LEARNING POLYNOMIALS WITH QUERIES: THE HIGHLY NOISY CASE*

ODED GOLDREICH†, RONITT RUBINFELD‡, AND MADHU SUDAN§

Abstract. Given a function f mapping n-variate inputs from a finite field F into F, we consider the task of reconstructing a list of all n-variate degree d polynomials that agree with f on a tiny but non-negligible fraction, δ, of the input space. We give a randomized algorithm for solving this task. The algorithm accesses f as a black box and runs in time polynomial in $\frac{n}{\delta}$ and exponential in d, provided δ is $\Omega(\sqrt{d/|F|})$. For the special case when $d = 1$, we solve this problem for all $\epsilon \stackrel{\text{def}}{=} \delta - \frac{1}{|F|} > 0$. In this case the running time of our algorithm is bounded by a polynomial in $\frac{1}{\epsilon}$ and n. Our algorithm generalizes a previously known algorithm, due to Goldreich and Levin, that solves this task for the case when $F = \mathrm{GF}(2)$ (and $d = 1$).

In the process we provide new bounds on the number of degree d polynomials that may agree with any given function on $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This result is derived by generalizing a well-known bound from coding theory on the number of codewords from an error-correcting code that can be "close" to an arbitrary word; our generalization works for codes over arbitrary alphabets, while the previous result held only for binary alphabets.

Key words. Multivariate polynomials, noisy reconstruction, error-correcting codes.

AMS subject classifications. 68Q15

1. Introduction.

We consider the following archetypal reconstruction problem:

Given: An oracle (black box) for an arbitrary function $f : F^n \to F$, a class of functions $\mathcal{C}$, and a parameter δ.
Output: A list of all functions $g \in \mathcal{C}$ that agree with f on at least δ fraction of the inputs.

The reconstruction problem can be interpreted in several ways within the framework of computational learning theory. First, it falls within the framework of learning with persistent noise. Here one assumes that the function f is derived from some function in the class $\mathcal{C}$ by "adding" noise to it. Typical works in this direction either tolerate only small amounts of noise [2, 42, 21, 39] (i.e., that the function is modified only at a small fraction of all possible inputs) or assume that the noise is random [1, 26, 20, 25, 33, 13, 36] (i.e., that the decision of whether or not to modify the function at any given input is made by a random process). In contrast, we take the setting to an extreme, by considering a *very large amount* of (possibly adversarially chosen) noise. In particular, we consider situations in which the noise disturbs the outputs for almost all inputs.

---

* A preliminary version of this paper appeared in *36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995. IEEE.

†Weizmann Institute of Science, Department of Computer Science, Rehovot, Israel. email: oded@wisdom.weizmann.ac.il. Supported by grant No. 92-00226 from the United States – Israel Binational Science Foundation (BSF), Jerusalem, Israel.

‡NEC Research Institute, 4 Independence Way, Princeton, NJ, 08540. email: ronitt@research.nj.nec.com. This work was done when this author was at Cornell University. Research supported in part by ONR Young Investigator Award N00014-93-1-0590 and grant No. 92-00226 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

§ MIT Laboratory for Computer Science, Cambridge, MA 02139. Part of this work was done when this author was at the IBM Thomas J. Watson Research Center. Research supported in part by a Sloan Foundation Fellowship and NSF Career Award CCR-9875511. Email: madhu@mit.edu.

A second interpretation of the reconstruction problem is within the framework of "agnostic learning" introduced by Kearns et al. [23] (see also [29, 30, 24]). In the setting of agnostic learning, the learner is to make no assumptions regarding the natural phenomenon underlying the input/output relationship of the function, and the goal of the learner is to come up with a simple explanation that best fits the examples. Therefore the best explanation may account for only part of the phenomenon. In some situations, when the phenomenon appears very irregular, providing an explanation that fits only part of it is better than nothing. Kearns et al. did not consider the use of queries (but rather examples drawn from an arbitrary distribution), since they were skeptical that queries could be of any help. We show that queries do seem to help (see below).

Yet another interpretation of the reconstruction problem, which generalizes the "agnostic learning" approach, is the following. Suppose that the natural phenomena can be explained by several simple explanations that together cover most of the input-output behavior but not all of it. Namely, suppose that the function f agrees almost everywhere with one of a small number of functions $g_i \in \mathcal{C}$. In particular, assume that each g_i agrees with f on at least a δ fraction of the inputs but that for some (say 2δ) fraction of the inputs f does not agree with any of the g_i's. This setting was investigated by Ar et al. [3]. The reconstruction problem described above may be viewed as a (simpler) abstraction of the problem considered in [3]. As in the case of learning with noise, there is no explicit requirement in the setting of [3] that the noise level be small, but all their results require that the fraction of inputs left unexplained by the g_i's be smaller than the fraction of inputs on which each g_i agrees with f. Our relaxation (and results) do not impose such a restriction on the noise and thus make the setting more appealing and closer in spirit to "agnostic learning".

1.1. Our Results. In this paper, we consider the special case of the reconstruction problem when the hypothesis class is the set of n-variate polynomials of bounded total degree d. (The total degree of a monomial $\prod_i x_i^{d_i}$ is $\sum_i d_i$; that is, the sum of the degrees of the variables in the monomial. The total degree of a polynomial is the maximum total degree of monomials with non-zero coefficient in the polynomial. For example, the total degree of the polynomial $x_1^2 x_2^3 + 5x_2^4$ is 5.) The most interesting aspect of our results is that they relate to *very small* values of the parameter δ (the fraction of inputs on which the hypothesis has to fit the function f). Our main results are:

- An algorithm that given d, F and $\delta = \Omega(\sqrt{d/|F|})$, and provided oracle access to an arbitrary function $f : F^n \to F$, runs in time $(n/\delta)^{O(d)}$ and outputs a list including all degree d polynomials that agree with f on at least a δ fraction of the inputs.
- An algorithm that given F and $\epsilon > 0$, and provided oracle access to an arbitrary function $f : F^n \to F$, runs in time $\mathrm{poly}(n/\epsilon)$ and outputs a list including all linear functions (degree $d = 1$ polynomials) that agree with f on at least a $\delta \stackrel{\mathrm{def}}{=} \frac{1}{|F|} + \epsilon$ fraction of the inputs.
- A new bound on the number of degree d polynomials that may agree with a given function $f : F^n \to F$ on a $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This bound is derived from a more general result about the number of codewords from an error-correcting code that may be close to a given word.

A special case of interest is when the function f is obtained by picking an arbitrary degree d polynomial p, and letting f agree with p on an *arbitrary* $\delta = \Omega(\sqrt{\frac{d}{|F|}})$ fraction

of the inputs and be set at random otherwise.[1] In this case, with high probability, only one polynomial (i.e., p) agrees with f on a δ fraction of the inputs (see Section 5). Thus, in this case, the above algorithm will output only the polynomial p.

Additional Remarks:

1. Any algorithm for the explicit reconstruction problem as stated above would need to output all the coefficients of such a polynomial, requiring time at least $\binom{n+d}{d}$. Moreover the number of such polynomials could grow as a function of $\frac{1}{\delta}$. Thus it seems reasonable that the running time of such a reconstruction procedure should grow as a polynomial function of $\frac{1}{\delta}$ and $\binom{n}{d}$.
   We stress that the above comment does not apply to "implicit reconstruction" algorithms as discussed in Section 1.4.
2. For $d < |F|$, the condition $\delta > \frac{d}{|F|}$ seems a natural barrier for our investigation, since there are *exponentially many* (in n) degree d polynomials that are at distance $\approx \frac{d}{|F|}$ from some functions (see Proposition 4.8).
3. Queries seem essential to our learning algorithm. We provide two indications to our belief, both referring to the special case of $F = \mathrm{GF}(2)$ and $d = 1$. First, if queries are not allowed, then a solution to the reconstruction problem yields a solution to the longstanding open problem of "decoding random (binary) linear codes". (Note that each random example given to the reconstruction algorithm corresponds to a random linear equation on the information variables. We admit that the longstanding open problem is typically stated for a linear number of equations, but nothing is known even in case the number of equations is polynomial in the information length.)
   Another well-studied problem that reduces to the problem of noisy reconstruction is the problem of "learning parity with noise" [20], which is commonly believed to be hard when one is only allowed uniformly and independently chosen examples [20, 7, 22]. Learning parity with noise is considered hard even for random noise, whereas here the noise is adversarial.
4. In Section 6, we give evidence that the reconstruction problem may be hard, for δ very close to $d/|F|$, even in the case where $n = 2$. A variant is shown to be hard even for $n = 1$.

1.2. A Coding Theory Perspective. We first introduce the formal definition of an error-correcting code (see, e.g. [31]). For positive integers N, K, D and q, an $[N, K, D]_q$ error-correcting code is a collection of q^K sequences of N-elements each from $\{1, \ldots, q\}$, called codewords, in which no two sequences have a "Hamming distance" of less than D (i.e., every pair of codewords disagree on at least D locations).

Polynomial functions lead to some of the simplest known constructions of error-correcting codes: A function from F^n to F may be viewed as an element of $F^{|F|^n}$ — by writing down explicitly the function's value on all $|F|^n$ inputs. Then the "distance property" of polynomials yields that the set of sequences corresponding to bounded-degree polynomial functions form an error-correcting code with non-trivial parameters (for details, see Proposition 4.3). Specifically, the set of n-variate polynomial of total degree d over $F = \mathrm{GF}(q)$ yields a $[N, K, D]_q$ error-correcting code with $N = |F|^n$, $K = \binom{n+d}{d}$ and $D = (|F| - d) \cdot |F|^{n-1}$. These constructions have been studied in the coding theory literature. The case $n = 1$ leads to "Reed-Solomon codes", while the case of general n is studied under the name "Reed-Muller codes".

[1] This is different from "random noise" as the set of corrupted inputs is selected adversarially – only the values at these inputs are random.

Our reconstruction algorithm may be viewed as an algorithm that takes an arbitrary word from $F^{|F|^n}$ and finds a list of all codewords from the Reed-Muller code that agree with the given word in δ fraction of the coordinates (i.e., $1-\delta$ fraction of the coordinates have been corrupted by errors). This task is referred to in the literature as the "list-decoding" problem [11]. For codes achieved by setting d such that $d/|F| \to 0$, our list decoding algorithm recovers from errors when the rate of errors approaches 1. We are not aware of any other case where an approach other (and better) than brute-force can be used to perform list decoding with the error-rate approaching 1. Furthermore, our decoding algorithm works without examining the entire codeword.

1.3. Related Previous Work. For sake of scholarly interest, we discuss several related areas in which related work has been done. In this subsection, it would be more convenient to refer to the error-rate $1-\delta$ rather than to the rate of agreement δ.

Polynomial interpolation: When the noise rate is 0, our problem is simply that of polynomial interpolation. In this case the problem is well analyzed and the reader is referred to [48], for instance, for a history of the polynomial interpolation problem.

Self-Correction: In the case when the noise rate is positive but small, one approach used to solving the reconstruction problem is to use *self-correctors*, introduced independently in [8] and [28]. Self-correctors convert programs that are known to be correct on a fraction δ of inputs into programs that are correct on each input. Self-correctors for values of δ that are larger than $3/4$ have been constructed for several algebraic functions [5, 8, 9, 28, 34], and in one case this was done for $\delta > 1/2$ [15].[2] We stress that self-correctors correct a given program using only the information that the program is supposed to be computing a function from a given class (e.g., a low-degree polynomial). Thus, when the error is larger than $\frac{1}{2}$ and the class contains more than a single function, such self-correction is no longer possible since there could be more than one function in the class that agrees with the given program on an $\delta < 1/2$ fraction of the inputs.

Cryptography and Learning Theory: In order to prove the security of a certain "hardcore predicate" relative to any "one-way function", Goldreich and Levin solved a special case of the (explicit) reconstruction problem [17]. Specifically, they considered the linear case (i.e., $d = 1$) for the Boolean field (i.e., $F = \mathrm{GF}(2)$) and any agreement rate that is bigger than the error-rate (i.e., $\delta > \frac{1}{2}$). Their ideas were subsequently used by Kushilevitz and Mansour [25] to devise an algorithm for learning Boolean decision trees.

1.4. Subsequent work. At the time this work was first published [18] no algorithm other than brute force was known for reconstructing a list of degree d polynomials agreeing with an arbitrary function on a vanishing fraction of inputs, for any $d \geq 2$. Our algorithm solves this problem with exponential dependence on d, but with polynomial dependence on n, the number of variables. Subsequently some new reconstruction algorithms for polynomials have been developed. In particular, Sudan [40], and Guruswami and Sudan [19] have provided new algorithms for reconstructing *univariate* polynomials from large amounts of noise. Their running time depends only

[2]Specifically, self-correctors correcting $\frac{1}{\Theta(d)}$ fraction of error for f that are degree d polynomial functions over a finite field F, $|F| \geq d+2$, were found by [5, 28]. For $d/|F| \to 0$, the fraction of errors that a self-corrector could correct was improved to almost $1/4$ by [14] and then to almost $1/2$ by [15] (using a solution for the univariate case given by [45]).

polynomially in d and works for $\delta = \Omega(\sqrt{d/|F|})$. Notice that the agreement required in this case is larger than the level at which our NP-hardness result (of Section 6) holds. The results of [40] also provide some reconstruction algorithms for multivariate polynomials, but not for as low an error as given here. In addition, the running time grows exponentially with n. Wasserman [44] gives an algorithm for reconstructing polynomials from noisy data that works *without* making queries. The running time of the algorithm of [44] also grows exponentially in n and polynomially in d.

As noted earlier (see Remark 1 in Section 1.1), the running time of any *explicit* reconstruction algorithm has to have an exponential dependence on either d or n. However this need not be true for *implicit* reconstruction algorithms: By the latter term we mean algorithms that produce as output a sequence of oracle machines, such that for every multivariate polynomial that has agreement δ with the function f, one of these oracle machines, given access to f, computes that polynomial. Recently, Arora and Sudan [4] gave an algorithm for this implicit reconstruction problem. The running time of their algorithm is bounded by a polynomial in n and d, and it works correctly provided that $\delta \geq (d^{O(1)})/|F|^{\Omega(1)}$; that is, their algorithm needs a much higher agreement, but works in time polynomial in all parameters. The reader may verify that such an implicit reconstruction algorithm yields an algorithm for the explicit reconstruction problem with running time that is polynomial in $\binom{n+d}{d}$. (E.g., by applying noise-free polynomial-interpolation to each of the oracle machines provided above, and testing the resulting polynomial for agreement with f.) Finally, Sudan, Trevisan, and Vadhan [41], have recently improved the result of [4], further reducing the requirement on δ to $\delta > 2\sqrt{d/|F|}$. The algorithm of [41] thus subsumes the algorithm of this paper for all choices of parameters, except $d = 1$.

1.5. Rest of this paper. The rest of the paper is organized as follows. In Section 2 we motivate our algorithm, starting with the special case of the reconstruction of linear polynomials. The general case algorithm is described formally in Section 3, along with an analysis of its correctness and running time assuming an upper bound on the number of polynomials that agree with a given function at δ fraction of the inputs. In Section 4 we provide two such upper bounds. These bounds do not use any special (i.e., algebraic) property of polynomials, but rather apply in general to collections of functions that have large distance between them. In Section 5 we consider a random model for the noise applied to a function. Specifically, the output either agrees with a fixed polynomial or is random. In such a case we provide a stronger upper bound (specifically, 1) on the number of polynomials that may agree with the black box. In Section 6 we give evidence that the reconstruction problem may be hard for small values of the agreement parameter δ, even in the case when $n = 1$. We conclude with an application of the linear-polynomial reconstruction algorithm to complexity theory: Specifically, we use it in order to prove the security of new, generic, hard-core functions (see Section 7).

Notations: In what follows, we use GF(q) to denote the finite field on q elements. We assume arithmetic in this field (addition, subtraction, multiplication, division and comparison with zero) may be performed at unit cost. For a finite set A, we use the notation $a \in_R A$ to denote that a is a random variable chosen uniformly at random from A. For a positive integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

2. Motivation to the algorithm. We start by presenting the algorithm for the linear case, and next present some of the ideas underlying the generalization to higher degrees. We stress that whereas Section 2.1 provides a full analysis of the

linear case, Section 2.2 merely introduces the additional ideas that will be employed in dealing with the general case. The presentation in Section 2.1 is aimed to facilitate the generalization from the linear case to the general case.

2.1. Reconstructing linear polynomials. We are given oracle access to a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ and need to find a polynomial (or actually all polynomials) of degree d that agrees with f on at least a $\delta = \frac{d}{q} + \epsilon$ fraction of the inputs, where $\epsilon > 0$. Our starting point is the linear case (i.e., $d = 1$); namely, we are looking for a polynomial of the form $p(x_1, ..., x_n) = \sum_{i=1}^{n} p_i x_i$. In this case our algorithm is a generalization of an algorithm due to Goldreich and Levin [17][3]. (The original algorithm is regained by setting $q = 2$.)

We start with a definition: The i-prefix of a linear polynomial $p(x_1, ..., x_n)$ is the polynomial that results by summing up all of the monomials in which only the first i variables appear. That is, the i-prefix of the polynomial $\sum_{j=1}^{n} p_j x_j$ is $\sum_{j=1}^{i} p_j x_j$. The algorithm proceeds in n rounds, so that in the i^{th} round we find a list of candidates for the i-prefixes of p.

In the i^{th} round, the list of i-prefixes is generated by extending the list of $(i-1)$-prefixes. A simple (and inefficient) way to perform this extension is to first extend each $(i-1)$-prefix in all q possible ways, and then to "screen" the resulting list of i-prefixes. A good screening is the essence of the algorithm. It should guarantee that the i-prefix of a correct solution p does pass and that not too many other prefixes pass (as otherwise the algorithm consumes too much time).

The screening is done by subjecting each candidate prefix, $(c_1, ..., c_i)$, to the following test. Pick $m = \mathrm{poly}(n/\epsilon)$ sequences uniformly from $\mathrm{GF}(q)^{n-i}$. For each such sequence $\bar{s} = (s_{i+1}, ..., s_n)$ and for every $\sigma \in \mathrm{GF(q)}$, estimate the quantity

$$P_{\bar{s}}(\sigma) \stackrel{\text{def}}{=} \mathbf{P}_{r_1,...,r_i \in \mathrm{GF}(q)} \left[f(\bar{r}, \bar{s}) = \sum_{j=1}^{i} c_j r_j + \sigma \right] \tag{2.1}$$

where $\bar{r} = (r_1, \ldots, r_i)$. The value σ can be thought of as a guess for $\sum_{j=i+1}^{n} p_j s_j$. For every fixed suffix $\bar{s}$, all these probabilities can be approximated simultaneously by using a sample of $\mathrm{poly}(n/\epsilon)$ sequences $(r_1, ..., r_i)$, regardless of q. If one of these probabilities is significantly larger than $1/q$ then the test accepts due to this suffix, and if no suffix makes the test accept then it rejects. The actual algorithm is presented in Figure 2.1.

Observe that a candidate $(c_1, ..., c_i)$ passes the test of Figure 2.1 if for at least one sequence of $\bar{s} = (s_{i+1}, ..., s_n)$ there exists a σ so that the estimate for $P_{\bar{s}}(\sigma)$ is greater than $\frac{1}{q} + \frac{\epsilon}{3}$. Clearly, for a correct candidate (i.e., $(c_1, ..., c_i)$ that is a prefix of a polynomial $p = (p_1, ..., p_n)$ having at least $\frac{1}{q} + \epsilon$ agreement with f) and $\sigma = \sum_{j=i+1}^{n} p_j s_j$, it holds that $\mathbf{E}_{\bar{s}}[P_{\bar{s}}(\sigma)] \geq \frac{1}{q} + \epsilon$. Using Markov's inequality, it follows that for a correct candidate, an $\epsilon/2$ fraction of the suffixes are such that for each such suffix $\bar{s}$ and some σ, it holds that $P_{\bar{s}}(\sigma) \geq \frac{1}{q} + \frac{\epsilon}{2}$; thus the correct candidate passes the test with overwhelming probability. On the other hand, for any suffix $\bar{s}$, if a prefix $(c_1, \ldots, c_i)$ is to pass the test (with non-negligible probability) due to suffix $\bar{s}$, then it must be the case that the polynomial $\sum_{j=1}^{i} c_j x_j$ has at least agreement-rate of $\frac{1}{q} + \frac{\epsilon}{4}$ with the function $f'(x_1, \ldots, x_i) \stackrel{\text{def}}{=} f(x_1, \ldots, x_i, s_{i+1}, \ldots, s_n)$. It is possible to

[3]We refer to the original algorithm as in [17], not to a simpler algorithm that appears in later versions (cf., [27, 16]).

Test-prefix$(f, \epsilon, n, (c_1, \ldots, c_i))$
  Repeat $\text{poly}(n/\epsilon)$ times:
    Pick $\bar{s} = s_{i+1}, \ldots, s_n \in_R \text{GF(q)}$.
    Let $t \stackrel{\text{def}}{=} \text{poly}(n/\epsilon)$.
    for $k = 1$ to t do
      Pick $\bar{r} = r_1, \ldots, r_i \in_R \text{GF(q)}$
      $\sigma^{(k)} \leftarrow f(\bar{r}, \bar{s}) - \sum_{j=1}^{i} c_j r_j$.
    endfor
    If $\exists\ \sigma$ s.t. $\sigma^{(k)} = \sigma$ for at least $\frac{1}{q} + \frac{\epsilon}{3}$ fraction of the k's
      then output accept and halt.
  endRepeat.
  If all iterations were completed without accepting, then reject.

FIG. 2.1. *Implementing the screening process*

bound the number of (i-variate) polynomials that have so much agreement with any function f'. (Sections 4 contains some such bounds.) Thus, in each iteration, only a small number of prefixes pass the test, thereby limiting the total number of prefixes that may pass the test in any one of the $\text{poly}(n/\epsilon)$ iterations.

The above yields a $\text{poly}(nq/\epsilon)$-time algorithm. In order to get rid of the q factor in running-time, we need to modify the process by which candidates are formed. Instead of extending each $(i-1)$-prefix, $(c_1, ..., c_{i-1})$, in all q possible ways, we do the following: We pick uniformly $\bar{s} \stackrel{\text{def}}{=} (s_{i+1}, ..., s_n) \in \text{GF}(q)^{n-i}$, $\bar{r} \stackrel{\text{def}}{=} (r_1, ..., r_{i-1}) \in \text{GF}(q)^{i-1}$ and $r', r'' \in \text{GF}(q)$, and solve the following system of equations

$$r'x + y = f(r_1, ..., r_{i-1}, r', s_{i+1}, ..., s_n) - \sum_{j=1}^{i-1} c_j r_j \tag{2.2}$$

$$r''x + y = f(r_1, ..., r_{i-1}, r'', s_{i+1}, ..., s_n) - \sum_{j=1}^{i-1} c_j r_j \tag{2.3}$$

using the solution for x as the value of the i^{th} coefficient (i.e., set $c_i = x$). This extension process is repeated $\text{poly}(n/\epsilon)$ many times, obtaining at most $\text{poly}(n/\epsilon)$ candidate i-prefixes, per each candidate $(i-1)$-prefix. We then subject each i-prefix in the list to the screening test (presented in Figure 2.1), and keep only the candidates that pass the test.

We need to show that if the $(i-1)$-prefix of a correct solution is in the list of candidates (at the beginning of round i) then the i-prefix of this solution will be found in the extension process. Let $p = (p_1, ..., p_n)$ be a correct solution (to the reconstruction problem for f). Then $\mathbf{P}_{\bar{r},r,\bar{s}}[p(\bar{r}, r, \bar{s}) = f(\bar{r}, r, \bar{s})] \geq \frac{1}{q} + \epsilon > \epsilon$. It follows that for at least an $\epsilon/2$ fraction of the sequences $(\bar{r}, \bar{s})$, the polynomial p satisfies $p(\bar{r}, r, \bar{s}) = f(\bar{r}, r, \bar{s})$ for at least an $\epsilon/2$ fraction of the possible r's. Let σ represent the value of the sum $\sum_{j=i+1}^{n} p_j s_j$, and note that $p(\bar{r}, r, \bar{s}) = \sum_{j=1}^{i-1} p_j r_j + p_i r + \sigma$. Then, with probability $\Omega(\epsilon^3)$ over the choices of $r_1, \ldots, r_{i-1}, s_{i+1}, \ldots, s_n$ and r', r'', the following two equations hold:

$$r'p_i + \sigma = f(r_1, ..., r_{i-1}, r', s_{i+1}, ..., s_n) - \sum_{j=1}^{i-1} p_j r_j$$

$$r''p_i + \sigma = f(r_1, ..., r_{i-1}, r'', s_{i+1}, ..., s_n) - \sum_{j=1}^{i-1} p_j r_j$$

and $r' \neq r''$. (I.e., with probability at least $\frac{\epsilon}{2}$, the pair $(\bar{r}, \bar{s})$ is good, and conditioned on this event r' is good with probability at least $\frac{\epsilon}{2}$, and similarly for r'' losing a term of $\frac{1}{q} < \frac{\epsilon}{4}$ to account for $r'' \neq r'$. We may assume that $1/q < \epsilon/4$, since otherwise $q < 4/\epsilon$ and we can afford to perform the simpler procedure above.) Thus, with probability $\Omega(\epsilon^3)$, solving the system (2.2)-(2.3) with $(c_1, ..., c_{i-1}) = (p_1, ..., p_{i-1})$ yields $x = p_i$. Since we repeat the process $\text{poly}(n/\epsilon)$ times for each $(i-1)$-prefix, it follows that the correct prefix always appears in our candidate list.

Recall that correct prefixes pass the screening process with overwhelmingly high probability. Using Theorem 4.5 (of Section 4) to bound the number of prefixes passing the screening process, we have:

THEOREM 2.1. *Given oracle access to a function f and parameters ϵ, k, our algorithm runs in $\text{poly}(\frac{k \cdot n}{\epsilon})$-time and outputs, with probability at least $1 - 2^{-k}$, a list satisfying the following properties:*

1. *The list contains all linear polynomials that agree with f on at least a $\delta = \frac{1}{q} + \epsilon$ fraction of the inputs.*
2. *The list does not contain any polynomial that agrees with f on less than a $\frac{1}{q} + \frac{\epsilon}{4}$ fraction of the inputs.*

2.2. Generalizing to higher degree. We remind the reader that in this subsection we merely introduce the additional ideas used in extending the algorithm from the linear case to the general case. The algorithm itself is presented and analyzed in Section 3.

Dealing with polynomials of degree $d > 1$ is more involved than dealing with linear polynomials, still we employ a similar strategy. Our plan is again to "isolate" the terms/monomials in the first i variables and find candidates for their coefficients. In particular, if $p(x_1, \ldots, x_n)$ is a degree d polynomial on n variables then $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is a degree $\leq d$ polynomial on i variables that has the same coefficients as p on all monomials involving only variables in $\{1, \ldots, i\}$. Thus, $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is the i-prefix of p.

We show how to extend a list of candidates for the $(i-1)$-prefixes of polynomials agreeing with f into a list of candidates for the i-prefixes. Suppose we get the $(i-1)$-prefix p that we want to extend. We select $d+1$ distinct elements $r^{(1)}, ..., r^{(d+1)} \in \text{GF}(q)$, and consider the functions

$$f^{(j)}(x_1, ..., x_{i-1}) \stackrel{\text{def}}{=} f(x_1, ..., x_{i-1}, r^{(j)}, 0, ..., 0) - p(x_1, ..., x_{i-1}). \tag{2.4}$$

Suppose that f equals some degree d polynomial and that p is indeed the $(i-1)$-prefix of this polynomial. Then $f^{(j)}$ is a polynomial of degree $d-1$ (since all the degree d monomials in the first i variables have been canceled by p). Furthermore, *given an explicit representation* of $f^{(1)}, ..., f^{(d+1)}$, we can find (by interpolation) the extension of p to a i-prefix. The last assertion deserves some elaboration.

Consider the i-prefix of f, denoted $p' = p'(x_1, ..., x_{i-1}, x_i)$. In each $f^{(j)}$, the monomials of p' that agree on the exponents of $x_1, ..., x_{i-1}$ are collapsed together (since x_i is instantiated and so monomials containing different powers of x_i are added together). However, using the $d+1$ collapsed values, we can retrieve the coefficients of the different monomials (in p'). That is, for each sequence of exponents $(e_1, ..., e_{i-1})$

such that $\sum_{j=1}^{i-1} e_j \leq d$, we retrieve the coefficients of all the $(\prod_{j=1}^{i-1} x^{e_j}) \cdot x_i^k$ in p', by interpolation that refers to the coefficients of $\prod_{j=1}^{i-1} x^{e_j}$ in the $f^{(\ell)}$'s.[4]

To complete the high level description of the procedure we need to show how to obtain the polynomials representing the $f^{(j)}$'s. Since in reality we only have access to a (possibly highly noisy) oracle for the $f^{(j)}$'s, we use the main procedure for finding a list of candidates for these polynomials. We point out that the recursive call is to a problem of degree $d-1$, which is lower than the degree we are currently handling.

The above description ignores a real difficulty that may occur: Suppose that the agreement rate of f with some p^* is at least δ, and so we need to recover p^*. For our strategy to work, the agreement rate of the $f^{(j)}$'s with $p^*(\ldots, 0^{n-i})$ must be close to δ. However, it may be the case that p^* does not agree with f at all on the inputs in $\mathrm{GF}(q)^i 0^{n-i}$, although p^* does agrees with f on a δ fraction of inputs in $\mathrm{GF}(q)^n$. Then solving the subproblem (i.e., trying to retrieve polynomials close to the $f^{(j)}$'s) gives us no information about p^*. Thus, we must make sure that the agreement rate on the subproblems on which we recurse is close to the original agreement rate. This can be achieved by applying a random linear transformation to the coordinate system as follows: Pick a random nonsingular matrix R and define new variables $y_1, \ldots, y_n$ as $(y_1, \ldots, y_n) = \bar{y} \equiv R\bar{x}$ (each y_i is a random linear combination of the x_i's and vice versa). This transformation can be used to define a new instance of the reconstruction problem in terms of the y_i's, and for the new instance the agreement rate on the subproblems on which we recurse is indeed close to the original agreement rate. Observe that

1. the total degree of the problem is preserved;
2. the points are mapped pairwise independently, and so the fraction of agreement points in all subspaces of the new problem is close to the agreement rate in the original space; and
3. one can easily transform the coordinate system back to the x_i's, and so it is possible to construct a new black box consistent with f that takes $\bar{y}$ as an input.

(It may be noted that the transformation does not preserve other properties of the polynomial; e.g., its sparsity.)

Comment: The above solution to the above difficulty is different than the one in the original version of this paper [18]. The solution there was to pick many different suffixes (instead of 0^{n-i}), and to argue that at least in one of them the agreement rate is preserved. However, picking many different suffixes creates additional problems, which needed to be dealt with carefully. This resulted in a more complicated algorithm in the original version.

3. Algorithm for degree $d > 1$ polynomials. Recall that we are given oracle access to a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$, and need to find all polynomials of degree d that agrees with f on at least a δ fraction of the inputs.

The main algorithm Find-all-poly will use several subroutines: Compute-coefficients, Test-valid, Constants, Brute-force, and Extend. The main algorithm is recursive, in n (the number of variables) and d (the degree), with the base case $d = 0$ being handled by the subroutine Constants and the other bases cases corresponding to $n \leq 4$ being handled by the subroutine Brute-force. Most of the work is done in Find-all-poly and

[4] Let c_k be the coefficient of $(\prod_{j=1}^{i-1} x^{e_j}) \cdot x_i^k$ in p', and v_ℓ be the coefficient of $\prod_{j=1}^{i-1} x^{e_j}$ in $f^{(\ell)}$. Then, $v_\ell = \sum_{k=0}^{d} (r^{(\ell)})^k c_k$, and the c_k's can be found given the v_ℓ's.

Extend, which are mutually recursive.

The algorithms have a number of parameters in their input. We describe the commonly occurring parameters first:

- q is the size of the field we will be working with; i.e., $F = \mathrm{GF(q)}$. (Unlike other parameters, the field never changes in the recursive calls.)
- f will be a function from $\mathrm{GF(q)}^n$ to GF(q) given as an oracle to the current procedure, and n will denote the number of variables of f.
- d will denote the degree of the polynomial we are hoping to reconstruct, and δ will denote the agreement parameter. Typically, the algorithm will have to reconstruct all degree d polynomials having agreement at least δ with f.

Many of the algorithms are probabilistic and make two-sided error.

- ψ will be the error parameter controlling the probability with which a valid solution may be omitted from the output.
- ϕ will be the error parameter controlling the error with which an invalid solution is included in the output list.

Picking a random element of GF(q) is assumed to take unit time, as are field operations and calls to the oracle f.

The symbol x will typically stand for a vector in $\mathrm{GF(q)}^n$, while the notation x_i will refer to the ith coordinate of x. When picking a sequence of vectors, we will use superscripts to denote the vectors in the sequence. Thus, $x_i^{(j)}$ will denote the ith coordinate of the jth element of the sequence of vectors $x^{(1)}, x^{(2)}, \ldots$. For two polynomials p_1 and p_2, we write $p_1 \equiv p_2$ if p_1 and p_2 are identical. (In this paper, we restrict ourselves to polynomials of degree less than the field size; thus identity of polynomials as functions is equivalent to identity of polynomials as a formal sum of monomials.) We now generalize the notion of the prefix of a polynomial in two ways. We extend it to arbitrary functions, and then extend it to arbitrary suffixes (and not just 0^i).

DEFINITION 3.1. *For $1 \le i \le n$ and $a_1, \ldots, a_{n-i} \in F$, the $(a_1, \ldots, a_{n-i})$-*prefix *of a function $f : F^n \to F$, denoted $f|_{a_1,\ldots,a_{n-i}}$, is the i-variate function $f|_{a_1,\ldots,a_{n-i}} : F^i \to F$, given by $f|_{a_1,\ldots,a_{n-i}}(x_1, ..., x_i) = f(x_1, \ldots, x_i, a_1, \ldots, a_{n-i})$. The i-*prefix *of f is the function $f|_{0^{n-i}}$.*

When specialized to a polynomial p, the i-prefix of p yields a polynomial on the variables $x_1, \ldots, x_i$ whose coefficients are exactly the coefficients of p on monomials involving only $x_1, \ldots, x_i$.

Fixing a field GF(q), we will use the notation $N_{n,d,\delta}$ to denote the maximum (over all possible f) of the number of polynomials of degree d in n variables that have agreement δ with f. In this section we will first determine our running time as a function of $N_{n,d,\delta}$, and only next use bounds on $N_{n,d,\delta}$ (proven in Section 4) to derive the absolute running times. We include the intermediate bounds since it is possible that the bounds of Section 4 may be improved, and this would improve our running time as well. By definition, $N_{n,d,\delta}$ is monotone non-decreasing in d and n, and monotone non-increasing in δ. These facts will be used in the analysis.

3.1. The subroutines. We first axiomatize the behavior of each of the subroutines. Next we present an implementation of the subroutine, and analyze it with respect to the axiomatization.

(P1) Constants(f, δ, n, q, ψ), with probability at least $1 - \psi$, returns every degree 0 (i.e., constant) polynomial p such that f and p agree on δ fraction of the

points.[5]

Constants works as follows: Set $k = O(\frac{1}{\delta^2}\log\frac{1}{\psi})$ and pick $x^{(1)},\ldots,x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. Output the list of all constants a (or equivalently the polynomial $p_a = a$) such that $|\{i \in [k] | f(x^{(i)}) = a\}| \geq \frac{3}{4}\delta k$.

An easy application of Chernoff bounds indicates that the setting $k = O(\frac{1}{\delta^2}\log\frac{1}{\psi})$ suffices to ensure that the error probability is at most ψ. Thus the running time of Constants is bounded by the time to pick $x^{(1)},\ldots,x^{(k)} \in \mathrm{GF(q)}^n$ which is $O(kn) = O(\frac{1}{\delta^2} n \log\frac{1}{\psi})$.

PROPOSITION 3.2. Constants(f,δ,n,q,ψ) *satisfies* **(P1)**. *Its running time is* $O(\frac{1}{\delta^2} n \log\frac{1}{\psi})$.

Another simple procedure is the testing of agreement between a given polynomial and a black box.

(P2) Test-valid$(f,p,\delta,n,d,q,\psi,\phi)$ returns true, with probability at least $1-\psi$, if p is an n-variate degree d polynomial with agreement at least δ with f. It returns false with probability at least $1-\phi$ if the agreement between f and p is less than $\frac{\delta}{2}$. (It may return anything if the agreement is between $\frac{\delta}{2}$ and δ.)

Test-valid works as follows: Set $k = O(\frac{1}{\delta^2}\log\frac{1}{\min\{\psi,\phi\}})$ and pick $x^{(1)},\ldots,x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. If $f(x^{(i)}) = p(x^{(i)})$ for at least $\frac{3}{4}\delta$ fraction of the values of $i \in [k]$ then output true else false.

Again an application of Chernoff bounds yields the correctness of Test-valid. The running time of Test-valid is bounded by the time to pick the k points from $\mathrm{GF(q)}^n$ and the time to evaluate p on them, which is $O(\frac{1}{\delta^2}(\log\frac{1}{\min\{\psi,\phi\}})\binom{n+d}{d})$.

PROPOSITION 3.3. Test-valid$(f,p,\delta,n,d,q,\psi,\phi)$ *satisfies* **(P2)**. *Its running time is bounded by* $O(\frac{1}{\delta^2}(\log\frac{1}{\min\{\psi,\phi\}})\binom{n+d}{d})$.

Next we describe the properties of a "brute-force" algorithm for reconstructing polynomials.

(P3) Brute-force$(f,\delta,n,d,q,\psi,\phi)$ returns a list that includes, with probability $1-\psi$, every degree d polynomial p such that f and p agree on δ fraction of the points. With probability at least $1-\phi$ it does not output any polynomial p whose agreement with f is less than $\frac{\delta}{2}$.

Notice that the goal of Brute-force is what one would expect to be the goal of Find-all-poly. Its weakness will be its running time, which is doubly exponential in n and exponential in d. However, we only invoke it for $n \leq 4$. In this case its running time is of the order of δ^{-d^4}. The description of Brute-force is given in Figure 3.1.

LEMMA 3.4. Brute-force$(f,\delta,n,d,q,\psi,\phi)$ *satisfies* **(P3)**. *Its running time is at most* $O(\frac{kl^3}{\delta^2}(\log\frac{k}{\phi}))$ *where* $l = \binom{n+d}{d}$ *and* $k = O\left((\delta - \frac{d}{q})^{-l}\left(\log\frac{1}{\psi}\right)\right)$.

Proof. The running time of Brute-force is immediate from its description (using the fact that a naive interpolation algorithm for a (multivariate) polynomial with l coefficients runs in time $O(l^3)$ and the fact that each call to Test-valid takes at most $O(\frac{l}{\delta^2}\log\frac{k}{\phi})$ time). If a polynomial p that is the output of the multivariate interpolation step has agreement less than $\frac{\delta}{2}$ with f, then by the correctness of Test-valid it follows that p is passed with probability at most ϕ/k. Summing up over the

[5]Notice that we do not make any claims about the probability with which constants that do not have significant agreement with f may be reported. In fact we do not need such a condition for our analysis. If required, such a condition may be explicitly enforced by "testing" every constant that is returned for sufficient agreement. Note also that the list is allowed to be empty if no polynomial has sufficiently large agreement.

Brute-force$(f, \delta, n, d, q, \psi, \phi)$
  Set $l = \binom{n+d}{d}$
      $k = O\left((\delta - \frac{d}{q})^{-l}\left(\log \frac{1}{\psi}\right)\right)$
      $\mathcal{L} \leftarrow \lambda$.
  Repeat k times
      Pick $x^{(1)}, \ldots, x^{(l)} \in_R \mathrm{GF(q)}^n$.
      Multivariate interpolation step:
        Find $p : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ of degree d s.t. $\forall i \in [l]$, $p(x^{(i)}) = f(x^{(i)})$.
      If Test-valid$(f, p, \delta, n, d, q, \frac{1}{2}, \phi/k)$ then $\mathcal{L} \leftarrow \mathcal{L} \cup \{p\}$.
  endRepeat
return$(\mathcal{L})$

FIG. 3.1. Brute-force

k iterations, we have that the probability that any polynomial with agreement less than $\frac{\delta}{2}$ is included in the output list is at most ϕ.

To prove that with probability at least $1 - \psi$, Test-valid outputs every polynomial p with δ agreement f, let us fix p and argue that in any one of the k iterations, p is likely to be added to the output list with probability $\zeta = \frac{1}{2(\delta - \frac{d}{q})^l}$. The lemma follows from the fact that the number of iterations is a sufficiently large multiple of $\frac{1}{\zeta}$.

To prove that with probability at least ζ the polynomial p is added to $\mathcal{L}$ (in a single iteration), we show that with probability at least 2ζ the polynomial interpolated in the iteration equals p. The lemma follows from the fact that Test-valid will return true with probability at least $\frac{1}{2}$.

To show that p is the polynomial returned in the interpolation step, we look at the task of finding p as the task of solving a linear system. Let $\vec{p}$ denote the l dimensional vector corresponding to the coefficients of p. Let M be the $l \times l$ dimensional matrix whose rows correspond to the points $x^{(1)}, \ldots, x^{(l)}$ and whose columns correspond to the monomials in p. Specifically, the entry $M_{i,j}$, where j corresponds to the monomial $x_1^{d_1} \ldots x_n^{d_n}$, is given by $(x_1^{(i)})^{d_1} \ldots (x_n^{(i)})^{d_n}$. Finally let $\vec{f}$ be the vector $(f(x^{(1)}), \ldots, f(x^{(l)}))$. To show that p is the polynomial returned in this step, we show that with high probability, M is of full rank and $p(x^{(i)}) = f(x^{(i)})$ for every i.

The last assertion is proven by induction on i. Let $M^{(i)}$ denote the $i \times l$ matrix with the first i rows of M. Fix $x^{(1)}, \ldots, x^{(i-1)}$ such that $p(x^{(j)}) = f(x^{(j)})$ for every $j \in [i-1]$. We argue that with probability at least $\delta - \frac{d}{q}$ over the choice of $x^{(i)}$, it holds that $p(x^{(i)}) = f(x^{(i)})$ AND the rank of $M^{(i)}$ is greater than that of $M^{(i-1)}$. It is easy to see that $f(x^{(i)}) = p(x^{(i)})$ with probability at least δ. To complete the proof it suffices to establish that the probability, over a random choice of $x^{(i)}$, that $M^{(i)}$ has the same rank as $M^{(i-1)}$ is at most $\frac{d}{q}$. Consider two polynomials p_1 and p_2 such that $p_1(x^{(j)}) = p_2(x^{(j)})$ for every $j \in [i-1]$. Then for the rank of $M^{(i)}$ to be the same as the rank of $M^{(i-1)}$ it must be that $p_1(x^{(i)}) = p_2(x^{(i)})$ (else the solutions to the ith system are not the same as the solutions to the $i-1$th system). But for distinct polynomials p_1 and p_2 the event $p_1(x^{(i)}) = p_2(x^{(i)})$ happens with probability at most $\frac{d}{q}$ for randomly chosen $x^{(i)}$. This concludes the proof of the lemma. □

As an extension of univariate interpolations, we have:

(P4) Compute-coefficients$(p^{(1)}, \ldots, p^{(d+1)}, r^{(1)}, \ldots, r^{(d+1)}, n, d, q, \psi)$ takes as input $d+1$ polynomials $p^{(j)}$ in $n-1$ variables of degree $d-1$ and $d+1$ values $r^{(j)} \in \mathrm{GF(q)}$

and returns a degree d polynomial $p : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ such that $p|_{r^{(j)}} \equiv p^{(j)}$ for every $j \in [d+1]$, if such a polynomial p exists (otherwise it may return anything).

Compute-coefficients works as a simple interpolation algorithm: Specifically it finds $d+1$ univariate polynomials $h_1, \ldots, h_{d+1}$ such that $h_i(r^{(j)})$ equals 1 if $i = j$ and 0 otherwise and then returns the polynomial $p(x_1, \ldots, x_n) = \sum_{j=1}^{d+1} h_j(x_n) \cdot p^{(j)}(x_1, \ldots, x_{n-1})$. Note that indeed

$$\begin{aligned} p(x_1, ..., x_{n-1}, r^{(j)}) &= \sum_{k=1}^{d+1} h_k(x_n) \cdot p^{(k)}(x_1, \ldots, x_{n-1}) \\ &= p^{(j)}(x_1, \ldots, x_{n-1}) \end{aligned}$$

Note that the polynomials $h_i(x) = \prod_{j \in \{1,\ldots,d+1\}, j \neq i} \left(\frac{x - r^{(j)}}{r^{(i)} - r^{(j)}} \right)$ depend only on the $r^{(j)}$'s. (Thus, it suffices to compute them once, rather than computing them from scratch for each monomial of p as suggested in Section 2.2.)

PROPOSITION 3.5. Compute-coefficients$(p^{(1)}, \ldots, p^{(d+1)}, r^{(1)}, \ldots, r^{(d+1)}, n, d, q, \psi)$ *satisfies* **(P4)**. *Its running time is* $O(d^2 \binom{n+d}{d})$.

3.2. The main routines. As mentioned earlier, the main subroutines are Find-all-poly and Extend, whose inputs and properties are described next. They take, among other inputs, a special parameter α which will be fixed later. For the sake of simplicity, we do not require Find-all-poly and Extend at this point to output only polynomials with good agreement. We will consider this issue later, when analyzing the running times of Find-all-poly and Extend.

(P5) Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$ returns a list of polynomials containing every polynomial of degree d on n variables that agrees with f on at least a δ fraction of the inputs. Specifically, the output list contains every degree d polynomial p with agreement δ with f, with probability at least $1 - \psi$.

The algorithm is described formally in Figure 3.2. Informally, the algorithm uses the ("trivial") subroutines for the base cases $n \leq 4$ or $d = 0$, and in the remaining (interesting) cases it iterates a randomized process several times. Each iteration is initiated by a random linear transformation of the coordinates. Then in this new coordinate system, Find-all-poly finds (using the "trivial" subroutine Brute-force) a list of all 4-variate polynomials having significant agreement with the 4-prefix of the oracle.[6] It then extends each polynomial in the list one variable at a time till it finds the n-prefix of the polynomial (which is the polynomial itself). Thus the crucial piece of the work is relegated to the subroutine Extend, which is supposed to extend a given $(i-1)$-prefix of a polynomial with significant agreement with f to its i-prefix. The goals of Extend are described next.

To simplify our notation, here onwards we assume that the elements of the field are named $0, \ldots, q-1$. In particular, we often use $p|_j$, for some integer $0 \leq j \leq d$, to represent the (j)-prefix of a function p.

[6] In principle we could apply Brute-force for any constant number of variables (and not just 4). However, since the running time is doubly-exponential in the number of variables, we try to use Brute-force only for a small number of variables. The need for using Brute-force when the number of variables is very small comes about due to the fact that in such a case (e.g., two variables) the randomization of the coordinate system does not operate well. Furthermore, applying Brute-force for univariate polynomials seems unavoidable. For simplicity of exposition, we choose to apply Brute-force also for 2, 3 and 4-variate polynomials. This allows better settings of some parameters and simplifies the calculations at the end of the proof of Lemma 3.6.

(P6) Extend$(f, p, \delta, n, d, q, \psi, \phi, \alpha)$ takes as input a degree d polynomial p in $n-1$ variables and with probability at least $1-\psi$ returns a list of degree d polynomials in n variables that includes every polynomial p^* that satisfies the following conditions:

1. p^* has agreement at least δ with f.
2. $p^*|_j$ has agreement at least $\alpha \cdot \delta$ with $f|_j$ for every $j \in \{0, \ldots, d\}$.
3. $p^*|_0 \equiv p$.

Figure 3.3 describes the algorithm formally. Extend returns all n-variable extensions p^*, of a given $(n-1)$-variable polynomial p, provided p^* agrees with f in a strong sense: p^* has significant agreement with f and each $p^*|_j$ has significant agreement with $f|_j$ (for every $j \in \{0, \ldots, d\}$). (The latter agreement requirement is slightly lower than the former.) To recover p^*, Extend first invokes Find-all-poly to find the polynomials $p^*|_j$ for $d+1$ values of j. This is feasible only if a polynomial $p^*|_j$ has good agreement with $f|_j$, for every $j \in \{0, \ldots, d\}$. Thus, it is crucial that when Extend is called with f and p, all p^*'s with good agreement with f also satisfy the stronger agreement property (above). We will show that the calling program (i.e., Find-all-poly at the higher level of recursion) will, with high probability, satisfy this property, by virtue of the random linear transformation of coordinates.

All the recursive calls (of Find-all-poly within Extend) always involve a smaller degree parameter, thereby ensuring that the algorithms terminate (quickly). Having found a list of possible values of $p^*|_j$, Extend uses a simple interpolation (subroutine Compute-coefficients) to find a candidate for p^*. It then uses Test-valid to prune out the many invalid polynomials that are generated this way, returning only polynomials that are close to f.

We now go on the formal analysis of the correctness of Find-all-poly and Extend.

3.3. Correctness of Find-all-poly and Extend. LEMMA 3.6. *If* $\alpha \leq 1 - \frac{1}{q}$, $\delta \geq \frac{d+1}{q}$, *and* $q \geq 3$ *then* Find-all-poly *satisfies* **(P5)** *and* Extend *satisfies* **(P6)**.

Proof. We prove the lemma by a double induction, first on d and for any fixed d, we perform induction on n. We shall rely on the properties of Compute-coefficients, Test-valid, Constants, and Brute-force, as established above.

Assume that Find-all-poly is correct for every $d' < d$ (for every $n' \leq n$ for any such d'.) We use this to establish the correctness of Extend$(f, p, n', d, q, \psi, \alpha)$ for every $n' \leq n$. Fix a polynomial p^* satisfying the hypothesis in **(P6)**. We will prove that p^* is in the output list with probability $1 - \frac{\psi}{N_{n',d,\delta}}$. The correctness of Extend follows from the fact that there are at most $N_{n',d,\delta}$ such polynomials p^* and the probability that there exists one for which the condition is violated is at most ψ.

To see that p^* is part of the output list, notice that, by the inductive hypothesis on Find-all-poly, when invoked with agreement parameter $\alpha \cdot \delta$, it follows that for any fixed $j \in \{0, \ldots, d\}$, the polynomial $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2(d+1)N_{n',d,\delta}}$. This follows from the fact that $p^*|_j - p$ and $f|_j - p$ have agreement at least $\alpha \cdot \delta$, the fact that $p^*|_j - p = p^*|_j - p^*|_0$ is a degree $d-1$ polynomial[7], and thus, by the inductive hypothesis on the correctness of Find-all-poly, such a polynomial should be in the output list. By the union bound, we have that for every $j \in \{0, \ldots, d\}$, the polynomial $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$, and in such a case $p^* - p$ will be one of the polynomials returned by an invocation of Compute-coefficients.

[7] To see that $p^*|_j - p^*|_0$ is a polynomial of total degree at most $d-1$, notice that $p^*(x_1, \ldots, x_n)$ can be expressed uniquely as $r(x_1, \ldots, x_{n-1}) + x_n q(x_1, \ldots, x_n)$, where degree of q is at most $d-1$. Thus $p^*|_j - p^*|_0 = j \cdot q(x_1, \ldots, x_{n-1}, j)$ is also of degree $d-1$.

Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$;
If $d = 0$ return(Constants(f, δ, n, q, ψ));
If $n \leq 4$ return(Brute-force$(f, \delta, n, d, q, \psi, \phi)$);

$\mathcal{L} \leftarrow \{\}$;
Repeat $O(\log \frac{N_{n,d,\delta}}{\psi})$ times:
  Pick a random nonsingular $n \times n$ matrix R over GF(q)
  Pick a random vector $b \in \mathrm{GF(q)}^n$.
  Let g denote the oracle given by $g(y) = f(R^{-1}(y - b))$.
  $\mathcal{L}_4 \leftarrow$ Brute-force$(g|_{0^{n-4}}, \delta, 4, d, q, \frac{1}{10n}, \phi)$.

  for $i = 5$ to n do
    $\mathcal{L}_i \leftarrow \{\}$ /* List of (d, i)-prefixes */
    for every polynomial $p \in \mathcal{L}_{i-1}$ do
      $\mathcal{L}_i = \mathcal{L}_i \cup$ Extend$(g|_{0^{n-i}}, p, \delta, i, d, q, \frac{1}{10n}, \phi, \alpha)$
    endfor
  endfor

  Untransform $\mathcal{L}_n$: $\mathcal{L}'_n \leftarrow \{p'(x) \stackrel{\text{def}}{=} p(Rx + b) | p \in \mathcal{L}_n\}$.
  $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}'_n$.
endRepeat
return($\mathcal{L}$)

FIG. 3.2. *Find-all-poly*

Extend$(f, \delta, p, n, d, q, \psi, \phi, \alpha)$.

  $\mathcal{L}' \leftarrow \{\}$.
  $\mathcal{L}^{(0)} \leftarrow \{\bar{0}\}$ (where $\bar{0}$ is the constant 0 polynomial).
  for $j = 1$ to d do
    $f^{(j)} \leftarrow f|_j - p$.
    $\mathcal{L}^{(j)} \leftarrow$Find-all-poly$(f^{(j)}, \alpha \cdot \delta, n, d - 1, q, \frac{\psi}{2N_{n,d,\alpha\cdot\delta}(d+1)}, \phi, \alpha)$.
  endfor

  for every $(d+1)$-tuple $(p^{(0)}, \ldots, p^{(d)})$ with $p^{(k)} \in \mathcal{L}^{(k)}$ do
    $p' \leftarrow$ Compute-coefficients$(p^{(0)}, \ldots, p^{(d)}, 0, \ldots, d;\ n, d, q)$.
    if Test-valid$(f, p + p', \delta, n, d, q, \psi/(2N_{n,d,\alpha\cdot\delta}), \phi)$ then
      $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{p + p'\}$;
  endfor
  return($\mathcal{L}'$).

FIG. 3.3. Extend

In such a case p^* will be tested by Test-valid and accepted with probability at least $1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$. Again summing up all the error probabilities, we have that p^* is in the output list with probability at least $1 - \frac{\psi}{N_{n',d,\alpha\cdot\delta}}$. This concludes the correctness of Extend.

We now move on to the correctness of Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$. Here we will try to establish that for a fixed polynomial p with agreement δ with f, the polynomial p is added to the list $\mathcal{L}$ with constant probability in each iteration of the Repeat loop. Thus the probability that it is not added in any of the iterations is at most $\frac{\psi}{N_{n,d,\delta}}$ and thus the probability that there exists a polynomial that is not added in any iteration is at most ψ. We may assume that $n \geq 5$ and $d \geq 1$ (or else correctness is guaranteed by the trivial subroutines).

Fix a degree d polynomial p with agreement δ with the function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$. We first argue that (R, b) form a "good" linear transformation with constant probability. Recall that from now onwards Find-all-poly works with the oracle $g : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ given by $g(y) = f(R^{-1}(y - b))$. Analogously define $p'(y) = p(R^{-1}(y - b))$, and notice p' is also a polynomial of degree d. For any $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$, we say that (R, b) is *good for* (i, j) if the agreement between $g|_{j,0^{n-i}}$ and $p'|_{j,0^{n-i}}$ is at least $\alpha\delta$. Lemma 3.7 (below) shows that the probability that (R, b) is good for (i, j) with probability at least $1 - \frac{1}{q^{i-1}} \cdot \left(2 + \frac{1}{\delta(1-\alpha)^2}\right)$. Now call (R, b) *good* if it is good for every pair (i, j), where $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$. Summing up the probabilities that (R, b) is not good for (i, j) we find that (R, b) is not good with probability at most

$$\begin{aligned}
&\sum_{j=0}^{d} \sum_{i=5}^{n} \left(2 + \frac{1}{\delta(1-\alpha)^2}\right) \cdot q^{-i+1} \\
&= (d+1) \cdot \left(2 + \frac{1}{\delta(1-\alpha)^2}\right) \cdot \sum_{i=5}^{n} q^{-i+1} \\
&< (d+1) \cdot \left(2 + \frac{1}{\delta(1-\alpha)^2}\right) \cdot \frac{q^{-3}}{q-1} \\
&\leq \frac{2}{q^2(q-1)} + \frac{1}{q-1} \quad \text{(Using } \alpha \leq 1 - \tfrac{1}{q}\text{, } \delta \geq \tfrac{d+1}{q}\text{, and } d+1 \leq q.\text{)} \\
&\leq \frac{11}{18} \quad \text{(Using } q \geq 3.\text{)}
\end{aligned}$$

Conditioned upon (R, b) being good and relying on the property of Brute-force, it follows that $\mathcal{L}_4$ contains the 4-prefix of p with probability at least $1 - \frac{1}{10n}$. Inductively, we have that the i-prefix of p is not contained in the list $\mathcal{L}_i$ with probability at most $\frac{i}{10n}$. (By the inductive hypothesis on Extend, with probability at most $\frac{1}{10n}$ the $(i-1)$-prefix of p is in $\mathcal{L}_{i-1}$ and yet the i-prefix is not returned by Extend.) Thus, with probability at most $\frac{1}{10}$, the polynomial p is not included in $\mathcal{L}_n$ (conditioned upon (R, b) being good). Adding back the probability that (R, b) is not good, we conclude that with probability at most $\frac{11}{18} + \frac{1}{10} < \frac{3}{4}$, the polynomial p is not in $\mathcal{L}_n$ in any single iteration. This concludes the proof of the correctness of Find-all-poly. □

3.4. Analysis of the random linear transformation. We now fill in the missing lemma establishing the probability of the "goodness" of a random linear transformation.

LEMMA 3.7. *Let f and g be functions mapping* $\mathrm{GF(q)}^n$ *to* $\mathrm{GF(q)}$ *that have δ agreement with each other, and let R be a random non-singular $n \times n$ matrix and b be a random element of* $\mathrm{GF(q)}^n$*. Then, for every $i \in \{1, \ldots, n\}$ and $j \in \mathrm{GF(q)}$:*

$$\Pr_{R,b}\left[f'|_{j,0^{n-i}} \text{ and } g'|_{j,0^{n-i}} \text{ have less than } \alpha\delta \text{ agreement}\right] \leq \frac{1}{q^{i-1}} \cdot \left(2 + \frac{1}{\delta(1-\alpha)^2}\right),$$

where $f'(y) = f(R^{-1}(y-b))$ *and* $g'(y) = g(R^{-1}(y-b))$.

Proof. Let $G = \{x \in \mathrm{GF(q)}^n | f(x) = g(x)\}$, be the set of "good" points. Observe that $\delta = |G|/q^n$. Let $S_{R,b} = \{x \in \mathrm{GF(q)}^n | Rx + b$ has $j0^{n-i}$ as suffix$\}$. Then we wish to show that

$$\Pr_{R,b}\left[\frac{|S_{R,b} \cap G|}{|S_{R,b}|} < \alpha \cdot \frac{|G|}{q^n}\right] \leq \frac{1}{q^{i-1}}\left(2 + \frac{1}{\delta(1-\alpha)^2}\right). \tag{3.1}$$

Observe that the set $S_{R,b}$ can be expressed as the pre-image of $(j, 0^{n-i})$ in the map $\pi : \mathrm{GF(q)}^n \to \mathrm{GF(q)}^m$, where $m = n-i+1$, given by $\pi(x) = R'x + b'$ where R' is the $m \times n$ matrix obtained by taking the bottom m rows of R and b' is the vector obtained by taking the last m elements of b. Note that R' is a uniformly distributed $m \times n$ matrix of full rank over GF(q) and b' is just a uniformly distributed m-dimensional vector over GF(q). We first analyze what happens when one drops the full-rank condition on R'.

Claim 3.8. *Let* R' *be a random* $m \times n$ *matrix over* GF(q) *and* b' *be a random element of* $\mathrm{GF(q)}^m$. *For some fixed vector* $\vec{s} \in \mathrm{GF(q)}^m$ *let* $S = \{x | R'x + b' = \vec{s}\}$. *Then, for any set* $G \subseteq \mathrm{GF(q)}^n$,

$$\Pr_{R',b'}\left[\frac{|S \cap G|}{|S|} < \alpha \cdot \frac{|G|}{q^n}\right] \leq \frac{q^m}{(1-\alpha)^2|G|} + q^{-(n-m)}.$$

Proof. We rewrite the probability in the claim as

$$\begin{aligned}
&\Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G| \cdot |S|}{q^n}\right] \\
&\leq \Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G| \cdot q^{n-m}}{q^n} \text{ or } |S| > q^{n-m}\right] \\
&\leq \Pr_{R',b'}\left[|S \cap G| < \alpha \cdot \frac{|G|}{q^m}\right] + \Pr_{R',b'}\left[|S| > q^{n-m}\right]
\end{aligned}$$

The event in the second term occurs only if the matrix R' is not full rank, and so the second term is bounded by $q^{-(n-m)}$ (see Claim 3.9). We thus focus on the first term.

For $x \in G \subseteq \mathrm{GF(q)}^n$, let $I(x)$ denote an indicator random variable that is 1 if $x \in S$ (i.e., $R'x + b' = \vec{s}$) and 0 otherwise. Then, the expected value of $I(x)$, over the choice of (R', b'), is q^{-m}. Furthermore, the random variables $I(x_1)$ and $I(x_2)$ are independent, for any distinct x_1 and x_2. Now, $|S \cap G| = \sum_{x \in G} I(x)$, and we are interested in the probability that the sum $\sum_{x \in G} I(x)$ is smaller than $\alpha \cdot |G| \cdot q^{-m}$ (whereas the expected value of the sum is $|G| \cdot q^{-m}$). A standard application of Chebychev's inequality yields the desired bound.[8] □

To fill the gap caused by the "full rank clause" (in the above discussion), we use the following claim.

Claim 3.9. *The probability that a randomly chosen* $m \times n$ *matrix over* GF(q) *is not of full rank is at most* $q^{-(n-m)}$.

Proof. We can consider the matrix as being chosen one row at a time. The probability that the jth row is dependent on the previous $j-1$ rows is at most q^{j-1}/q^n. Summing up over j going from 1 to m we get that the probability of getting a matrix not of full rank is at most $q^{-(n-m)}$. □

[8] Specifically, we obtain a probability bound of $\frac{|G| \cdot q^{-m}}{\left((1-\alpha) \cdot (|G| \cdot q^{-m})\right)^2} = \frac{q^m}{(1-\alpha)^2 \cdot |G|}$ as required.

Finally we establish (3.1). Let $E_{R',b'}$ denote the event that $\frac{|S\cap G|}{|S|} < \alpha \cdot \frac{|G|}{q^n}$ (recall that $S = S_{R',b'}$) and let $F_{R',b'}$ denote the event that R' is of full row rank. Then considering the space of uniformly chosen matrices R' and uniformly chosen vectors b' we are interested in the quantity:

$$\begin{aligned}\Pr_{R',b'}[E_{R',b'}|F_{R',b'}] &\le \Pr_{R',b'}[E_{R',b'}] + \Pr_{R',b'}[\neg(F_{R',b'})] \\ &\le \frac{q^m}{(1-\alpha)^2|G|} + 2\cdot q^{-(n-m)}.\end{aligned}$$

The lemma follows by substituting $m = n - i + 1$ and $|G| = \delta \cdot 2^n$. □

3.5. Analysis of the running time of Find-all-poly.

LEMMA 3.10. *For integers d_0, n_0, q and $\alpha, \delta_0 \in [0,1]$ satisfying $\alpha^{d_0}\delta_0 \ge 2d_0/q$, let $M = \max_{0\le d\le d_0}\{N_{n_0,d,(\alpha^{d_0-d})\cdot(\delta_0/2)}\}$. Then, with probability $1 - \phi\cdot(n_0^2(d_0+1)^2 M\log M)^{d_0+1}\cdot\log(1/\psi_0)$, the running time of* Find-all-poly$(f,\delta_0,n_0,d_0,q,\psi_0,\phi,\alpha)$ *is bounded by a polynomial in M^{d_0+1}, $(n_0+d_0)^{d_0}$, $(\frac{1}{\alpha^{d_0}\delta_0})^{(d_0+4)^4}$, $\log\frac{1}{\psi_0}$ and $\log\frac{1}{\phi}$.*

Proof. We fix n_0 and d_0. Observe that in all recursive calls to Find-all-poly, δ and d are related by the invariant $\delta = \alpha^{d_0-d}\delta_0$. Now, assuming the algorithms run correctly, they should only return polynomials with agreement at least $\delta/2$ (which motivates the quantity M). Further, in all such calls, we have that $\alpha^{d_0}\delta_0 - \frac{d}{q} \ge \alpha^{d_0}\delta_0/2$. Observe further that the parameter ϕ never changes and the parameter ψ only affects the number of iterations of the outermost call to Find-all-poly. In all other calls, this parameter (i.e., ψ) is at least $\psi_1 \stackrel{\text{def}}{=} \frac{1}{20n_0(d_0+1)M}$. Assume for simplicity that $\psi_0 \le \psi_1$. Let T_1, T_2, T_3, and T_4 denote the maximum running time of any of the subroutine calls to Constants, Test-valid, Brute-force, and Compute-coefficients, respectively. Let $T = \max\{T_1,T_2,T_3,T_4\}$. Then

$$T_1 = O\left(\frac{n}{\alpha^{2d_0}\delta_0^2}\cdot\log\frac{1}{\psi_0}\right)$$

$$T_2 = O\left(\frac{1}{\alpha^{2d_0}\delta_0^2}\cdot\binom{n_0+d_0}{d_0}\cdot\log\frac{1}{\min\{\psi_0,\phi\}}\right)$$

$$T_3 = O\left(\frac{kl^3}{(\alpha^{d_0}\delta_0/2)^2}\cdot\log\frac{k}{\phi}\right)$$

where $l = O((d_0+4)^4)$ and $k = O\left(\alpha^{-(d_0+4)^4}\cdot(\delta_0/2)^{-(d_0+4)^4}\cdot\log\frac{1}{\psi_0}\right)$.

$$T_4 = O\left(d_0^2\cdot\binom{n_0+d_0}{d_0}\right)$$

Note that all the above quantities are upper bounded by polynomials in $(n_0+d_0)^{d_0}$, $(\frac{2}{\alpha^{d_0}\delta_0})^{(d_0+4)^4}$, $\log M$, $\log\frac{1}{\phi}$, and thus so is T. In what follows we show that the running time is bounded by some polynomial in $(n_0d_0M)^{(d_0+1)}$ and T and this will suffice to prove the lemma.

Let $P(d)$ denote an upper bound on the probability that any of the recursive calls made to Find-all-poly by Find-all-poly$(f,\alpha^{d_0-d}\delta_0,n,d,q,\psi,\phi,\alpha)$ returns a list of length greater than M, maximized over f, $1\le n\le n_0$, $\psi\ge\psi_0$. Let $F(d)$ denote an upper bound on the running time on Find-all-poly$(f,\alpha^{d_0-d}\delta_0,n,d,q,\psi,\phi,\alpha)$, conditioned

upon the event that no recursive call returns a list of length greater than M. Similarly let $E(d)$ denote an upper bound on the running time of Extend, under the same condition.

We first derive recurrences for P. Notice that the subroutine Constants never returns a list of length greater than $\frac{2}{\alpha^{d_0}\delta_0}$ (every constant output must have a fraction of $\frac{\alpha^{d_0}\delta_0}{2}$ representation in the sampled points). Thus $P(0) = 0$. To bound $P(d)$ in other cases, we observe that every iteration of the Repeat loop in Find-all-poly contributes an error probability of at most ϕ from the call to Brute-force, and at most $n_0 - 4$ times the probability that Extend returns an invalid polynomial (i.e., a polynomial with agreement less than $\delta_d/2$ with its input function f). The probability that Extend returns such an invalid polynomial is bounded by the sum of $(d+1)\cdot P(d-1)$ [from the recursive calls to Find-all-poly] and $M^{d+1}\cdot\phi$ [from the calls to Test-valid]. (Notice that to get the final bound we use the fact that we estimate this probability only when previous calls do not produce too long a list.) Finally the number of iterations of the Repeat loop in Find-all-poly is at most $\log(M/\psi)$, by the definition of M. Recall that in the outer most call of Find-all-poly, we have $\psi = \psi_0$ whereas in all other calls $\psi \geq \psi_1$, where $\log(1/\psi_1) = \log(20n_0(d_0+1)M) < n_0(d_0+1)\log M$, for sufficiently large n_0. Thus summing up all the error probabilities , we have

$$P(d) \;<\; \log(M/\psi)\cdot n_0\cdot\big((d+1)\cdot P(d-1) + M^{d+1}\cdot\phi\big)$$

where for $d = d_0$ we use $\psi = \psi_0$ and otherwise $\psi = \psi_1$. It follows that

$$\begin{aligned}
P(d_0) &< \log(M/\psi_0)\cdot n_0\cdot\big((d_0+1)\cdot P(d_0-1) + M^{d+1}\cdot\phi\big)\\
&< \log(M/\psi_0)\cdot n_0\cdot(d_0+1)\cdot\big((n_0\cdot(d_0+1))^2\log M\big)^{d_0}\cdot M^{d+1}\cdot\phi\\
&< \big(n_0^2\cdot(d_0+1)^2\cdot M\log M\big)^{d_0+1}\cdot\phi\cdot\log(1/\psi_0)
\end{aligned}$$

A similar analysis for F and E yields the following recurrences:

$$\begin{aligned}
&F(0) \leq T\\
&F(d) \leq n_0^2(d_0+1)(\log M)\cdot E(d)\\
&E(d) \leq (d+1)F(d-1) + M^{d+1}T
\end{aligned}$$

Solving the recurrence yields $F(d) \leq (n_0^2(d_0+1)^2 M\log M)^{d+1}T$. This concludes the proof of the lemma. □

LEMMA 3.11. *For integers d_0, n_0 and $\alpha, \delta_0 \in [0,1]$, let*

$$M = \max_{0\leq d\leq d_0}\{N_{n_0,d,(\alpha^{d_0-d})\cdot(\delta_0/2)}\}.$$

If $\alpha \geq 1 - \frac{1}{d_0+1}$ and $\delta_0 \geq 2e\sqrt{\frac{d_0}{q}}$ then $M \leq O(\frac{1}{\delta_0^2})$.

Proof. We use Part (2) of Theorem 4.4, which claims that $N_{n,d,\delta} \leq \frac{1}{\delta^2-(d/q)}$, provided $\delta^2 \geq d/q$. Let $\delta_d = \alpha^{d_0-d}\delta_0$. Then $\delta_d/2 \geq (1-\frac{1}{d_0+1})^{d_0+1}\cdot(\delta_0/2) \geq \delta_0/2e \geq \sqrt{d/q}$, by the condition in the lemma. Thus M is at most $\frac{1}{\delta_d^2-(d/q)} \leq \frac{2}{\delta_d^2} = O(\frac{1}{\delta_0^2})$. □

THEOREM 3.12. *Given oracle access to a function f and suppose δ, k, d and q are parameters satisfying $\delta \geq \max\{\frac{d+1}{q}, 2e\sqrt{d/q}\}$ and $q \geq 3$. Let $\alpha = 1 - \frac{1}{d+1}$, $\psi = 2^{-k}$ and $\phi = 2^{-k}\cdot(n(d+1)\frac{1}{\delta_0^2})^{-2(d+1)}$. Then, given oracle access to a function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$, the algorithm* Find-all-poly$(f,\delta,n,d,q,\psi,\phi,\alpha)$ *runs in*

$\mathrm{poly}((k \cdot nd/\delta)^{O(d^4)})$-time and outputs, with probability at least $1-2^{-k}$, a list containing all degree d polynomials that agree with f on at least an δ fraction of the inputs. Furthermore, the list does not contain any polynomials that agree with f on less than an $\frac{\delta}{2}$ fraction of the inputs.

Remarks:

1. Thus, combining Theorems 2.1 and 3.12, we get reconstruction algorithms for all $d < q$, provided δ is large enough. Specifically, for the case $q = 2$ and $d = 1$, we invoke Theorem 2.1.
2. The constant $2e$ in the lower bound on δ can be replaced by $(1+\epsilon)e^{d/q}$, for any $\epsilon > 0$, by re-calibrating the subroutine Test-valid and by setting $\alpha = 1-\frac{1}{q}$.

Proof. The main part of the correctness claim follows from Lemma 3.6, and the running-time bound follows from Lemmas 3.10 and 3.11. (In particular, note that the condition $\alpha^{d_0}\delta_0 \geq 2d/q$ from Lemma 3.10 is met, since $\alpha^{d_0} = \frac{1}{e}$ and $\delta_0 \geq 2\sqrt{d/q} \geq 2d/q$.) The furthermore part follows from the proof of Lemma 3.10. □

4. Counting: Worst Case. In this section we give a worst-case bound on the number of polynomials that agree with a given function f on δ fraction of the points. In the case of linear polynomials our bound works for any $\delta > \frac{1}{q}$, while in the general case our bound works only for δ that is large enough. The bounds are derived using a very elementary property of polynomial functions, namely that two of them do not agree on too many points. In fact we first state and prove bounds for any generic "error correcting code" and then specialize the bound to the case of polynomials.

4.1. General error-correcting bounds. We first recall the standard definition of error-correcting codes. To do so we refer to strings over an alphabet $[q]$. For a string $R \in [q]^N$ (R for received word) and $i \in [N]$, we let $R(i)$ denote the ith coordinate of R. The Hamming distance between strings R_1 and R_2, denoted $\Delta(R_1, R_2)$, is the number of coordinates i where $R_1(i) \neq R_2(i)$.

DEFINITION 4.1 (Error correcting code). *For integers N, K, D and q an $[N, K, D]_q$ code is a family of q^K strings from $[q]^N$ such that for any two distinct strings in the family, the Hamming distance between them is at least D. That is, if $\mathcal{C} \subseteq [q]^N$ is an $[N, K, D]_q$ code then $|\mathcal{C}| = q^K$ and for every $C_1 \neq C_1 \in \mathcal{C}$ it holds that $\Delta(C_1, C_2) \geq D$.*

In the following theorem we take an arbitrary word $R \in [q]^N$ and consider the number of codeword that may have a Hamming distance of at most $(1-\delta) \cdot N$ from R (i.e., codewords that agree with R on at least $\delta \cdot N$ coordinates). We give an upper bound provided δ is sufficiently large (as a function of D/N).

THEOREM 4.2. *Let N, D and q satisfy $\frac{D}{N} < 1$ and define $\gamma \stackrel{\text{def}}{=} 1 - \frac{D}{N} > 0$. Let $\delta > 0$ and $R \in [q]^N$. Suppose that $C_1, \ldots, C_m \in [q]^N$ are distinct codewords from an $[N, K, D]_q$ code that satisfy $\Delta(R, C_j) \leq (1-\delta) \cdot N$, for all $j \in \{1, \ldots, m\}$. Then the following bounds hold:*

1. *If $\delta > \sqrt{2+\frac{\gamma}{4}} \cdot \sqrt{\gamma} - \frac{\gamma}{2}$ then $m < \frac{2}{\delta + \frac{\gamma}{2}}$.*
   It follows that if $\delta > \sqrt{2\gamma}$ then $m < 2/\delta$.
2. *If $\gamma \geq \frac{1}{q}$ and $\delta > \frac{1}{q} + \sqrt{(\gamma - \frac{1}{q}) \cdot (1 - \frac{1}{q})}$ then $m \leq \frac{(1-\gamma)\cdot\left(1-\frac{1}{q}\right)}{(\delta-(1/q))^2-(1-\frac{1}{q})(\gamma-\frac{1}{q})}$.*
   It follows that if ($\gamma \geq \frac{1}{q}$ and) $\delta > \min\{\sqrt{\gamma}, \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}\}$ then $m \leq \frac{1-\gamma}{\delta^2-\gamma} < \frac{1}{\delta^2-\gamma}$. In particular, for $\gamma = \frac{1}{q}$, the bounds hold for every $\delta > \frac{1}{q}$.

For small γ, the latter (simpler) expressions given in each of the two parts of the theorem provide good approximations to the former (tighter) expressions. The fact

that the former expressions imply the latter ones is obvious for Part (1), and is proved below for Part (2).

Additional Remarks:

1. The bounds in the two parts of the theorem apply in different situations and yield different bounds on m. The first bound applies for somewhat larger values of δ and yields a stronger bound that is $O(\frac{1}{\delta})$. The second bound applies also for smaller values of δ and yields a bound that grows as $\Theta(\frac{1}{\delta^2})$.
2. Note that Part (2) only considers codes with distance $D \leq (1-1/q)\cdot N$ (i.e., $\gamma \geq 1/q$). Still, the bound $m \leq \frac{(1-\gamma)\cdot\left(1-\frac{1}{q}\right)}{(\delta-(1/q))^2-(1-\frac{1}{q})(\gamma-\frac{1}{q})}$, holds also in case $\gamma < 1/q$, provided $\delta \geq 1/q$. (See Footnote 9 at the end of the proof of Part (2).) We mention that it is well known that codes with distance $D \geq (1-1/q)\cdot N$ have at most qN codewords, which immediately implies $m \leq qN \leq N/\gamma$ (for any $\gamma \geq 1/q$ regardless of δ).

Proof. [(of Part 1)] The bound in Part (1) is proven by a simple inclusion-exclusion argument. For any $m' \leq m$, we count the number of coordinates $i \in [N]$ that satisfy the property that one of the first m' codewords agree with R on coordinate i. Namely, let $\chi_j(i) = 1$ if $C_j(i) = R(i)$ and $\chi_j(i) = 0$ otherwise. Then, by inclusion-exclusion we get

$$\begin{aligned} N &\geq |\{i : \exists j\ \chi_j(i) = 1\}| \\ &\geq \sum_{j=1}^{m'} \sum_i \chi_j(i) - - \sum_{1\leq j_1<j_2\leq m'} \sum_i \chi_{j_1}(i)\chi_{j_2}(i) \\ &\geq m'\cdot\delta N - \binom{m'}{2} \cdot \max_{1\leq j_1<j_2\leq m'} |\{i : C_{j_1}(i) = C_{j_2}(i)\}| \end{aligned}$$

where the last inequality is due to the fact that C_j agrees with R on at least δN coordinates. Since two codewords R_1 and R_2 can agree on at most $N-D$ coordinates, we get:

$$\forall m' \leq m, \quad m'\delta N - \frac{m'(m'-1)}{2}\cdot(N-D) \leq N. \tag{4.1}$$

Consider the function $g(y) \stackrel{\text{def}}{=} \frac{\gamma}{2}\cdot y^2 - (\delta+\frac{\gamma}{2})\cdot y + 1$. Then (4.1) says that $g(m') \geq 0$, for every integer $m' \leq m$. Let α_1 and α_2 be the roots of g. To establish Part (1) we show that

- The roots α_1 and α_2 are both real numbers.
- The roots are both non-negative.
- $|\alpha_1 - \alpha_2| > 1$.
- $\min(\alpha_1, \alpha_2) < \frac{2}{\delta+\frac{\gamma}{2}}$.

Without loss of generality, suppose $\alpha_1 \leq \alpha_2$. It follows that $m \leq \alpha_1$, since otherwise $g(m') < 0$ for every $m' \in (\alpha_1, \alpha_2)$ and in particular for the integer $m' = \lfloor\alpha_1\rfloor + 1$, in contradiction to the above (i.e., $g(m') \geq 0$ for every $m' \leq m$).

Let $\beta = \gamma/2$. Then $g(y) = \beta y^2 - (\beta+\delta)\cdot y + 1$. The roots, α_1 and α_2 are real, provided that $\zeta \stackrel{\text{def}}{=} (\beta+\delta)^2 - 4\beta$ is positive which follows from a stronger requirement (see below). Without loss of generality, suppose $\alpha_1 \leq \alpha_2$. To guarantee $\alpha_2 - \alpha_1 > 1$, we require $2\cdot\frac{\sqrt{\zeta}}{2\beta} > 1$ which translates to $\zeta > \beta^2$ (and hence $\zeta > 0$ as required above). We need to show that

$$(\beta+\delta)^2 - 4\beta > \beta^2$$

which occurs if $\delta > \sqrt{\beta^2 + 4\beta} - \beta$. Plugging in the value of β we find that the last inequality is exactly what is guaranteed in the hypothesis of Part (1) of the theorem statement. Thus α_1 and α_2 are real and $\alpha_2 - \alpha_1 > 1$. Lastly, we bound the smaller root α_1. First we prove the upper bound.

$$\begin{aligned}
\alpha_1 &= \frac{\beta + \delta - \sqrt{(\beta+\delta)^2 - 4\beta}}{2\beta} \\
&= \frac{\beta+\delta}{2\beta} \cdot \left[1 - \left(1 - \frac{4\beta}{(\beta+\delta)^2}\right)^{1/2}\right] \\
&< \frac{\beta+\delta}{2\beta} \cdot \left[1 - \left(1 - \frac{4\beta}{(\beta+\delta)^2}\right)\right] \\
&= \frac{2}{\beta+\delta}
\end{aligned}$$

where the inequality follows by $\zeta > 0$. Again by plugging in the value of β we get the desired bound. For the lower bound, consider the first equality in the above displayed set of inequalities and note that since $\beta > 0$, we have

$$\alpha_1 \;=\; \frac{\beta + \delta - \sqrt{(\beta+\delta)^2 - 4\beta}}{2\beta} \;>\; 0.$$

□

Proof. [(of Part 2)] We first introduce some notation. In what follows we will use the arithmetic of integers modulo q to simplify some of our notation. This arithmetic will be used on the letters of the alphabet, i.e., the set $[q]$. For $j \in \{1, \ldots, m\}$ and $i \in [N]$ let $\Gamma_j(i) = 1$ if $C_j(i) \neq R(i)$ and 0 otherwise. (Notice that $\Gamma_j(i) = 1 - \chi_j(i)$.) For $j \in \{1, \ldots, m\}$, $t \in \{0, \ldots, q-1\}$ and $i \in [N]$ let $\Gamma_j^{(t)}(i) = 1$ if $C_j(i) - R(i) \equiv t \pmod q$ and 0 otherwise. Thus $\Gamma_j(i) = 1$ if and only if there exists $t \neq 0$ such that $\Gamma_j^{(t)}(i) = 1$. Let $w_j \stackrel{\text{def}}{=} |\{i : C_j(i) \neq R(i)\}| = \sum_i \Gamma_j(i)$ and let $\overline{w} = \frac{\sum_{j=1}^m w_j}{m}$. The fact that the C_j's are close to R implies that $w_j \leq (1-\delta) \cdot N$, for all j.

Our proof generalizes a proof due to S. Johnson (c.f., MacWilliams and Sloane [31]) for the case $q = 2$. The central quantity used to bound m in the binary case can be generalized in one of the two following ways:

$$S \equiv \sum_{j_1, j_2, i} \Gamma_{j_1}(i)\Gamma_{j_2}(i).$$

$$S' \equiv \sum_{j_1, j_2, i} \sum_{t \neq 0} \Gamma_{j_1}^{(t)}(i)\Gamma_{j_2}^{(t)}(i).$$

The first quantity sums, over all j_1, j_2, the number of coordinates for which C_{j_1} and C_{j_2} both differ from R. The second quantity sums, over all j_1, j_2, the number of coordinate where C_{j_1} and C_{j_2} agree with each other, but disagree from R by t. (Notice that the two quantities are the same for the case $q = 2$.) While neither one of the two quantities are sufficient for our analysis, their sum provides good bounds.

Lower bound on $S + S'$: The following bound is shown using counting arguments which consider the worst way to place a given number of differences between the C_j's

and R. Let $N_i = |\{j|C_j(i) \neq R(i)\}| = \sum_j \Gamma_j(i)$ and let $N_i^{(t)} = |\{j|C_j(i) - R(i) \equiv t \pmod q\}| = \sum_j \Gamma_j^{(t)}(i)$. Note that $\sum_i N_i = \sum_i \sum_{t\neq 0} N_i^{(t)} = m\overline{w}$. We can lower bound S as follows:

$$S = \sum_{j_1,j_2,i} \Gamma_{j_1}(i)\Gamma_{j_2}(i) = \sum_i N_i^2 \geq \frac{(m\overline{w})^2}{N}.$$

where the last inequality above follows from the fact that subject to the condition $\sum_i N_i = m\overline{w}$, the sum of N_i's squared is minimized when all the N_i's are equal. Similarly, using $\sum_i \sum_{t\neq 0} N_i^{(t)} = m\overline{w}$, we lower bound S' as follows:

$$S' = \sum_{j_1,j_2,i} \sum_{t\neq 0} \Gamma_{j_1}^{(t)}(i)\Gamma_{j_2}^{(t)}(i) = \sum_i \sum_{t\neq 0} (N_i^{(t)})^2 \geq \frac{(m\overline{w})^2}{(q-1)N}.$$

By adding the two lower bounds above we obtain:

$$S + S' \geq \frac{(m\overline{w})^2}{N} + \frac{(m\overline{w})^2}{(q-1)N} = \frac{\frac{q}{q-1} m^2\overline{w}^2}{N}. \tag{4.2}$$

Upper bound on $S + S'$: For the upper bound we perform a careful counting argument using the fact that the C_j's are codewords from an error-correcting code. For fixed $j_1, j_2 \in \{1, \ldots, m\}$ and $t_1, t_2 \in [q]$, let

$$M_{t_1t_2}^{(j_1j_2)} \equiv |\{i|\Gamma_{j_1}^{(t_1)}(i) = \Gamma_{j_2}^{(t_2)}(i) = 1\}|.$$

For every j_1, j_2, we view the $M_{t_1t_2}^{(j_1j_2)}$'s as elements of a $q \times q$ matrix $M^{(j_1j_2)}$. Now, S and S' can be expressed as sums of some of the elements of the matrices $M^{(j_1j_2)}$. Summing over the $(q-1) \times (q-1)$ minors of all the matrices we get:

$$S = \sum_{j_1,j_2} \sum_{t_1\neq 0} \sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)}$$

and summing the diagonal elements of $M^{(j_1j_2)}$ over all j_1j_2, we get

$$S' = \sum_{j_1j_2} \sum_{t\neq 0} M_{tt}^{(j_1j_2)}.$$

We start by upper bounding the internal sum above for fixed pair (j_1, j_2), $j_1 \neq j_2$. Since the C_j's are codewords from an $[N, K, D]_q$ code we have $R_{j_1}(i) = R_{j_2}(i)$ for at most $N - D$ values of i, so

$$\sum_{t\neq 0} M_{tt}^{(j_1j_2)} \;\leq\; N - D - M_{00}^{(j_1j_2)} \;=\; \gamma N - M_{00}^{(j_1j_2)}.$$

Note that the sum of the values of all elements of $M^{(j_1j_2)}$ equals N, and $N - w_{j_1}$ (resp. $N - w_{j_2}$) is equal to the sum of the values of the 0^{th} column (resp. row) of $M^{(j_1j_2)}$. To bound the remaining term in the summation above we use inclusion-exclusion as follows:

$$\begin{aligned}
&\sum_{t_1\neq 0} \sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \\
&= \sum_{t_1} \sum_{t_2} M_{t_1t_2}^{(j_1j_2)} - - \sum_{t_1} M_{t_10}^{(j_1j_2)} - - \sum_{t_2} M_{0t_2}^{(j_1j_2)} + M_{00}^{(j_1j_2)} \\
&= N - (N - w_{j_1}) - (N - w_{j_2}) + M_{00}^{(j_1j_2)} \\
&= w_{j_1} + w_{j_2} - N + M_{00}^{(j_1j_2)}.
\end{aligned}$$

Combining the bounds above we have (for $j_1 \neq j_2$)

$$\sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \leq (\gamma N - M_{00}^{(j_1j_2)}) + (w_{j_1} + w_{j_2} - N + M_{00}^{(j_1j_2)})$$
$$= w_{j_1} + w_{j_2} - (1-\gamma)\cdot N.$$

(The key point above is the cancellation of $M_{00}^{(j_1j_2)}$.) Observe that if $j_1 = j_2 = j$, then the quantity $\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)} = \sum_{t\neq 0} M_{tt}^{(jj)} = w_j$.

We now combine the bounds above as follows:

$$\begin{aligned} S + S' &= \sum_j \left(\sum_{t\neq 0} M_{tt}^{(jj)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)} \right) + \sum_{j_1\neq j_2} \left(\sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \right) \\ &\leq 2\sum_j w_j + \sum_{j_1\neq j_2} (w_{j_1} + w_{j_2} - (1-\gamma)N) \\ &= 2m^2\overline{w} - m(m-1)(1-\gamma)N. \end{aligned}$$

Thus, we get:

$$S + S' \;\leq\; (2\overline{w} - (1-\gamma)\cdot N)\cdot m^2 + (1-\gamma)\cdot N\cdot m. \tag{4.3}$$

Putting it together: Combining (4.2) and (4.3) and letting $\overline{\delta} = 1 - \overline{w}/N$, we get

$$\begin{aligned} m &\leq (1-\gamma)\cdot \frac{1}{(\frac{\overline{w}}{N})^2 \frac{q}{q-1} + 1 - \gamma - 2\cdot\frac{\overline{w}}{N}} \\ &= (1-\gamma)\cdot \frac{1}{(1-\overline{\delta})^2\frac{q}{q-1} + 1 - \gamma - 2(1-\overline{\delta})}. \end{aligned}$$

provided $(1-\overline{\delta})^2\frac{q}{q-1} + 1 - \gamma - -2(1-\overline{\delta}) \geq 0$. Let $g(x) \stackrel{\text{def}}{=} \frac{q}{q-1}x^2 - 2x + (1-\gamma)$. Note that $g(x)$ is monotone decreasing when $x \leq \frac{q-1}{q}$. Note further that $\frac{1}{q} \leq \delta \leq \overline{\delta}$ and thus we get:

$$m \leq (1-\gamma)\cdot \frac{1}{g(1-\delta)},$$

provided $g(1-\delta) > 0$. We need to bound δ so that $g(1-\delta) > 0$. Observe first that $g(x) = \frac{q}{q-1}\cdot\left(\frac{q-1}{q} - x\right)^2 - \left(\gamma - \frac{1}{q}\right)$. Thus $g(x) > 0$ if $\frac{q-1}{q} - x > \sqrt{\frac{q-1}{q}\cdot(\gamma - \frac{1}{q})}$. (Note that the expression in the square root is non-negative, since $\gamma \geq \frac{1}{q}$.)[9] In other words, $g(1-\delta) > 0$, provided $\delta > \frac{1}{q} + \sqrt{\left(1-\frac{1}{q}\right)\cdot\left(\gamma - \frac{1}{q}\right)}$. In this case the bound obtained on m is $\frac{1-\gamma}{g(1-\delta)} = \frac{1-\gamma}{\frac{q}{q-1}\cdot\left(\delta - \frac{1}{q}\right)^2 - \left(\gamma - \frac{1}{q}\right)}$. This is exactly as claimed in the main part of Part (2).

We now move on to prove secondary bounds claimed in Part (2). Firstly, we show that $g(1-\delta) > 0$ for $\delta > \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}$. This follows immediately from the above and

[9] For $\gamma < \frac{1}{q}$, the function g is positive everywhere. However to use the inequality $g(1-\delta) \leq g(1-\overline{\delta})$, we need $\delta \geq \frac{1}{q}$. This gives the bound claimed in Additional Remark 2 after Theorem 4.2.

the inequality:

$$\frac{1}{q}+\sqrt{\gamma-\frac{1}{q}} \;>\; \frac{1}{q}+\sqrt{\left(1-\frac{1}{q}\right)\cdot\left(\gamma-\frac{1}{q}\right)}.$$

Next, we verify that $g(1-\delta)>0$ for every $\delta>\sqrt{\gamma}$. Let $x=1-\delta$. Then $1-x=\delta>\sqrt{\gamma}$. In this case we have:

$$\begin{aligned} g(x) &= \left(1+\frac{1}{q-1}\right)x^2-2x+1-\gamma \\ &= (1-x)^2+\frac{1}{q-1}x^2-\gamma \\ &\geq (1-x)^2-\gamma \\ &> 0 \end{aligned}$$

Thus $g(1-\delta)>0$ provided $\delta>\min\{\sqrt{\gamma},\frac{1}{q}+\sqrt{\gamma-\frac{1}{q}}\}$. We now derive the claimed upper bounds on m. Setting $x=1-\delta$, and using $g(x)\geq(1-x)^2-\gamma$, we get $g(1-\delta)\geq\delta^2-\gamma$. Thus $m\leq\frac{1-\gamma}{g(1-\delta)}\leq\frac{1-\gamma}{\delta^2-\gamma}<\frac{1}{\delta^2-\gamma}$. □

4.2. The special case of polynomials. Recall that a function $f:\mathrm{GF(q)}^n\to\mathrm{GF(q)}$ may be viewed as a string of length q^n with letters from the set $[q]$. Viewed in this way we get the following construction of a code using multivariate polynomials. These codes are known as Reed-Muller codes in the coding theory literature.

PROPOSITION 4.3. *The collection of degree d polynomials in n variables over* GF(q) *form an $[N,K,D]_q$ code, for $N=q^n$, $K=\binom{n+d}{d}$ and $D=(q-d)\cdot q^{n-1}$.*

Proof. The parameters N and K follow by definition. The distance bound D is equivalent to the well-known fact [10, 38, 46] that two degree d (multivariate) polynomials over GF(q) may agree in at most d/q fraction of the inputs. □

Combining Theorem 4.2 with Proposition 4.3 (and using $\gamma=\frac{d}{q}$ in the theorem), we get the following upper bound on the number of polynomials with δ agreement with an arbitrary function.

THEOREM 4.4. *Let $\delta>0$ and $f:\mathrm{GF}(q)^n\to\mathrm{GF}(q)$. Suppose that $p_1,\ldots,p_m:\mathrm{GF}(q)^n\to\mathrm{GF}(q)$ are distinct degree d polynomials that satisfy*

$$\Pr_{x\in\mathrm{GF}(q)^n}[f(x)=p_i(x)]\geq\delta,\quad\forall\ i\in\{1,\ldots,m\}.$$

Then the following bounds hold:

1. *If $\delta>\sqrt{2+\frac{d}{4q}}\cdot\sqrt{\frac{d}{q}}-\frac{d}{2q}$ then $m<\frac{2}{\delta+\frac{d}{2q}}$.*
   In particular, if $\delta>\sqrt{2d/q}$ then $m<2/\delta$.
2. *If $\delta>\frac{1+\sqrt{(d-1)(q-1)}}{q}$ then $m\leq\frac{(q-d)(q-1)}{q^2}\cdot\frac{1}{\left(\delta-\frac{1}{q}\right)^2-\frac{(q-1)(d-1)}{q^2}}$.*
   In particular, if $\delta>\min\{\sqrt{\frac{d}{q}},\frac{1}{q}+\sqrt{\frac{d-1}{q}}\}$ then $m<\frac{1}{\delta^2-(d/q)}$.

We emphasize the special case of linear polynomials (i.e., $d=1$):

THEOREM 4.5. *Let $\epsilon>0$ and $f:\mathrm{GF}(q)^n\to\mathrm{GF}(q)$. Suppose that $p_1,\ldots,p_m:\mathrm{GF}(q)^n\to\mathrm{GF}(q)$ are distinct linear functions that satisfy $\Pr_{x\in\mathrm{GF}(q)^n}[f(x)=p_i(x)]\geq\frac{1}{q}+\epsilon$, for all $i\in\{1,\ldots,m\}$. Then $m\leq\left(1-\frac{1}{q}\right)^2\cdot\frac{1}{\epsilon^2}\leq\frac{4}{\epsilon^2}$.*

Proof. Just substitute $d=1$ and $\delta=\frac{1}{q}+\epsilon$ in the main part of Part (2) of Theorem 4.4. □

4.3. On the tightness of the upper bounds. We show that several aspects of the bounds presented above are tight. We start with the observation that Theorem 4.2 can not be extended to smaller δ without (possibly) relying on some special properties of the code.

PROPOSITION 4.6. *Let δ_0, γ_0 satisfy the identity*

$$\delta_0 = \frac{1}{q} + \sqrt{\left(\gamma_0 - \frac{1}{q}\right) \cdot \left(1 - \frac{1}{q}\right)} \tag{4.4}$$

Then for any $\epsilon > 0$, and for sufficiently large N, there exists an $[N, K, D]_q$ code $\mathcal{C}$, with $\frac{N-D}{N} \leq \gamma_0 + \epsilon$, a word $R \in [q]^N$ and $M \geq 2^{\Omega(\epsilon^2 N)}$ codewords $C_1, \ldots, C_M \in \mathcal{C}$ such that $\Delta(R, C_j) \leq (1 - (\delta_0 - \epsilon)) \cdot N$, for every $j \in [M]$.

Remark: The proposition above should be compared against Part (2) of Theorem 4.2. That part says that for δ_0 and γ_0 satisfying (4.4) and *any* $[N, K, D]_q$ code with $\frac{N-D}{N} = \gamma_0$, there exist at most $O(\frac{1}{\delta_0^2})$ codewords at distance at most $(1 - \delta_0) \cdot N$ from any string of length N. In contrast, the proposition says that if δ_0 is reduced slightly (to $\delta_0 - \epsilon$) and γ_0 increased slightly (to $\gamma_0 + \epsilon$), then there could be exponentially many codewords at this distance.

Proof. The bound is proven by a standard probabilistic argument. The code $\mathcal{C}$ will consist only of the codewords $C_1, \ldots, C_M$ that will be close to the string R. The codewords C_j's are chosen randomly and independently by the following process. Let $p \in [0, 1]$, to be determined shortly.

For every codeword C_j, each coordinate is chosen independently as follows: With probability p it is set to be 1, and with probability $1 - p$ it is chosen uniformly from $\{2, \ldots, q\}$. The string R is simply 1^N.

Observe that for any fixed j, the expected number of coordinates where R and C_j agree is pN. Thus with probability at most $2^{-\Omega(\epsilon^2 N)}$, the agreement between R and C_j is less than $(p - \epsilon)N$. It is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ so that the probability that there exists such a word C_j is less than $\frac{1}{2}$.

Similarly the expected agreement between C_i and C_j is $\left(p^2 + \frac{(1-p)^2}{q-1}\right) \cdot N$. Thus the probability that the agreement between a fixed pair is ϵN larger than this number is at most $2^{-\Omega(\epsilon^2 N)}$. Again it is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ such that the probability that such a pair C_i and C_j exists is less than $\frac{1}{2}$.

Thus there is a positive probability that the construction yields an $[N, \Omega(\frac{\epsilon^2 N}{\log q}), D]_q$ code with $\frac{N-D}{N} = p^2 + \frac{(1-p)^2}{q-1} + \epsilon$, so that all codewords are within a distance of $(1 - (p - \epsilon))N$ of the word R. Thus, the setting $\delta_0 = p$ and $\gamma_0 = p^2 + \frac{(1-p)^2}{q-1}$ would yield the proposition, once it is verified that this setting satisfies (4.4). The latter fact is easily verified by the following algebraic manipulations, starting with our setting of δ_0 and γ_0.

$$\begin{aligned}
\gamma_0 = \delta_0^2 + \frac{(1-\delta_0)^2}{q-1} & \\
\Leftrightarrow \frac{q}{q-1} \cdot \delta_0^2 - \frac{2}{q-1} \cdot \delta_0 + \frac{1}{q-1} - \gamma_0 = 0 & \\
\Leftrightarrow \delta_0^2 - \frac{2}{q} \cdot \delta_0 + \frac{1}{q} - \frac{q-1}{q} \cdot \gamma_0 = 0 &
\end{aligned}$$

$$\Leftrightarrow\left(\delta_0-\frac{1}{q}\right)^2=\left(\gamma_0-\frac{1}{q}\right)\cdot\left(1-\frac{1}{q}\right)$$
$$\Leftrightarrow\delta_0=\frac{1}{q}+\sqrt{\left(\gamma_0-\frac{1}{q}\right)\cdot\left(1-\frac{1}{q}\right)}$$

This concludes the proof. □

Next we move on to the tightness of the bounds regarding polynomials. We show that Theorem 4.5 is tight for $\delta=O(1/q)$, whereas Part (1) of Theorem 4.4 is tight for $\delta=\Theta(1/\sqrt{q})$ and $d=1$. The results below show that for a given value of δ that meets the conditions of the appropriate theorem, the value of m can not be made much smaller.

PROPOSITION 4.7. *Given a prime p, and an integer k satisfying $1<k\leq p/3$, let $\delta=k/p$. Then, there exists a function $f:\mathrm{GF}(p)\to\mathrm{GF}(p)$ and at least $m\stackrel{\text{def}}{=}\frac{1}{18(k-1)\delta^2}$ linear functions $f_1,\ldots,f_m:\mathrm{GF}(p)\to\mathrm{GF}(p)$ such that $|\{x|f_i(x)=f(x)\}|\geq\delta p=k$, for all $i\in\{1,\ldots,m\}$. Furthermore, if $\delta>\sqrt{1/p}$ then $m>\frac{1}{\delta}-1$.* For $\delta=\frac{2}{p}=\frac{1}{p}+\epsilon$, we get $m=\frac{1}{18\delta^2}$ (which establishes tightness of the bound $m\leq\frac{4}{\epsilon^2}=\frac{16}{\delta^2}$ given in Theorem 4.5). For $\delta=\sqrt{\frac{2}{p}}+\frac{1}{p}>\sqrt{\frac{2}{p}}$, we get $m>\frac{1}{\delta}-1$ (which establishes tightness of the bound $m\leq\frac{2}{\delta}$ given for $d=1$ in Part (1) of Theorem 4.4).

Proof. We start by constructing a relation $R\subset\mathrm{GF}(p)\times\mathrm{GF}(p)$ such that $|R|\leq p$ and there exist many linear functions $g_1,\ldots,g_m$ such that $|R\cap\{(x,g_i(x))|x\in\mathrm{GF}(p)\}|\geq k$ for all i. Later we show how to transform R and the g_i's so that R becomes a function that still agrees with each transformed g_i on k inputs.

Let $l=\lfloor p/k\rfloor$ and recall that $\delta=k/p$. Notice $l\approx\frac{1}{\delta}$ and $l\geq\frac{1}{\delta}-1$. The relation R consists of the $k\cdot l\leq p$ pairs in the square $\{(i,j)|0\leq i<k,0\leq j<l\}$. Let $\mathcal{G}$ be the set of all linear functions that agree with R in at least k places. We shall show that $\mathcal{G}$ has size at least $1/(18\delta^2(k-1))$. Given non-negative integers a,b s.t. $a\cdot(k-1)+b<l$, consider the linear function $g_{a,b}(x)=ax+b \bmod p$. Then, $g_{a,b}(i)\in\{0,\ldots,l-1\}$, for ever such (a,b) and $i\in\{0,\ldots,k-1\}$. Thus, $g_{a,b}(i)$ intersects R in k places. Lastly, we observe that there are at least $1/(18\delta^2(k-1))$ distinct pairs (a,b) s.t. $a\cdot(k-1)+b<l$: Fixing any $a<l$, there are at least $l-(k-1)a-1$ possible values for b, and so that the total number of pairs is at least

$$\begin{aligned}\sum_{a=0}^{\frac{l-1}{k-1}}l-(k-1)a-1&=\left(\frac{l-1}{k-1}+1\right)\cdot(l-1)-(k-1)\cdot\frac{\frac{l-1}{k-1}\cdot\left(\frac{l-1}{k-1}+1\right)}{2}\\&>\frac{(l-1)^2}{2(k-1)}\\&\geq\frac{(1-2\delta)^2}{2\delta^2(k-1)}\qquad(\text{Using } l\geq\tfrac{1-\delta}{\delta}.)\\&\geq\frac{1}{18\delta^2(k-1)}\qquad(\text{Using } \delta\leq\tfrac{1}{3}.)\end{aligned}$$

Next, we convert the relation R into a function in two stages. First we *stretch* the relation by a factor of l to get a new relation R'. That is, $R'\stackrel{\text{def}}{=}\{(l\cdot i,j)|(i,j)\in R\}$. We modify the functions $g_{a,b}\in\mathcal{G}$ accordingly: That is, $g'_{a,b}(x)\stackrel{\text{def}}{=}g_{a,b}(l^{-1}\cdot x)=(a\cdot l^{-1})x+b$, where l^{-1} is the multiplicative inverse of $l \pmod p$ and $g_{a,b}(x)=ax+b$. Thus, if $g_{a,b}(i)=j$, then $g'_{a,b}(l\cdot i)=j$, and so if $(i,g_{a,b}(i))\in R$ then $(l\cdot i,g'_{a,b}(l\cdot i))\in R'$.

It follows that is $g_{a,b}$ agrees with R on at least k places then $g'_{a,b}$ agrees with R' on at least k places. Thus, letting $\mathcal{G}'$ denote the set of linear functions that agree with R' in k places, we have $g'_{a,b} \in \mathcal{G}'$ if $g_{a,b} \in \mathcal{G}$. Moreover the map from $\mathcal{G}$ to $\mathcal{G}'$ is one-to-one (i.e., $g_{a,b}$ is mapped to $g'_{a,b} \equiv g_{l^{-1}\cdot a,b}$), implying $|\mathcal{G}'| \geq |\mathcal{G}|$. (Actually, the argument above extends to show that $|\mathcal{G}'| = |\mathcal{G}|$.)

We note that for all $a < l$ (which in turn is smaller than $p/2$), it holds that $l^{-1} \cdot a \not\equiv -1 \pmod{p}$. (This is the case since otherwise $a \equiv -l \equiv p - l \pmod{p}$, in contradiction to $a < p/2$.)

Last we introduce a slope to R', so that it becomes a function. Specifically, $R'' \stackrel{\text{def}}{=} \{(i+j, j) | (i,j) \in R'\} = \{(l \cdot i + j, j) | (i,j) \in R\}$. Notice that for any two distinct $(i_1, j_1), (i_1, j_2) \in R''$, we have $i_1 \neq i_2$ (since $i_1 = l \cdot i'_1 + j_1$, $i_2 = l \cdot i'_2 + j_2$, and $j_1, j_2 \in \{0, ..., l-1\}$), and so R'' can be extended to a function $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$ (i.e., if $(i,j) \in R''$ then $j = f(i)$). Now for every function $g'(x) = a'x + b' \in \mathcal{G}'$, consider the function $g''(x) = a''x + b''$, where $a'' = a'/(1+a')$ and $b'' = b'/(1+a')$ (and recalling that $a' \not\equiv -1 \pmod{p}$). Observe that if $g'(x) = y$, then

$$\begin{aligned} g''(x+y) &= \frac{a'}{1+a'} \cdot (x + g'(x)) + \frac{b'}{1+a'} \\ &= \frac{a'}{1+a'} \cdot (x + a'x + b') + \frac{1}{1+a'} \cdot b' \\ &= a'x + b' \ = \ y \end{aligned}$$

Thus, if g' agrees with R' in at least k places then g'' agrees with R'' in at least k places (since $(x, g'(x)) \in R'$ implies $(x + g'(x), g''(x + g'(x))) \in R''$ and $x_1 + g'(x_1) = (a'+1) \cdot x_1 + b'_1 \neq (a'+1) \cdot x_2 + b'_1 = x_2 + g'(x_2)$ for all $x_1 \neq x_2$), and hence g'' agrees with f in at least k places. Again, the mapping of g' to g'' is one-to-one (since the system $a'' = a'/(1+a')$ and $b'' = b'/(1+a')$ has at most one solution in (a', b')). Thus, if we use $\mathcal{G}''$ to denote the set of linear functions that agree with f in k places, then we have $|\mathcal{G}''| \geq |\mathcal{G}'| \geq |\mathcal{G}| \geq \frac{1}{18\delta^2(k-1)}$, as desired.

For the furthermore clause, observe that if $\delta > \sqrt{1/p}$ then our setting dictates $l - 1 < \sqrt{p} < k$ and so $\frac{l-1}{k-1} < 1$. Actually, in this case we may use $\{g_{0,b} : b = 0, ..., l-1\}$ in role of $\mathcal{G}$, $\mathcal{G}'$ and $\mathcal{G}''$, and derive $|\mathcal{G}| \geq l \geq \frac{1}{\delta} - 1$. □

Finally we note that the bounds in Theorem 4.4 always require δ to be larger than d/q. Such a threshold is also necessary, or else there can be exponentially many degree d polynomials close to the given function. This is shown in the following proposition.

PROPOSITION 4.8. *Let q be a prime-power, $d < q$ and $\delta = \frac{d}{q} - \frac{d-1}{q^2}$. Then, there exist an n-variate function f over $\mathrm{GF}(q)$, and at least q^{n-1} degree d polynomials that agree with f on at least a δ fraction of the inputs.* Note that for $d = 1$ we have $\delta = \frac{1}{q}$. Also, by a minor extension of the following proof, we may use in role of f any n-variate degree d polynomial over $\mathrm{GF}(q)$.

Proof. We use the all-zero function in role of f. Consider the family of polynomials having the form $\prod_{i=1}^{d-1}(x_1 - i) \cdot \sum_{i=2}^{n} c_i x_i$, where $c_2, ..., c_n \in \mathrm{GF}(q)$. Clearly, each member of this family is a degree d polynomial and the family contains q^{n-1} different polynomials. Now, each polynomial in the family is zero on inputs $(a_1, ..., a_n)$ satisfying either $a_1 \in \{1, ..., (d-1)\}$ or $\sum_{i=2}^{n} c_i a_i = 0$, where the c_i's are these specifying the polynomial in the collection. Since at least a $\frac{d-1}{q} + (1 - \frac{d-1}{q}) \cdot \frac{1}{q}$ fraction of the inputs satisfy this condition, the proposition follows. □

5. Counting: A Random Case. In this section we present a bound on the number of polynomials that can agree with a function f if f is chosen to look like

a polynomial p on some domain D and random on other points. Specifically, for $|D| \geq 2(d+1)\cdot q^{n-1}$, we show that with high probability p itself is the only polynomial that agrees with f on at least $|D|$ (and even $|D|/2$) points.

THEOREM 5.1. *Let $\delta \geq \frac{2(d+1)}{q}$. Suppose that D is an arbitrary subset of density δ in $\mathrm{GF}(q)^n$, and $p(x_1, ..., x_n)$ is a degree d polynomial. Consider a function f selected as follows:*

1. *f agrees with p on D;*
2. *the value of f on each of the remaining points is uniformly and independently chosen. That is, for every $x \in \overline{D} \stackrel{\text{def}}{=} \mathrm{GF}(q)^n \setminus D$, the value of $f(x)$ is selected at random in* GF(q).

Then, with probability at least $1 - \exp\{(n^d \log_2 q) - \delta^2 q^{n-2}\}$, the polynomial p is the only degree d polynomial that agrees with f on at least a $\delta/2$ fraction of the inputs.

Thus, for functions constructed in this manner, the output of our reconstruction algorithm will be a single polynomial; namely, p itself.

Proof. We use the fact that for two polynomials $p_1 \neq p_2$ in $\mathrm{GF}(q)^n$, $p_1(x) = p_2(x)$ on at most d/q fraction of the points in $\mathrm{GF}(q)^n$ [10, 38, 46]. Thus, except for p, no other degree d polynomial can agree with f on more than $\frac{d}{q} \cdot q^n$ points in D. The probability that any polynomial p' agrees with f on more than a $\frac{1}{q} + \epsilon$ fraction of the points in $\overline{D}$ is at most $\exp\{-\epsilon^2 q^n\}$. Furthermore, in order to agree with f on more than an $\frac{\delta}{2}$ fraction of all points, p' must agree with f on at least $\left(\frac{\delta}{2} - \frac{d}{q}\right) \cdot q^n$ of the points in $\overline{D}$, and so we can use $\epsilon \geq \frac{(\delta/2)-(d/q)}{1-\delta} - \frac{1}{q} > \frac{\delta}{2} - \frac{d+1}{q} + \frac{\delta\cdot((\delta/2)-(d/q))}{q} \geq \frac{\delta}{q}$. Thus, the probability that there exists a degree d n-variate polynomial, other than p, that agrees with f on at least an $\delta/2$ fraction of all points is at most $q^{n^d} \cdot \exp\{-\left(\frac{\delta}{q}\right)^2 q^n\}$, and the theorem follows. □

6. Hardness Results. In this section we give evidence that the (explicit or implicit) reconstruction problem may be hard for some choices of d and the agreement parameter δ, even in the case when $n = 1$. We warn the reader that the problems shown to be hard does differ in some very significant ways from the reconstruction problems considered in previous sections. In particular, the problems will consider functions and relations defined on some finite subset of a large field, either the field of rational numbers or a sufficiently large field of prime order, where the prime is specified in binary. The hardness results use the "large" field size crucially.

Furthermore, the agreement threshold for which the problem is shown hard is very small. For example, the hardness results of Section 6.2, defines a function $f : H_1 \times H_2 \to F$, where F is a large field and H_1, H_2 are small subsets of F. In such a hardness result, one should compare the threshold δ of agreement that is required, against $\frac{d}{\max\{|H_1|,|H_2|\}}$, since the latter ratio that determines the "distance" between two polynomials on this subset of the inputs. Our hardness results typically hold for $\delta \approx \frac{d+2}{\max\{|H_1|,|H_2|\}}$. We stress that the agreement is measured as a fraction of the subset mentioned above, rather than as a fraction of the n-tuples over the field (in case it is finite), which is much smaller.

6.1. NP-hardness for a variant of the univariate reconstruction problem. We define the following (*variant* of the) interpolation problem PolyAgree:

Input: Integers d, k, m, and a set of pairs $P = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ such that $\forall i \in [m]$, $x_i \in F$, $y_i \in F$, where F is either the field of rationals or a prime field given

by its size in binary.[10]

Question: Does there exist a degree d polynomial $p : F^n \to F$ for which $p(x_i) = y_i$ for at least k different i's?

We stress that the pairs in P are *not* required to have distinct x-components (i.e., $x_i = x_j$ may hold for some $i \neq j$). Our result takes advantage of this fact.

THEOREM 6.1. *PolyAgree is NP-hard.*

Remark: This result should be contrasted with the results of [40, 19]. They show that PolyAgree is easy provided $k \geq \sqrt{dm}$, while our result shows it is hard without this condition. In particular, the proof uses $m = 2d+3$ and $k = d+2$ (and so $k < \sqrt{dm}$). Furthermore, our result is established using a set of pairs in which $x_i = x_j$ holds for some $i \neq j$, whereas this never happens when given oracle access to a function (as in previous sections and in [40, 19]). In the next subsection, we show that in it is hard even in the oracle setting when the number of variables is at least two.

Proof. We present the proof for the case of the field of rational numbers only. It is easy to verify that the proof also holds if the field F has prime order that is sufficiently large (see parenthetical comments at the end of the proof for further details.)

We reduce from subset sum: Given integers $B, a_1, \ldots, a_\ell$, does there exist a subset of the a_i's that sum to B (without loss of generality, $a_i \neq 0$ for all i).

In our reduction we use the fact that degree d polynomials satisfy certain interpolation identities. In particular, let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$ for $1 \leq i \leq d+1$ and $\alpha_0 = -1$. Then $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ if and only if $(0, f(0)), (1, f(1)), \ldots, (d+1, f(d+1))$ lies on a degree d univariate polynomial.

We construct the following instance of PolyAgree. Set $d = l - 1$, $m = 2d + 3$ and $k = d+2$. Next, set $x_i \leftarrow i$, $x_{d+1+i} \leftarrow i$, $y_i \leftarrow a_i/\alpha_i$, and $y_{d+1+i} \leftarrow 0$ for $1 \leq i \leq d+1$. Finally, set $x_{2d+3} \leftarrow 0$ and $y_{2d+3} \leftarrow B$.

No polynomial can pass through both $(x_i, y_i) = (i, a_i/\alpha_i)$ and $(x_{d+1+i}, y_{d+1+i}) = (i, 0)$ for any i, since $a_i \neq 0$. We show that there is a polynomial of degree d that passes through $(0, B)$ and one of either $(i, 0)$ or $(i, a_i/\alpha_i)$ for each $1 \leq i \leq d+1$ if and only if there is a subset of $a_1, \ldots, a_{d+1}$ whose sum is B.

Assume that there is a polynomial p of degree d that passes through $(0, B)$ and one of $(i, 0)$ and $(i, a_i/\alpha_i)$ for each $1 \leq i \leq d+1$. Let S denote the set of indices for which $p(i) = a_i/\alpha_i$ (and $p(i) = 0$ for $i \in [d+1]\backslash S$). Then

$$0 = \sum_{i=0}^{d+1} \alpha_i p(i) = \alpha_0 \cdot B + \sum_{i \in S} \alpha_i \cdot \frac{a_i}{\alpha_i} = -B + \sum_{i \in S} a_i \tag{6.1}$$

Similarly, if there is set of indices S such that $\sum_{i \in S} a_i = B$, then we define f so that $f(0) = B$, $f(i) = a_i/\alpha_i$ for $i \in S$ and $f(i) = 0$ for $i \in [d+1]\backslash S$. Observing that $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ it follows that there is a degree d polynomial that agrees with f on $i = 0, ..., d+1$.

For the case where F is a finite field of order q, we assume that the integers B and $a_1, \ldots, a_{d+1}$ are all multiples of α_i for every i. (This assumption can be realized easily by multiplying all integers in the input by $\mathrm{lcm}(|\alpha_0|, \ldots, |\alpha_{d+1}|)$.) Further we pick $q > |B| + \sum_{i=1}^{d+1} |a_i|$. The only change to the proof is that the equalities in Equation (6.1) directly hold only modulo q. At this stage, we use the condition $q > |B| + \sum_{i=1}^{d+1} |a_i|$ to conclude that $B = \sum_{i \in S} a_i$.

[10] When F is the field of rational numbers, the input elements are assumed to be given as a ratio of two N-bit integers. In such a case the input size is measured in terms of the total bit length of all inputs.

□

6.2. NP-hardness of the reconstruction problem for $n \geq 2$. In the above problem, we did not require that the x_i's be distinct. Thus this result does not directly relate to the black box model used in this paper. The following result applies to our black box model for n-variate functions, for any $n \geq 2$.

We define a multivariate version of PolyAgree that requires that the x_i's be distinct. We actually define a parameterized family $\mathsf{FunctionalPolyAgree}_n$, for any $n \geq 1$.
Input: Integer d, a field F, a finite subset $H \subseteq F^n$, a rational number δ, and a function $f : H \to F$, given as a table of values.
Question: Does there exist a degree d polynomial $p : F^n \to F$ for which $p(x) = f(x)$ for at least δ fraction of the x's from H?

THEOREM 6.2. *For every $n \geq 2$, $FunctionalPolyAgree_n$ is NP-hard.*

Proof. We prove the theorem for $n = 2$. The other cases follow by simply making an instance where only the values of first two variables vary in the set H and the remaining variables are assigned some fixed value (say 0).

The proof of this theorem builds on the previous proof. As above we reduce from subset sum. Given an instance $B, a_1, \ldots, a_l$ of the subset sum problem, we set $d = l - 1$ and $k = 2(d + 1)$ and F to be the field of rationals. (We could also work over any prime field $\mathrm{GF}(q)$, provided $q \geq |B| + \sum_{i=1}^{n} |a_i|$.) Let $\delta = \frac{d+3}{2(d+2)}$. We set $H_1 = \{0, \ldots, d + 1\}$, $H_2 = [2k]$. and let $H = H_1 \times H_2$. For $i \in H_1$ we let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$ as before. For $i \in H_1 - \{0\}$, let $y_i = a_i/\alpha_i$ as before. The function f is defined as follows:

$$f(i,j) = \begin{cases} B & \text{if } i = 0 \\ y_i & \text{if } i \in H_1 - \{0\} \text{ and } j \in [k] \\ 0 & \text{otherwise (i.e., if } i \in H_1 - \{0\} \text{ and } j \in \{k+1, \ldots, 2k\} \end{cases}$$

This completes the specification of the instance of the FunctionalPolyAgree$_2$ problem. We now argue that if the subset sum instance is satisfiable then there exists a polynomial p with agreement δ (on inputs from H) with f. Let $S \in [l]$ be a subset such that $\sum_{i \in S} a_i = B$. Then the function

$$p(i,j) \stackrel{\text{def}}{=} p'(i) \stackrel{\text{def}}{=} \begin{cases} B & \text{if } i = 0 \\ y_i & \text{if } i \in S \\ 0 & \text{if } i \in H_1 \setminus S \end{cases}$$

is a polynomial in i of degree d (since $\sum_{i=0}^{d+1} \alpha_i p'(i) = -B + \sum_{i \in S} a_i = 0$). Furthermore, p and f agree in $2k + k(d+1)$ inputs from H. In particular $p(0, j) = f(0, j) = B$ for every $j \in [2k]$, $p(i, j) = f(i, j) = y_i$ if $i \in S$ and $j \in [k]$ and $p(i, j) = f(i, j) = 0$ if $i \notin S$ and $j \in \{k+1, \ldots, 2k\}$. Thus p and f agree on a fraction $\frac{2k+k(d+1)}{2(d+2)k} = \frac{d+3}{2(d+2)} = \delta$ of the inputs from H, as required.

We now argue that if instance of the FunctionalPolyAgree$_2$ problem produced by the reduction is satisfiable then the subset sum instance is satisfiable. Fix a polynomial p that has agreement δ with f; i.e., $p(i, j) = f(i, j)$ for at least $2k + k(d + 1)$ inputs from H. We argue first that in such a case $p(i, j) = p'(i)$ for some polynomial $p'(i)$ and then the proof will be similar to that of Theorem 6.1. The following claim is crucial in this proof.

CLAIM 6.3. *For any $i \in [d + 1]$, if $|\{j | p(i, j) = f(i, j)\}| \geq k$, then there exists $c_i \in \{0, y_i\}$ s.t. $p(i, j) = c_i$ for every $j \in [2k]$.*

Proof. Consider the function $p^{(i)}(j) \stackrel{\text{def}}{=} p(i,j)$. $p^{(i)}$ is a degree d polynomial in j. By hypothesis (and the definition of $f(i,j)$) we have, $p^{(i)}(j) \in \{0, y_i\}$ for k values of $j \in [2k]$. Hence $p^{(i)}(j) = 0$ for $k/2$ values of j or $p^{(i)}(j) = y_i$ for $k/2$ values of j. In either case we have that $p^{(i)}$, a degree d polynomial, equals a constant polynomial for $k/2 = d+1$ points implying that $p^{(i)}$ is a constant. That $p^{(i)}(j) = c_i \in \{0, y_i\}$ follows from the hypothesis and definition of f. □

From the claim above it follows immediately that for any $i \in [d+1]$, $|\{j | f(i,j) = p(i,j)\}| \leq k$. Now using the fact that f and p agree on $2k + k(d+1)$ inputs it follows that for every $i \in [d+1]$, $f(i,j) = p(i,j)$ for exactly k values of j; and $f(0,j) = p(0,j) = B$ for all values of j. Using the above claim again we conclude that we can define a function $p'(i) \stackrel{\text{def}}{=} c_i \in \{0, y_i\}$ if $i \in [d+1]$ and $p'(0) = B$ such that $p(i,j) = p'(i)$ for every $(i,j) \in H$. Furthermore $p'(i)$ is a degree d polynomial, since p is a degree d polynomial; and hence $\sum_{i=0}^{d+1} \alpha_i p'(i) = 0$. Letting $S = \{i \in [d+1] | y_i \neq 0\}$, we get $-B + \sum_{i \in S} \alpha_i y_i = 0$ which in turns implies $B = \sum_{i \in S} a_i$. Thus the instance of the subset sum problem is satisfiable. This concludes the proof. □

7. An application to complexity theory. In this section we use the reconstruction algorithm for linear-polynomials to prove the security of new, generic, hard-core *functions*. This generalizes the result of Goldreich and Levin [17] that provided hard-core *predicates*. We comment that an alternative construction of generic hard-core *functions* was also presented in [17], where its security was reduced to the security of a specific hard-core predicate via a "computational XOR lemma" due to [43]. For further details, see [16].

Loosely speaking, a function $h : \{0,1\}^* \to \{0,1\}^*$ is called a *hard-core* of a function $f : \{0,1\}^* \to \{0,1\}^*$ if h is polynomial-time, but given $f(x)$ it is infeasible to distinguish $h(x)$ from a random $|h(x)|$-bit long string. Thus, not only is $h(x)$ hard to find given $f(x)$, it is even hard to recognize pairs of the form $(h(x), f(x))$. Intuitively, if h is a hard-core of f then it must be hard to invert f (i.e., given $f(x)$ find x). We formulate all these notions below, assuming for simplicity that f is length-preserving, i.e., $|f(x)| = |x|$ for all $x \in \{0,1\}^*$.

DEFINITION 7.1 (one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called* one-way *if the following two conditions hold:*

1. *The function f is computable in polynomial-time.*
2. *For every probabilistic polynomial-time algorithm A, every polynomial p and all sufficiently large n*

$$\mathbf{P}_{x \in \{0,1\}^n}[A(f(x)) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

DEFINITION 7.2 (hard-core function). *A function $h : \{0,1\}^* \to \{0,1\}^*$ is called a* hard-core *of a function $f : \{0,1\}^* \to \{0,1\}^*$ if the following two conditions hold:*

1. *The function h is computable in polynomial-time.*
2. *For every probabilistic polynomial-time algorithm D, every polynomial p and all sufficiently large n*

$$\left|\mathbf{P}_{x \in \{0,1\}^n}[D(f(x), h(x)) = 1] - \mathbf{P}_{x \in \{0,1\}^n,\, r \in \{0,1\}^{|h(x)|}}[D(f(x), r) = 1]\right| < \frac{1}{p(n)}$$

THEOREM 7.3. *Let f' be a one-way function, and ℓ be a polynomial-time computable integer function satisfying $\ell(m) = O(\log m)$. For $x = (x_1, ..., x_m) \in \mathrm{GF}(2^{\ell(m)})^m$ and $y = (y_1, ..., y_m) \in \mathrm{GF}(2^{\ell(m)})^m$, let $f(x,y) \stackrel{\text{def}}{=} (f'(x), y)$ and $h(x,y) \stackrel{\text{def}}{=} \sum_{i=1}^m x_i y_i$.*

Then h is a hard-core of f. This theorem generalizes the result of Goldreich and Levin [17] from $\ell \equiv 1$ to any logarithmically bounded ℓ.

Proof. Assume for contradiction that there exists a probabilistic polynomial-time algorithm D and a polynomial p so that for infinitely many m's

$$\left| \begin{array}{l} \mathbf{P}_{x,y\in \mathrm{GF}(2^{\ell(m)})^m}[D(f(x,y),h(x,y))=1] \\ \quad - \quad \mathbf{P}_{x,y\in \mathrm{GF}(2^{\ell(m)})^m\,,\,r\in \mathrm{GF}(2^{\ell(m)})}[D(f(x,y),r)=1] \end{array} \right| \geq \frac{1}{p(m)}$$

Let M denote the set of integers m for which the above inequality holds; and let $\epsilon(m)$ denote the difference (inside the absolute value), and assume, without loss of generality, that it is positive. Using the above D we first prove the following:

Claim: There exists a probabilistic polynomial-time algorithm A that satisfies

$$\mathbf{P}_{x,y\in \mathrm{GF}(2^{\ell(m)})^m}[A(f(x,y))=h(x,y)] \geq 2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)}-1}$$

for every $m \in M$.

Proof: The claim may be established by analyzing the success probability of the following algorithm A: On input $z = f(x,y)$, uniformly select $r \in \mathrm{GF}(2^{\ell(m)})$, invoke $D(z,r)$, and return r if $D(z,r)=1$ and a uniformly selected value in $\mathrm{GF}(2^{\ell(m)}) \setminus \{r\}$ otherwise. Details follow.

$$\begin{aligned} \mathbf{P}_{x,y}[A(f(x,y))=h(x,y)] &= \mathbf{P}_{x,y,r}[A(f(x,y))=h(x,y)\ \&\ r=h(x,y)] \\ &\quad +\mathbf{P}_{x,y,r}[A(x,y)=h(x,y)\ \&\ r\neq h(x,y)] \\ &= \mathbf{P}_{x,y,r}[D(f(x,y),r)=1\ \&\ r=h(x,y)] \\ &\quad +\mathbf{P}_{x,y,r,r'}[D(f(x,y),r)=0\ \&\ r'=h(x,y)\neq r] \end{aligned}$$

where r' denotes a uniformly selected value in $\mathrm{GF}(2^{\ell(m)}) \setminus \{r\}$. Letting $L = 2^{\ell(m)}$, we have:

$$\begin{aligned} \mathbf{P}_{x,y,r}[D(f(x,y),r)=1\ \&\ r=h(x,y)] &= \frac{1}{L}\cdot \mathbf{P}_{x,y,r}[D(f(x,y),r)=1 \,|\, r=h(x,y)] \\ &= \frac{1}{L}\cdot \mathbf{P}_{x,y,r}[D(f(x,y),h(x,y))=1] \end{aligned}$$

On the other hand

$$\begin{aligned} &\mathbf{P}_{x,y,r,r'}[D(f(x,y),r)=0\ \&\ r'=h(x,y)\neq r] \\ &= \frac{1}{L-1}\cdot \mathbf{P}_{x,y,r}[D(f(x,y),r)=0\ \&\ r\neq h(x,y)] \\ &= \frac{1}{L-1}\cdot (\mathbf{P}_{x,y,r}[D(f(x,y),r)=0] - \mathbf{P}_{x,y,r}[D(f(x,y),r)=0\ \&\ r=h(x,y)]) \\ &= \frac{1}{L-1}\cdot \left(1-\mathbf{P}_{x,y,r}[D(f(x,y),r)=1] - \frac{1}{L} + \frac{1}{L}\cdot \mathbf{P}_{x,y}[D(f(x,y),h(x,y))=1]\right) \end{aligned}$$

Combining the above, we get

$$\begin{aligned}
&\mathbf{P}_{x,y}[A(f(x,y)) = h(x,y)] \\
&= \frac{1}{L} \cdot \mathbf{P}_{x,y,r}[D(f(x,y),h(x,y)) = 1] \\
&\quad + \frac{1}{L} + \frac{1}{L-1} \cdot \left(\frac{1}{L} \cdot \mathbf{P}_{x,y}[D(f(x,y),h(x,y)) = 1] - \mathbf{P}_{x,y,r}[D(f(x,y),r) = 1] \right) \\
&= \frac{1}{L} + \frac{1}{L-1} \cdot (\mathbf{P}_{x,y}[D(f(x,y),h(x,y)) = 1] - \mathbf{P}_{x,y,r}[D(f(x,y),r) = 1]) \\
&\geq \frac{1}{L} + \frac{1}{L-1} \cdot \epsilon(m)
\end{aligned}$$

and the claim follows. QED

Let A be as in the above claim. Recalling that $f(x,y) = (f'(x), y)$, observe that A gives rise to a function $F_{z'} : \mathrm{GF}(2^{\ell(m)})^m \to \mathrm{GF}(2^{\ell(m)})$ defined by $F_{z'}(y) \stackrel{\mathrm{def}}{=} A(z', y)$, and it holds that

$$\mathbf{P}_{x,y \in \mathrm{GF}(2^{\ell(m)})^m}[F_{f'(x)}(y) = \sum_{i=1}^{m} x_i y_i] \geq 2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)} - 1}$$

Thus, for at least an $\frac{\epsilon(m)}{2^{\ell(m)+1}}$ fraction of the possible $x = (x_1, ..., x_m)$'s it holds that

$$\mathbf{P}_{y \in \mathrm{GF}(2^{\ell(m)})^m}[F_{f'(x)}(y) = \sum_{i=1}^{m} x_i y_i] \geq 2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)+1}}$$

Applying Theorem 2.1 to such $F_{f'(x)}$, with $\delta = 2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)+1}}$, we obtain a list of all polynomials that agree with $F_{f'(x)}$ on at least a $2^{-\ell(m)} + \frac{\epsilon(m)}{2^{\ell(m)+1}}$ fraction of the inputs. This list includes x, and hence we have inverted the function f' in time $\mathrm{poly}(m/\delta) = \mathrm{poly}(|x|)$. This happens on a polynomial fraction of the possible x's, and thus we have reached a contradiction to the hypothesis that f' is a one-way function. □

Acknowledgments. We are grateful to Mike Kearns and Dana Ron for discussion regarding agnostic learning. We thank Eyal Kushilevitz and Yishay Mansour for comments regarding an early version of this paper. We thank Salil Vadhan for clarifying discussions. We thank Felix Wu for pointing out an error in a previous version of the paper. We thank the anonymous referees for pointing out numerous errors and other helpful comments.

REFERENCES

[1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[2] Dana Angluin and Mārtiņš Krikis. Learning with malicious membership queries and exceptions. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 57–66, New Brunswick, New Jersey, 12-15 July 1994. ACM Press.

[3] Sigal Ar, Richard J. Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing* 28(2): 487–510, April 1999.

[4] Sanjeev Arora and Madhu Sudan. Improved low degree testing and its applications. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 485–495, El Paso, Texas, 4-6 May 1997.

[5] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of Lecture Notes in Computer Science, pages 37–48, Rouen, France, 22-24 February 1990. Springer.

[6] Avrim Blum and Prasad Chalasani. Learning switching concepts. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 231-242, Pittsburgh, Pennsylvania, 27-29 July 1992. ACM Press.

[7] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of Lecture Notes in Computer Science, pages 278-291, 22-26 August 1993. Springer-Verlag.

[8] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, December 1993.

[9] Richard Cleve and Michael Luby. A note on self-testing/correcting methods for trigonometric functions. *International Computer Science Institute Technical Report*, TR-90-032, July, 1990.

[10] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, June 1978.

[11] P. Elias. List decoding for noisy channels. *1957-IRE WESCON Convention Record*, Pt. 2, pages 94-104, 1957.

[12] Yoav Freund and Dana Ron. Learning to model sequences generated by switching distributions. *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 41–50, Santa Cruz, California, ACM Press, 1995.

[13] Sally A. Goldman, Michael J. Kearns, and Robert E. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing*, 22(4):705-726, August 1993.

[14] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 32–42, New Orleans, Louisiana, 6-8 May 1991.

[15] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4): 169–174, September 1992.

[16] Oded Goldreich. Foundations of Cryptography – Fragments of a Book. Weizmann Institute of Science, February 1995. Available from the ECCC, `http://www.eccc.uni-trier.de/eccc/`.

[17] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25-32, Seattle, Washington, 15-17 May 1989.

[18] Oded Goldreich, Ronitt Rubinfeld and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995. IEEE.

[19] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757-1767, September 1999.

[20] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM,* 45(6): 983–1006, November 1998.

[21] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4): 807–837, August 1993.

[22] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions (extended abstract). *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pp. 273-282, Montreal, Quebec, Canada, 23-25 May 1994.

[23] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning (extended abstract). *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 341-352, Pittsburgh, Pennsylvania, ACM Press, 1992.

[24] Pascal Koiran. Efficient learning of continuous neural networks. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 348-355, New Brunswick, New Jersey, 12-15 July 1994. ACM Press.

[25] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331-1348, December 1993.

[26] Phillip D. Laird. *Learning From Good and Bad Data*. Kluwer Academic Publishers, Boston,

1988.

[27] Leonid A. Levin, Randomness and non-determinism. *Journal of Symbolic Logic*, 58(3):1102-1103, 1993. Also in *International Congress of Mathematicians: Abstracts of Invited Lectures.* p.155, Zurich, August 4, 1994.

[28] Richard J. Lipton. New directions in testing. *Distributed Computing and Cryptography: Proceedings of a DIMACS Workshop*, J. Feigenbaum and M. Merritt (Eds.), pages 191-202, American Mathematical Society, Providence, RI, 1991.

[29] Wolfgang Maass. Efficient agnostic PAC-learning with simple hypotheses. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pp. 67-75, New Brunswick, New Jersey, ACM Press, 1994.

[30] Wolfgang Maass. Agnostic PAC-learning of functions on analog neural nets. *Proceedings of the 6th conference on Advances in Neural Information Processing Systems*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector (Eds.), pp. 311–318, Morgan Kaufmann Publishers, Inc., 1994.

[31] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1981.

[32] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, April 1995.

[33] Dana Ron and Ronitt Rubinfeld. Learning fallible deterministic finite automata. *Machine Learning*, 18(2/3):149-185, Kluwer Academic Publishers, Boston, 1995.

[34] Ronitt Rubinfeld. On the robustness of functional equations. *SIAM Journal on Computing*, 28(6):1972–1997, December 1999.

[35] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.

[36] Yasubumi Sakakibara. On learning from queries and counterexamples in the presence of noise. *Information Processing Letters*, 37(5):279–284, 14 March 1991.

[37] Yasubumi Sakakibara and Rani Siromoney. A noise model on learning sets of strings. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 295–302, Pittsburgh, Pennsylvania, 27-29 July 1992. ACM Press.

[38] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[39] Robert Sloan. Types of noise in data for concept learning (extended abstract). *Proceedings of the 1988 Workshop on Computational Learning Theory*, pp. 91-96, MIT, ACM Press, 1988.

[40] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1): 180–193, March 1997.

[41] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pages 537-546, Atlanta, Georgia, 1-4 May 1999.

[42] Leslie G. Valiant. Learning disjunctions of conjunctions. *Proceedings of the 9th International Joint Conference on Artificial Intelligence,* pp. 560–566, Morgan Kauffman Publishers, 1985.

[43] Umesh V. Vazirani and Vijai V. Vazirani. Efficient and Secure Pseudo-Random Number Generation. In *Proc. 25th IEEE Symp. on Foundation of Computer Science*, pages 458–463, 1984.

[44] Hal Wasserman. Reconstructing randomly sampled multivariate polynomials from highly noisy data. *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, San Francisco, California, 25-27 January 1998.

[45] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. *US Patent*, Number 4,633,470, December 30, 1986.

[46] Richard E. Zippel. Probabilistic algorithms for sparse polynomials. *Proceedings of EUROSAM '79: International Symposium on Symbolic and Algebraic Manipulation*, E. Ng (Ed.), Lecture Notes in Computer Science, vol. 72, pp. 216–226, Springer 1979.

[47] Richard E. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, March 1990.

[48] Richard E. Zippel. *Effective Polynomial Computation*. Kluwer Academic Publishers, Boston, 1993.