SIAM J. COMPUT.
Vol. 42, No. 1, pp. 84–111

ERROR-CORRECTING DATA STRUCTURES*

VICTOR CHEN†, ELENA GRIGORESCU‡, AND RONALD DE WOLF§

Abstract. We study data structures in the presence of adversarial noise. We want to encode a given object in a succinct data structure that enables us to efficiently answer specific queries about the object, even if the data structure has been corrupted by a constant fraction of errors. We measure the efficiency of a data structure in terms of its *length* (the number of bits in its representation) and query-answering time, measured by the number of *bit-probes* to the (possibly corrupted) representation. The main issue is the trade-off between these two. This new model is the common generalization of (static) data structures and locally decodable error-correcting codes (LDCs). We prove a number of upper and lower bounds on various natural error-correcting data structure problems. In particular, we show that the optimal length of t-probe error-correcting data structures for the MEMBERSHIP problem (where we want to store subsets of size s from a universe of size n such that membership queries can be answered efficiently) is approximately the optimal length of t-probe LDCs that encode strings of length s. It has been conjectured that LDCs with small t must be superpolynomially long. This bad probes-versus-length trade-off carries over to error-correcting data structures for MEMBERSHIP and many other data structure problems. We then circumvent this problem by defining so-called *relaxed* error-correcting data structures, inspired by the notion of "relaxed locally decodable codes" developed in the PCP literature. Here the decoder is required to answer *most* queries correctly with high probability, and for the remaining queries the decoder with high probability either answers correctly or declares "don't know." Furthermore, if there is no noise on the data structure, it answers *all* queries correctly with high probability. We obtain positive results for the following two data structure problems: (1) MEMBERSHIP. We construct a relaxed error-correcting data structure for this problem with length nearly linear in $s \log n$ that answers membership queries with $O(1)$ bit-probes. This nearly matches the asymptotically optimal parameters for the noiseless case: length $O(s \log n)$ and one bit-probe, due to Buhrman et al. (2) UNIVARIATE POLYNOMIAL EVALUATION (namely, we want to store a univariate polynomial g of degree $\deg(g) \leq s$ over the integers modulo n such that evaluation queries can be answered efficiently; i.e., we can evaluate the output of g on a given integer modulo n). We construct a relaxed error-correcting data structure for this problem with length nearly linear in $s \log n$ that answers evaluation queries with $\text{polylog}(s) \cdot \log^{1+o(1)}(n)$ bit-probes. This nearly matches the parameters of the best known noiseless construction due to Kedlaya and Umans.

Key words. data structures, fault-tolerance, error-correcting codes, locally decodable codes, membership problem, polynomial evaluation

AMS subject classifications. 68P05, 68P30, 94B65

DOI. 10.1137/110834949

1. Introduction.

1.1. Background.

Data structures deal with one of the most fundamental questions of computer science: how can we store objects in a way that both is space-efficient and that enables us to efficiently answer questions about the object? Thus, for in-

*Received by the editors May 23, 2011; accepted for publication (in revised form) September 6, 2012; published electronically January 10, 2013. This paper is based on two conference papers [15, 14].

http://www.siam.org/journals/sicomp/42-1/83494.html

†Oracle Corp., Redwood Shores, CA 94065 (victor.chen@alum.mit.edu). The work of this author was done while he was a Ph.D. student at MIT.

‡Department of Computer Science, Purdue University, West Lafayette, IN 47907 (elena-g@purdue.edu). This work was initiated when this author visited CWI in the summer of 2008 as an MIT Ph.D. student. The work of this author was supported in part by NSF award CCF-0829672 and NSF award 1019343 to the Computing Research Association for the Computing Innovation Fellowship Program.

§CWI and University of Amsterdam, Amsterdam, The Netherlands (rdewolf@cwi.nl). The work of this author was partially supported by a Vidi grant from the Netherlands Organization for Scientific Research (NWO) and by the European Commission under projects QAP (015848) and QCS (255961).

stance, it makes sense to store a set as an ordered list or as a heap-structure, because this is space-efficient and allows us to determine quickly (in time logarithmic in the size of the set) whether a certain element is in the set or not.

From a complexity-theoretic point of view, the aim is usually to study the trade-off between the two main resources of the data structure: the length/size of the data structure (storage space) and the efficiency with which we can answer specific queries about the stored object. To make this precise, we measure the length of the data structure in bits, and measure the efficiency of query-answering in the number of *probes*, i.e., the number of bit-positions in the data structure that we look at in order to answer a query.

The following definition is adapted from Miltersen's survey [31].

DEFINITION 1 (data structure). *Let X be a set of data items, Q be a set of queries, A be a set of answers, and $f : X \times Q \to A$. Let $t > 0$ be an integer and $\varepsilon \in [0, 1/2]$. A (t, ε)-*data structure *for f of length N is a map $\mathcal{E} : X \to \{0,1\}^N$ (the "encoding") for which there is a randomized algorithm $\mathcal{D}$ (the "decoder") with the following properties: for every $x \in X$,*

1. *$\mathcal{D}$ makes at most t probes to its oracle for $\mathcal{E}(x)$,*
2. *$\Pr[\mathcal{D}^{\mathcal{E}(x)}(q) = f(x, q)] \geq 1 - \varepsilon$ for every $q \in Q$.*

Here $\mathcal{D}^{\mathcal{E}(x)}(q)$ denotes the random variable which is the decoder's output on inputs $\mathcal{E}(x)$ and q. The notation indicates that $\mathcal{D}$ accesses its two inputs in different ways: while it has full access to the query q, it has only bit-probe access (or "oracle access") to the string $\mathcal{E}(x)$.

Most standard data structures taught in undergraduate computer science are deterministic, and hence have error probability $\varepsilon = 0$. However, it is convenient for us to already allow for some error probability in the above definition, as it will facilitate generalizing this definition to the noisy case later. As mentioned, the main complexity issue here is the trade-off between N and t.

Some data structure problems that we will consider are the following:

- EQUALITY. $X = Q = \{0,1\}^n$, and $f(x, y) = 1$ if $x = y$, $f(x, y) = 0$ if $x \neq y$. This is not a terribly interesting data structure problem in itself, since for every x there is only one query y for which the answer is "1"; we merely mention it here because it will be used to illustrate some definitions later on.
- s-out-of-n MEMBERSHIP. $X = \{x \in \{0,1\}^n : \text{Hamming weight } |x| \leq s\}$, $Q = [n] := \{1, \ldots, n\}$, and $f(x, i) = x_i$. In other words, x corresponds to a set of size at most s from a universe of size n, and we want to store the set in a way that easily allows us to make membership queries. This is probably the most basic and widely studied data structure problem [19, 39, 10, 34]. Note that for $s = 1$ this is EQUALITY on $\log n$ bits, while for $s = n$ it is the general MEMBERSHIP problem without constraints on the set.
- POLYNOMIAL EVALUATION. Let $\mathbb{Z}_n$ denote the set of integers modulo n, and let $s \leq n$ be some nonnegative integer. Given a univariate polynomial $g \in \mathbb{Z}_n[X]$ of degree at most s, we would like to store g in a compact representation so that for each evaluation query $a \in \mathbb{Z}_n$, $g(a)$ can be computed efficiently. Formally, $X = \{g \in \mathbb{Z}_n[X] : \deg(g) \leq s\}$, $Q = \mathbb{Z}_n$, and $A = \mathbb{Z}_n$, and the function is $\text{POLYEVAL}_{n,s}(g, a) = g(a)$.
- SUBSTRING. $X = \{0,1\}^n$, $Q = \{y \in \{0,1\}^n : |y| \leq r\}$, $f(x, y) = x_y$, where x_y is the $|y|$-bit substring of x indexed by the 1-bits of y (e.g., $1010_{0110} = 01$). For $r = 1$ this is MEMBERSHIP.
- INNER PRODUCT ($\text{IP}_{n,r}$). $X = \{0,1\}^n$, $Q = \{y \in \{0,1\}^n : |y| \leq r\}$, and $f(x, y) = x \cdot y \bmod 2$. This problem is among the hardest Boolean problems,

where the answer depends on at most r bits of x (again, for $r = 1$ it is MEMBERSHIP).

Many other data structure problems (RANK, PREDECESSOR, NEAREST NEIGHBOR, etc.) have also been studied widely (see, for example, [4, 21, 33, 1]), but we will not consider them here.

In this work we extend Definition 1 to two natural settings: *error-correcting data structures* and *relaxed error-correcting data structures.* We provide efficient constructions of such data structures for the problems that we described here. We start by introducing error-correcting data structures and then describe our results and techniques employed in the proofs. We then further motivate the generalization to *relaxed* error-correcting data structures and discuss our second set of results and techniques.

1.2. Error-correcting data structures. One issue that Definition 1 ignores is the issue of *noise*. Memory and storage devices are not perfect: the world is full of cosmic rays, small earthquakes, random (quantum) events, bypassing trams, etc., that can cause a few errors here and there. Another potential source of noise is transmission of the data structure over some noisy channel. Of course, better hardware can partly mitigate these effects, but in many situations it is realistic to expect a small fraction of the bits in the storage space to become corrupted over time. Our goal in this paper is to study *error-correcting* data structures. These still enable efficient computation of $f(x, q)$ from the stored data structure $\mathcal{E}(x)$, even if the latter has been corrupted by a constant fraction of errors. In analogy with the usual setting for error-correcting codes [28, 27], we will take a pessimistic, adversarial view of errors here: we want to be able to deal with a constant fraction of errors *no matter where they are placed*. We consider only bit-flip-errors here, not erasures. Since erasures are easier to handle than bit-flips, for the purposes of upper bounds it suffices to design a data structure dealing with bit-flips.

Formally, we define error-correcting data structures as follows.

DEFINITION 2 (error-correcting data structure (ECDS)). *Let X be a set of data items, Q be a set of queries, A be a set of answers, and $f : X \times Q \to A$. Let $t > 0$ be an integer, $\delta \in [0, 1]$, and $\varepsilon \in [0, 1/2]$. A (t, δ, ε) ECDS for f of length N is a map $\mathcal{E} : X \to \{0,1\}^N$ for which there is a randomized algorithm $\mathcal{D}$ with the following properties: for every $x \in X$ and every $w \in \{0,1\}^N$ at Hamming distance $\Delta(w, \mathcal{E}(x)) \leq \delta N$ from $\mathcal{E}(x)$,*

1. *$\mathcal{D}$ makes at most t probes to its oracle for w,*
2. *$\Pr[\mathcal{D}^w(q) = f(x, q)] \geq 1 - \varepsilon$ for every $q \in Q$.*

Definition 1 is the special case of Definition 2 where $\delta = 0$.[1] Note that if $\delta > 0$, then the adversary can always set the errors in a way that gives the decoder $\mathcal{D}$ a nonzero error probability. Hence the setting with bounded error probability is the natural one for ECDSs. This contrasts with the standard noiseless setting, where one usually considers deterministic structures.

For the data structure problems considered in this paper, our decoding procedures make only *nonadaptive* probes; i.e., the positions of the probes are determined all at once and sent simultaneously to the oracle. For other data structure problems it may

[1] As [10, end of section 1.1] notes, a data structure can be viewed as a locally decodable source code. With this information-theoretic point of view, an *error-correcting* data structure is a locally decodable combined source-*channel* code, and our results for MEMBERSHIP show that one can sometimes do better than combining the best source code with the best channel code. We thank one of the anonymous referees of [15] for pointing this out.

be natural for decoding procedures to be adaptive. Thus, we do not require $\mathcal{D}$ to be nonadaptive in condition 1 of Definition 2.

To illustrate the definition, here is a simple yet efficient ECDS for EQUALITY: encode x with an error-correcting code $\mathcal{E} : \{0,1\}^n \to \{0,1\}^N$ that has constant rate and minimal distance close to $N/2$ (for instance, take a random linear code). Then $N = O(n)$, and we can decode by one probe: given query y, probe $\mathcal{E}(x)_j$ for uniformly chosen $j \in [N]$, compare it with $\mathcal{E}(y)_j$, and output 1 iff these two bits are equal. Suppose up to a δ-fraction of the bits in $\mathcal{E}(x)$ are corrupted. If $x = y$, then we will give the correct answer 1 with probability $1 - \delta$, and if $x \neq y$, then we will give the correct answer 0 with probability roughly $1/2 - \delta$. By outputting 0 with probability $1/3$ and running the above 1-probe procedure with probability $2/3$, these two error probabilities can be balanced to 2-sided error $\varepsilon = 1/3 + O(\delta)$. The error can be reduced further by allowing more than one bit-probe.

We deal only with so-called *static* data structures here: we do not worry about updating the x that we have encoded. What about *dynamic* data structures, which allow efficient updates as well as efficient queries to the encoded object? Note that if data items x and x' are distinguishable in the sense that $f(x,q) \neq f(x',q)$ for at least one query $q \in Q$, then their respective encodings $\mathcal{E}(x)$ and $\mathcal{E}(x')$ must have distance greater than $2\delta N$, for otherwise δN errors on $\mathcal{E}(x)$ and δN errors on $\mathcal{E}(x')$ could yield the same string w.[2] Hence updating the encoded data from x to x' will require $\Omega(N)$ changes in the data structure $\mathcal{E}(x)$, which shows that a dynamical version of our model of error-correcting data structures with efficient updates is not possible.

ECDSs and locally decodable codes. Error-correcting data structures not only generalize the standard (static) data structures (Definition 1), but they also generalize *locally decodable codes*, which are error-correcting data structures for MEMBERSHIP with $s = n$. These are defined as follows:

DEFINITION 3 (locally decodable code (LDC)). *Let $t > 0$ be an integer, $\delta \in [0,1]$, and let $\varepsilon \in [0, 1/2]$. A (t, δ, ε)-LDC of length N is a map $\mathcal{E} : \{0,1\}^n \to \{0,1\}^N$ for which there is a randomized algorithm $\mathcal{D}$ with the following properties: for every $x \in \{0,1\}^n$ and every $w \in \{0,1\}^N$ at Hamming distance $\Delta(w, \mathcal{E}(x)) \leq \delta N$,*

1. *$\mathcal{D}$ makes at most t probes to its oracle for w,*
2. *$\Pr[\mathcal{D}^w(i) = x_i] \geq 1 - \varepsilon$ for every $i \in [n]$.*

Much work has been done on LDCs, but their length-versus-probes trade-off is still poorly understood. For $t = 2$ probes, the exponential-length Hadamard code is essentially optimal.[3] For $t = 3$ there are some beautiful constructions of LDCs of length roughly $2^{2^{\sqrt{\log n}}}$ [40, 16], which is much less than the Hadamard code's exponential length, but still far from polynomial length. For constant t we do not know LDCs of polynomial length, and it seems reasonable to conjecture that none exist (the best lower bounds for constant $t > 2$ are superlinear but not much stronger than that [23, 25, 37]). Once we allow polylog queries to our LDC-decoder, the length becomes very efficient. In particular, for fixed $\eta > 0$, there exist LDCs with

[2] Hence if all pairs $x, x' \in X$ are distinguishable (which is usually the case), then $\mathcal{E}$ is an error-correcting code.

[3] The Hadamard code of $x \in \{0,1\}^n$ is the codeword of length 2^n obtained by concatenating the bits $x \cdot y \pmod 2$ for all $y \in \{0,1\}^n$. It can be decoded by two probes, since for every $y \in \{0,1\}^n$ we have $(x \cdot y) \oplus (x \cdot (y \oplus e_i)) = x_i$. Picking y at random, decoding from a δ-corrupted codeword will be correct with probability $\geq 1 - 2\delta$, because both probes y and $y \oplus e_i$ are individually random and hence probe a corrupted entry with probability $\leq \delta$. This exponential length is optimal for 2-probe LDCs [25].

$t = (\log n)^{O(1/\eta)}$ probes and length $n^{1+\eta}$; see, for instance, [41, section 2.3]. We refer the reader to [36, 41] and the references therein for further context.[4]

LDCs address only a very simple type of data structure problem: we have an n-bit "database" and want to be able to retrieve individual bits from it. In practice, databases have more structure and complexity, and one usually asks more complicated queries, such as retrieving all records within a certain range. Our more general notion of error-correcting data structures enables a study of more practical data structure problems in the presence of adversarial noise. If one subscribes to the approach towards errors taken in the area of error-correcting codes, then our definition of error-correcting data structures seems a very natural one. Yet, to our knowledge, this definition is new and has not been studied before (see section 1.4 for other approaches).

Our results on ECDSs. Let us start with a useful general result. One obvious way to construct error-correcting data structures is to take a good nonerror-correcting data structure $\mathcal{E}$ for the problem at hand, say with t probes and length N, and encode it with a good t'-probe LDC $C : \{0,1\}^N \to \{0,1\}^{N'}$; i.e., define $\mathcal{E}'(x) = C(\mathcal{E}(x))$. If the LDC has error probability at most ε/t, then we can simulate each of the t probes to $\mathcal{E}(x)$ by t' probes to the LDC-encoding of $\mathcal{E}(x)$, and by a union bound, the overall error probability will be at most $\varepsilon + t \cdot \varepsilon/t = 2\varepsilon$. The resulting number of probes of $\mathcal{E}'$ will be $t \cdot t'$, and the length N will go up to the length N' of the LDC. Thus we have proved the following.

PROPOSITION 1. *If data structure problem $f : X \times Q \to A$ has a (t, ε)-data structure of length N and there exists a $(t', \delta, \varepsilon/t)$-LDC that encodes N-bit strings into N'-bit strings, then f has a $(t \cdot t', \delta, 2\varepsilon)$-error-correcting data structure of length N'.*

In particular, suppose we use a $t' = (\log N)^{O(1/\eta)} \log(t/\varepsilon)$-probe LDC that encodes N bits into $N^{1+\eta}$ bits and has error probability $\leq \varepsilon/t$. This results in an error-correcting data structure $\mathcal{E}'$ for f of length $N' = N^{1+\eta}$, which is only slightly longer than the length of the nonerror-correcting starting point; the number of probes used is $t\text{polylog}(N)$. If the original data structure already used $t = \text{polylog}(N)$ probes, then this blow-up in the number of probes may well be a price worth paying for the error-correcting capacity.

Membership. General constructions such as Proposition 1 can yield very suboptimal results when the number of probes t of the original data structure is small. For example for the s-out-of-n MEMBERSHIP problem, the optimal nonerror-correcting data structure of Buhrman et al. [10] (explained below) uses only 1 probe and $O(s \log n)$ bits. Encoding this with the best possible 2-probe LDC gives a $(2, \delta, \varepsilon)$-error-correcting data structure of length $N' = 2^{O(s \log n)}$. Encoding it with the best known 3-probe LDCs gives length $N' \approx 2^{2^{\sqrt{\log s + \log\log n}}}$. Fortunately, something substantially better can be done. Fix some number of probes t, noise level δ, and allowed error probability ε, and consider the minimal length of t-probe error-correcting data structures

[4]The terminologies used in the data-structure and LDC literature conflict at various points, and we needed to reconcile them somehow. To avoid confusion, let us repeat here the choices we have made. We reserve the term "query" for the question q one asks about the encoded data x, while accesses to bits of the data structure are called "probes" (in contrast, these are usually called "queries" in the LDC literature). The number of probes is denoted by t. We use n for the number of bits of the data item x (in contrast with the literature about MEMBERSHIP, which mostly uses m for the size of the universe and n for the size of the set). We use N for the length of the data structure (while the LDC literature often uses m). We use the term "decoder" for the algorithm $\mathcal{D}$. Another issue is that ε is sometimes used as the error probability (in which case one wants $\varepsilon \approx 0$) and sometimes as the bias away from 1/2 (in which case one wants $\varepsilon \approx 1/2$). We use the former.

for s-out-of-n MEMBERSHIP. Let us call this minimal length $\mathrm{MEM}(t, s, n)$. A first observation is that such a data structure includes a locally decodable code for s bits: just restrict attention to n-bit strings whose last $n - s$ bits are all 0. Hence, with $\mathrm{LDC}(t, s)$ denoting the minimal length among all t-probe LDCs that encode s bits (for our fixed ε, δ), we immediately get the lower bound

$$\mathrm{LDC}(t, s) \leq \mathrm{MEM}(t, s, n).$$

This bound is close to optimal if $s \approx n$. Another trivial lower bound comes from the observation that our data structure for MEMBERSHIP is a map with domain of size $B(n, s) := \sum_{i=0}^{s} \binom{n}{i}$ and range of size 2^N that has to be injective. Hence we get another obvious lower bound,

$$\Omega(s \log(n/s)) \leq \log B(n, s) \leq \mathrm{MEM}(t, s, n).$$

What about *upper* bounds, beyond the above-mentioned consequences of Proposition 1? Our main positive result in section 2 (Theorem 7) says that the maximum of the above two lower bounds is not far from optimal. Slightly simplifying,[5] in Theorem 7 we prove

$$\mathrm{MEM}(t, s, n) \leq O(\mathrm{LDC}(t, 1000s) \log n). \tag{1}$$

In other words, if we have a decent t-probe LDC for encoding $O(s)$-bit strings, then we can use this to encode sets of size s from a much larger universe $[n]$, at the expense of blowing up the length of our data structure by only a factor of $\log n$.

For instance, for $t = 2$ probes we now get $\mathrm{MEM}(2, s, n) \leq 2^{O(s)} \log n$ from the Hadamard code, which is much better than the earlier $2^{O(s \log n)}$. For $t = 3$ probes, we get $\mathrm{MEM}(3, s, n) \leq 2^{2^{\sqrt{\log s}}} \log n$, which is much better than the earlier $2^{2^{\sqrt{\log s + \log\log n}}}$ if $\log s \ll \log\log n$.

Techniques for building ECDSs for MEMBERSHIP. Our construction for the upper bound of (1) relies heavily on the nonerror-correcting MEMBERSHIP construction of Buhrman et al. [10]. Their structure is obtained using the probabilistic method. (Explicit but slightly less efficient structures were subsequently given by Ta-Shma [35].) We sketch its main properties here.

The construction of Buhrman et al. [10] *(BMRV-structure).* The encoding can be represented as a bipartite graph $\mathcal{G} = (L, R, E)$ with $|L| = n$ left vertices, $|R| = m := \frac{100}{\varepsilon^2} s \log n$ right vertices, and regular left degree $d = \frac{\log n}{\varepsilon}$. This $\mathcal{G}$ is an *expander graph* in the following sense: for each set $S \subseteq L$ with $|S| \leq 2s$, its neighborhood $\Gamma(S) \subseteq R$ satisfies $|\Gamma(S)| \geq \left(1 - \frac{\varepsilon}{2}\right) |S| d$. For each assignment of bits to the left vertices with at most s ones, the encoding specifies an assignment (not explained here) of bits to the right vertices. In other words, each $x \in \{0,1\}^n$ of weight $|x| \leq s$ corresponds to an assignment to the left vertices, and the m-bit encoding of x corresponds to an assignment to the right vertices. For each $i \in [n]$ we write $\Gamma_i := \Gamma(\{i\})$ to denote the set of d neighbors of i. A crucial property of the encoding function $\mathcal{E}_{bmrv}$ is that for every x of weight $|x| \leq s$, for each $i \in [n]$, if $y = \mathcal{E}_{bmrv}(x) \in \{0,1\}^m$, then $\Pr_{j \in \Gamma_i}[x_i = y_j] \geq 1 - \varepsilon$. Hence the decoder for this data structure can just probe a random index $j \in \Gamma_i$ and return the resulting bit. For fixed ε, the length $m = O(s \log n)$ of the BMRV-structure is optimal up to a constant factor, because clearly $\log \binom{n}{s}$ is a lower bound.

[5] Our actual result, Theorem 7, is a bit weaker, with some deterioration in the error and noise parameters.

Note that this construction is not error-correcting at all, since $|\Gamma_i|$ errors in the data structure suffice to erase all information about the ith bit of the encoded x. Our main idea to make it error-correcting is to randomly partition the m indices of $\mathcal{E}_{bmrv}(x)$ into roughly $\log n$ blocks of $O(s)$ indices each. For a typical i, most blocks will intersect with Γ_i in one point. The error-correcting data structure now encodes each of the $\log n$ blocks of $O(s)$ separately, using an optimal t-probe LDC that encodes $O(s)$ bits. The resulting overall length is $N = \mathrm{LDC}(t, O(s)) \log n$. The decoder chooses one of the $\log n$ blocks at random, decodes the Γ_i-index in that block using the LDC-decoder for the encoding of that block, and outputs the resulting bit. Since a randomly chosen block is expected to have the same fraction of errors as the whole N-bit string, the fraction of noise in the chosen block is probably small enough for the LDC-decoder to work. The details of the analysis are given in section 2. Note that we did not need the expander properties of the encoding graph $\mathcal{G}$ for this; we will, however, need those for later results.

Inner product. Another problem that we consider here is Inner product. We show constructions for this problem where the error-correcting data structures are not much longer than their nonerror-correcting counterparts—though that is mostly because even the noiseless case already requires large length. Our main results are the following comparable upper and lower bounds.

Theorem 2. *Every (t, ε)-data structure for $\mathrm{IP}_{n,r}$ needs length*

$$N \geq \frac{1}{2} 2^{(\log(B(n,r)) - 2\log(1/(1-2\varepsilon)) - 1)/t}.$$

Theorem 3. *For every $p \geq 2$, there exists a $(t, \delta, t\delta)$-error-correcting data structure for $\mathrm{IP}_{n,r}$ of length $N \leq p \cdot 2^{r(t-1)^2 n^{1/(t-1)}}$.*

Some of the techniques used in the proofs here draw on communication complexity arguments as well as on previous literature on private information retrieval schemes. We defer all proofs and further discussions on this problem to Appendix A.

1.3. Relaxed error-correcting data structures. The fact that (1) means that progress (i.e., better upper and/or lower bounds) on LDCs for any constant number of probes is *equivalent* to progress on error-correcting data structures for s-out-of-n Membership. If we believe the conjecture that there are no polynomial-length LDCs with a constant number of probes, then the same negative conclusion also applies to error-correcting data structures for Membership, as well as for many other data structure problems to which Membership can be reduced (that is, in fact, most data structure problems). Accordingly, the length-versus-probes trade-offs for error-correcting data structures seem to be much worse than in the noiseless model. We thus ask the following question:

> What is a clean model of data structures that allows efficient representations *and* has error-correcting capabilities?

The model of Definition 2 imposes a rather stringent requirement on decoding: *every* query must be answered correctly with high probability from the possibly corrupted encoding. While this requirement is natural in the definition of LDCs due to their connection to complexity theory and cryptography, for data structures it seems somewhat restrictive. The alternative models discussed in section 1.4 do not require this either. Even in the noiseless model, some famous data structures such as Bloom filters [7] allow incorrect answers on a small fraction of the possible queries.

We will relax our notion of error-correcting data structures as follows. For *most* queries, the decoder still has to return the correct answer with high probability. How-

ever, for the few remaining queries, the decoder may claim ignorance, i.e., declare the data item unrecoverable from the (corrupted) data structure. Still, for *every* query, the probability of an incorrect answer should be small. In fact, just as Definition 2 is a generalization of LDCs, our relaxed model is a generalization of the "relaxed" locally decodable codes (RLDCs) introduced by Ben-Sasson et al. [6]. They relax the usual definition of an LDC by requiring the decoder to return the correct answer on *most* rather than all queries. For the remaining queries it is allowed to claim ignorance, i.e., to output a special symbol "$\perp$" interpreted as "don't know" or "unrecoverable." As shown in [6], relaxing the LDC definition like this allows for constructions of RLDCs with $O(1)$ bit-probes of *nearly linear* length.

Using RLDCs as building blocks, we construct error-correcting data structures that are very efficient in terms of time as well as space: their length and number of probes are not much bigger than in the noiseless setting. Before we describe our results, let us formally define our model. In addition to the earlier parameters t, δ, and ε, we now add a fourth parameter λ, which is an upper bound on the fraction of queries in Q that are not answered correctly with high probability (the "λ" stands for "lost").

DEFINITION 4 (relaxed error-correcting data structure (RECDS)). *Let $f : X \times Q \to A$ be a data structure problem. Let $t > 0$ be an integer, $\delta \in [0,1]$, $\varepsilon \in [0,1/2]$, and $\lambda \in [0,1]$. A $(t,\delta,\varepsilon,\lambda)$-RECDS for f of length N is a map $\mathcal{E} : X \to \{0,1\}^N$ for which there exists a randomized algorithm $\mathcal{D}$ with the following properties: for every $x \in X$ and every $w \in \{0,1\}^N$ at Hamming distance $\Delta(w, \mathcal{E}(x)) \le \delta N$,*

1. *$\mathcal{D}$ makes at most t bit-probes to w;*
2. *$\Pr[\mathcal{D}^w(q) \in \{f(x,q), \perp\}] \ge 1 - \varepsilon$ for every $q \in Q$;*
3. *the set $G := \{q : \Pr[\mathcal{D}^w(q) = f(x,q)] \ge 1 - \varepsilon\}$ has size at least $(1-\lambda)|Q|$ ("G" stands for "good");*
4. *if $w = \mathcal{E}(x)$, then $G = Q$.*

Just as an LDC is an error-correcting data structure for MEMBERSHIP (with $s = n$), a $(t,\delta,\varepsilon,\lambda)$-RLDC is a $(t,\delta,\varepsilon,\lambda)$-RECDS for MEMBERSHIP.

Our results on RECDSs. We focus on two common data structure problems for which we obtain almost optimal RECDS in terms of length and bit-probe complexity: MEMBERSHIP and POLYNOMIAL EVALUATION.

Membership. First, it is not hard to show that by composing the BMRV-structure with an RLDC, one already obtains an error-correcting data structure of length $O((s \log n)^{1+\eta})$, where η is an arbitrarily small constant. However, following the approach sketched at the end of section 1.2, with RLDCs instead of LDCs, we obtain a data structure of length $O(s^{1+\eta} \log n)$, which is substantially shorter if $s = o(\log n)$.

THEOREM 4. *For every $\varepsilon \in (0,1/2)$, $\eta \in (0,1)$, there exist an integer $t > 0$ and real $\tau > 0$ such that for all s and n, and every $\delta \le \tau$, s-out-of-n MEMBERSHIP has a $(t,\delta,\varepsilon,\frac{s}{2n})$-RECDS of length $O(s^{1+\eta} \log n)$.*

We will prove Theorem 4 in section 3.1. Note that the size of the good set G is at least $n - \frac{s}{2}$. Hence corrupting a δ-fraction of the bits of the data structure may cause a decoding failure for at most half of the positive queries (i where $x_i = 1$) but not for all or most positive queries. One may replace this factor $\frac{1}{2}$ easily by another constant (though the parameters t and τ will then change).

Polynomial evaluation. Since there are n^{s+1} polynomials of degree at most s, with each polynomial requiring a different instantiation of the data structure, the information-theoretic lower bound on the space of any data structure for this problem is at least $\log(n^{s+1}) \approx s \log n$ bits. Since each answer is an element of $\mathbb{Z}_n$ and must be

represented by $\lfloor \log n \rfloor + 1$ bits, $\lfloor \log n \rfloor + 1$ is the information-theoretic lower bound on the bit-probe complexity.

Consider the following two naive ways to construct a data structure. On one hand, one can simply record the evaluations of g in a table with n entries, each with $\lfloor \log n \rfloor + 1$ bits. The length of this data structure is $O(n \log n)$, and each query requires reading only $\lfloor \log n \rfloor + 1$ bits. On the other hand, g can be stored as a table of its $s+1$ coefficients. This gives a data structure of length and bit-probe complexity $(s+1)(\lfloor \log n \rfloor + 1)$.

A natural question is whether one can construct a data structure that is optimal in terms of both space and time, i.e., has length $O(s \log n)$ and answers queries with $O(\log n)$ bit-probes. No such constructions are known to exist. However, some lower bounds are known in the weaker cell-probe model, where each cell is a sequence of $\lfloor \log n \rfloor + 1$ bits. For instance, as noted in [31], any data structure for Polynomial Evaluation that stores $O(s^2)$ cells (which takes $O(s^2 \log n)$ bits) requires reading at least $\Omega(s)$ cells ($\Omega(s \log n)$ bits). Moreover, by [30], if $\log n \gg s \log s$ and the data structure is constrained to store $s^{O(1)}$ cells, then its query complexity is $\Omega(s)$ cells. This implies that the second trivial construction described above is essentially optimal in the cell-probe model.

Recently, Kedlaya and Umans [24] obtained a data structure that has length $s^{1+\eta} \log^{1+o(1)} n$ (where η is an arbitrarily small constant) that answers evaluation queries with $O(\text{polylog}(s) \cdot \log^{1+o(1)}(n))$ bit-probes. These parameters exhibit the best trade-off between s and n known so far. When $s = n^\eta$ for some $0 < \eta < 1$, the data structure of Kedlaya and Umans [24] is far superior to the trivial solution: its length is nearly optimal, and the query complexity drops from $\text{poly}(n)$ to only $\text{polylog}(n)$ bit-probes.

Here we construct an error-correcting data structure for the polynomial evaluation problem that works even in the presence of adversarial noise, with length nearly linear in $s \log n$ and bit-probe complexity $O(\text{polylog}(s) \cdot \log^{1+o(1)}(n))$. Formally, we have the following.

Theorem 5. *For every $\varepsilon \in (0, 1/2)$, $\lambda, \eta \in (0,1)$, there exists $\tau \in (0,1)$ such that for all positive integers $s \le n$, for all $\delta \le \tau$, the data structure problem* $\text{PolyEval}_{n,s}$ *has a $(\text{polylog}(s) \cdot \log^{1+o(1)}(n), \delta, \varepsilon, \lambda)$-RECDS of length $O((s \log n)^{1+\eta})$.*

Theorem 5 easily holds when $s = (\log n)^{o(1)}$. As we discussed previously, one can just store a table of the $s+1$ coefficients of g. To make this error-correcting, encode the entire table by a standard error-correcting code. This has length and bit-probe complexity $O(s \log n) = O(\log^{1+o(1)} n)$. Accordingly, the proof in section 3.2 will focus on the case where $s = (\log n)^{\Omega(1)}$.

Time-complexity of decoding and encoding. In our results we have used the number of bit-probes as a proxy for the actual time the decoder needs for query-answering. This is fairly standard and usually justified by the fact that the actual time-complexity of decoding is not much worse than its number of bit-probes. This is also the case for our constructions. For Membership, it can be shown that the decoder uses $O(1)$ probes and $\text{polylog}(n)$ time (as do the RLDCs of [6]). For Polynomial Evaluation, the decoder uses $\text{polylog}(s) \log^{1+o(1)}(n)$ probes and $\text{polylog}(sn)$ time.

The efficiency of *encoding*, i.e., the "preprocessing" of the data into the form of a data structure, is more problematic. The problems are, first, that the BMRV-structure is itself constructed using the probabilistic method, and, second, both of our relaxed error-correcting data structures (for Membership and Polynomial Evaluation) use the RLDC of [6]. Its encoding-efficiency is not addressed explicitly in [6] and needs further study.

Techniques for building RECDSs. At a high level, for both data structure problems we build our constructions by composing an RLDC with an appropriate noiseless data structure. If the underlying probe-accessing scheme in the noiseless data structure is "pseudorandom," then the noiseless data structure can be made error-correcting by appropriate compositions with other data structures. By pseudorandom, we mean that if a query is chosen uniformly at random from Q, then the positions of the probes selected also "behave" as if they are chosen uniformly at random. Such a property allows us to analyze the error-tolerance of our constructions.

More specifically, for MEMBERSHIP we use the same idea as for nonrelaxed error-correcting data structures, but using an RLDC instead of an LDC. In order to bound λ in our new construction, we now use the above-mentioned expander properties of the BMRV-structure. The left side of the expander represents the set of queries, and a neighborhood of a query (a left node) represents the set of possible bit-probes that can be chosen to answer this query. The expansion property of the graph essentially implies that for a random query, the distribution of a bit-probe chosen to answer this query is close to uniform.[6] This property allows us to construct an efficient relaxed error-correcting data structure for this problem.

For the polynomial evaluation problem, we rely on the noiseless data structure of Kedlaya and Umans [24], which has a decoding procedure that uses the reconstructive algorithm from the Chinese Remainder Theorem (CRT). The property that we need is the simple fact that if a is chosen uniformly at random from $\mathbb{Z}_n$, then for any $m \leq n$, a modulo m is uniformly distributed in $\mathbb{Z}_m$. This implies that for a random evaluation point a, the distribution of certain tuples of cell-probes used to answer this evaluation point is close to uniform. This observation allows us to construct an efficient, error-correcting data structure for polynomial evaluation. Our construction follows fairly closely the nonerror-correcting one of [24]; the main new ingredient is to add redundancy to their CRT-based reconstruction by using more primes, which gives us the error-correcting features we need.

1.4. Related work. Much work has been done on LDCs, a.k.a. error-correcting data structures for the MEMBERSHIP problem without constraints on the set size. However, the error-correcting version of s-out-of-n MEMBERSHIP ("storing sparse tables") or of other possible data structure problems has not been studied before.[7] Here we briefly describe a number of other approaches to data structures in the presence of memory errors. There is also work on data structures with faulty *processors*, but we will not discuss that here.

Fault-tolerant pointer-based data structures. Aumann and Bender [3] study fault-tolerant versions of *pointer-based* data structures. They define a pointer-based data structure as a directed graph where the edges are pointers, and the nodes come in two types: information nodes carry real data, while auxiliary nodes carry auxiliary or structural data. An *error* is the destruction of a node and its outgoing edges. They assume such an error is detected when accessing the node. Even a few errors may be very harmful to pointer-based data structures: for instance, losing one pointer in the middle of a standard linked list means we lose the second half of the list. They

[6] We remark that this is different from the notion of smooth decoding in the LDC literature, which requires that for every *fixed* query, each bit-probe by itself is chosen with probability close to uniform (though not independent of the other bit-probes).

[7] Using the connection between information-theoretical private information retrieval and locally decodable codes, one may derive some error-correcting data structures from the private information retrieval (PIR) results of [13]. However, the resulting structures seem fairly weak.

call a data structure (d, g)-*fault-tolerant* (where d is an integer that upper bounds the number of errors and g is a function) if $f \leq d$ errors cause at most $g(f)$ information nodes to be lost.

Aumann and Bender present fault-tolerant *stacks* with $g(f) = O(f)$, and fault-tolerant *linked lists* and *binary search trees* with $g(f) = O(f \log d)$, with only a constant-factor overhead in the size of the data structure, and small computational overhead. Notice, however, that their error-correcting demands are much weaker than the ones of Definition 2: we require that *no* part of the data be lost (every query should be answered with a high probability of success), even in the presence of a constant fraction of errors. Of course, we pay for that in terms of the length of the data structure.

Faulty-memory RAM model. An alternative model of error-correcting data structures is the "faulty-memory RAM model," introduced by Finocchi and Italiano [18]. In this model, one assumes there are $O(1)$ incorruptible memory cells available. This is justified by the fact that CPU registers are much more robust than other kinds of memory. On the other hand, all other memory cells can be faulty—including the ones used by the algorithm that is answering queries (something our model does not consider). The model assumes an upper bound Δ on the number of errors.

Finocchi, Grandoni, and Italiano [17] described essentially optimal resilient algorithms for *sorting* that work in $O(n \log n + \Delta^2)$ time with Δ up to about $\sqrt{n}$, and for *searching* in $\Theta(\log n + \Delta)$ time. There is a lot of fairly recent work in this model: Jørgenson, Moruz, and Mølhave [22] study resilient *priority queues*, Finocchi, Grandoni, and Italiano [17] study resilient *search trees*, Caminiti, Finocchi, and Fusco [12] study *dynamic programming*, Brodal et al. [8] study resilient *dictionaries*, and Brodal et al. [9] study *counting*. This interesting model allows for more efficient data structures than our model, but its disadvantages are also clear: it assumes a small number of incorruptible cells, which may not be available in many practical situations (for instance, when the whole data structure is stored on a hard disk), and the constructions mentioned above cannot deal well with a constant noise rate. Furthermore, correctness can only be guaranteed for keys that are not affected by noise.

1.5. Organization. In section 2 we construct our ECDS for the MEMBERSHIP problem (Theorem 7). Our further results and discussions regarding the INNER PRODUCT problem are deferred to Appendix A. In section 3.1 we construct our efficient RECDSs for MEMBERSHIP (Theorem 4) and for POLYNOMIAL EVALUATION (Theorem 5). In section 4 we describe a few interesting open problems that emerge from this work.

2. Error-correcting data structures.

2.1. MEMBERSHIP**: 1 probe.** We start by commenting on the known data structures for the MEMBERSHIP problem with 1 probe. For the noiseless case, the BMRV-structure sketched at the end of section 1.2 has information-theoretically optimal length $m = O(s \log n)$ and decodes with the minimal number of probes (one). This can also be achieved in the error-correcting case if $s = 1$; then we just have the EQUALITY problem (for this see the remark following Definition 2). For larger s, one can observe that the BMRV-structure still works with high probability if $\delta \ll 1/s$: in that case the total number of errors is $\delta m \ll \log n$, so for each i, most bits in the $\Theta(\log n)$-set Γ_i are uncorrupted.

THEOREM 6 (see [10]). *There exist* $(1, \Omega(1/s), 1/4)$*-error-correcting data structures for* MEMBERSHIP *of length* $N = O(s \log n)$.

This works only if $\delta \ll 1/s$, which is actually close to optimal, as follows. An s-bit LDC can be embedded in an error-correcting data structure for MEMBERSHIP, and hence it follows from Katz and Trevisan [23, Theorem 3] that there are no 1-probe error-correcting data structures for MEMBERSHIP if $s > 1/(\delta(1 - H(\varepsilon)))$ (where $H(\cdot)$ denotes binary entropy). In sum, there are 1-probe error-correcting data structures for MEMBERSHIP of information-theoretically optimal length if $\delta \ll 1/s$. In contrast, if $\delta \gg 1/s$, then there are no 1-probe error-correcting data structures at all, not even of exponential length.

2.2. MEMBERSHIP**: $t > 1$ probes.** As we argued in section 1.2, for fixed ε and δ there is an easy lower bound on the length N of t-probe error-correcting data structures for s-out-of-n MEMBERSHIP:

$$N \geq \max\left(\mathrm{LDC}(t, s), \log \sum_{i=0}^{s} \binom{n}{i}\right).$$

Our nearly matching upper bound $N = O(\mathrm{LDC}(t, 1000s) \log n)$, described below, uses the BMRV-structure for a small fixed error, combined with an LDC.

THEOREM 7. *If there is a $(t, 100\delta, 1/100)$-LDC of length ℓ that encodes $1000s$ bits, then there is a $(t, \delta, 49/100)$-error-correcting data structure of length $O(\ell \log n)$ for the s-out-of-n* MEMBERSHIP *problem.*

In this section we prove Theorem 7. We start by showing a useful property of the BMRV-structure.

PROPOSITION 8. *For all positive integers s, n with $s \leq n$, the BMRV bipartite graph $\mathcal{G} = ([20n], [10000 \cdot s \log(20n)], E)$ for s-out-of-$20n$* MEMBERSHIP *with error parameter $\frac{1}{10}$ has the following property: there exists a partition of $[10000s \log(20n)]$ into $b = 10 \log(20n)$ disjoint sets $B_1, \ldots, B_b$ of $1000s$ vertices each, such that for each $i \in [n]$, there are at least $\frac{b}{4}$ sets B_k satisfying $|\Gamma_i \cap B_k| = 1$.*

Proof. We use the probabilistic method. Apply a random permutation π to the indices of $y = \mathcal{E}_{bmrv}(x)$ (we show below that π can be fixed to a specific permutation). The intuition is as follows. Thanks to the random permutation and the fact that $|\Gamma_i|$ equals the number of blocks b, the expected intersection between Γ_i and a block is exactly 1. Hence for many $i \in [20n]$, many blocks will contain exactly one index $j \in \Gamma_i$.

To make this precise, call $k \in [b]$ "good for i" if block k contains *exactly one* $j \in \Gamma_i$, and let X_{ik} be the indicator random variable for this event. Call $i \in [20n]$ "good" if at least $b/4$ of the blocks are good for i (i.e., $\sum_{k \in [b]} X_{ik} \geq b/4$), and let X_i be the indicator random variable for this event. The expected value (over uniformly random π) of each X_{ik} is the probability that if we randomly place b balls into ab positions (a is the block-size $1000s$), then there is exactly one ball among the a positions of the first block, and the other $b - 1$ balls are in the last $ab - a$ positions. This is

$$\frac{a\binom{ab-a}{b-1}}{\binom{ab}{b}} = \frac{ab-a}{ab-1} \cdot \frac{ab-a-1}{ab-2} \cdot \ldots \cdot \frac{ab-a-b+2}{ab-b+1} = \prod_{i=0}^{b-2} \left(1 - \frac{a-1}{ab-1-i}\right)$$
$$\geq \left(1 - \frac{a-1}{ab-b+1}\right)^{b-1} \geq \left(1 - \frac{1}{b-1}\right)^{b-1} \geq \frac{3}{10}$$

for $b \geq 5$.

Then, using linearity of expectation we obtain

$$20n \cdot b \cdot \frac{3}{10} \leq \text{Exp}\left[\sum_{i\in[20n],k\in[b]} X_{ik}\right] \leq b \cdot \text{Exp}\left[\sum_{i\in[20n]} X_i\right] + \frac{b}{4}\left(20n - \text{Exp}\left[\sum_{i\in[20n]} X_i\right]\right),$$

which implies

$$\text{Exp}\left[\sum_{i\in[20n]} X_i\right] \geq n.$$

Hence we can fix one permutation π such that at least n of the indices i are good. By renaming we can assume that the first n indices are good. □

Proof of Theorem 7. To construct an error-correcting data structure, we proceed as follows. Let $\delta \ll 1/100$ be a small constant; this is the noise level that we want our final data structure for MEMBERSHIP to protect against. Consider the BMRV-structure for s-out-of-$20n$ MEMBERSHIP, with error probability at most $1/10$. Then $m = 10000s\log(20n)$ is its length, and $b = 10\log(20n)$ is the size of each of the sets Γ_i. Using Proposition 8, we will treat this as consisting of roughly $\log n$ blocks of $O(s)$ bits each.

Encoding. Let $m = 10000s\log(20n)$ and $B_1, \ldots, B_b$ be a partition of $[m]$ as guaranteed by Proposition 8. For a string $y \in \{0,1\}^m$, we abuse notation and write $y = y^{(1)} \cdots y^{(b)}$, where $y^{(k)}$ denotes the $1000s$-bit string corresponding to the indices in B_k. Let $\mathcal{E}_{bmrv}$ be the encoder of the BMRV-structure. Let $C : \{0,1\}^{1000s} \to \{0,1\}^\ell$ be a $(t, 100\delta, 1/100)$-LDC, and let $\mathcal{D}_C$ be its decoder. On input $x \in \{0,1\}^n$, $|x| \leq s$, the encoder $\mathcal{E}$ does the following:

1. Let $\mathcal{E}_{bmrv}\left(x0^{19n}\right) = y = y^{(1)} \cdots y^{(b)}$.
2. Output the concatenation $\mathcal{E}(x) = C(y^{(1)}) \ldots C(y^{(b)})$.

The length of $\mathcal{E}(x)$ is $N = b \cdot \ell = O(\ell \log n)$.

Decoding. Given a string $w \in \{0,1\}^N$, we write $w = w^{(1)} \ldots w^{(b)}$, where the ℓ-bit strings $w^{(k)}$ correspond to the (possibly corrupted) LDC-encoding $C(y^{(k)})$ of $y^{(k)}$. Given input i and oracle access to w, the decoder $\mathcal{D}$ does the following:

1. Pick a random $k \in [b]$.
2. If $|\Gamma_i \cap B_k| \neq 1$, then output a random bit.
   Else, let $\Gamma_i \cap B_k = \{j\}$ and output $\mathcal{D}_C^{w^{(k)}}(j)$.

For every index i, at least 90% of all $j \in \Gamma_i$ satisfy $y_j = x_i$. Hence for a good index i, with probability at least $1/4 - 1/10$ we will pick a k such that the kth block is good for i *and* the unique $j \in \Gamma_i$ in the kth block satisfies $y_j = x_i$. By Markov's inequality, the probability that the block that we picked has more than a 100δ-fraction of errors is less than $1/100$. If the fraction of errors is at most 100δ, then our LDC-decoder recovers the relevant bit y_j with probability $99/100$. Hence the overall probability of outputting the correct value x_i is at least

$$\frac{3}{4} \cdot \frac{1}{2} + \left(\frac{1}{4} - \frac{1}{10} - \frac{1}{100}\right) \cdot \frac{99}{100} > \frac{51}{100}.$$ □

3. Relaxed error-correcting data structures.

3.1. RECDs for MEMBERSHIP**.** In this section we construct a *relaxed* error-correcting data structure for MEMBERSHIP, overcoming the main drawback of the above construction: the fact that we don't know short LDCs with few probes. To

prove Theorem 4 we use the *relaxed* LDC of Ben-Sasson et al. [6], which has nearly linear length.

THEOREM 9 (see [6]). *For every $\varepsilon \in (0, 1/2)$ and $\eta > 0$, there exist an integer $t > 0$ and reals $c > 0$ and $\tau > 0$ such that for every n and every $\delta \le \tau$, there is a $(t, \delta, \varepsilon, c\delta)$-RLDC encoding n bits of length $O(n^{1+\eta})$.*

Note that by picking the error-rate δ to be a sufficiently small constant, one can set $\lambda = c\delta$ (the fraction of unrecoverable queries) to be very close to 0.

Proof of Theorem 4. By Theorem 9, for every $\varepsilon \in (0, 1/2)$, $\eta > 0$, for some $t = O(1)$, and $\tau_0 > 0$, for every $\delta < \tau_0$, there is a $(t, 10^5\delta, \frac{1}{100}, c'\delta)$-RLDC encoding $1000s$ bits with length $\ell = O(s^{1+\eta})$, where $c' = 10^5 c$. Let $\mathcal{E}_{bghsv}$ and $\mathcal{D}_{bghsv}$ be its encoder and decoder, respectively. The encoding and decoding procedure will be the same as in the proof of Theorem 7, except that $\mathcal{E}_{bghsv}$ and $\mathcal{D}_{bghsv}$ replace the LDC-encoder C and decoder $\mathcal{D}_C$, respectively. The resulting length is $O(\ell \log n)$. We will use the same notation as defined in the proof of Theorem 7.

Analysis of correctness. Fix $x \in X$ and $w \in \{0,1\}^N$ such that $\Delta(w, \mathcal{E}(x)) \le \delta N$, where δ is less than some small constant τ to be specified later. Recall that $y = \mathcal{E}_{bmrv}(x0^{19n}) = y^{(1)} \cdots y^{(b)}$. We now verify the four conditions of Definition 4. For condition 1, the number of probes the decoder $\mathcal{D}$ makes is the number of probes the decoder $\mathcal{D}_{bghsv}$ makes, which is at most t, a fixed integer.

We now examine condition 2. Fix $i \in [n]$. By Markov's inequality, for a random $k \in [b]$, the probability that the relative Hamming distance between $\mathcal{E}(y_{B_k})$ and $w^{(k)}$ is greater than $10^5\delta$ is at most 10^{-5}. If k is chosen such that the fraction of errors in $w^{(k)}$ is at most $10^5\delta$ and $\Gamma_i \cap B_k = \{j\}$, then with probability at least 0.99, $\mathcal{D}_{bghsv}$ outputs y_j or $\perp$. Let $\beta \ge \frac{1}{4}$ be the fraction of $k \in [b]$ such that $|\Gamma_i \cap B_k| = 1$. Then

$$\Pr[\mathcal{D}(i) \in \{x_i, \perp\}] \ge (1-\beta)\frac{1}{2} + \beta\frac{99}{100} - \frac{1}{10^5} > 0.624. \tag{2}$$

To prove condition 3, we need the expansion property of the BMRV-structure. For $k \in [b]$, define $G_k \subseteq B_k$ so that $j \in G_k$ if $\Pr[\mathcal{D}_{bghsv}^{w^{(k)}}(j) = y_j] \ge 0.99$. In other words, G_k consists of indices in block B_k that are answered correctly by $\mathcal{D}_{bghsv}$ with high probability. By Theorem 9, if the fraction of errors in $w^{(k)}$ is at most $10^5\delta$, then $|G_k| \ge (1 - c'\delta)|B_k|$ for fixed constant c'. Set $A = \cup_{k\in[b]} B_k \backslash G_k$. Since we showed above that for a $(1 - 10^{-5})$-fraction of $k \in [b]$, the fraction of errors in $w^{(k)}$ is at most $10^5\delta$, we have $|A| \le c'\delta m + 10^{-5}m$.

Recall that the BMRV expander has left degree $d = 10\log(20n)$. Take δ small enough that $|A| < \frac{1}{40}sd$. This determines the value of τ of the theorem: say, $\tau = \min\{\tau_0, \frac{sd - 10^{-5}m}{45c'm}\}$. Note that this is a positive constant since $m = 10000s\log(20n) = 1000sd$. We need to show that for any such small set A, most queries $i \in [n]$ are answered correctly with probability at least 0.51. It suffices to show that for most i, most of the set Γ_i falls outside of A. To this end, let $B(A) = \{i \in [n] : |\Gamma_i \cap A| \ge \frac{d}{10}\}$. We show that if A is small, then $B(A)$ is small.

CLAIM 10. *For every $A \subseteq [m]$ with $|A| < \frac{sd}{40}$, it is the case that $|B(A)| < \frac{s}{2}$.*

Proof. Suppose, by way of contradiction, that $B(A)$ contains a set W of size $s/2$. This W is a set of left vertices in the underlying expander graph $\mathcal{G}$, and since $|W| < 2s$, we must have

$$|\Gamma(W)| \ge \left(1 - \frac{1}{20}\right) d|W|.$$

By construction, each vertex in W has at most $\frac{9}{10}d$ neighbors outside A. Thus, we can bound the size of $\Gamma(W)$ from above as follows:

$$\begin{aligned}|\Gamma(W)| &\leq |A| + \frac{9}{10}d|W| \\ &< \frac{1}{40}ds + \frac{9}{10}d|W| \\ &= \frac{1}{20}d|W| + \frac{9}{10}d|W| \\ &= \left(1 - \frac{1}{20}\right)d|W|.\end{aligned}$$

This is a contradiction. Hence no such W exists, and $|B(A)| < \frac{s}{2}$. □

Define $G = [n]\backslash B(A)$ and notice that $|G| > n - \frac{s}{2}$. It remains to show that each query $i \in G$ is answered correctly with probability > 0.51. To this end, we have

$$\begin{aligned}\Pr[\mathcal{D}(i) = \perp] &\leq \Pr[\mathcal{D} \text{ probes a block with noise-rate} > 10^5\delta] \\ &\quad + \Pr[\mathcal{D} \text{ probes a } j \in A] + \Pr[\mathcal{D}(i) = \perp \mid \mathcal{D} \text{ probes a } j \notin A] \\ &\leq \frac{1}{10^5} + \frac{1}{10} + \frac{1}{100} < 0.111.\end{aligned}$$

Combining with (2), for all $i \in G$ we have

$$\Pr[\mathcal{D}(i) = x_i] = \Pr[\mathcal{D}(i) \in \{x_i, \perp\}] - \Pr[\mathcal{D}(i) = \perp] \geq 0.51.$$

Finally, condition 4 follows from the corresponding condition of the RLDC. □

3.2. RECDs for polynomial evaluation. In this section we prove Theorem 5. Given a polynomial g of degree s over $\mathbb{Z}_n$, our goal is to write down a data structure of length roughly linear in $s \log n$ so that for each $a \in \mathbb{Z}_n$, the value $g(a)$ can be computed with approximately $\text{polylog}(s) \cdot \log(n)$ bit-probes. Our data structure is built on the work of Kedlaya and Umans [24]. Since we cannot quite use their construction as a black-box, we first give a high-level overview of our proof, motivating each of the proof ingredients that we need.

Encoding based on reduced polynomials. The most naive construction, by recording $g(a)$ for each $a \in \mathbb{Z}_n$, has length $n \log n$ and answers an evaluation query with $\log n$ bit-probes. As explained in [24], one can reduce the length by using the CRT: if P_1 is a collection of distinct primes, then a nonnegative integer $m < \prod_{p \in P_1} p$ is uniquely specified by (and can be reconstructed efficiently from) the values $[m]_p$ for each $p \in P_1$, where $[m]_p$ denotes $m \bmod p$.

Consider the value $g(a)$ over $\mathbb{Z}$, which can be bounded above by n^{s+2}, for $a \in \mathbb{Z}_n$. Let P_1 consist of the first $\log(n^{s+2})$ primes. For each $p \in P_1$, compute the *reduced polynomial* $g_p \in \mathbb{Z}_p[X]$, defined as $g_p := g \mod p$ (i.e., g_p is obtained from g by reducing the coefficients modulo p,) and write down $g_p(b)$ for each $b \in \mathbb{Z}_p$. Consider the data structure that simply concatenates the evaluation table of every reduced polynomial. This data structure has length $|P_1|(\max_{p \in P_1} p)^{1+o(1)}$, which is $s^{2+o(1)} \log^{2+o(1)} n$ by the Prime Number Theorem. Note that $g(a) < \prod_{p \in P_1} p$. So to compute $[g(a)]_n$, it suffices to apply CRT to reconstruct $g(a)$ over $\mathbb{Z}$ from the values $[g(a)]_p = g_p([a]_p)$ for each $p \in P_1$. The number of bit-probes is $|P_1| \log(\max_{p \in P_1} p)$, which is $s^{1+o(1)} \log^{1+o(1)} n$.

Error-correction with reduced polynomials. The above CRT-based construction has terrible parameters, but it serves as an important building block from which we can obtain a data structure with better parameters. For now, we explain how the above CRT-based encoding can be made error-correcting. One can protect the bits of the evaluation tables of each reduced polynomial by an RLDC as provided by Theorem 9. However, the evaluation tables can have nonbinary alphabets, and a bit-flip in just one "entry" of an evaluation table can destroy the decoding process. To remedy this, one can first encode each entry by a standard error-correcting code and then encode the concatenation of all the tables by an RLDC. This is encapsulated in Lemma 11, which can be viewed as a version of Theorem 9 over a nonbinary alphabet. We prove Lemma 11 in Appendix B.

LEMMA 11. *Let $f : X \times Q \to \{0,1\}^\ell$ be a data structure problem. For every $\varepsilon \in (0,1/2)$, $\eta, \lambda \in (0,1)$, there exists $\tau \in (0,1)$ such that for every $\delta \le \tau$, f has an $(O(\ell), \delta, \varepsilon, \lambda)$-relaxed error-correcting data structure of length $O((\ell|Q|)^{1+\eta})$.*

We use the following instantiation in Lemma 11: let X be the set of degree-s polynomials over $\mathbb{Z}_n$, let Q be the set of all evaluation points of all the reduced polynomials of g (each specified by a pair (a,p)), and the data structure problem f outputs evaluations of some reduced polynomial of g (i.e., $f(g,(a,p)) = [g(a)]_p$).

By itself, Lemma 11 cannot guarantee resilience against noise. In order to apply the CRT to reconstruct $g(a)$, all the values $\{[g(a)]_p : p \in P_1\}$ must be correct, which is not guaranteed by Lemma 11. To fix this, we add redundancy, taking a larger set of primes than necessary so that the reconstruction via CRT can be made error-correcting. Specifically, we apply a CRT code to the encoding process.

DEFINITION 5 (CRT code). *Let $p_1 < p_2 < \cdots < p_N$ be distinct primes, $K < N$, and $T = \prod_{i=1}^K p_i$. The* CRT code *with basis $p_1, \ldots, p_N$ and rate $\frac{K}{N}$ over message space $\mathbb{Z}_T$ encodes $m \in \mathbb{Z}_T$ as $\langle [m]_{p_1}, [m]_{p_2}, \ldots, [m]_{p_N} \rangle$.*

By the CRT, for distinct $m_1, m_2 \in \mathbb{Z}_T$, their encodings agree on at most $K-1$ coordinates. Hence the CRT code with basis $p_1 < \cdots < p_N$ and rate $\frac{K}{N}$ has distance $N - K + 1$.

It is known that good families of CRT codes exist and that unique decoding algorithms for CRT codes can correct up to almost half of the distance of the code (see, e.g., [20]). The following statement can be easily derived from known facts, and we include a proof in Appendix C.

THEOREM 12. *For every positive integer T, there exists a set P consisting of distinct primes, with (1) $|P| = O(\log T)$, and (2) for all $p \in P$, $\log T < p < 500 \log T$ such that a CRT code with basis P and message space $\mathbb{Z}_T$ has rate $\frac{1}{2}$ and relative distance $\frac{1}{2}$, and can correct up to a $(\frac{1}{4} - O(\frac{1}{\log\log T}))$-fraction of errors.*

We apply Theorem 12 to a message space of size n^{s+2} to obtain a set of primes P_1 with the properties described above. Note that these primes are all within a constant factor of one another, and in particular, the evaluation table of each reduced polynomial has the same length, up to a constant factor. This fact and Lemma 11 will ensure that our CRT-based encoding is error-correcting.

Reducing the bit-probe complexity. We now explain how to reduce the bit-probe complexity of the CRT-based encoding, using an idea from [24]. Write $s = d^m$, where $d = (\log s)^C$, $m = \frac{\log s}{C \log\log s}$, and $C > 1$ is a sufficiently large constant. Consider the following multivariate extension map $\psi_{d,m} : \mathbb{Z}_n[X] \to \mathbb{Z}_n[X_0, \ldots, X_{m-1}]$ that sends a univariate polynomial of degree at most s to an m-variate polynomial of degree less than d in each variable. For every $i \in [s]$, write $i = \sum_{j=0}^{m-1} i_j d^j$ in base d.

Define $\psi_{d,m}$ which sends the monomial X^i to $X_1^{i_0} \cdots X_m^{i_{m-1}}$ and extends linearly to polynomials in $\mathbb{Z}_n[X]$.

To simplify our notation we write $\tilde{g}$ to denote the multivariate polynomial $\psi_{d,m}(g)$. For every $a \in \mathbb{Z}_n$, define $\tilde{a} \in \mathbb{Z}_n^m$ to be $([a]_n, [a^d]_n, [a^{d^2}]_n, \ldots, [a^{d^{m-1}}]_n)$. Note that for every $a \in \mathbb{Z}_n$, $g(a) = \tilde{g}(\tilde{a}) \pmod{n}$. Now the trick is to observe that the total degree of the multivariate polynomial $\tilde{g}$ is less than the degree of the univariate polynomial g, and hence its maximal value over the integers is much reduced. In particular, for every $a \in \mathbb{Z}_n^m$, the value $\tilde{g}(a)$ over the integers is bounded above by $d^m n^{dm+1}$.

We now work with the reduced polynomials of $\tilde{g}$ for our encoding. Let P_1 be the collection of primes guaranteed by Theorem 12 when $T_1 = d^m n^{dm+1}$. Then both $|P_1|$ and $\max_{p \in P_1} p$ are $O(\log T_1) = O(dm \log n)$. For $a, p \in \mathbb{Z}$, define $[a]_{n,p} = [a]_n \mod p$. For $p \in P_1$, let $\tilde{g}_p$ denote $\tilde{g} \mod p$ and $\tilde{a}_p$ denote point $([a]_{n,p}, [a^d]_{n,p}, \ldots, [a^{d^{m-1}}]_{n,p})$. Consider the data structure that concatenates the evaluation table of $\tilde{g}_p$ for each $p \in P_1$. For each $a \in \mathbb{Z}_n$, to compute $g(a)$, it suffices to compute $\tilde{g}(\tilde{a})$ over $\mathbb{Z}$, which by Theorem 12 can be reconstructed (even with noise) from the set $\{\tilde{g}_p(\tilde{a}_p) : p \in P_1\}$.

Since the maximum value of $\tilde{g}$ is at most $T_1 = d^m n^{dm+1}$ (whereas the maximum value of g is at most $d^m n^{d^m+1}$), the number of primes we now use is significantly less. This effectively reduces the bit-probe complexity. In particular, each evaluation query can be answered with $|P_1| \cdot \max_{p \in P_1} \log p = (dm \log n)^{1+o(1)}$ bit-probes, which by our choice of d and m is equal to $\text{polylog}(s) \cdot \log^{1+o(1)}(n)$. However, the *length* of this encoding, which is at most

$$|P_1| \max_{p \in P_1} p^m \log p = O(dm \log n)^{m+1} \log p = O\left(s^{1+1/C} (\log n)^{\frac{\log s}{\log \log s}}\right),$$

is still far from the information-theoretically optimal $s \log n$ bits. We shall explain how to reduce the length, but since encoding with multivariate reduced polynomials introduces potential complications in error-correction, we first explain how to circumvent these complications.

Error-correction with reduced multivariate polynomials. There are two complications that arise from encoding with reduced multivariate polynomials. The first is that not all the points in the evaluation tables are used in the reconstructive CRT algorithm. Lemma 11 only guarantees that *most* of the entries of the table can be decoded, not all of them. So if the entries that are used in the reconstruction via CRT are not decoded by Lemma 11, then the whole decoding procedure fails.

More specifically, to reconstruct $\tilde{g}(\tilde{a})$ over $\mathbb{Z}_n$, it suffices to query the point $\tilde{a}_p$ in the evaluation table of $\tilde{g}_p$ for each $p \in P_1$. Typically the set $\{\tilde{a}_p : a \in \mathbb{Z}_n\}$ will be much smaller than $\mathbb{Z}_p^m$, so not all the points in $\mathbb{Z}_p^m$ are used. To circumvent this issue, we store only the query points that are used in the CRT reconstruction. Let $B^p = \{\tilde{a}_p : a \in \mathbb{Z}_n\}$. For each $p \in P_1$, the encoding stores only the evaluation of $\tilde{g}_p$ at the points B^p instead of the entire domain $\mathbb{Z}_p^m$. The disadvantage of computing the evaluation at the points in B^p is that the encoding stage takes time proportional to n. We thus give up on encoding efficiency (which was one of the main goals of Kedlaya and Umans) in order to guarantee error-correction.

The second complication is that the sizes of the evaluation tables may no longer be within a constant factor of each other. (This is true even if the evaluation points come from all of $\mathbb{Z}_p^m$.) If one of the tables has length significantly longer than the others, then a constant fraction of noise may completely corrupt the entries of all the other small tables, rendering decoding via CRT impossible. This potential problem is

easy to fix; we apply a repetition code to each evaluation table so that all the tables have essentially equal length.

Reducing the length. Now we explain how to reduce the length of the data structure to nearly $s \log n$, along the lines of Kedlaya and Umans [24]. To reduce the length, we need to reduce the magnitude of the primes used by the CRT reconstruction. We can effectively achieve that by applying the CRT twice. Instead of storing the evaluation table of $\tilde{g}_p$, we apply CRT again and store evaluation tables of the reduced polynomials of $\tilde{g}_p$ instead. Whenever an entry of $\tilde{g}_p$ is needed, we can apply the CRT reconstruction to the reduced polynomials of $\tilde{g}_p$.

Fix m and d as before. Note that for $p_1 \in P_1$, the maximum value of $\tilde{g}_{p_1}$ (over the integers rather than mod n) is at most $T_2 = d^m p_1^{dm+1}$. Now apply Theorem 12 with T_2 the size of the message space to obtain a collection of primes P_2. Recall that each $p_1 \in P_1$ is at most $O(dm \log n)$. So each $p_2 \in P_2$ is at most $O((dm)^{1+o(1)} \log\log n)$, which also bounds the cardinality of P_2 from above.

For each query, the number of bit-probes made is at most $|P_1||P_2| \max_{p_2 \in P_2} \log p_2$, which is at most $(dm)^{2+o(1)} \log^{1+o(1)} n$. Recall that by our choice of d and m, $dm = \frac{(\log s)^{C+1}}{C \log\log s}$. Thus, the bit-probe complexity is $\mathrm{polylog}(s) \cdot \log^{1+o(1)}(n)$. Now, by Lemma 11, the length of the encoding is nearly linear in $|P_1||P_2| \max_{p_2 \in P_2} p_2^m \log p_2$, which is at most $\mathrm{polylog}(s) \cdot \log^{1+o(1)}(n) \cdot \max_{p_2 \in P_2} p_2^m$. So it suffices to bound $\max_{p_2 \in P_2} p_2^m$ from above. To this end, recall that by the remark following Theorem 5, we may assume without loss of generality that $s = \Omega((\log n)^\zeta)$ for some $0 < \zeta < 1$. This implies that $\log\log\log n \le \log\log s - \log\zeta$. Then for each $p_2 \in P_2$,

$$\begin{aligned} p_2^m &\le \left(O\left((dm)^{1+o(1)} \log\log n\right)\right)^m \\ &\le (dm)^{(1+o(1))m} \cdot s^{\frac{1}{C}+o(1)}. \end{aligned}$$

It is easy to see that $(dm)^{(1+o(1))m}$ can be bounded above by $s^{(1+o(1))(1+\frac{1}{C}-o(1))}$. Thus, $p_2^m = s^{1+\frac{2}{C}+o(1)}$. Putting everything together and choosing C sufficiently large, the length of the encoding can be made $O(s^{1+\eta} \log n)$ bits for any constant $\eta > 0$ of our choice.

Below we give the rather technical formal proof that all this works.

Proof of Theorem 5. We construct only an error-correcting data structure with error probability $\varepsilon = \frac{1}{4}$. By a standard amplification technique (i.e., $O(\log(1/\varepsilon'))$ repetitions) we can reduce the error probability to any other positive constant ε'. We now give a formal description of the encoding and decoding algorithms.

Encoding. Apply Theorem 12 with $T = d^m n^{dm+1}$ to obtain a collection of primes P_1. Apply Theorem 12 with $T = d^m (\max_{p \in P_1} p)^{dm+1}$ to obtain a collection of primes P_2. Set $p_{max} = \max_{p_2 \in P_2} p_2$.

Now, for each $p_1 \in P_1$, $p_2 \in P_2$, define a collection of evaluation points $B^{p_1,p_2} = \{\tilde{a}_{p_1,p_2} : a \in \mathbb{Z}_n\}$. Fix a univariate polynomial $g \in \mathbb{Z}_n[x]$ of degree at most s. For every $p_1 \in P_1$, $p_2 \in P_2$, view each evaluation of the reduced multivariate polynomial $\tilde{g}_{p_1,p_2}$ as a bit-string of length exactly $\lceil \log p_{max} \rceil$. Let $L = \max_{p_1 \in P_1, p_2 \in P_2} |B^{p_1,p_2}|$, and for each $p_1 \in P_1$, $p_2 \in P_2$, set $r^{p_1,p_2} = \lceil \frac{L}{|B^{p_1,p_2}|} \rceil$. Define f^{p_1,p_2} to be the concatenation of r^{p_1,p_2} copies of the string $\langle \tilde{g}(q) \rangle_{q \in B^{p_1,p_2}}$. Define the string $f = \langle f^{p_1,p_2} \rangle_{p_1 \in P_1, p_2 \in P_2}$.

We want to apply Lemma 11 to protect the string f, which we can do since f may be viewed as a data structure problem, as follows. The set of data items is the set of polynomials g as above. The set of queries Q is the disjoint union of the sets $B^{p_1,p_2} \times [r^{p_1,p_2}]$ over all $p_1 \in P_1, p_2 \in P_2$. The answer to query $(q^{p_1,p_2}, i^{p_1,p_2})$ is the i^{p_1,p_2}th copy of $\tilde{g}_{p_1,p_2}(q^{p_1,p_2})$.

Fix $\lambda \in (0,1)$. By Lemma 11, for every $\eta > 0$, there exists $\tau_0 \in (0,1)$ such that for every $\delta \leq \tau_0$, there is a $(O(\log p_{max}), \delta, 2^{-10}, \lambda^3 2^{-36})$-RECDS for the data structure problem corresponding to f. Let $\mathcal{E}_0, \mathcal{D}_0$ be its encoder and decoder, respectively. Finally, the encoding of the polynomial g is simply

$$\mathcal{E}(g) = \mathcal{E}_0(f).$$

Note that the length of $\mathcal{E}(g)$ is at most $(|P_1||P_2| \max_{p_2 \in P_2} p_2^m \log p_2)^{1+\eta}$, which as we computed earlier is bounded above by $O((s \log n)^{1+\zeta})$ for some arbitrarily small constant ζ.

Decoding. We may assume, without loss of generality, that the CRT-decoder $\mathcal{D}_{crt}$ from Theorem 11 outputs $\perp$ when more than a $\frac{1}{16}$-fraction of its input are erasures (i.e., $\perp$ symbols).

The decoder $\mathcal{D}$, with input $a \in \mathbb{Z}_n$ and oracle access to w, does the following:

1. Compute $\tilde{a} = (a, a^d, \ldots, a^{d^{m-1}}) \in \mathbb{Z}_n^m$, and for every $p_1 \in P_1$, $p_2 \in P_2$, compute the reduced evaluation points $\tilde{a}_{p_1,p_2}$.
2. For every $p_1 \in P_1$, $p_2 \in P_2$, pick $j \in [r^{p_1,p_2}]$ uniformly at random and run the decoder $\mathcal{D}_0$ with oracle access to w to obtain the answers $v^{(a)}_{p_1,p_2} = \mathcal{D}_0(\tilde{a}_{p_1,p_2}, j)$.
3. For every $p_1 \in P_1$ obtain $v^{(a)}_{p_1} = \mathcal{D}_{crt}((v^{(a)}_{p_1,p_2})_{p_2 \in P_2})$.
4. Output $v^{(a)} = \mathcal{D}_{crt}((v^{(a)}_{p_1})_{p_1 \in P_1})$.

Analysis. Fix a polynomial g with degree at most s. Fix a bit-string w at relative Hamming distance at most δ from $\mathcal{E}(g)$, where δ is at most τ_0. We proceed to verify that the above encoding and decoding satisfy the conditions of Definition 4. Conditions 1 and 4 are easily verified.

Condition 1: Observe that for each $p_1 \in P_1$, $p_2 \in P_2$, $\mathcal{D}_0$ makes at most $O(\log p_{max})$ bit-probes. So $\mathcal{D}$ makes at most $O(|P_1||P_2| \log p_{max})$ bit-probes, which, as we calculated earlier, is at most $\text{polylog}(s) \cdot \log^{1+o(1)}(n)$.

Condition 4: Note that since $\mathcal{D}_0$ decodes correctly when no noise is present, $v^{(a)}_{p_1,p_2}$ equals $\tilde{g}_{p_1,p_2}(\tilde{a}_{p_1,p_2})$. By our choice of P_1 and P_2, after two applications of the CRT, it is easy to see that $\mathcal{D}$ outputs $v = \tilde{g}(\tilde{a})$, which equals $g(a)$.

Condition 2: Fix $a \in \mathbb{Z}_n$. We want to show that with oracle access to w, with probability at least $\frac{3}{4}$, the decoder $\mathcal{D}$ on input a outputs either $g(a)$ or $\perp$. For $\pi \in P_1 \cup (P_1 \times P_2)$, we say that a point $v^{(a)}_\pi$ is *incorrect* if $v^{(a)}_\pi \notin \{\tilde{g}_\pi(\tilde{a}_\pi), \perp\}$.

By Lemma 11, for each $p_1 \in P_1$ and $p_2 \in P_2$, $v^{(a)}_{p_1,p_2}$ is incorrect with probability at most 2^{-10}. Now fix $p_1 \in P_1$. On expectation (over the decoder's randomness), at most a 2^{-10}-fraction of the points in the set $\{v^{(a)}_{p_1,p_2} : p_2 \in P_2\}$ are incorrect. By Markov's inequality, with probability at least $1 - 2^{-6}$, the fraction of points in the set $\{v^{(a)}_{p_1,p_2} : p_2 \in P_2\}$ that are incorrect is at most $\frac{1}{16}$. If the fraction of blank symbols in the set $\{v^{(a)}_{p_1,p_2} : p_2 \in P_2\}$ is at least $\frac{1}{16}$, then $\mathcal{D}_{crt}$ outputs $\perp$, which is acceptable. Otherwise, the fraction of errors and erasures (i.e., $\perp$ symbols) in the set $\{v^{(a)}_{p_1,p_2} : p_2 \in P_2\}$ is at most $\frac{1}{8}$. By Theorem 12, the decoder $\mathcal{D}_{crt}$ will output an incorrect $v^{(a)}_{p_1}$ with probability at most 2^{-6}. Thus, on expectation, at most a 2^{-6}-fraction of the points in $\{v^{(a)}_{p_1} : p_1 \in P_1\}$ are incorrect. By Markov's inequality again, with probability at least $\frac{3}{4}$, at most a $\frac{1}{16}$-fraction of the points in $\{v^{(a)}_{p_1} : p_1 \in P_1\}$ are incorrect, which by Theorem 12 implies that $\mathcal{D}^w(a)$ is either $\perp$ or $g(a)$. This establishes condition 2.

Condition 3: We show the existence of a set $G \subseteq \mathbb{Z}_n$ such that $|G| \geq (1 - \lambda)n$, and for each $a \in G$ we have $\Pr[\mathcal{D}(a) = g(a)] \geq \frac{3}{4}$. Our proof relies on the

following observation: for any $p_1 \in P_1$ and $p_2 \in P_2$, if $a \in \mathbb{Z}_n$ is chosen uniformly at random, then the evaluation point $\tilde{a}_{p_1,p_2}$ is like a uniformly chosen element from $q \in B^{p_1,p_2}$. This observation implies that if a few entries in the evaluation tables of the multivariate reduced polynomials are corrupted, then for most $a \in \mathbb{Z}_n$, the output of the decoder $\mathcal{D}$ on input a remains unaffected. We now formalize this observation.

CLAIM 13. *Fix $p_1 \in P_1$, $p_2 \in P_2$, and a point $q \in B^{p_1,p_2}$. Then*

$$\Pr_{a \in \mathbb{Z}_n}[\tilde{a}_{p_1,p_2} \equiv q] \leq \frac{4}{p_2}.$$

Proof. It is not hard to see that for a fixed $q_1 \in B^{p_1}$, the number of integers $a \in \mathbb{Z}_n$ such that $\tilde{a}_{p_1} \equiv q_1$ is at most $\lfloor \frac{n}{p_1} \rfloor + 1$. Furthermore, for a fixed $q_2 \in B^{p_1,p_2}$, the number of points in B^{p_1} that are congruent to q_2 mod p_2 is at most $\lfloor \frac{p_1}{p_2} \rfloor + 1$. Thus, for a fixed $q \in B^{p_1,p_2}$, the number of integers $a \in \mathbb{Z}_n$ such that $\tilde{a}_{p_1,p_2} \equiv q$ is at most $(\lfloor \frac{n}{p_1} \rfloor + 1)(\lfloor \frac{p_1}{p_2} \rfloor + 1)$, which is at most $4\frac{n}{p_2}$ since $n \geq p_1 \geq p_2$. ∎

Now, for every $p_1 \in P_1$ and $p_2 \in P_2$, we say that a query $(q, j) \in B^{p_1,p_2} \times [r^{p_1,p_2}]$ is *bad* if the probability that $\mathcal{D}_0^w(q, j) \neq \tilde{g}_{(p_1,p_2)}(q)$ is greater than 2^{-10}. By Lemma 11, the fraction of bad queries in $\cup_{p_1,p_2} B^{p_1,p_2} \times [r^{p_1,p_2}]$ is at most $\lambda_0 := \lambda^3 2^{-36}$. We say that a tuple of primes $(p_1, p_2) \in P_1 \times P_2$ is *bad* if more than a $2^{11}\lambda_0\lambda^{-1}$-fraction of queries in $B^{p_1,p_2} \times [r^{p_1,p_2}]$ are bad (below, *good* always denotes not bad). By averaging, the fraction of bad tuples (p_1, p_2) is at most $2^{-11}\lambda$.

For a fixed good tuple (p_1, p_2), we say that an index i^{p_1,p_2} is *bad* if more than a $2^{-11}\lambda$-fraction of queries in the copy $B^{p_1,p_2} \times \{i^{p_1,p_2}\}$ are bad. Since (p_1, p_2) is good, by averaging, at most a $2^{22}\lambda_0\lambda^{-2}$-fraction of $[r^{p_1,p_2}]$ are bad. Recall that in step 2 of the decoder $\mathcal{D}$, the indices $\{j^{p_1,p_2} : p_1 \in P_1, p_2 \in P_2\}$ are chosen uniformly at random. So on expectation, the set of indices $\{j^{p_1,p_2} : (p_1, p_2) \text{ is good}\}$ has at most a $2^{22}\lambda_0\lambda^{-2}$-fraction of bad indices. By Markov's inequality, with probability at least $\frac{7}{8}$, the fraction of bad indices in the set $\{j^{p_1,p_2} : (p_1, p_2) \text{ is good}\}$ is at most $2^{25}\lambda_0\lambda^{-2}$. We condition on this event occurring and fix the indices j^{p_1,p_2} for each $p_1 \in P_1$, $p_2 \in P_2$.

Fix a good tuple (p_1, p_2) and a good index j^{p_1,p_2}. By Claim 13, for a uniformly random $a \in \mathbb{Z}_n$, the query $(\tilde{a}_{p_1,p_2}, j^{p_1,p_2})$ is bad with probability at most $2^{-9}\lambda$. By linearity of expectation, for a random $a \in \mathbb{Z}_n$, the expected fraction of bad queries in the set $S^a = \{(\tilde{a}_{p_1,p_2}, j^{p_1,p_2}) : p_1 \in P_1, p_2 \in P_2\}$ is at most $2^{-11}\lambda + 2^{25}\lambda_0\lambda^{-2} + 2^{-9}\lambda$, which is at most $2^{-8}\lambda$ by definition of λ_0. Thus, by Markov's inequality, for a random $a \in \mathbb{Z}_n$, with probability at least $1 - \lambda$, the fraction of bad queries in the set S^a is at most 2^{-8}. By linearity of expectation, there exists some subset $G \subseteq \mathbb{Z}_n$ with $|G| \geq (1 - \lambda)n$ such that for every $a \in G$, the fraction of bad queries in S^a is at most 2^{-8}.

Now fix $a \in G$. By definition, the fraction of bad queries in S^a is at most 2^{-8}, and furthermore, each of the good queries in S^a is incorrect with probability at most 2^{-10}. So on expectation, the fraction of errors and erasures in S^a is at most $2^{-8} + 2^{-10}$. By Markov's inequality, with probability at least $\frac{7}{8}$, the fraction of errors and erasures in the set $\{v^{(a)}_{p_1,p_2} : p_1 \in P_1, p_2 \in P_2\}$ is at most $2^{-5} + 2^{-7}$, which is at most $\frac{1}{25}$. We condition on this event occurring. By averaging, for more than a $\frac{4}{5}$-fraction of the primes $p_1 \in P_1$, the set $\{v^{(a)}_{p_1,p_2} : p_2 \in P_2\}$ has at most a $\frac{1}{5}$-fraction of errors and erasures, which can be corrected by the CRT-decoder $\mathcal{D}_{crt}$. Thus, after step 3 of the decoder $\mathcal{D}$, the set $\{v^{(a)}_{p_1}\}$ has at most a $\frac{1}{5}$-fraction of errors and erasures, which again will be corrected by the CRT-decoder $\mathcal{D}_{crt}$. Hence, by the union bound, the two events that we conditioned on earlier occur simultaneously with probability at least $\frac{3}{4}$, and $\mathcal{D}^w(a)$ will be $g(a)$. ∎

4. Future work. Many questions are opened up by our model of error-correcting data structures. We mention a few:

- There are plenty of other natural data structure problems, such as RANK, PREDECESSOR, versions of NEAREST NEIGHBOR, etc. [31]. What about the length-versus-probes trade-offs for their error-correcting versions? The obvious approach is to put the best known LDC on top of the best known nonerror-correcting data structures (Proposition 1). This is not always optimal, though; for instance, in the case of s-out-of-n MEMBERSHIP one can do significantly better, as we showed here.
  The problem of computing RANK within a sparse ordered set is a good target. Suppose we are given a universe $[n]$, some nonnegative integer $s \leq n$, and a subset $S \subseteq [n]$ of size at most s. The rank problem is to store S compactly so that on input $i \in [n]$, the value $|\{j \in S : j \leq i\}|$ can be computed efficiently. For obvious information-theoretic reasons, any data structure for this problem needs length at least $\log \binom{n}{s} \geq s \log(n/s)$ and makes $\Omega(\log s)$ bit-probes for each query. If $s = O(\log n)$, one can trivially obtain an error-correcting data structure of optimal length $O(s \log n)$ with $O(\log^2 n)$ bit-probes, which is only quadratically worse than optimal: write down S as a string of $s \log n$ bits, encode it with a good error-correcting code, and read the entire encoding when an index is queried. However, it may be possible to do something smarter.
- It is often natural to assume that a memory cell contains not a bit, but some number from, say, a polynomial-size universe. This is called the *cell-probe* model [39], in contrast to the *bit-probe* model we considered here. Probing a cell gives $O(\log n)$ bits at the same time, which can significantly improve the length-versus-probes trade-off and is worth studying. Still, we view the bit-probe approach taken here as more fundamental than the cell-probe model: a t-probe cell-probe structure is an $O(t \log n)$-probe bit-probe structure, but not vice versa. Also, the way memory is addressed in actual computers in constant chunks of, say, 8 or 16 bits at a time, is closer in spirit to the bit-probe model than to the cell-probe model.
- Zvi Lotker suggested to us the following connection with distributed computing, which deserves further study. Suppose the data structure is distributed over N processors, each holding one bit. Interpreted in this setting, an error-correcting data structure allows an honest party to answer queries about the encoded object while communicating with at most t processors. The answer will be correct with probability $1 - \varepsilon$, even if up to a δ-fraction of the N processors are faulty or even malicious (the querier need not know where the faulty/malicious sites are).

Appendix A. Constructions for the INNER PRODUCT problem.

A.1. INNER PRODUCT: Noiseless case. Here we show bounds for INNER PRODUCT, first for the case where there is no noise ($\delta = 0$).

Upper bound. Consider all strings z of weight at most $\lceil r/t \rceil$. The number of such z is $B(n, \lceil r/t \rceil) = \sum_{i=0}^{\lceil r/t \rceil} \binom{n}{i} \leq (etn/r)^{r/t}$. We define our codeword by writing down, for all z in lexicographic order, the inner product $x \cdot z$ mod 2. If we want to recover the inner product $x \cdot y$ for some y of weight at most r, we write $y = z_1 \oplus \cdots \oplus z_t$ for z_j's of weight at most $\lceil r/t \rceil$ and recover $x \cdot z_j$ for each $j \in [t]$, using one probe for each. Summing the results of the t probes gives $x \cdot y \pmod 2$. In particular, for $t = 1$ probes, the length is $B(n, r)$.

Lower bound. To prove a nearly matching lower bound, we use Miltersen's technique of relating a data structure to a two-party communication game [29]. We refer to [26] for a general introduction to communication complexity. Suppose Alice gets string $x \in \{0,1\}^n$, Bob gets string $y \in \{0,1\}^n$ of weight $\leq r$, and they need to compute $x \cdot y \pmod 2$ with bounded error probability and minimal communication between them. Call this communication problem $\mathrm{IP}_{n,r}$. Let $B(n,r) = \sum_{i=0}^{r} \binom{n}{i}$ be the size of Q, i.e., the number of possible queries y. The proof of our communication complexity lower bound below uses a fairly standard discrepancy argument, but we have not found this specific result anywhere.

THEOREM 14. *Every communication protocol for* $\mathrm{IP}_{n,r}$ *with worst-case (or even average-case) success probability* $\geq 1/2 + \beta$ *needs at least* $\log(B(n,r)) - 2\log(1/2\beta)$ *bits of communication.*

Proof. Let μ be the uniform input distribution: each x has probability $1/2^n$ and each y of weight $\leq r$ has probability $1/B(n,r)$. We show a lower bound on the communication c of *deterministic* protocols that compute $\mathrm{IP}_{n,r}$ with μ-probability at least $1/2 + \beta$. By Yao's principle [38], this lower bound then also applies to (public-coin) randomized protocols.

Consider a deterministic c-bit protocol. Assume the last bit communicated is the output bit. It is well known that this partitions the input space into *rectangles* $R_1, \ldots, R_{2^c}$, where $R_i = A_i \times B_i$, and the protocol gives the same output bit a_i for each $(x,y) \in R_i$.[8] The *discrepancy* of rectangle $R = A \times B$ under μ is the difference between the weight of the 0's and the 1's in that rectangle:

$$\delta_\mu(R) = \left|\mu(R \cap \mathrm{IP}_{n,r}^{-1}(1)) - \mu(R \cap \mathrm{IP}_{n,r}^{-1}(0))\right|.$$

We can show for every rectangle that its discrepancy is not very large.

LEMMA 15. $\delta_\mu(R) \leq (\sqrt{|R|})/(\sqrt{2^n}B(n,r))$.

Proof. Let M be the $2^n \times B(n,r)$ matrix whose (x,y)-entry is $(-1)^{\mathrm{IP}_{n,r}(x,y)} = (-1)^{x \cdot y}$. It is easy to see that $M^T M = 2^n I$, where I is the $B(n,r) \times B(n,r)$ identity matrix. This implies, for every $v \in \mathbb{R}^{B(n,r)}$,

$$\| Mv \|^2 = (Mv)^T \cdot (Mv) = v^T M^T M v = 2^n v^T v = 2^n \| v \|^2.$$

Let $R = A \times B$, and let $v_A \in \{0,1\}^{2^n}$ and $v_B \in \{0,1\}^{B(n,r)}$ be the characteristic (column) vectors of the sets A and B. Note that $\| v_A \| = \sqrt{|A|}$ and $\| v_B \| = \sqrt{|B|}$. The sum of M-entries in R is $\sum_{a \in A, b \in B} M_{ab} = v_A^T M v_B$. We can bound this by using the Cauchy–Schwarz inequality:

$$|v_A^T M v_B| \leq \| v_A \| \cdot \| Mv_B \| = \| v_A \| \cdot \sqrt{2^n} \| v_B \| = \sqrt{|A| \cdot |B| \cdot 2^n}.$$

Observing that $\delta_\mu(R) = |v_A^T M v_B|/(2^n B(n,r))$ and $|R| = |A| \cdot |B|$ concludes the proof. □

Define the success and failure probabilities (under μ) of the protocol as

$$P_s = \sum_{i=1}^{2^c} \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) \quad \text{and} \quad P_f = \sum_{i=1}^{2^c} \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1 - a_i)).$$

[8] See [26, section 1.2]. The number of rectangles may be smaller than 2^c, but we can always add empty ones.

Then

$$\begin{aligned}
2\beta &\le P_s - P_f \\
&= \sum_i \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) - \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1-a_i)) \\
&\le \sum_i \left|\mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(a_i)) - \mu(R_i \cap \mathrm{IP}_{n,r}^{-1}(1-a_i))\right| \\
&= \sum_i \delta_\mu(R_i) \;\le\; \frac{\sum_i \sqrt{|R_i|}}{\sqrt{2^n}B(n,r)} \;\le\; \frac{\sqrt{2^c}\sqrt{\sum_i |R_i|}}{\sqrt{2^n}B(n,r)} \;=\; \sqrt{2^c/B(n,r)},
\end{aligned}$$

where the last inequality is an application of the Cauchy–Schwarz inequality, and the last equality holds because $\sum_i |R_i|$ is the total number of inputs, which is $2^n B(n,r)$.

Rearranging gives $2^c \ge (2\beta)^2 B(n,r)$; hence $c \ge \log(B(n,r)) - 2\log(1/2\beta)$. ∎

Armed with this communication complexity bound we can prove Theorem 2: every (t,ε)-data structure for $\mathrm{IP}_{n,r}$ needs length $N \ge \frac{1}{2}2^{(\log(B(n,r))-2\log(1/(1-2\varepsilon))-1)/t}$.

Proof of Theorem 2. Consider a data structure $\mathcal{E}$ of length N. We will use this to obtain a communication protocol for $\mathrm{IP}_{n,r}$ that uses $t(\log(N)+1)+1$ bits of communication, and then we will invoke Theorem 14 to obtain the lower bound.

Alice holds x, and hence $\mathcal{E}(x)$, while Bob simulates the decoder. Bob starts the communication. He picks his first probe to the data structure and sends it over in $\log N$ bits. Alice sends back the 1-bit answer. After t rounds of communication, all t probes have been simulated and Bob can give the same output as the decoder would have given. Bob's output will be the last bit of the communication. Theorem 14 now implies

$$t(\log(N)+1)+1 \ge \log(B(n,r)) - 2\log(1/(1-2\varepsilon)).$$

Rearranging gives the bound on N. ∎

For fixed ε, the lower bound is $N = \Omega\left(B(n,r)^{1/t}\right)$. This is $\Omega((n/r)^{r/t})$, which (at least for small t) is not too far from the upper bound of approximately $(etn/r)^{r/t}$ mentioned above. Note that in general our bound on N is superpolynomial in n whenever $t = o(r)$. For instance, when $r = \alpha n$ for some constant $\alpha \in (0, 1/2)$, $N = \Omega(2^{nH(\alpha)/t})$, which is nontrivial whenever $t = o(n)$. Finally, note that the proof also works if Alice's messages are longer than 1 bit (i.e., if the code is over a larger-than-binary alphabet).

A.2. Inner product: Noisy case.

A.2.1. Constructions for Substring. It is easy to construct error-correcting data structures for Substring, which also suffice for Inner product. Note that since we are recovering r bits, and each probe gives at most one bit of information, by information theory we need at least roughly r probes to the data structure. Our solutions below will use $O(r \log r)$ probes. View x as a concatenation $x = x^{(1)} \dots x^{(r)}$ of r strings of n/r bits each (we ignore rounding for simplicity), and define $\mathcal{E}(x)$ as the concatenation of the Hadamard codes of these r pieces. Then $\mathcal{E}(x)$ has length $N = r \cdot 2^{n/r}$.

If $\delta \ge 1/4r$, then the adversary could corrupt one of the r Hadamard codes by 25% noise, ensuring that some of the bits of x are irrevocably lost even when we allow the full N probes. However, if $\delta \ll 1/r$, then we can recover each bit x_i with small constant error probability by 2 probes in the Hadamard codeword where i sits,

and with error probability $\ll 1/r$ using $O(\log r)$ probes. Hence we can compute $f(x,y) = x_y$ with error close to 0 using $t = O(r \log r)$ probes (or with $2r$ probes if $\delta \ll 1/r^2$).[9] This also implies that *any* data structure problem where $f(x,q)$ depends on at most some fixed constant r bits of x, has an error-correcting data structure of length $N = r \cdot 2^{n/r}$ and $t = O(r \log r)$ probes, and that works if $\delta \ll 1/r$. Alternatively, we can take Efremenko's 3-probe LDC [16] of length $N \approx 2^{2^{\sqrt{\log n}}}$, and just decode each of the r bits separately. Using $O(\log r)$ probes to recover a bit with error probability $\ll 1/r$, we recover the r-bit string x_y using $t = O(r \log r)$ probes even if δ is a constant independent of r.

A.2.2. Constructions for INNER PRODUCT**.** Going through the proof of [16], one can see that it allows us to compute the parity of any set of r bits from x using at most $3r$ probes with error ε, if the noise rate δ is at most $\varepsilon/(3r)$ (just add the results of the 3 probes one would make for each bit in the parity). To get error-correcting data structures even for small constant t (independent of r), we can adapt the polynomial schemes from [5] to prove Theorem 3: for every $p \geq 2$, there exists a $(t, \delta, t\delta)$-error-correcting data structure for $\mathrm{IP}_{n,r}$ of length $N \leq p \cdot 2^{r(t-1)^2 n^{1/(t-1)}}$.

Proof of Theorem 3. Here we construct t-probe error-correcting data structures for the inner product problem, inspired by the approach to LDCs of [5]. Let d be an integer to be determined later. Pick $m = \lceil dn^{1/d} \rceil$. Then $\binom{m}{d} \geq n$, so there exist n distinct sets $S_1, \ldots, S_n \subseteq [m]$, each of size d. For each $x \in \{0,1\}^n$, define an m-variate polynomial p_x of degree d over $\mathbb{F}_2$ by

$$p_x(z_1, \ldots, z_m) = \sum_{i=1}^{n} x_i \prod_{j \in S_i} z_j.$$

Note that if we identify S_i with its m-bit characteristic vector, then $p_x(S_i) = x_i$. For $z^{(1)}, \ldots, z^{(r)} \in \{0,1\}^m$, define an rm-variate polynomial $p_{x,r}$ over $\mathbb{F}_2$ by

$$p_{x,r}(z^{(1)}, \ldots, z^{(r)}) = \sum_{j=1}^{r} p_x(z^{(j)}).$$

This polynomial $p_{x,r}(z)$ has rm variables and degree d, and allows us to evaluate parities of any set of r of the variables of x: if $y \in \{0,1\}^n$ (of weight r) has its 1-bits at positions $i_1, \ldots, i_r$, then

$$p_{x,r}(S_{i_1}, \ldots, S_{i_r}) = \sum_{j=1}^{r} x_{i_j} = x \cdot y \pmod 2.$$

To construct an error-correcting data structure for $\mathrm{IP}_{n,r}$, it thus suffices to give a structure that enables us to evaluate $p_{x,r}$ at any point w of our choice.[10]

Let $w \in \{0,1\}^{rm}$. Suppose we "secret-share" this into t pieces $w^{(1)}, \ldots, w^{(t)} \in \{0,1\}^{rm}$ which are uniformly random subject to the constraint $w = w^{(1)} + \cdots + w^{(t)}$. Now consider the trm-variate polynomial $q_{x,r}$ defined by

$$q_{x,r}(w^{(1)}, \ldots, w^{(t)}) = p_{x,r}(w^{(1)} + \cdots + w^{(t)}). \tag{3}$$

[9] It follows from Buhrman et al. [11] that if we allow a *quantum* decoder, the factor of $\log r$ is not needed.

[10] If we want to be able to compute $x \cdot y \pmod 2$ for $|y| < r$, we can just add a dummy 0 as $(n+1)$st variable to x, and use its index $r - |y|$ times as inputs to $p_{x,r}$.

Each monomial M in this polynomial has at most d variables. If we pick $d = t-1$, then for every M there will be a $j \in [t]$ such that M does not contain variables from $w^{(j)}$. Assign all such monomials to a new polynomial $q_{x,r}^{(j)}$, which is independent of $w^{(j)}$. This allows us to write

$$q_{x,r}(w^{(1)}, \ldots, w^{(t)}) = q_{x,r}^{(1)}(w^{(2)}, \ldots, w^{(t)}) + \cdots + q_{x,r}^{(t)}(w^{(1)}, \ldots, w^{(t-1)}). \tag{4}$$

Note that each $q_{x,r}^{(j)}$ has domain of size $2^{(t-1)rm}$. The data structure is defined as the concatenation, for all $j \in [t]$, of the values of $q_{x,r}^{(j)}$ on all possible inputs. This has length

$$N = t \cdot 2^{(t-1)rm} \leq t \cdot 2^{r(t-1)^2 n^{1/(t-1)}}.$$

This length is $2^{O(rn^{1/(t-1)})}$ for $t = O(1)$.

Answering a query works as follows: the decoder would like to evaluate $p_{x,r}$ on some point $w \in \{0,1\}^{rm}$. He picks $w^{(1)}, \ldots, w^{(t)}$ as above, and for all $j \in [t]$, in the jth block of the code probes the point $w^{(1)}, \ldots, w^{(j-1)}, w^{(j+1)}, \ldots, w^{(t)}$. This, if uncorrupted, returns the value of $q_{x,r}^{(j)}$ at that point. The decoder outputs the sum of his t probes (mod 2). If none of the probed bits were corrupted, then the output is $p_{x,r}(w)$ by (3) and (4). Note that the probe within the jth block is uniformly random in that block, so its error probability is exactly the fraction δ_j of errors in the jth block. Hence by the union bound, the total error probability is at most $\sum_{j=1}^{t} \delta_j$. If the overall fraction of errors in the data structure is at most δ, then we have $\frac{1}{t}\sum_{j=1}^{t} \delta_j \leq \delta$, and hence the total error probability is at most $t\delta$. □

For the $t = 2$ case, we get something simpler and better from the Hadamard code. This code, of length 2^n, actually allows us to compute $x \cdot y$ (mod 2) for any $y \in \{0,1\}^n$ of our choice, with 2 probes and error probability at most 2δ (just probe z and $y \oplus z$ for uniformly random $z \in \{0,1\}^n$ and observe that $(x \cdot z) \oplus (x \cdot (z \oplus y)) = x \cdot y$). Note that for $r = \Theta(n)$ and $t = O(1)$, even nonerror-correcting data structures need length $2^{\Theta(n)}$ (Theorem 2). This is an example where error-correcting data structures are not significantly longer than the nonerror-correcting kind.

Appendix B. Nonbinary answer set. We prove Lemma 11, a version of Theorem 9 when the answer set A is nonbinary. We first encode the $\ell|Q|$-bit string $\langle f(x,q)\rangle_{q \in Q}$ by an RLDC and use the decoder of the RLDC to recover each of the ℓ bits of $f(x,q)$. Now it is possible that for each $q \in Q$, the decoder outputs some blank symbols $\perp$ for some of the bits of $f(x,q)$, and no query could be answered correctly. To circumvent this, we first encode each ℓ-bit string $f(x,q)$ with a good error-correcting code, then encode the entire string by the RLDC. Now if the decoder does not output too many errors or blank symbols among the bits of the error-correcting code for $f(x,q)$, we can recover it. We need a family of error-correcting codes with the following property; see, e.g., page 668 in [32].

FACT 16. *For every $\delta \in (0, 1/2)$ there exists $R \in (0,1)$ such that for all n, there exists a binary linear code of block length n, information length Rn, and Hamming distance δn such that the code can correct from e errors and s erasures, as long as $2e + s < \delta n$.*

Proof of Lemma 11. We construct only a relaxed error-correcting data structure with error probability $\varepsilon = \frac{1}{4}$. By a standard amplification technique (i.e., $O(\log(1/\varepsilon'))$ repetitions) we can reduce the error probability to any other positive constant ε'. Let $\mathcal{E}_{ecc} : \{0,1\}^{\ell} \to \{0,1\}^{\ell'}$ be an asymptotically good binary error-correcting code (from

Fact 16), with $\ell' = O(\ell)$ and relative distance $\frac{3}{8}$, and decoder $\mathcal{D}_{ecc}$. By Theorem 9, there exist $c_0, \tau_0 > 0$ such that for every $\delta \le \tau_0$, there is an $(O(1), \delta, \frac{1}{32}, c_0\delta)$-RLDC. Let $\mathcal{E}_0$ and $\mathcal{D}_0$ denote its encoder and decoder, respectively.

Encoding. We construct a data structure for f as follows. Without loss of generality, we may impose an ordering on the set Q and identify each $q \in Q$ with an integer in $[Q]$. Define the encoder $\mathcal{E} : X \to \{0,1\}^N$, where $N = O((\ell' \cdot |Q|)^{1+\eta})$, as

$$\mathcal{E}(x) = \mathcal{E}_0\left(\langle\, \mathcal{E}_{ecc}(f(x,q))\,\rangle_{q\in Q}\right).$$

Decoding. With input $q \in Q$ and oracle access to $w \in \{0,1\}^N$, the decoder $\mathcal{D}$ does the following:

1. For each $j \in [\ell']$, let $r_j = \mathcal{D}_0^w\left((q-1)\ell' + j\right)$ and set $r = r_1 \dots r_{\ell'} \in \{0,1,\perp\}^{\ell'}$.
2. If the number of blank symbols $\perp$ in r is at least $\frac{\ell'}{8}$, then output $\perp$. Else, output $\mathcal{D}_{ecc}(r)$.

Analysis. Fix $x \in X$ and $w \in \{0,1\}^N$ such that $\Delta(w, \mathcal{E}(x)) \le \delta N$, and $\delta \le \tau$, where τ is the minimum of τ_0 and $\lambda/(64c_0)$. We need to argue that the above encoding and decoding satisfy the four conditions of Definition 4. For condition 1, since $\mathcal{D}_0$ makes $O(1)$ bit-probes and $\mathcal{D}$ runs this ℓ' times, $\mathcal{D}$ makes $O(\ell') = O(\ell)$ bit-probes into w.

We now show $\mathcal{D}$ satisfies condition 2. Fix $q \in Q$. We want to show $\Pr[\mathcal{D}^w(q) \in \{f(x,q), \perp\}] \ge \frac{3}{4}$. By Theorem 9, for each $j \in [\ell']$, with probability at most $\frac{1}{32}$, $r_j = f(x,q)_j \oplus 1$. So on expectation, for at most a $\frac{1}{32}$-fraction of the indices j, $r_j = f(x,q)_j \oplus 1$. By Markov's inequality, with probability at least $\frac{3}{4}$, the number of indices j such that $r_j = f(x,q)_j \oplus 1$ is at most $\frac{\ell'}{8}$. If the number of $\perp$ symbols in r is at least $\frac{\ell'}{8}$, then $\mathcal{D}$ outputs $\perp$, so assume the number of $\perp$ symbols is less than $\frac{\ell'}{8}$. Those $\perp$'s are viewed as erasures in the codeword $\mathcal{E}_{ecc}(f(x,q))$. Since $\mathcal{E}_{ecc}$ has relative distance $\frac{3}{8}$, by Fact 16, $\mathcal{D}_{ecc}$ will correct these errors and erasures and output $f(x,q)$.

For condition 3, we show there exists a large subset G of q's satisfying $\Pr[\mathcal{D}^w(q) = f(x,q)] \ge \frac{3}{4}$. Let $y = \langle\, \mathcal{E}_{ecc}(f(x,q))\,\rangle_{q\in Q}$, which is an $\ell'|Q|$-bit string. Call an index i in y *bad* if $\Pr[\mathcal{D}_0^w(i) = y_i] < \frac{3}{4}$. By Theorem 9, at most a $c_0\delta$-fraction of the indices in y are bad. We say that a query $q \in Q$ is *bad* if more than a $\frac{1}{64}$-fraction of the bits in $\mathcal{E}_{ecc}(f(x,q))$ are bad. By averaging, the fraction of bad queries in Q is at most $64c_0\delta$, which is at most λ by our choice of τ. We define G to be the set of $q \in Q$ that are not bad. Clearly $|G| \ge (1-\lambda)|Q|$.

Fix $q \in G$. On expectation, the fraction of indices j in r such that $r_j \ne f(x,q)_j$ is at most $\frac{1}{64} + \frac{1}{32}$. Hence by Markov's inequality, with probability at least $\frac{3}{4}$, the fraction of indices in r such that $r_j \ne f(x,q)_j$ is at most $\frac{3}{16}$. Thus, by Fact 16, $\mathcal{D}_{ecc}(r)$ will recover from these errors and erasures, and output $f(x,q)$.

Finally, condition 4 follows since the pair $(\mathcal{E}_0, \mathcal{D}_0)$ satisfies Condition 4, finishing the proof. □

Appendix C. CRT codes. In this section we explain how Theorem 12 follows from known facts. In [20], Goldreich, Ron, and Sudan designed a unique decoding algorithm for the CRT code.

THEOREM 17 (from [20]). *Given a CRT code with basis $p_1 < \cdots < p_N$ and rate K/N, there exists a polynomial-time algorithm that can correct up to $\frac{\log p_1}{\log p_1 + \log p_N}(N-K)$ errors.*

By choosing the primes appropriately, we can establish Theorem 12. In particular, the following well-known estimate, essentially a consequence of the Prime Number Theorem, is useful. See, for instance, Theorem 4.7 in [2] for more details.

FACT 18. *For an integer $\ell > 0$, the ℓth prime (denoted q_ℓ) satisfies $\frac{1}{6}\ell \log \ell < q_\ell < 13\ell \log \ell$.*

Proof of Theorem 12. Let $K = \lfloor \frac{12 \log T}{\log \log T} \rfloor$ and q_ℓ denote the ℓth prime. By Fact 18, $q_K > \frac{1}{6} K \log K > \log T$ and $q_{3K-1} < 39K \log 3K < 500 \log T$. Also, notice that $\prod_{i=K}^{2K-1} q_i > q_K^K > (\log T)^{\frac{\log T}{\log \log T}} = T$. Thus, the CRT code with basis $q_K, \ldots, q_{3K-1}$ has message space $\mathbb{Z}_T$, rate $\frac{1}{2}$, and relative distance $\frac{1}{2}$. Lastly, by Theorem 17, the code can correct a fraction $\frac{1}{4} - O(\frac{1}{\log \log T})$ of errors. ∎

Acknowledgments. Thanks to Harry Buhrman, Zvi Lotker, Peter Bro Miltersen, Gabriel Moruz, Jaikumar Radhakrishnan, Nitin Saxena, and Madhu Sudan for discussions, comments, pointers, and other help with the two conference papers that this paper is based on. We also thank the anonymous referees for many comments that helped improve the presentation.

REFERENCES

[1] A. ANDONI AND P. INDYK, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, Comm. ACM, 51 (2008), pp. 117–122.
[2] T. M. APOSTOL, *Introduction to Analytic Number Theory*, Springer-Verlag, New York, 1979.
[3] Y. AUMANN AND M. BENDER, *Fault-tolerant data structures*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1996, pp. 580–589.
[4] P. BEAME AND F. E. FICH, *Optimal bounds for the predecessor problem and related problems*, J. Comput. System Sci., 65 (2002), pp. 38–72.
[5] A. BEIMEL, Y. ISHAI, AND E. KUSHILEVITZ, *General constructions for information-theoretical private information retrieval*, J. Comput. System Sci., 72 (2005), pp. 247–281.
[6] E. BEN-SASSON, O. GOLDREICH, P. HARSHA, M. SUDAN, AND S. VADHAN, *Robust PCPs of proximity, shorter PCPs, and applications to coding*, SIAM J. Comput., 36 (2006), pp. 889–974.
[7] B. H. BLOOM, *Space/time trade-offs in hash coding with allowable errors*, Comm. ACM, 13 (1970), pp. 422–426.
[8] G. BRODAL, R. FAGERBERG, I. FINOCCHI, F. GRANDONI, G. ITALIANO, A. JØRGENSON, G. MORUZ, AND T. MØLHAVE, *Optimal resilient dynamic dictionaries*, in Proceedings of the 15th European Symposium on Algorithms (ESA), Springer-Verlag, Berlin, 2007, pp. 347–358.
[9] G. BRODAL, A. JØRGENSON, G. MORUZ, AND T. MØLHAVE, *Counting in the presence of memory faults*, in Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC), Springer-Verlag, Berlin, 2009, pp. 842–851.
[10] H. BUHRMAN, P. B. MILTERSEN, J. RADHAKRISHNAN, AND S. VENKATESH, *Are bitvectors optimal?*, SIAM J. Comput., 31 (2002), pp. 1723–1744.
[11] H. BUHRMAN, I. NEWMAN, H. RÖHRIG, AND R. DE WOLF, *Robust polynomials and quantum algorithms*, Theory Comput. Syst., 40 (2007), pp. 379–395.
[12] S. CAMINITI, I. FINOCCHI, AND E. G. FUSCO, *Local dependency dynamic programming in the presence of memory faults*, in Proceedings of the 28th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Dagstuhl Publishing, Wodein, Germany, 2011, pp. 45–56.
[13] R. CANETTI, Y. ISHAI, R. KUMAR, M. REITER, R. RUBINFELD, AND R. WRIGHT, *Selective private function evaluation with applications to private statistics*, in Proceedings of 20th Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2001, pp. 293–304.
[14] V. CHEN, E. GRIGORESCU, AND R. DE WOLF, *Efficient and error-correcting data structures for membership and polynomial evaluation*, in Proceedings of 27th Annual Symposium on Theoretical Aspects of Computer Science (STACS), 2010, pp. 203–214. Available online at hal.inria.fr/STACS2010.
[15] R. DE WOLF, *Error-correcting data structures*, in Proceedings of the 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS), 2009, pp. 313–324. Available online at stars2009.informatif.uni-freiberg.de/proceedings.php.

[16] K. Efremenko, *3-query locally decodable codes of subexponential length*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, ACM, New York, 2009, pp. 39–44.

[17] I. Finocchi, F. Grandoni, and G. F. Italiano, *Resilient search trees*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadelphia, 2007, pp. 547–554.

[18] I. Finocchi and G. Italiano, *Sorting and searching in the presence of memory faults (without redundancy)*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, New York, 2004, pp. 101–110.

[19] M. Fredman, M. Komlós, and E. Szemerédi, *Storing a sparse table with $O(1)$ worst case access time*, J. ACM, 31 (1984), pp. 538–544.

[20] O. Goldreich, D. Ron, and M. Sudan, *Chinese remaindering with errors*, IEEE Trans. Inform. Theory, 46 (2000), pp. 1330–1338.

[21] A. Golynski, *Optimal lower bounds for rank and select indexes*, Theoret. Comput. Sci., 387 (2007), pp. 348–359.

[22] A. G. Jørgenson, G. Moruz, and T. Mølhave, *Priority queues resilient to memory faults*, in Proceedings of the 10th International Workshop on Algorithms and Data Structures (WADS), Lecture Notes in Comput. Sci. 4619, Springer, Berlin, 2007, pp. 127–138.

[23] J. Katz and L. Trevisan, *On the efficiency of local decoding procedures for error-correcting codes*, in Proceedings the of 32nd Annual ACM Symposium on Theory of Computing, ACM, New York, 2000, pp. 80–86.

[24] K. S. Kedlaya and C. Umans, *Fast modular composition in any characteristic*, in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 2008, pp. 146–155.

[25] I. Kerenidis and R. de Wolf, *Exponential lower bound for 2-query locally decodable codes via a quantum argument*, J. Comput. System Sci., 69 (2004), pp. 395–420.

[26] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, Cambridge, UK, 1997.

[27] J. H. van Lint, *Introduction to Coding Theory*, 3rd ed., Springer, Berlin, 1998.

[28] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[29] P. B. Miltersen, *Lower bounds for union-split-find related problems on random access machines*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, ACM, New York, 1994, pp. 625–634.

[30] P. B. Miltersen, *On the cell probe complexity of polynomial evaluation*, Theoret. Comput. Sci., 143 (1995), pp. 167–174.

[31] P. B. Miltersen, *Cell Probe Complexity—A Survey*, Invited paper at Advances in Data Structures Workshop, 1999. Available online at www.daimi.au.dk/∼bromille/Papers/index.html

[32] V. S. Pless, W. C. Huffman, and R. A. Brualdi, eds., *Handbook of Coding Theory, Vol.* 1, Elsevier Science, New York, 1998.

[33] M. Pătraşcu and M. Thorup, *Time-space trade-offs for predecessor search*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, ACM, New York, 2006, pp. 232–240.

[34] J. Radhakrishnan, P. Sen, and S. Venkatesh, *The quantum complexity of set membership*, Algorithmica, 34 (2002), pp. 462–479.

[35] A. Ta-Shma, *Storing information with extractors*, Inform, Process. Lett., 83 (2002), pp. 267–274.

[36] L. Trevisan, *Some applications of coding theory in computational complexity*, Quaderni di Matematica, 13 (2004), pp. 347–424.

[37] D. Woodruff, *New Lower Bounds for General Locally Decodable Codes*, ECCC Report TR07-006, eccc.hpi-web.de/eccc-reports/2007/TR07-006/index.html (2007).

[38] A. C-C. Yao, *Probabilistic computations: Toward a unified measure of complexity*, in Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1977, pp. 222–227.

[39] A. C-C. Yao, *Should tables be sorted?*, J. ACM, 28 (1981), pp. 615–628.

[40] S. Yekhanin, *Towards 3-query locally decodable codes of subexponential length*, J. ACM, 55 (2008), 1.

[41] S. Yekhanin, *Locally decodable codes*, Found. Trends Theor. Comput. Sci., 6 (2012), pp. 139–255.