, we change the oracle slightly and show there is a subexponential time algorithm for some parameters in the new model, including some of the parameters where the original problem is considered to be intractable.

In our 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 $4d < q$. When queried, the oracle first picks $a \in \mathrm{GF}(q)^n$ randomly, then picks η from Ψ_α repeatedly until $|\eta| \leq d$ (we always interpret η as an integer between $-(q-1)/2$ and $(q-1)/2$), and returns a and $b = a \cdot u + \eta$.

The algorithm works similarly as the algorithms in previous sections. In this section we will define a new way to do linearization to handle non-multilinear polynomials, and we will use different sets of variables. However we will use the same notations (such as z, Y^d, etc.).

We first write the polynomial P such that $P(\eta) = 0$ is always satisfied by η, notice that here it is a single-variate polynomial and has degree $2d + 1$.

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

For some technical reasons that we'll explain later the algorithm will pick a random vector $r \in \mathrm{GF}(q)^{2d}$, and consider it as a degree $2d - 1$ polynomial

$$R(\eta) = \sum_{i=0}^{2d-1} r_i \eta^i.$$

The randomness of vector r gives us the following useful property: each coefficient of $\eta^i (1 \leq i \leq 4d)$ in $P(\eta)R(\eta)$ is nonzero with probability $1 - 1/q$.

We use z (an n dimensional vector) as variables, and intuitively the system of equations generated by the algorithm will have solution $z = u$. Then we substitute $\eta = a \cdot z + b$ in the polynomial $P(\eta)R(\eta)$ to obtain a degree $4d$ polynomial over the variables z_i, and force it to be 0.

$$\left((a \cdot z + b) \prod_{i=1}^{d} (a \cdot z + b + i)(a \cdot z + b - i). \right) \left(\sum_{i=0}^{2d-1} r_i (a \cdot z + b)^i \right) = 0.$$

This constraint is always satisfied if $z = u$. Let $D = 4d$, so D is the degree of the polynomial $P(\eta)R(\eta)$. Finally the algorithm linearizes this equation using variables y_v. The vector y 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 as $deg(v)$ or $deg(y_v)$. It's easy to see $deg(v) = deg(y_v) = \sum_{i=1}^{n} v_i$. 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,

$$L\left(P(a \cdot z + b)R(a \cdot z + b)\right) = 0. \tag{13}$$

The algorithm queries the oracle $O(N\alpha q^2)$ times and generates a linear system of equations over the variables y's (the algorithm picks a different polynomial R for each constraint). We will show that with high probability this system of equations has a unique solution.

Theorem 5.3 *With high probability (over the randomness of both the oracle and the algorithm), the system of linear equations generated by the algorithm will have a unique solution. Moreover, $y_{e_i} = u_i$ where e_i is the vector that has 1 in the i-th coordinate and 0 elsewhere.*

Proof: It's easy to see when $y_v = \prod_{i=1}^{n} u_i^{v_i}$, constraint (13) is equivalent to $P(\eta)R(\eta) = 0$, and is always satisfied. In this solution we have $y_{e_i} = u_i$. So now we only need to prove that the system of equations has at most one solution.

The proof has a similar structure as the proof for Theorem 2.2. We will consider a new set of variables $\tilde{y}$ that is indexed in the same way as the vector y. Define $\tilde{y}^k$ similarly to y^k as the degree k components of $\tilde{y}$. The coordinate of $\tilde{y}$ that corresponds to vector v is defined as the linearization of $\prod_{i=1}^{n} (z_i + u_i)^{v_i}$. 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 $\tilde{y}$.

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 of the form

$$\sum_{i=1}^{D} c_i a^{\otimes i} \cdot \tilde{Y}^i. \tag{14}$$

Here $c_i \in \mathrm{GF}(q)$ are coefficients that are considered to be constants. We prove something similar to Lemma 2.4:

Lemma 5.4 *The Polynomial (14) 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 $\tilde{y}^i = \mathbf{0}$.*

Proof: Assume that there is a k such that $y^k \neq \mathbf{0}$ and $c_k \neq 0$. Let v be a coordinate 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. ∎

Let $P(\eta) = \sum_{i=1}^{2d+1} p_i \eta^i$ and $R(\eta) = \sum_{i=0}^{2d-1} r_i \eta^i$, represent $P(\eta)R(\eta)$ as

$$P(\eta)R(\eta) = \sum_{i=1}^{D} c_i \eta^i.$$

As we claimed before, we have the following Lemma.

Lemma 5.5 *Each $c_i (1 \leq i \leq D)$ is nonzero with probability $1 - 1/q$.*

Proof: Computing the product shows $c_i = \sum_{j=0}^{2d-1} r_j p_{i-j}$, undefined coordinates of p are assumed to be 0. Notice that, for each c_i, either p_1 or p_{2d+1} appears in the sum. We know $p_1 = \prod_{i=1}^{d} i(-i) \neq 0$ and $p_{2d+1} = 1 \neq 0$. Since q is a prime, $r_j p_1$ and $r_j p_{2d+1}$ are distributed uniformly for any j. Therefore each c_i is distributed uniformly in $\mathrm{GF}(q)$ (over the randomness of the vector r), and is nonzero with probability $1 - 1/q$. ∎

For any $\tilde{y} \neq \mathbf{0}$, assume $\tilde{y}^k \neq \mathbf{0}$. For any constraint that the algorithm writes, with probability at least αq we have $\eta = 0$ (this is by Lemma 5.2). Substitute $b = a \cdot u$ into Constraint (13), it will have the form

$$L(P(a \cdot (z + u))R(a \cdot (z + u))) = 0.$$

We expand the polynomial, think of $(z+u)$ as the variables, and do linearization by replacing $(z + u)^{\otimes k}$ with $\tilde{Y}^k$, the formula becomes

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

Now the left hand side has the same form as Polynomial (14). By Lemma 5.4 when $c_k \neq 0$ the left hand side is a nonzero polynomial of degree at most D. Here we use the canonical version of Schwartz-Zippel Lemma which states a nonzero polynomial of degree D in $\mathrm{GF}(q)$ is 0 with probability at most D/q when the variables are chosen from uniformly random distribution.

Thus by $D < q$ 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 $\Omega(1/\alpha q)1/q(1-1/q)$. The probability that it satisfies all $CN\alpha q^2$ constraints is at most e^{-10N} for sufficiently large constant C. Finally by union bound we know the probability that the system of equations has more than one solution is at most $e^{-10N}2^N \ll 1$.

■

Notice that the oracle $Q'(u, \Psi_\alpha, d)$ is very similar to $Q(u, \Psi_\alpha)$ when d is large enough because Ψ_α concentrates around 0. We use this fact to prove the following theorem, which appears to be the first nontrivial algorithm for LWE:

Theorem 5.6 *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 an $2^{\tilde{O}(n^{2\varepsilon})}$ time algorithm algorithm that learns u when given access to the oracle $Q(u, \Psi_\alpha)$.*

Proof: We show this by a reduction to the structured noise model. Let $d = C \log n(\alpha q)^2$ where C is a large enough constant. We run the same algorithm for oracle $Q'(u, \Psi_\alpha, d)$ and claim that the algorithm learns u with high probability even if we replace $Q'(u, \Psi_\alpha, d)$ by $Q(u, \Psi_\alpha)$. By Lemma 5.1, the probability that $|\eta|$ is greater than d is $e^{-O((C\alpha q \log n)^2)}$. When C is large enough this is much smaller than $1/N\alpha q^2$. The statistical difference between outputs of oracle $Q(u, \Psi_\alpha)$ and $Q'(u, \Psi_\alpha, d)$ is at most the probability that $|\eta|$ is greater than d, and is much smaller than $1/N\alpha q^2$. It's easy to see that $O(N\alpha q^2)$ independent queries will have statistical distance much smaller than 1. Therefore even if we replace $Q'(u, \Psi_\alpha, d)$ by $Q(u, \Psi_\alpha)$, the algorithm can still learn the secret u with high probability. The algorithm will run in time $n^{O(d)} = 2^{\tilde{O}(n^{2\varepsilon})}$ time which is subexponential for $\varepsilon < 1/2$. ■

6 Can LPN be Solved by Solving Linear Equations?

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 "solving linear equations" 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, where the size of the linear system is *quasipolynomial* in the size of the algebraic algorithm.

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 6.1 (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 6.2 *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 6.3 *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 [6] showed that any arithmetic circuit of size s and degree 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 [10], 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 $q_i (1 \le i \le 2s)$. 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 $\det M = 0$, $\mathbf{Pr}[Py = q \text{ has a solution}] \le 1/2^{2s}$. Since the algorithm looks at no more than s components from (A, b), by union bound we have

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

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 $L(D_1) - L(D_2) > \delta$. ∎

Acknowledgements.

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

References

[1] 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.

[2] 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.

[3] 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 Cryp-*

tology 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.

[4] 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

[5] 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.

[6] 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

[7] 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

[8] 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.

[9] 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

[10] 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.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de