Electronic Colloquium on Computational Complexity, Report No. 195 (2018)

Erasures vs. Errors in Local Decoding and Property Testing

Sofya Raskhodnikova[1]
Department of Computer Science, Boston University, USA
sofya@bu.edu

Noga Ron-Zewi
Department of Computer Science, University of Haifa, Israel
noga@cs.haifa.il

Nithin Varma[2]
Department of Computer Science, Boston University, USA
nvarma@bu.edu
https://orcid.org/0000-0002-1211-2566

Abstract

We initiate the study of the role of erasures in local decoding and use our understanding to prove a separation between erasure-resilient and tolerant property testing. Local decoding in the presence of errors has been extensively studied, but has not been considered explicitly in the presence of erasures.

Motivated by applications in property testing, we begin our investigation with local *list* decoding in the presence of erasures. We prove an analog of a famous result of Goldreich and Levin on local list decodability of the Hadamard code. Specifically, we show that the Hadamard code is locally list decodable in the presence of a constant fraction of erasures, arbitrary close to 1, with list sizes and query complexity better than in the Goldreich-Levin theorem. We use this result to exhibit a property which is testable with a number of queries independent of the length of the input in the presence of erasures, but requires a number of queries that depends on the input length, n, for tolerant testing. We further study *approximate* locally list decodable codes that work against erasures and use them to strengthen our separation by constructing a property which is testable with a constant number of queries in the presence of erasures, but requires $n^{\Omega(1)}$ queries for tolerant testing.

Next, we study the general relationship between local decoding in the presence of errors and in the presence of erasures. We observe that every locally (uniquely or list) decodable code that works in the presence of errors also works in the presence of twice as many erasures (with the same parameters up to constant factors). We show that there is also an implication in the other direction for locally decodable codes (with unique decoding): specifically, that the existence of a locally decodable code that works in the presence of erasures implies the existence of a locally decodable code that works in the presence of errors and has related parameters. However, it remains open whether there is an implication in the other direction for locally *list* decodable codes[3]. We relate this question to other open questions in local decoding.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms, Mathematics of computing → Coding theory

Keywords and phrases Erasures vs. Errors, Erasure-Resilient Property Testing, Tolerant Property Testing, Hadamard Code, Local List Decoding

[1] Supported by National Science Foundation under Grant No. CCF-142297
[2] Supported by National Science Foundation under Grant No. CCF-142297
[3] Our Hadamard result shows that there has to be some difference in parameters for some settings.

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ISSN 1433-8092

Digital Object Identifier 10.4230/LIPIcs...

Acknowledgements The authors would like to thank Venkatesan Guruswami for helping us tighten the analysis of our local list erasure-decoder for the Hadamard code and also for making a suggestion that led us to Observation 4.2. The authors are grateful to Prahladh Harsha, Or Meir, Ramesh Krishnan S. Pallavoor, Adam Smith, Sergey Yekhanin, and Avi Wigderson for useful discussions. Last but not least, the authors would like to express their gratitude to the sponsors and organizers of the Workshop on Local Algorithms 2018 for making this collaboration possible.

1 Introduction

The contributions of this work are two-fold: on one hand, we initiate the investigation of erasures in local decoding; on the other hand, we apply our understanding of local list decoding to study the relative difficulty with which sublinear algorithms can cope with erasures and errors in their inputs.

Intuitively, a family of codes is *locally decodable* in the presence of a specified type of corruptions (erasures or errors) if there exists an algorithm that, given oracle access to a codeword with a limited fraction of specified corruptions, can decode each desired character of the encoded message with high probability after querying a small number of characters in the corrupted codeword. In other words, we can simulate oracle access to the message by using oracle access to a corrupted codeword. This notion can be extended to local *list* decoding by requiring the algorithm to output a list of descriptions of local decoders. Intuitively, a family of codes is *locally list decodable* in the presence of a specified type of corruptions if there exists an algorithm that, given oracle access to a corrupted codeword w, outputs a list of algorithms such that for each message x whose encoding sufficiently agrees with w, there is an algorithm in the list that, given oracle access to w, can simulate oracle access to x. In addition to the usual quantities studied in the literature on error-correcting codes (such as the fraction of corruptions a code can handle, its rate and efficiency of decoding), the important parameters in local decoding are the number of queries that the algorithms make to w and, in the case of local list decoding, list size.

The notion of locally decodable codes (LDCs) arose in the 1990s, motivated by numerous applications in complexity theory, such as program checking [43, 12, 21, 22], probabilistically checkable proofs [4, 3, 2, 47], derandomization [5, 52, 53], and private information retrieval [14]. Locally decodable codes that work in the presence of errors have been extensively studied [4, 12, 21, 22, 47, 6, 55, 19, 18, 7]. The related notion of locally list decodable codes (LLDCs) has also received a lot of attention [28, 52, 36, 7, 40, 38, 31, 29] and found applications in cryptography [28], learning theory [41], average-to-worst-case reductions [42, 13, 23], and hardness amplification and derandomization [5, 52]. The literature on decoding in the presence of erasures is too vast to survey here. *List* decoding in the presence of erasures (without the locality restriction) has been addressed by Guruswami [32] and Guruswami and Indyk [33]. In particular, Guruswami [32] constructed an asymptotically good family of binary linear codes that can be list decoded from an arbitrary fraction of erasures with lists of constant size. Even though decoding in the presence of erasures is an important and well established problem, to the best of our knowledge, local (unique and list) decoding from erasures has not been studied before.

Motivated by applications in property testing [27, 50], we begin our investigation of effects of erasures with local *list* decoding. Our first result is a local list *erasure-decoder*

for the Hadamard code. Local list decodability of the Hadamard code in the presence of errors is a famous result of Goldreich and Levin [28]. However, (local list) decoding of the Hadamard code is impossible when the fraction of errors reaches or exceeds 1/2. In contrast, we show that the Hadamard code is locally list decodable in the presence of any constant fraction of erasures in $[0, 1)$. Moreover, the list size and the query complexity for our decoder is better than for the Goldreich-Levin decoder: for our decoder, both quantities are inversely proportional to the fraction of input that has not been corrupted, whereas for the Goldreich-Levin decoder they are quadratically larger and are known to be optimal for that setting. Thus, our Hadamard decoder demonstrates that a square-root reduction in the list size and query complexity in local list decoding can be achieved for some settings of parameters when we move from errors to erasures.

The second thrust of our work, enabled by our local list decoding results, is investigating the effects of adversarial corruption to inputs on the complexity of sublinear-time algorithms. Understanding the relative difficulty of designing algorithms that work in the presence of input errors and in the presence of input erasures is a problem of fundamental importance. The motivation of investigating adversarial input corruption spurred the generalization of property testing, one of the most widely studied models of sublinear-time algorithms [24, 25, 48, 49, 26], to (error) tolerant testing [46] and erasure-resilient testing [17].

Erasure-resilient property testing falls between (standard) property testing and tolerant testing. Specifically, an erasure-resilient tester for a property, in the special case when no erasures occur, is a standard tester for this property. Also, a tolerant tester for a property implies the existence of an erasure-resilient tester with comparable parameters for the same property. Fischer and Fortnow [20] separated standard and tolerant testing by describing a property that is *easy* to test in the standard model and *hard* to test tolerantly. Dixit et al. [17] showed that the property defined by Fischer and Fortnow separates standard property testing from erasure-resilient testing in the same sense. Dixit et al. [17] asked whether it is possible to obtain a separation between erasure-resilient and tolerant testing.

In this work, we provide such a separation. Specifically, we describe a property of binary strings that is easy to test in the erasure-resilient model, but hard to test tolerantly.

The key idea in our construction of the separating property is to encode *sensitive regions* of strings (without which testing becomes hard) with an error correcting code. We need a code that exhibits a difference in its local list decoding capabilities for the same fraction of erasures and errors. Specifically, we want, for some constant α, q and L, a code that can be decoded from an α fraction of erasures with q queries and lists of size L, but cannot be decoded from an α fraction of errors. We first define a property where the sensitive regions are encoded with the Hadamard code and show that it is testable in the erasure-resilient model (with a constant number of queries), but is not testable tolerantly.

Next, we want to strengthen the separation to obtain a property that is testable with erasures, but requires as many queries as possible to test tolerantly. In our construction, the lower bound on the number of queries needed for tolerant testing is determined by the rate of the code. Since the Hadamard code has low rate, we only get a polylogarithmic lower bound on the query complexity of tolerant testing. To obtain a lower bound of $n^{\Omega(1)}$, we would need a code of polynomial rate. The question of whether there is a locally list erasure decodable code (with constant α, q and L) of polynomial rate remains open. An LLDC with such parameters is the holy grail of research on local decoding.

We circumvent the above difficulty by starting out with a property of binary strings that has a tester whose queries to a sensitive region of the input are *nearly uniformly* distributed. This implies that testing remains easy even if a constant fraction of the sensitive region is

corrupted. We construct a new separating property by encoding the sensitive region using a code that is *approximate locally list decodable* from erasures, where an approximate locally list decodable code (ALLDC) is defined identically to an LLDC except that the algorithms output by a decoder for such a code simulate oracle access to strings that are close to the original messages. We show that the resulting property can be erasure-resiliently tested using a constant number of queries but needs $n^{\Omega(1)}$ queries in order to be tested tolerantly, thus obtaining a strengthened separation.

Next, we study the general relationship between local decoding in the presence of errors and in the presence of erasures. One can observe that every LLDC that works in the presence of errors also works in the presence of twice as many erasures (with the same parameters up to constant factors). We ask if LLDCs or ALLDCs that work in the presence of erasures can have significantly smaller list sizes and query complexity than LLDCs or ALLDCs of the same rate that work in the presence of errors. We also prove that such a statement cannot hold for the case of local unique decoding: specifically, we show that if a code is locally unique erasure-decodable, then there exists another comparable code that is locally unique decodable (up to minor losses in parameters).

1.1 Model Definitions and Our Results

This section contains descriptions and definitions of the codes and property testing models we study, and also statements and discussion of our main results.

Local List Erasure-Decoding and the Hadamard Code

In this paper, we restrict our attention to binary codes. A binary code is an infinite family of maps $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$. The parameter n is called the message length, N is the block length, and n/N is the rate of the code. Corruptions in codewords can either be in the form of erasures (missing entries, denoted by the symbol $\perp$) or in the form of errors (wrong values from $\mathbb{F}_2$).

Recall that a local list decoder outputs a list of algorithms which give oracle access to decoded messages or, in other words *implicitly compute* the decoded messages. This, and the notion of local list erasure-decoders are formalized in the following definitions.

▶ **Definition 1.1** (Implicit Computation)**.** An algorithm A is said to implicitly compute $x \in \mathbb{F}_2^n$ if, for all $i \in [n]$, the algorithm A on input i, outputs the i^{th} bit of x.

▶ **Definition 1.2** (Locally List Erasure-Decodable Codes (LLEDCs))**.** A family of codes $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is (α, q, L)-*locally list erasure-decodable* if there exists a randomized algorithm A such that, for every $n \in \mathbb{N}$ and every $w \in (\mathbb{F}_2 \cup \{\perp\})^N$ with at most an α fraction of erasures, the algorithm A makes at most q queries to w and outputs a list of randomized algorithms $\{T_1, T_2, \ldots, T_L\}$ such that the following hold:

1. With probability at least $2/3$, for all $x \in \mathbb{F}_2^n$ such that $C_n(x)$ agrees with w on all nonerased bits, there exists an index $j \in [L]$ such that T_j with oracle access to w implicitly computes x.
2. For all $j \in [L]$ and $i \in [n]$, the expected number of queries that the algorithm T_j makes to w on input i is at most q.

The definition of an (α, q, L)-LLDC is identical to Definition 1.2 except that the input word has no erasures, and the list is required to contain, with probability at least $2/3$, algorithms that implicitly compute messages corresponding to codewords disagreeing with

the input word on at most an α fraction of bits. The celebrated Goldreich-Levin theorem [28] states that the Hadamard code, defined next, is an LLDC that has an efficient decoder.

▶ **Definition 1.3** (Hadamard code)**.** For $a \in \mathbb{F}_2^n$, let $H_a : \mathbb{F}_2^n \to \mathbb{F}_2$ be defined as follows: $H_a(x) = \bigoplus_{i\in[n]} a_i \cdot x_i$ for all $x \in \mathbb{F}_2^n$. The Hadamard code, denoted by $\{\mathcal{H}_n : \mathbb{F}_2^n \to \mathbb{F}_2^{2^n}\}_{n\in\mathbb{N}}$, is such that for $a \in \mathbb{F}_2^n$, the encoding $\mathcal{H}_n(a)$ is the string of evaluations of H_a over $\mathbb{F}_2^n$.

Our first result is about the local list erasure-decodability of the Hadamard code. It is an analogue of the Goldeich-Levin Theorem [28] for corruptions in the form of erasures.

▶ **Theorem 1.4** (Local List Erasure-Decoder for Hadamard)**.** *There is an* $\left(\alpha, \Theta(\frac{1}{1-\alpha}), \Theta(\frac{1}{1-\alpha})\right)$*-local list erasure-decoder for the Hadamard code that works for every* $\alpha \in [0, 1)$*.*

The Goldreich-Levin theorem holds for any fraction of errors in $[0, 1/2)$. In contrast, our local list erasure-decoder works for any fraction of erasures less than 1. However, it is impossible to decode the Hadamard code in the presence of $1/2$ fraction of errors because every Hadamard codeword has relative distance at most $1/2$ from the all-zero codeword. Another improvement in Theorem 1.4 as compared to Golreich-Levin is in the list size and the query complexity: from $\Theta(\frac{1}{(1/2-\alpha)^2})$ to $\Theta(\frac{1}{1-\alpha})$. Such an improvement is impossible if we are decoding against errors as opposed to erasures. Specifically, for the list size, Blinovsky [11] and Guruswami and Vadhan [35] show that every list decoder for every binary code that is list decodable in the presence of an α fraction of errors must output lists of size $\Omega(\frac{1}{(1/2-\alpha)^2})$. Grinberg, Shaltiel, and Viola [30] show that the same lower bound holds for query complexity.

Observation 4.4 states that every (α, q, L)-LLDC is also an $(2\alpha, 4q, 4L)$-LLEDC. By combining this observation with the Goldreich-Levin theorem, one can obtain a local list erasure-decoder for the Hadamard code that works for every $\alpha \in [0, 1)$ and has list size and query complexity $\Theta(\frac{1}{(1-\alpha)^2})$. However, we obtain strictly better list size and query complexity in Theorem 1.4.

Separation between Erasure-Resilient and Tolerant Testing

We first describe the erasure-resilient and tolerant models of testing. A *property* $\mathcal{P}$ is a set of strings. Given $\alpha \in [0, 1)$, a string is α*-erased* if at most an α fraction of its values are erasures (denoted by $\perp$). A *completion* of an α-erased string $x \in \{0, 1, \perp\}^n$ is a string $y \in \{0, 1\}^n$ that agrees with x on all the positions where x is nonerased. An α*-erasure-resilient* ε*-tester* [17] for a property $\mathcal{P}$ is a randomized algorithm that, given parameters $\alpha \in [0, 1), \varepsilon \in (0, 1)$ and oracle access to an α-erased string x, accepts with probability at least $2/3$ if x has a completion in $\mathcal{P}$ and rejects with probability at least $2/3$ if, in every completion of x, at least an ε fraction of the nonerased values has to be changed to get a string in $\mathcal{P}$. The property $\mathcal{P}$ is α*-erasure-resiliently* ε*-testable* if there exists an α-erasure-resilient ε-tester for $\mathcal{P}$ with query complexity that depends only on the parameters α and ε (but not on n).

A string $x \in \{0, 1\}^n$ is ε'-far (α-close) from (to, respectively) a property $\mathcal{P}$, if the normalized Hamming distance of x from $\mathcal{P}$ is at least ε' (at most α, respectively). An (α, ε')*-tolerant tester* [46] for $\mathcal{P}$ is a randomized algorithm that, given parameters $\alpha \in (0, 1), \varepsilon' \in (\alpha, 1)$ and oracle access to a string x, accepts with probability at least $\frac{2}{3}$ if x is α-close to $\mathcal{P}$ and rejects with probability at least $\frac{2}{3}$ if x is ε'-far from $\mathcal{P}$. The property $\mathcal{P}$ is (α, ε')*-tolerantly testable* if there exists an (α, ε')-tolerant tester for $\mathcal{P}$ with query complexity that depends only on α and ε' (but not n).

Comparison of parameters. We remark that, while comparing the two models, it is appropriate to compare $(\alpha, \alpha + \varepsilon(1 - \alpha))$-tolerant testing of a property $\mathcal{P}$ with α-erasure-resilient ε-testing of $\mathcal{P}$ for the same values of α and ε. The parameter α in both models is

an upper bound on the fraction of corruptions (erasures, or errors) that an adversary can make to an input. An α-erasure-resilient ε-tester rejects with probability at least $\frac{2}{3}$ if, for for every way of completing an input string, one needs to change at least an ε fraction of the remaining part of the input to make it satisfy $\mathcal{P}$. Similarly, an $(\alpha, \alpha+\varepsilon(1-\alpha))$-tolerant tester rejects with probability at least $\frac{2}{3}$ if, for every way of *correcting* an α fraction of the input values, one needs to change at least an ε fraction of the remaining $(1-\alpha)$ fraction of the input to make it satisfy $\mathcal{P}$.

Separation. The following theorem states that there exists a property that is erasure-resiliently testable but is not tolerantly testable. This proves that tolerant testing is, in general, harder problem than erasure-resilient testing.

▶ **Theorem 1.5** (Separation). *There exists a property $\mathcal{P}$ and constants $\varepsilon, \alpha \in (0,1)$ such that*
- *$\mathcal{P}$ is α-erasure-resiliently ε-testable;*
- *$\mathcal{P}$ is not $(\alpha, \alpha+\varepsilon(1-\alpha))$-tolerantly testable.*

Approximate Local List Erasure-Decoding and Strengthened Separation

We obtain a separation better than in Theorem 1.5 with the help of a variant of LLEDCs, called approximate locally list erasure-decodable codes (ALLEDC). An approximate local list erasure-decoder is identical to a local list erasure-decoder in all aspects except that the algorithms in its list are required to implicitly compute strings that are just "close" to the actual messages. More formally, (α, β, q, L)-ALLEDCs are defined as (α, q, L)-LLEDCs in Definition 1.2, except that we replace "implicitly computes x" at the end of Item 1 with "implicitly computes a string $x' \in \mathbb{F}_2^n$ that is β-close to x".

The definition of an (α, β, q, L)-approximate locally list decodable code (ALLDC) is identical to that of an (α, β, q, L)-ALLEDC except that the input word has no erasures, and the list is required to contain, with probability at least 2/3, algorithms that implicitly compute strings that are β-close to messages corresponding to codewords which are α-close to the input word.

We observe (Observation 4.2) that every (α, β, q, L)-ALLDC is also a $(2\alpha, \beta, 4q, 4L)$-ALLEDC, and combine this observation with existing constructions for ALLDCs [37, 8] to obtain efficient ALLEDCs. We use them and get our strengthened separation.

▶ **Theorem 1.6** (Strengthened Separation). *There exists a property $\mathcal{P}'$ and constants $\varepsilon, \alpha \in (0,1)$ such that*
- *$\mathcal{P}'$ is α-erasure-resiliently ε-testable;*
- *every $(\alpha, \alpha+\varepsilon(1-\alpha))$-tolerant tester for $\mathcal{P}'$ makes $n^{\Omega(1)}$ queries.*

Relationship between Local Erasure-Decoding and Local Decoding

We investigate the general relationship between the erasures and errors in the context of local unique and list decoding. We show that local (unique) decoding from erasures implies local (unique) decoding from errors, up to some loss in parameters.

▶ **Definition 1.7** (Locally Erasure-Decodable Codes (LEDCs)). A code family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is (α, q)-*locally erasure-decodable* if there exists an algorithm A that, given an index $i \in [n]$ and oracle access to an input word $w \in (\{\perp\} \cup \mathbb{F}_2)^N$ with at most α fraction of erasures, makes at most q queries to w and outputs x_i with probability at least $\frac{2}{3}$.

An (α, q)-locally decodable code (LDC) is defined similarly to an (α, q)-LEDC except that the input word w contains at most α fraction of errors instead of erasures. We observe

(Observation 6.4) that an LDC is also locally erasure-decodable from (nearly) twice as many erasures. We also show that constant-query LEDCs are constant-query locally decodable (up to constant loss in parameters).

▶ **Theorem 1.8.** *For every $\alpha \in [0,1)$, if a code family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n \in N}$ is (α, q)-locally erasure-decodable, then it is $(\frac{\alpha}{O(q^2 \cdot 81^q)}, O(q \cdot 9^q))$ locally decodable.*

We note that although our final code has small decoding radius (that is, it tolerates only a small fraction of errors), the decoding radius can be amplified to any constant arbitrarily close to $1/4$ at the cost of increasing the query complexity and encoding length by a constant factor. Specifically, using a local version of the AEL transformation [1] (see [39, Lemma 3.1]), one can amplify the decoding radius to any constant arbitrarily close to $1/2$ at the cost of increasing the query complexity, alphabet size, and length by constant factors. The alphabet then can be reduced back to binary by encoding the binary representation of each alphabet symbol with the Hadamard code. The length will grow by another constant factor, and using a local version of the GMD decoder [39, Corollary 3.9], one can show that final decoding radius is arbitrarily close to $1/4$ and query complexity grows only by a constant factor.

1.2 Open Questions

The main open question raised by our work is whether local list decoding is significantly easier in terms of the query complexity, the list size, or the rate of codes when corruptions are in the form of erasures. The same question can be asked about approximate local list decoding. Our local list erasure-decoder for the Hadamard code shows that there is some advantage for having erasures over errors, in terms of the list size and query complexity, for some settings of parameters. A positive or negative answer to this question, combined with our result on the equivalence of errors and erasures in the local decoding regime, will enhance the understanding of whether local list decoding is an inherently more powerful model when compared to local decoding.

We remark that our proof that the existence of a locally decodable code that works in the presence of erasures implies the existence of a locally decodable code that works in the presence of errors and has related parameters does not directly extend to the setting of local list decoding. However, it can be extended with an additional assumption that the output lists contain only valid algorithms (those that correspond to the original messages). We discuss this in more detail in Section 6. This raises the question about the power of this assumption.

2 Local List Erasure-Decoding of the Hadamard Code

In this section, we describe a local list erasure-decoder for the Hadamard code and prove Theorem 1.4. We follow the style of the proof of the Goldreich-Levin theorem given in a tutorial by Luca Trevisan [54] on the applications of coding theory to complexity.

Proof of Theorem 1.4. For $b_1, b_2 \in \mathbb{F}_2$, let $b_1 \oplus b_2$ denote the XOR of b_1 and b_2. For vectors $x, y \in \mathbb{F}_2^n$, let $x \odot y$ denote the bitwise XOR of x and y. Let $e_k \in \mathbb{F}_2^n$ denote the k^{th} standard basis vector. A codeword of the Hadamard code $\mathcal{H}_n$ (see Definition 1.3) is the string of all evaluations of a linear function mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$. A function $f : \mathbb{F}_2^n \to \mathbb{F}_2 \cup \{\perp\}$ is α-erased, if f evaluates to $\perp$ on at most α fraction of its domain. Our local list erasure-decoder, described in Algorithm 1, gets a parameter $\alpha \in [0,1)$ as its input and has oracle access to an

Algorithm 1 Local List Erasure-Decoder for the Hadamard code

Input: $\alpha \in [0,1)$; oracle access to α-erased linear function $f : \mathbb{F}_2^n \to \mathbb{F}_2 \cup \{\perp\}$

1: Let $t = \lceil \log_2(1 + \frac{12}{1-\alpha}) \rceil$.
2: Choose $z_1, z_2, \ldots, z_t \in \mathbb{F}_2^n$ uniformly and independently at random.
3: Let $z_S \leftarrow \bigodot_{i \in S} z_i$ for all nonempty $S \subseteq [t]$. Let $z_\phi \leftarrow \vec{0}$.
4: Set $B \leftarrow \{i \in [t] : f(z_i) = \perp\}$.
5: **for** all $b_1, b_2, \ldots, b_{|B|} \in \{0,1\}$ **do define**
6: ▷ Description of the local decoder $T_{b_1,\ldots,b_{|B|}}$ follows.
7: **function** $A_{b_1,\ldots,b_{|B|}}$
8: **input:** $x \in \mathbb{F}_2^n$; oracle access to $f : \mathbb{F}_2^n \to \mathbb{F}_2 \cup \{\perp\}$
9: **for** all $S \subseteq [t]$ **do**
10: **if** $f(x \odot z_S) \neq \perp$ **then return** $(\bigoplus_{j \in S \cap B} b_j) \oplus (\bigoplus_{j \in S \cap ([t] \setminus B)} f(z_j)) \oplus f(x \odot z_S)$.
11: **Return** $\perp$.
12: **function** $T_{b_1,\ldots,b_{|B|}}$
13: **input:** $k \in [n]$; oracle access to $f : \mathbb{F}_2^n \to \mathbb{F}_2 \cup \{\perp\}$
14: **repeat**
15: Pick $y \in \mathbb{F}_2^n$ uniformly and independently at random.
16: $u \leftarrow A_{b_1,\ldots,b_{|B|}}(y \odot e_k)$, $v \leftarrow A_{b_1,\ldots,b_{|B|}}(y)$.
17: **if** $v \neq \perp$ and $u \neq \perp$ **then return** $u \oplus v$.
18: **Return** the descriptions of $T_{b_1,\ldots,b_{|B|}}$ for all $b_1, b_2, \ldots, b_{|B|} \in \{0,1\}$.

α-erased linear function $f : \mathbb{F}_2^n \to \mathbb{F}_2 \cup \{\perp\}$ (or, equivalently, oracle access to an α-erased codeword of $\mathcal{H}_n$) .

Let $t = \lceil \log_2(1+\frac{12}{1-\alpha}) \rceil$. Consider $z_1, z_2, \ldots, z_t \in \mathbb{F}_2^n$ sampled uniformly and independently at random. For a nonempty set $S \subseteq [t]$, let z_S denote $\bigodot_{i \in S} z_i$. Let z_ϕ denote $\vec{0}$. For any two nonempty sets $R, S \subseteq [t]$ such that $R \neq S$, the vectors z_R and z_S are independently and uniformly distributed in $\mathbb{F}_2^n$. Recall that for a string $a \in \mathbb{F}_2^n$, let $H_a : \mathbb{F}_2^n \to \{0,1\}$ denotes the Hadamard encoding (see Definition 1.3) of a.

Let $a \in \mathbb{F}_2^n$ be such that for all $x \in \mathbb{F}_2^n$ where $f(x) \neq \perp$, the functions H_a and f agree with each other. There exists some iteration of Step 5 of Algorithm 1 such that $b_i = H_a(z_i)$ for all $i \in B$. Let T and A denote the algorithms whose descriptions are generated in Steps 12 and 7 of this iteration respectively.

First, we show that for x distributed uniformly in $\mathbb{F}_2^n$, the algorithm A on input x, returns $H_a(x)$ with probability at least $\frac{2}{3}$. Fix $x \in \mathbb{F}_2^n$. Consider a set $S \subseteq [t]$ such that $f(x \odot z_S) \neq \perp$. According to the description of A,

$$\begin{aligned} A(x) &= (\oplus_{j \in S \cap B} b_j) \oplus \left(\oplus_{j \in S \cap ([t] \setminus B)} f(z_j)\right) \oplus f(x \odot z_S) \\ &= (\oplus_{j \in S \cap B} H_a(z_j)) \oplus \left(\oplus_{j \in S \cap ([t] \setminus B)} H_a(z_j)\right) \oplus H_a(x \odot z_S) \\ &= (\oplus_{j \in S} H_a(z_j)) \oplus H_a(x) \oplus (\oplus_{j \in S} H_a(z_j)) = H_a(x). \end{aligned}$$

Let $\alpha^\star \leq \alpha$ denote the fraction of erasures in f. For each $S \subseteq [t]$ and $x \in \mathbb{F}_2^n$, we have that $f(x \odot z_S) \neq \perp$ with probability equal to $1 - \alpha^\star$, since $x \odot z_S$ is uniformly distributed in $\mathbb{F}_2^n$. Define an indicator random variable $Z_S = \mathbb{1}(f(x \odot Z_S) \neq \perp)$. Then $\mathbb{E}[Z_S] = 1 - \alpha^\star$ and $\mathrm{Var}(Z_S) = (1 - \alpha^\star) \cdot \alpha^\star$. Note that the collection $\{x \odot z_S | S \subseteq [t], S \neq \emptyset\}$ is pairwise independent, and hence the collection $\{Z_S | S \subseteq [t], S \neq \emptyset\}$ is also pairwise independent.

Let $Z = \sum_{S \subseteq [t] : S \neq \emptyset} Z_S$. The random variable Z denotes the number of nonerased values among $f(x \odot z_S)$ over all nonempty $S \subseteq [t]$. The event that $\forall S \subseteq [t], S \neq \emptyset, f(x \odot z_S) = \perp$ is

equivalent to the event that $Z < 1$. Also,

$$\begin{aligned}\mathbb{E}[Z] &= \sum_{S\subseteq[t],S\neq\emptyset} \mathbb{E}[Z_S] = (1-\alpha^\star)\cdot(2^t-1);\\ \mathrm{Var}[Z] &= \sum_{S\subseteq[t],S\neq\emptyset} \mathrm{Var}[Z_S] = (2^t-1)\cdot\alpha^\star(1-\alpha^\star).\end{aligned}$$

By Chebyshev's inequality,

$$\begin{aligned}\Pr[Z<1] &= \Pr[\mathbb{E}[Z]-Z>\mathbb{E}[Z]-1]\\ &\le \Pr[\mathbb{E}[Z]-Z\ge(1-\alpha^\star)\cdot(2^t-1)-1]\\ &\le \Pr\left[\mathbb{E}[Z]-Z>\frac{(1-\alpha^\star)\cdot(2^t-1)}{2}\right]\le\frac{4\mathrm{Var}(Z)}{(1-\alpha^\star)^2\cdot(2^t-1)^2}\\ &\le \frac{4\alpha^\star}{(1-\alpha^\star)\cdot(2^t-1)}\le\frac{4\alpha}{(1-\alpha)\cdot(2^t-1)}\le\frac{1}{3}.\end{aligned}$$

The last inequality follows from our setting of t. Therefore, for x distributed uniformly in $\mathbb{F}_2^n$, the algorithm A on input x, returns $H_a(x)$ with probability at least $\frac{2}{3}$.

We now prove that T implicitly computes $a\in\mathbb{F}_2^n$ and that the expected number of queries that it makes to f is $\Theta(\frac{1}{1-\alpha})$. It is clear that the output of T on input k is always $a[k]=H_a(y\odot e_k)\oplus H_a(y)=H_a(e_k)$. The number of queries made by T to A is a geometric random variable with success probability $\ge\frac{1}{3}$. Hence, the expected number of queries made by T to A is at most 3. Since the query complexity of A is at most 2^t, the expected number of queries made to f in one invocation of T is $\Theta(2^t)$, that is, $\Theta(\frac{1}{1-\alpha})$. The number of algorithms whose descriptions are generated is also at most 2^t, which is, $\Theta(\frac{1}{1-\alpha})$. ◀

3 Separation

In this section, we describe a property $\mathcal{P}$ that is erasure-resiliently testable using a constant number of queries, but not tolerantly testable using a constant number of queries, and prove Theorem 1.5. In fact, we prove the following (more general) statement and show that it implies Theorem 1.5.

▶ **Theorem 3.1.** *Let $\varepsilon^\star\in(0,\frac{1}{100})$ be a constant. There exists a property $\mathcal{P}\subseteq\{0,1\}^*$ such that*

- *for every $\alpha\in[0,\frac{3\varepsilon^\star}{16})$ and $\varepsilon\in(\frac{3\varepsilon^\star}{4(1-\alpha)},1)$, the property $\mathcal{P}$ can be α-erasure-resiliently ε-tested using $O(\frac{1}{\varepsilon(1-2\alpha)})$ queries.*
- *for all $\alpha\in(\frac{\varepsilon^\star}{8},1)$ and $\varepsilon'\in(\alpha,\varepsilon^\star-\frac{(\varepsilon^\star)^2}{4})$, the query complexity of (α,ε')-tolerant testing $\mathcal{P}$ on inputs of length N is $\tilde{\Omega}(\log N)$.*

3.1 Description of the Separating Property $\mathcal{P}$

The property $\mathcal{P}$ is defined in terms of a property $\mathcal{R}$ that is hard to test in the standard property testing model [27, 50], a probabilistically checkable proof system (PCP of proximity [9, 16][4]) for the problem of testing $\mathcal{R}$, and the Hadamard code. We discuss them below. The idea of using PCPs of proximity in separating property testing models comes from the work of

[4] PCPs of proximity are referred to as assignment testers by Dinur and Reingold [16]. Ben-Sasson et al. [9] and Dinur and Reingold [16] defined these objects concurrently and independently in order to obtain simpler and more efficient PCP constructions.

Fischer and Fortnow [20]. Our contribution is to use locally list decodable codes in this context.

Given a Boolean formula ϕ over n variables, let $\mathcal{R}_\phi \subseteq \{0,1\}^n$ denote the set of all satisfying assignments to ϕ, represented as n-bit strings. Ben-Sasson, Harsha and Raskhodnikova [10] showed that for infinitely many $n \in \mathbb{N}$, there exists a 3CNF formula ϕ_n on n variables such that every tester for $\mathcal{R}_{\phi_n}$ requires $\Omega(n)$ queries.

▶ **Lemma 3.2** ([10]). *There exists a parameter $\varepsilon^\star \in (0,1)$ and a countably infinite set $\aleph \subseteq \mathbb{N}$ such that for all $n \in \aleph$, there exists a 3CNF formula ϕ_n with n variables and $\Theta(n)$ clauses such that every $\varepsilon^\star$-tester for $\mathcal{R}_{\phi_n}$ has query complexity $\Omega(n)$.*

As mentioned before, another important ingredient in the description of the separating property $\mathcal{P}$ is a probabilistically checkable proof system for property testing problems, called PCP of proximity, defined and studied independently by Ben-Sasson et al. [9] and Dinur and Reingold [16]. PCPs of proximity were further studied by Dinur [15] and Meir [44, 45].

▶ **Definition 3.3** (PCP of proximity [9, 16]). Given a a property $P_n \subseteq \{0,1\}^n$, the PCP of proximity *(PCPP)* for P_n is a randomized algorithm V that takes a parameter $\varepsilon \in (0,1]$ as input, gets oracle access to a string $y \circ \pi$, where $y \in \{0,1\}^n$ is the input and $\pi \in \{0,1\}^m$ is the proof, and satisfies the following:

- if $y \in P_n$, then, for some π, the algorithm V always accepts $y \circ \pi$;
- if y is ε-far from P_n, then, for every π, the algorithm V rejects $y \circ \pi$ with probability at least $\frac{2}{3}$.

A result by Dinur [15, Corollary 8.4] states that there are efficient PCPPs (over a small constant alphabet Σ) for testing properties (over Σ) that are decidable using polynomial-sized circuits. By representing the symbols in Σ using the binary alphabet, we obtain the following lemma.

▶ **Lemma 3.4** ([15]). *If $P_n \subseteq \{0,1\}^n$ is a property decidable by a circuit of size $s(n)$, then there exists a* PCPP *V that works for every $\varepsilon \in (0,1]$, uses a proof of length at most $s(n) \cdot \text{polylog } s(n)$, and has query complexity $O(\frac{1}{\varepsilon})$. Moreover, the queries of V are nonadaptive.*

Claim 3.5 uses Lemma 3.4 in conjunction with the fact that the property $\mathcal{R} = \{\mathcal{R}_{\phi_n}\}_{n \in \aleph}$ can be decided using linear-sized circuits.

▶ Claim 3.5. There exists a constant $c > 0$ such that for every large enough $n \in \mathbb{N}$, there exists a *PCPP* V for the property $\mathcal{R}_{\phi_n}$ that works for all $\varepsilon \in (0,1]$, uses a proof of length at most $cn \cdot \textit{polylog } n$, and has query complexity $O(\frac{1}{\varepsilon})$.

Proof. One can observe that for all $n \in \aleph$, the circuit complexity of deciding $\mathcal{R}_{\phi_n}$ (described in Lemma 3.2) is $O(n)$. In other words, there exists a c'' such that for every large enough n, the property $\mathcal{R}_{\phi_n}$ can be decided using a circuit of size at most $c''n$. The claim follows by plugging this fact into Lemma 3.4. ◀

The following is the definition of our separating property $\mathcal{P}$. At a high level, the definition says that, for all $n \in \aleph$, a string of length $O(2^{n \cdot \text{polylog } n})$ satisfies $\mathcal{P}$ if its first part is the repetition of a string y satisfying $\mathcal{R}$, and the second part is the encoding (by the Hadamard code) of y concatenated with a proof π that makes the algorithm V in Claim 3.5 accept.

▶ **Definition 3.6** (Separating Property $\mathcal{P}$). Let $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2. For $n \in \aleph$, let $p(n) \leq cn \cdot \textit{polylog } n$ denote the length of proof that the algorithm V in Claim 3.5 has

oracle access to. A string $x \in \{0,1\}^N$ of length $N = \frac{4}{\varepsilon^\star} \cdot 2^{n+p(n)}$ satisfies $\mathcal{P}$ if the following conditions hold:

1. The first $(\frac{4}{\varepsilon^\star} - 1) \cdot 2^{n+p(n)}$ bits of x (called the *plain part* of x) consist of $(\frac{4}{\varepsilon^\star} - 1) \cdot \frac{2^{n+p(n)}}{n}$ repetitions of a string $y \in \mathcal{R}_{\phi_n}$ of length n, for ϕ_n from Lemma 3.2.
2. The remaining bits of x (called the *encoded part* of x) form the Hadamard encoding of a string $y \circ \pi(y)$ of length $n + p(n)$, where $\circ$ denotes the concatenation operation on strings. The string $y \in \{0,1\}^n$ is the same as the one in the description of the plain part. The string $\pi(y) \in \{0,1\}^{p(n)}$ is a proof such that the algorithm V (from Claim 3.5) accepts when given oracle access to y and $\pi(y)$.

3.2 Proof of Theorem 3.1

In this section, we prove Theorem 3.1, which in turn implies Theorem 1.5. Lemmas 3.7 and 3.10 prove the first and second parts of Theorem 3.1, respectively.

We first give a high level overview of the proof. The erasure-resilient tester for $\mathcal{P}$ first obtains a list of (implicit) decodings of the encoded part (see Definition 3.6) of an input string $x \in \{0,1\}^N$ using the local list erasure-decoder guaranteed by Theorem 1.4. If $x \in \mathcal{P}$, with high probability, at least one of the algorithms implicitly computes (see Definition 1.1) the string $y \circ \pi(y)$, where y is such that the plain part of x (see Definition 3.6) consists of repetitions of y, and $\pi(y)$ is a proof string such that the algorithm V (from Claim 3.5) accepts upon oracle access to $y \circ \pi(y)$. In case x is ε-far from $\mathcal{P}$ we show that for every algorithm T output by the local list erasure-decoder, the string $y' \circ \pi(y')$ implicitly computed by T is such that, (1) either the plain part of x is far from being the repetitions of y', (2) or y' is far from $\mathcal{R}$ (in which case, the algorithm V from Claim 3.5 rejects when given oracle access to $y' \circ \pi(y')$).

To show that tolerant testing of $\mathcal{P}$ is hard, we reduce $\varepsilon^\star$-testing of $\mathcal{R}_{\phi_n}$ to it. Specifically, given oracle access to a string $y \in \{0,1\}^n$ that we want to $\varepsilon^\star$-test, we simulate oracle access to a string $x \in \{0,1\}^N$ such that the plain part of x consists of repetitions of y, and every bit in the encoded part of x is 0. Since every Hadamard codeword has an equal number of 0s and 1s, the string x can be thought of as having 0.5 fraction of "errors" in the encoded part. If $y \in \mathcal{R}_{\phi_n}$, then the string x is close to being in $\mathcal{P}$, as the errors are only in the encoded part of x and the length of the encoded part is a small fraction of the length of x. If y is far from $\mathcal{R}_{\phi_n}$, then x is also far from $\mathcal{P}$, since the plain part of x, whose length is a large fraction of the length of x, is the repetitions of y. Thus, the decision of a tolerant tester for $\mathcal{P}$ on x can be used to test y for $\mathcal{R}_{\phi_n}$, implying that the complexity of tolerant testing of $\mathcal{P}$ is equal to the complexity of testing $\mathcal{R}_{\phi_n}$.

We now prove the existence of an efficient erasure-resilient tester for $\mathcal{P}$. An α-erased string x is ε-far from a property $\mathcal{P}$ if there is no way to complete x to a string that satisfies $\mathcal{P}$ without changing at least an ε fraction of the nonerased values in x.

▶ **Lemma 3.7.** *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2. For every $\alpha \in [0, \frac{3\varepsilon^\star}{16})$, and every $\varepsilon \in (\frac{3\varepsilon^\star}{4(1-\alpha)}, 1)$, the property $\mathcal{P}$ can be α-erasure-resiliently ε-tested using $O(\frac{1}{\varepsilon(1-2\alpha)})$ queries.*

Proof. The erasure-resilient tester for $\mathcal{P}$ is described in Algorithm 2. The query complexity of the tester is evident from its description. We now prove that the tester, with probability at least $\frac{2}{3}$, accepts strings in $\mathcal{P}$ and rejects strings that are ε-far from $\mathcal{P}$.

Let $\aleph, \varepsilon^\star \in (0,1)$ be as in Lemma 3.2. Fix $n \in \aleph$ and let $p(n)$ and N be as in Definition 3.6. Let s denote $(\frac{4}{\varepsilon^\star} - 1) \cdot \frac{2^{n+p(n)}}{n}$. Consider a string $x \in \{0,1\}^N$ that we want to erasure-resiliently test for $\mathcal{P}$. As in Definition 3.6, we refer to the substring $x[1 \dots sn]$ as the plain part of x and the substring $x[sn+1 \dots N]$ as the encoded part of x.

Algorithm 2 Erasure-resilient tester for separating property $\mathcal{P}$

Input: $\alpha, \varepsilon \in (0,1), N = \frac{4}{\varepsilon^\star} \cdot 2^{(n+p(n))}$; oracle access to $x \in \{0,1,\perp\}^N$

1: Set $s \leftarrow (\frac{4}{\varepsilon^\star} - 1) \cdot \frac{2^{(n+p(n))}}{n}$, $\varepsilon' \leftarrow \frac{\varepsilon(1-2\alpha)}{3}$.
2: Set $Q \leftarrow \frac{C}{\varepsilon(1-2\alpha)}$ for a large enough constant C.
3: **Accept** whenever the number of queries exceeds Q.
4: Let $T_1, T_2, \ldots, T_L$ be the list of algorithms returned by a $(\frac{3}{4}, q, L)$-local list erasure-decoder for the Hadamard code (Algorithm 1), given oracle access to $x[sn+1..N]$, the encoded part of x.
5: **for** each $k \in [L]$ **do**
6: ▷ Check if the plain part of x is the repetition of y, where y denotes the first n bits of the decoding (given by T_k) of the encoded part of x.
7: **repeat** $\left\lceil \frac{9 \log L}{\varepsilon(1-2\alpha)} \right\rceil$ times:
8: Pick $a \in_R [n], i \in_R [s]$.
9: **if** $x[(i-1)n+a] \neq \perp$ and $T_k(a) \neq x[(i-1)n+a]$ **then**
10: **Discard** the current k
11: ▷ Check if the string $y \in \mathcal{R}_{\phi_n}$, where y denotes the first n bits of the decoding (by T_k) of the encoded part of x.
12: **repeat** $\lceil 4 \log L \rceil$ times:
13: Run V, from Claim 3.5, with input ε' and oracle access to T_k.
14: **Discard** the current k if V rejects.
15: **Reject** if every $k \in [L]$ is **discarded**; otherwise, **accept**.

Assume that $x \in \mathcal{P}$. Since $\alpha < 3\varepsilon^\star/16$, the fraction of erasures in the encoded part of x is at most $3/4$. Hence, by Theorem 1.4, with probability at least $2/3$, there exists an algorithm T_k computed in Step 4 of Algorithm 2, such that T_k implicitly computes the string $y \circ \pi \in \{0,1\}^{n+p(n)}$, where $y \in \mathcal{R}_{\phi_n}$, the plain part of x can be completed to a repetition of y, and π is a proof such that the algorithm V (from Claim 3.5) accepts when given oracle access to $y \circ \pi$. Therefore k is not discarded in either Step 10 or Step 14. Thus, the tester will accept with probability at least $2/3$.

Now, assume that x is ε-far from $\mathcal{P}$. Let E denote the event that the number of queries made by the tester does not exceed its query budget. We will first show that, conditioned on E, the tester rejects x with probability at least $4/5$.

Let $\mathcal{N}_1$ denote the set of nonerased points in the plain part of x and $\mathcal{N}_2$ denote those in the encoded part and let $\mathcal{N}$ denote the set of nonerased points in x. Even if all of at most $N\alpha$ erased points in x are in the plain part of x, the total number of nonerased points in plain part of x, which is $|\mathcal{N}_1|$, is at least $sn - N\alpha$. Since, for large enough n, we have $N \leq 2sn$ by Definition 3.6, we can see that $|\mathcal{N}_1| \geq sn(1-2\alpha)$. We first prove two claims about the plain part of x, that is, $x[1 \ldots sn]$.

▶ Claim 3.8. The plain part of x is $\frac{2\varepsilon}{3}$-far from being s repetitions of a string $y \in \mathcal{R}_{\phi_n}$.

Proof. The fraction of nonerased points in $x[sn+1 \ldots N]$, the encoded part of x, is at most $\frac{N-sn}{|\mathcal{N}|} \leq \frac{N-sn}{N(1-\alpha)} = \frac{\varepsilon^\star}{4(1-\alpha)}$, which is at most $\varepsilon/3$ since $\varepsilon \in (\frac{3\varepsilon^\star}{4(1-\alpha)}, 1)$. Thus, the plain part of x is $\frac{2\varepsilon}{3}$-far from being s repetitions of a string $y \in \{0,1\}^n$ that satisfies $\mathcal{R}_{\phi_n}$. ◀

From Claim 3.8, it follows that at least $\frac{2\varepsilon \cdot |\mathcal{N}_1|}{3} \geq \frac{2\varepsilon \cdot sn(1-2\alpha)}{3}$ nonerased points need to be changed in the plain part of x for it to be s repetitions of a string $y \in \mathcal{R}_{\phi_n}$.

▶ Claim 3.9. For every $y \in \{0,1\}^n$, if the plain part of x can be changed to s repetitions of y by modifying less than $\frac{\varepsilon \cdot sn(1-2\alpha)}{3}$ nonerased values, then y is $\frac{\varepsilon(1-2\alpha)}{3}$-far from $\mathcal{R}_{\phi_n}$.

Proof. Consider $y \in \{0,1\}^n$ such that we can change less than $\varepsilon \cdot sn(1-2\alpha)/3$ nonerased points in the plain part of x and make it s repetitions of y. Assume that there exists $y' \in \mathcal{R}_{\phi_n}$ such that the Hamming distance of y' to y is at most $\varepsilon \cdot n(1-2\alpha)/3$. Then, the plain part of x, can be changed to being s repetitions of y' by first changing it to be s repetitions of y (modifying less than $\varepsilon \cdot sn(1-2\alpha)/3$ nonerased points) and then modifying at most $s \cdot \varepsilon \cdot n(1-2\alpha)/3$ nonerased points to make it s repetitions of y'. In other words, $x[1 \dots sn]$ can be modified in less than $2\varepsilon \cdot sn(1-2\alpha)/3$ nonerased points to make it s repetitions of a string y' in $\mathcal{R}_{\phi_n}$. This contradicts Claim 3.8. ◀

Fix $k \in [L]$. Let $y' \in \{0,1\}^n$ be the first n bits from the left in the decoding, using T_k, of the encoded part of x. We will show that the algorithm discards k with high probability. We split the analysis into two cases.

Case I: Suppose we need to change at least $\frac{\varepsilon|\mathcal{N}_1|}{3} \geq \frac{\varepsilon \cdot sn(1-2\alpha)}{3}$ nonerased points in the plain part of x for it to become s repetitions of y'. We show that in this case, Steps 7-10 discard k with probability at least $\frac{9}{10L}$. A point $(i-1)n+a$ for $i \in [s]$ and $a \in [n]$ is called a witness if $x[(i-1)n+a] \neq \perp$ and $x[(i-1)n+a] \neq y'[a]$. Since we need to change at least $\varepsilon \cdot sn(1-2\alpha)/3$ nonerased points in the plain part of x for it to become s repetitions of y', there are at least $\varepsilon \cdot sn(1-2\alpha)/3$ witnesses in the plain part of x. In each iteration of Steps 7-10, the point selected is a witness with probability at least $\frac{\varepsilon \cdot sn(1-2\alpha)}{3sn} = \frac{\varepsilon \cdot (1-2\alpha)}{3}$. Thus, in $\lceil \frac{9 \log L}{\varepsilon(1-2\alpha)} \rceil$ iterations, Algorithm 2 finds a witness (and discards k) with probability at least $9/10L$.

Case II: In this case, we assume that we can change less than $\varepsilon \cdot sn(1-2\alpha)/3$ nonerased points in the plain part of x and make it s repetitions of y'. Then, by Claim 3.9, y' is $\varepsilon \cdot (1-2\alpha)/3$-far from $\mathcal{R}_{\phi_n}$. Let $\varepsilon' = \frac{\varepsilon \cdot (1-2\alpha)}{3}$. By Claim 3.5, for every proof $\pi \in \{0,1\}^{p(n)}$, the algorithm V (from Claim 3.5), on input ε' and oracle access to $y' \circ \pi$ (obtained via T_k), rejects (causing k to be discarded) with probability at least $2/3$. Thus, the probability that tester fails to discard k in $\lceil 4 \log L \rceil$ independent iterations of Steps 12-14 is at most $1/16L$.

Therefore, the probability that the tester fails to discard k is at most $\frac{1}{10L} + \frac{1}{16L} < \frac{1}{5L}$. By the union bound, the probability that Algorithm 2 fails to discard some $k \in [L]$ is at most $1/5$. Thus, conditioned on the event E that the number of queries made by the tester does not exceed its query budget, with probability at least $4/5$, the tester rejects.

We now bound the probability of the event E. For this, we calculate the expected number of queries made by Algorithm 2. For all $k \in [L]$, the expected number of queries that each invocation of the algorithm T_k makes is at most q. Since the fraction of erasures (with respect to the encoded part $x[sn+1 \dots N]$) is at most $3/4$, the values q and L are both constants (by Theorem 1.4). Hence, the expected number of queries made in Steps 7-10 is $O(\frac{1}{\varepsilon(1-2\alpha)})$.

By Claim 3.5, the number of queries made by the algorithm V (from Claim 3.5) on input $\varepsilon' = \frac{\varepsilon \cdot (1-2\alpha)}{3}$ and oracle access to T_k, is $O(\frac{1}{\varepsilon(1-2\alpha)})$. Thus, the expected number of queries made in Steps 12-14 by Algorithm 2 is $O(\frac{1}{\varepsilon(1-2\alpha)})$.

Therefore the expected total number of queries made by the tester is $O(\frac{1}{\varepsilon(1-2\alpha)})$. Hence, for a large enough constant C, the probability that the number of queries exceed $\frac{C}{\varepsilon(1-2\alpha)}$ is at most $1/10$ by Markov's inequality. Hence, the probability that the tester accepts x that is ε-far from $\mathcal{P}$ is at most $1/3$. ◀

▶ **Lemma 3.10.** *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2. For every $\alpha \in (\frac{\varepsilon^\star}{8}, 1)$ and $\varepsilon' \in (\alpha, \varepsilon^\star - \frac{(\varepsilon^\star)^2}{4})$, the query complexity of (α,ε')-tolerant testing $\mathcal{P}$ on strings of length N is $\tilde{\Omega}(\log N)$.*

Proof. Let $\aleph, \varepsilon^\star \in (0,1)$ be as in Lemma 3.2. We will prove the lemma by showing a reduction from $\varepsilon^\star$-testing of $\mathcal{R}_{\phi_n}$. Fix $n \in \aleph$ and let $p(n)$ and N be as in Definition 3.6. Let s denote $(\frac{4}{\varepsilon^\star} - 1) \cdot \frac{2^{n+p(n)}}{n}$.

Consider a string $y \in \{0,1\}^n$ that we want to $\varepsilon^\star$-test for $\mathcal{R}_{\phi_n}$. Let $x \in \{0,1\}^N$ be the string where the first sn bits of x are s repetitions of y and the remaining bits are all 0s. We refer to the substring $x[1 \dots sn]$ as the plain part of x and the substring $x[sn+1 \dots N]$ as the encoded part of x.

Assume that A is an (α,ε')-tolerant tester for $\mathcal{P}$. We now describe an $\varepsilon^\star$-tester A' for $\mathcal{R}_{\phi_n}$ that has the same query complexity as A. Given oracle access to $y \in \{0,1\}^n$, the tester A' runs the tester A on the string $x \in \{0,1\}^N$ and accepts if and only if A accepts, where x is constructed from y as described above. Observe that one can simulate a query to x by making at most one query to y.

If $y \in \mathcal{R}_{\phi_n}$, then x is α-close to $\mathcal{P}$. Observe that the encoded part of x needs to be changed in at most $1/2$ fraction of its positions in order to make it the encoding of a string $y \circ \pi$, where π is a proof that makes a PCP of proximity for testing $\mathcal{R}_{\phi_n}$ accept. This follows from the fact that the normalized weight of every nonzero codeword in the Hadamard code is $1/2$. Thus, the fraction of bits in x that needs to be changed in order to make it satisfy $\mathcal{P}$ is at most $\frac{1}{2} \cdot \frac{N-sn}{N} = \frac{\varepsilon^\star}{8}$, which is less than α. Therefore, by definition, A' will accept x with probability at least $2/3$.

Assume now that y is $\varepsilon^\star$-far from $\mathcal{R}_{\phi_n}$. Then x needs to be changed in at least $\varepsilon^\star \cdot sn$ positions to make it satisfy $\mathcal{P}$. From this, one can observe that x is $(\varepsilon^\star - \frac{(\varepsilon^\star)^2}{4})$-far from $\mathcal{P}$. Hence, for all $\varepsilon' < \varepsilon^\star - \frac{(\varepsilon^\star)^2}{4}$, we have that A will reject x with probability at least $2/3$, and therefore A' will reject y with probability at least $2/3$.

Thus, we have shown that the query complexity of (α,ε')-tolerant testing $\mathcal{P}$ is at least the query complexity of $\varepsilon^\star$-testing $\mathcal{R}_{\phi_n}$. Hence, the query complexity of (α,ε')-tolerant testing $\mathcal{P}$ is $\Omega(n)$, which is equal to $\tilde{\Omega}(\log N)$. ◀

Proof of Theorem 1.5. Theorem 3.1 states that, for certain ranges of parameters $\alpha, \varepsilon, \varepsilon' \in (0,1)$ and for large enough $N \in \mathbb{N}$, the property $\mathcal{P}$ on binary strings of length N, is α-erasure-resiliently ε-testable, but is not (α,ε')-tolerantly testable. To prove Theorem 1.5, we need to show the existence of $\alpha, \varepsilon \in (0,1)$ such that the property $\mathcal{P}$ on binary strings of length N is α-erasure-resiliently ε-testable, but is not $(\alpha, \alpha + \varepsilon(1-\alpha))$-tolerantly testable. In other words, the following system of inequalities should have a solution.

$$\frac{\varepsilon^\star}{8} < \alpha < \frac{3\varepsilon^\star}{16}; \quad \frac{3\varepsilon^\star}{4(1-\alpha)} < \varepsilon < 1;$$

$$\varepsilon' = \alpha + \varepsilon(1-\alpha) < \varepsilon^\star - (\varepsilon^\star)^2/4$$

For every $0 < \varepsilon^\star < 1/100$, the above system has a feasible solution with $\alpha = \varepsilon^\star/6$ and $\varepsilon = 4\varepsilon^\star/5$. This implies the existence of $\alpha, \varepsilon \in (0,1)$ such that $\mathcal{P}$ is α-erasure-resiliently testable, but not $(\alpha, \alpha + \varepsilon(1-\alpha))$-tolerantly testable. Theorem 1.5 follows. ◀

4 Approximate Local List Erasure-Decoding

In this section, we prove the existence of an approximate locally list erasure-decodable code (ALLEDC) with inverse polynomial rate. Our starting point is an approximate locally

list decodable code (ALLDC) due to Impagliazzo et al. [37]. To this code, we apply an observation that every ALLDC that works in the presence of errors also works in the presence of twice as many erasures (with the same parameters up to constant factors). This gives us the required ALLEDC that we later use for our strengthened separation.

▶ **Theorem 4.1** ([37] as restated by [8]). *For every $\gamma, \beta > 0$, there exists a number $f(\gamma, \beta) > 0$ and a code family $\{C_k : \mathbb{F}_2^k \to \mathbb{F}_2^{f(\gamma,\beta)k^5}\}_{k\in\mathbb{N}}$ that is $(\gamma, \beta, O(\frac{\log(1/\beta)}{(\frac{1}{2}-\gamma)^3}), O(\frac{1}{(\frac{1}{2}-\gamma)^2}))$-approximate locally list decodable.*

For the sake of completeness, we state and prove the observation that every ALLDC that works in the presence of errors also works in the presence of twice as many erasures (with the same parameters up to constant factors).

▶ Observation 4.2. If a code family $\{C_k : \mathbb{F}_2^k \to \mathbb{F}_2^n\}_{k\in\mathbb{N}}$ is (α, β, q, L)-approximate locally list decodable, it is also $(2\alpha, \beta, 4q, 4L)$-approximate locally list erasure-decodable.

Proof. Consider a codeword $w \in (\mathbb{F}_2 \cup \{\perp\})^n$ with at most 2α fraction of erasures. Let A be an (α, β, q, L)-approximate local list decoder for C_k. Assume without loss of generality that the success probability of A is at least $4/5$. This can be ensured by running A twice and outputting the concatenation of lists obtained in both iterations. The approximate local list erasure-decoder A' for C_k first runs A on the word w_0 obtained by replacing each erasure in w with a 0, and then on the word w_1 obtained by replacing each erasure in w with a 1. The list output by algorithm A' is the concatenation of lists output by A in these two executions. Let E_1 be the event that the first execution of A succeeds and E_2 be the event that the second execution of A succeeds. Each codeword $w' = C_k(y')$ that agrees with w on all the nonerased points agrees with either w_0 or w_1 in at least $1 - \alpha$ fraction of points. If $E_1 \cap E_2$ holds, there exists an algorithm in the list output by A' that implicitly computes (see Definition 1.1) a string y'' that is β-close to y'. The probability of failure of A' is at most $\Pr[\overline{E_1} \cup \overline{E_2}] \leq \frac{2}{5}$. Hence, A' is a $(2\alpha, \beta, 4q, 4L)$-approximate local list erasure-decoder for C_k. ◀

Applying Observation 4.2 to Theorem 4.1, we get the ALLEDCs that we need.

▶ **Lemma 4.3.** *Let $c_3 > 0$ be a constant. For every $\gamma, \beta > 0$, there exists a number $f(\gamma, \beta) > 0$ and a code family $\{C_k : \mathbb{F}_2^k \to \{0,1\}^{f(\gamma,\beta)k^5}\}_{k\in\mathbb{N}}$ that is $(\gamma, \beta, \frac{c_3 \log(1/\beta)}{(1-\gamma)^3}, \frac{c_3}{(1-\gamma)^2})$-approximate locally list erasure-decodable.*

The proof of the following observation is identical to that of Observation 4.2.

▶ Observation 4.4. If a code family $\{C_k : \mathbb{F}_2^k \to \mathbb{F}_2^n\}_{k\in\mathbb{N}}$ is (α, q, L)-locally list decodable, it is also $(2\alpha, 4q, 4L)$-locally list erasure-decodable.

5 Strengthened Separation

In this section, we describe a property $\mathcal{P}'$ that can be erasure-resiliently tested using a constant number of queries, but for which every tolerant tester has query complexity $n^{\Omega(1)}$, and prove Theorem 1.6. The following theorem implies Theorem 1.6.

▶ **Theorem 5.1.** *There exists a property $\mathcal{P}'$ and constants $\varepsilon^\star \in (0,1), c_2 > 1$ such that,*

- *For every $\varepsilon \in \left(\frac{\varepsilon^\star}{8}, 1\right)$ and $\alpha \in (0, \frac{\varepsilon^\star}{57600 \cdot c_2})$, property $\mathcal{P}'$ can be α-erasure-resiliently ε-tested using a constant number of queries,*
- *For every $\alpha \in (\frac{\varepsilon^\star}{57600 \cdot c_2 + 2\varepsilon^\star}, 1)$, and $\varepsilon' \in \left(\alpha, \frac{28800 \cdot c_2 \cdot \varepsilon^\star}{28800 \cdot c_2 + \varepsilon^\star}\right)$, every (α, ε')-tolerant tester for $\mathcal{P}'$ on inputs of length N has query complexity $N^{\Omega(1)}$.*

5.1 Description of the Separating Property $\mathcal{P}'$

The property $\mathcal{P}'$ is very similar to the property $\mathcal{P}$ that we used in our first separation (see Definition 3.6). Like a string that satisfies $\mathcal{P}$, a string that satisfies $\mathcal{P}'$ can also be thought of as consisting of a plain part (that contains the repetition of a string $y \in \mathcal{R}_{\phi_n}$) and an encoded part. The encoded part of a string in $\mathcal{P}$ is the Hadamard encoding of a string $y \circ \pi$, where π is a proof that makes the algorithm V from Claim 3.5 accept. However, the encoded part of a string satisfying $\mathcal{P}'$ is the encoding of a string π', where π' is a proof (whose length is asymptotically equal to $|\pi|$) that makes a 'smoothened' PCPP accept. In addition, the encoding uses an ALLEDC (from Section 4) instead of the Hadamard code.

We first describe the 'smoothened' PCPP used in our construction. The following lemma by Ben-Sasson et al. [9] and Guruswami and Rudra [34] states that algorithms making nonadaptive queries can be transformed into algorithms that make nearly uniform queries.

▶ **Lemma 5.2** ([34, 9]). *For every nonadaptive algorithm T, there exists a mapping $\varphi : \{0,1\}^* \to \{0,1\}^*$ and an algorithm T' satisfying the following:*

- *For every $x \in \{0,1\}^*$, the decision of T with oracle access to x is identical to the decision of T' with oracle access to $\varphi(x)$. Moreover, $3|x| < |\varphi(x)| \leq 4|x|$, and the number of queries that T' makes to $\varphi(x)$ is at most twice the number of queries that T makes to x.*
- *Given oracle access to $x' \in \{0,1\}^n$, each query of T' is to location $j \in [n]$ with probability at most $2/n$.*

Combining Lemma 3.4 with Lemma 5.2 (along with the fact that $\mathcal{R} = \{\mathcal{R}_{\phi_n}\}_{n \in \aleph}$ can be decided using linear-sized circuits), we get the required 'smoothened' PCPP for $\mathcal{R}$.

▶ **Lemma 5.3.** *Let $c_1 > 0, c_2 > 1$ be constants. The property $\mathcal{R}_{\phi_n}$ has a* PCPP *V that works for all $\varepsilon \in (0,1]$, gets oracle access to an input y of length n and a proof π of length at most $c_1 n \cdot \text{polylog } n$, and makes at most $\frac{c_2}{\varepsilon}$ queries. Moreover, the queries of V are nonadaptive and satisfy the following:*

- *Each query V makes to y is to any particular location of y with probability at most $2/n$.*
- *Each query V makes to π is to any particular location of π with probability at most $2/|\pi|$.*

The following is the definition of our separating property $\mathcal{P}'$. Note that the encoded part of a string satisfying $\mathcal{P}'$ contains the encoding of a proof as well as the complement of that encoding. This is done in order to equalize the number of 0s and 1s in the encoded part.

▶ **Definition 5.4** (Separating Property $\mathcal{P}'$). Let $\aleph$ and $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2. Let $c_1 > 0$, $c_2 > 1$ be as in Lemma 5.3 and $c_3 > 0$ be as in Lemma 4.3. Let $m = \frac{28800 \cdot c_2}{\varepsilon^\star}$, $\gamma = \frac{1}{2} + \frac{\varepsilon^\star}{57600 \cdot c_2}$ and $\beta = \frac{\varepsilon^\star}{9000 c_2 \cdot \left\lceil \ln \frac{6c_3}{(1-\gamma)^2} \right\rceil}$.

For $n \in \aleph$, let $p(n) \leq c_1 \cdot n \cdot \text{polylog } n$ denote the length of the proof that makes the algorithm V in Lemma 5.3 accept. Let $f(\cdot,\cdot)$ be as in Lemma 4.3. Let $\mathcal{C} = \{C_k\}_{k \in \mathbb{N}}$ be the $(\gamma, \beta, \frac{c_3 \log(1/\beta)}{(1-\gamma)^3}, \frac{c_3}{(1-\gamma)^2})$-ALLEDC from Lemma 4.3.

A string $x \in \{0,1\}^N$ of length $N = (m+1) \cdot 2f(\gamma,\beta) \cdot (p(n))^5$ satisfies $\mathcal{P}'$ if the following conditions hold:

1. The first $m \cdot 2f(\gamma,\beta) \cdot (p(n))^5$ bits of x (called the *plain part* of x) consist of $m \cdot \frac{2f(\gamma,\beta) \cdot (p(n))^5}{n}$ repetitions of a string $y \in \mathcal{R}_{\phi_n}$ of length n, for $\mathcal{R}_{\phi_n}$ from Lemma 3.2.
2. The remaining $2f(\gamma,\beta) \cdot (p(n))^5$ bits of x is called the *encoded part*. Its first half is the encoding, using $\mathcal{C}$, of a string $\pi \in \{0,1\}^{p(n)}$ such that the PCPP V in Lemma 5.3 accepts when given oracle access to $y \circ \pi$. The second half of the encoded part is the complement of its first half.

5.2 Proof of Strengthened Separation

In this section, we prove Theorem 5.1. Lemmas 5.5 and 5.9 together imply the first and second parts of Theorem 5.1, respectively. The high level idea of the proof of Lemma 5.5 is very similar to that of Lemma 3.7. The differences arise mainly because of the way the encoded parts of strings satisfying $\mathcal{P}$ and $\mathcal{P}'$ differ. The erasure-resilient tester for $\mathcal{P}$ could first check whether the plain part is a repetition of the 'decoded input', and then check whether the 'decoded input' is in $\mathcal{R}$ with the help of the 'decoded PCPP proof'. Since the encoded part of $\mathcal{P}'$ is the encoding of just a PCPP proof, this is not possible. Instead, the erasure-resilient tester for $\mathcal{P}'$ samples a uniformly random nonerased point u from the plain part and uses the 'block' from which u is obtained as a 'candidate input' y. It then checks whether the plain part is a repetition of y and also checks whether $y \in \mathcal{R}$ using the 'approximately decoded proof'. In case a string is α-erased and ε-far from $\mathcal{P}'$, we show that the 'candidate input' y that we sample is $c\alpha$-erased and $c'\varepsilon$-far from $\mathcal{R}$, for some constants c, c'. Hence, the smoothened PCPP verifier rejects.

▶ **Lemma 5.5.** *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2 and $c_2 > 1$ be as in Lemma 5.3. For every $\varepsilon \in \left(\frac{\varepsilon^\star}{8}, 1\right)$ and $\alpha \in (0, \frac{\varepsilon^\star}{57600 \cdot c_2})$, the property $\mathcal{P}'$ is α-erasure-resiliently ε-testable using a constant number of queries.*

Proof. The erasure-resilient tester is presented in Algorithm 3. Let m denote $\frac{28800 \cdot c_2}{\varepsilon^\star}$. Let $\gamma = \frac{1}{2} + \frac{\varepsilon^\star}{57600 \cdot c_2}$, $\beta = \frac{\varepsilon^\star}{9000 c_2 \cdot \left\lceil \ln \frac{6c_3}{(1-\gamma)^2} \right\rceil}$, $q = \frac{c_3 \log(1/\beta)}{(1-\gamma)^3}$, and $L = \frac{c_3}{(1-\gamma)^2}$. For $n \in \aleph$, consider a string $x \in \{0,1\}^N$, where $N = (m+1) \cdot 2f(\gamma,\beta) \cdot (p(n))^5$. The plain part of x is m times larger than the encoded part. Let s denote the number $m \cdot \frac{2f(\gamma,\beta)\cdot(p(n))^5}{n}$.

Assume that x satisfies $\mathcal{P}'$. Since x satisfies $\mathcal{P}'$, the plain part of x is completable to the repetitions of some $y \in \mathcal{R}_{\phi_n}$. Therefore, Steps 5-9 never reject. By definition of $\mathcal{P}'$, the first half of the encoded part of x is the encoding (using the (γ,β,q,L)-ALLED code $\mathcal{C}$ from Lemma 4.3) of a proof $\pi(y) \in \{0,1\}^{p(n)}$ such that the PCPP V with oracle access to $y \circ \pi(y)$ always accepts. The second half of the encoding is the complement of the first half. The fraction of erasures in the encoded part (even if all of the erasures were there) is at most $(m+1)\alpha$. Therefore, the fraction of erasures in either the first half or the second half of the encoded part is at most $(m+1)\cdot\alpha = \frac{1}{2} + \frac{1}{2m} = \gamma$.

By the definition of a (γ,β,q,L)-ALLED code, with probability at least $2/3$, one of the algorithms $T_1, T_2, \ldots, T_L$ returned by the approximate local list decoder provides oracle access to $\pi(y)$ with at most β fraction of errors. Let T_k be that algorithm. The tester discards this k only if an erroneous point is queried in some iteration of Steps 14-18. Since each proof query of V (in Step 17) is made to a specific index in the proof with probability at most $2/|p(n)|$ and the string decoded by T_k is β-erroneous, by the union bound over queries of V, the probability of V querying an erroneous point is at most $2\beta \cdot \frac{c_2 \cdot 75}{24\varepsilon}$. Hence, by the union bound, the probability that the tester discards k is at most $\frac{1}{3} + 2 \cdot 6 \cdot \lceil \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\varepsilon} \cdot \beta \leq \frac{2}{5}$, where the inequality follows from our setting of β. Hence, Step 19 does not reject with probability at least $3/5$. That is, the tester accepts x with probability at least $3/5$.

Assume now that x is ε-far from $\mathcal{P}'$. Let $\mathcal{N}_{\text{pl}}$ denote the set of nonerased points in the plain part of x. Let $\mathcal{N}_{\text{en}}$ denote the set of nonerased points in the encoded part of x. Let α_{pl} denote the fraction (with respect to sn) of erased points in the plain part.

Let E denote the event that the number of queries made by the tester does not exceed the query budget Q. We first analyze the probability that Algorithm 3 rejects, conditioned on E. We prove later, in Claim 5.8, that $\Pr[\overline{E}] \leq 1/30$.

Case I: the plain part of x is $\varepsilon/144$-far from being the repetitions of every $y \in \{0,1\}^n$.

Algorithm 3 Erasure-resilient tester for separating property $\mathcal{P}'$

Input: $\alpha, \varepsilon \in (0,1), N = (m+1) \cdot 2f(\gamma,\beta) \cdot (p(n))^5$; oracle access to $x \in \{0,1,\perp\}^N$

1: Set $s \leftarrow m \cdot \frac{2f(\gamma,\beta)\cdot(p(n))^5}{n}$, $q \leftarrow \frac{c_3 \log(1/\beta)}{(1-\gamma)^3}$, and $L \leftarrow \frac{c_3}{(1-\gamma)^2}$.

2: Set the query budget $Q \leftarrow 30 \cdot \left(\lceil \frac{432}{\varepsilon} \rceil \cdot \frac{2}{1-(m+1)\alpha} + L\lceil 6 \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\varepsilon} \cdot q \right)$.

3: **Accept** whenever the number of queries exceeds Q.

4: ▷ Steps 5-9 check that the plain part of x is the repetition of a string $y \in \{0,1\}^n$.

5: **repeat** $\lceil \frac{432}{\varepsilon} \rceil$ times:

6: Repeatedly sample a uniformly random point u from the plain part until $x[u] \neq \perp$.

7: Let u be $(i-1)n + a$ for $i \in [s]$ and $a \in [n]$.

8: Repeatedly sample $j \in [s]$ uniformly at random until $x[(j-1)n + a] \neq \perp$.

9: **Reject** if $x[(i-1)n + a] \neq x[(j-1)n + a]$.

10: ▷ In order to query the i-th bit of the encoding, we query the i-th bits of both the first and second halves of the encoded part. We set the i-th bit of the encoding to the i-th bit of the first half if that is nonerased, and to the complement of the i-th bit of second half if that is nonerased. If both are erased, we set the i-th bit of the encoding to $\perp$.

11: Run the decoder for the (γ, β, q, L)-ALLED code (Lemma 4.3) with oracle access to the encoded part of x. Let $A_1, A_2, \ldots, A_L$ be the list of algorithms it returns.

12: ▷ Steps 13-19 check that $y \in \mathcal{R}_{\phi_n}$ using the PCPP V on decoded proofs.

13: **for** each $k \in [L]$ **do**

14: **repeat** $\lceil 6 \ln 6L \rceil$ times:

15: Repeatedly sample a uniformly random point u from the plain part until $x[u] \neq \perp$.

16: Let u be $(i-1)n + a$ for $i \in [s]$ and $a \in [n]$.

17: Run the PCPP V (guaranteed by Lemma 5.3) with proximity parameter $\frac{24\varepsilon}{75}$, and oracle access to $x[(i-1)n+1, \ldots, (i-1)n+n]$ as the input string and the string decoded by T_k as the proof.

18: **Discard** the current k if all query answers to V are nonerased and V rejects.

19: **Reject** if every $k \in [L]$ is **discarded**; otherwise, **accept**.

Let ε_{pl} denote the fraction of points (with respect to $|\mathcal{N}_{\text{pl}}|$) in $\mathcal{N}_{\text{pl}}$ whose values need to be changed in order to make the plain part a repetition of some string $y \in \{0,1\}^n$. Let $S_a = \{(i-1)n + a : i \in [s]\}$ for all $a \in [n]$. We use the term a-th segment to refer to the set S_a. For all $a \in [n]$, we have $|S_a| = s$. For all $a \in [n]$, let $\alpha_a = |\{u \in S_a : x[u] = \perp\}|/s$ denote the fraction of points in S_a that are erased. Let $\mathcal{N}_a \subseteq S_a$ denote the set of nonerased points in the a-th segment. Let ε_a for all $a \in [n]$ denote the fraction of points in $\mathcal{N}_a$ whose values need to be changed in order to satisfy $x[u] = x[v]$ for all $u, v \in S_a$ such that both $x[u]$ and $x[v]$ are nonerased.

For every $a \in [n]$ and $u \in \mathcal{N}_a$, the number of $v \in \mathcal{N}_a$ such that $x[u] \neq x[v]$, is at least $\varepsilon_a \cdot |\mathcal{N}_a|$. Let G_a for all $a \in [n]$ denote the (good) event that the tester samples a point from $\mathcal{N}_a$ in Step 6. Let F denote the event that the tester rejects in a single iteration of the loop in Steps 5-9.

$$\Pr[F|E] = \sum_{a \in [n]} \Pr[G_a|E] \cdot \Pr[F|G_a, E] \geq \sum_{a \in [n]} \frac{|\mathcal{N}_a|}{|\mathcal{N}_{\text{pl}}|} \cdot \varepsilon_a = \varepsilon_{\text{pl}} \geq \frac{\varepsilon}{144}.$$

Therefore, conditioned on E, in at least $432/\varepsilon$ iterations, the tester will reject with probability

at least 19/20. Hence, in Case I, the algorithm accepts with probability at most $\frac{1}{20} + \Pr[\overline{E}] \leq \frac{1}{20} + \frac{1}{30} \leq \frac{2}{5}$, where we prove later (Claim 5.8) $\Pr[\overline{E}] \leq 1/30$. Thus, the algorithm rejects with probability at least 3/5.

Case II: the plain part of x is $\varepsilon/144$-close to being repetitions of a string $y^* \in \{0,1\}^n$. We first show that y^* has to be far from $\mathcal{R}_{\phi_n}$.

▶ Claim 5.6. The string y^* is $\varepsilon/2$-far from $\mathcal{R}_{\phi_n}$.

Proof. Otherwise, one can transform the entire plain part of x to (be completable to) repetitions of y^* by making at most $|\mathcal{N}_{\text{pl}}| \cdot \frac{\varepsilon}{144} \leq N \cdot \frac{\varepsilon}{144}$ changes. This can then be transformed to repetitions of a string in $\mathcal{R}_{\phi_n}$ by making at most $sn \cdot \frac{\varepsilon}{2} \leq N \cdot \frac{\varepsilon}{2}$ changes. Thus, the string x can be made to satisfy $\mathcal{P}'$ by making at most $N \cdot \left(\frac{\varepsilon}{144} + \frac{\varepsilon}{2} + \frac{1}{m+1}\right)$ changes, where the term $\frac{N}{m+1}$ accounts for the number of changes in the encoded part. Since $\varepsilon > \frac{\varepsilon^\star}{8}$ and $c_2 > 1$, we have that $m = \frac{28800 \cdot c_2}{\varepsilon^\star} > \frac{144}{71\varepsilon}$. Hence, $\frac{1}{m} < \frac{71\varepsilon}{144}$ and, therefore, $N \cdot \left(\frac{\varepsilon}{144} + \frac{\varepsilon}{2} + \frac{1}{m+1}\right) < \varepsilon N$. Thus, the string x can be made to satisfy $\mathcal{P}'$ by making less than εN changes. This is a contradiction. ◀

Let $B_i = \{(i-1)n + a : a \in [n]\}$ for all $i \in [s]$. We use the term i-th block to refer to the set B_i. For all $i \in [s]$, we have, $|B_i| = n$. Let $\alpha_i = \frac{|\{u \in B_i : x[u] = \perp\}|}{n}$ for all $i \in [s]$ denote the fraction of points in B_i that are erased. Let $\mathcal{N}_i \subseteq S_i$ denote the set of nonerased points in the i-th block. Let ε_i for all $i \in [s]$ denote the fraction of points in $\mathcal{N}_i$ whose values need to be changed in order to satisfy $x[(i-1)n + a] = y^*[a]$ for all $a \in [n]$.

Fix $k \in [L]$. We show that Algorithm 3 discards k with high probability. Consider a single iteration of the repeat-loop in Steps 15-18. Let y' denote the (partially erased) string represented by the block that Algorithm 3 samples in Step 15. Let G_1 denote the (good) event that y' is $\varepsilon/6$-close to y^*. Let G_2 denote the (good) event that y' has at most 48α fraction of erasures. We first evaluate the probability that the tester discards k in Steps 15-18 conditioned on G_1 and G_2.

▶ Claim 5.7. Conditioned on G_1 and G_2, the string y' is $24\varepsilon/75$-far from $\mathcal{R}_{\phi_n}$.

Proof. Let y'' be a string in $\mathcal{R}_{\phi_n}$ closest to y'. Let d denote the number of nonerased bits in y' that need to be changed in order for it to be completable to y''. By our conditioning, y' is a 48α-erased string that is $\varepsilon/6$-close to y^*. Thus, one can convert y^* into y' and then y' into y'' by modifying at most $48\alpha n + \frac{\varepsilon n}{6} + d$ values in y^*. Since y^* is $\varepsilon/2$-far from $\mathcal{R}_{\phi_n}$, we get that $d \geq \frac{\varepsilon n}{2} - \frac{\varepsilon n}{6} - 48\alpha n$. From the restrictions on α and ε, one can verify that for all settings of these parameters, we have $\alpha \leq \frac{\varepsilon}{3600}$, which implies that $d \geq \frac{24\varepsilon n}{75}$. ◀

The PCPP V, with proximity parameter $\frac{24\varepsilon}{75}$, is run on y' and the proof decoded by T_k. Let B_1 denote the (bad) event that the PCPP V obtains an erased bit as the answer to some query. Let B_2 denote the (bad) event that V accepts. By Lemma 5.3, each query of V to the input part is made to each input index with probability at most $\frac{2}{n}$ uniformly distributed among the n input indices. Hence $\Pr[B_1 | E, G_1, G_2]$, the probability that some input query is made to an erased point, is at most $\frac{c_2 \cdot 75}{24\varepsilon} \cdot 96\alpha$.

The probability that the V accepts (even if there were no erased query answers) is $\Pr[B_2 | E, G_1, G_2]$ and is, by Definition 3.3, at most 1/3. Thus, the probability that the PCPP accepts, conditioned on E, G_1, and G_2, is by the union bound, at most $\frac{c_2 \cdot 75}{24\varepsilon} \cdot 96\alpha + \frac{1}{3} \leq \frac{1}{24} + \frac{1}{3}$, where the inequality follows from our setting of ε and α.

To bound the probability that the PCPP accepts in a single iteration of Steps 15-18, we now evaluate $\Pr[\overline{G_1}]$ and $\Pr[\overline{G_2}]$. Let the random variable X denote the relative Hamming

distance of y' from y^*. Then, $\mathbb{E}[X] = \sum_{i\in[s]} \frac{|\mathcal{N}_i|}{|\mathcal{N}_{\text{pl}}|} \cdot \varepsilon_i = \varepsilon_{\text{pl}} \le \frac{\varepsilon}{144}$. By Markov's inequality, $\Pr[\overline{G_1}] = \Pr[X \ge \frac{\varepsilon}{6}] \le \mathbb{E}[X]/(\varepsilon/6) \le 1/24$. To bound $\Pr[\overline{G_2}]$, let the random variable Y denote the fraction of erasures in y'. Then,

$$\begin{aligned}
\mathbb{E}[Y] &= \sum_{i\in[s]} \frac{|\mathcal{N}_i|}{|\mathcal{N}_{\text{pl}}|} \cdot \alpha_i = \frac{1}{sn(1-\alpha_{\text{pl}})} \sum_{i\in[s]} n(1-\alpha_i)\alpha_i = \frac{1}{s(1-\alpha_{\text{pl}})} \sum_{i\in[s]} \alpha_i(1-\alpha_i) \\
&= \frac{\sum_{i\in[s]} \alpha_i}{s(1-\alpha_{\text{pl}})} - \frac{\sum_{i\in[s]} \alpha_i^2}{s(1-\alpha_{\text{pl}})} = \frac{\alpha_{\text{pl}}}{1-\alpha_{\text{pl}}} - \frac{\sum_{i\in[s]} \alpha_i^2}{s(1-\alpha_{\text{pl}})} \\
&\le \frac{\alpha_{\text{pl}}}{1-\alpha_{\text{pl}}} - \frac{(\sum_{i\in[s]} \alpha_i)^2/s}{s(1-\alpha_{\text{pl}})} && \text{(Cauchy-Schwartz)} \\
&= \frac{\alpha_{\text{pl}}}{1-\alpha_{\text{pl}}} - \frac{\alpha_{\text{pl}}^2}{1-\alpha_{\text{pl}}} = \alpha_{\text{pl}}.
\end{aligned}$$

Even if all the erasures were in the plain part, $\alpha_{\text{pl}} \le \frac{\alpha N}{sn} \le \alpha \cdot (1 + \frac{1}{m})$. Again, by an application of Markov's inequality, we get $\Pr[\overline{G_2}] = \Pr[Y > 48\alpha] \le \frac{\mathbb{E}[Y]}{48\alpha} \le \frac{1+\frac{1}{m}}{48} \le 1/24$.

Therefore, conditioned on E, the probability that the PCPP accepts in one iteration of Steps 15-18 is at most $\Pr[B_1|E, G_1, G_2] + \Pr[B_2|E, G_1, G_2] + \Pr[\overline{G_2}] + \Pr[\overline{G_1}] \le \frac{1}{24} + \frac{1}{3} + \frac{1}{24} + \frac{1}{24} \le \frac{2}{3}$. That is, conditioned on E, for a fixed $k \in [L]$, in $\lceil 6 \ln 6L \rceil$ independent repetitions of Steps 15-18, the probability that the PCPP does not discard k is at most $\left(1 - \frac{1}{3}\right)^{\lceil 6 \ln 6L \rceil} \le \frac{1}{36L^2}$. Hence, conditioned on E, the probability that for some $k \in [L]$, Steps 14-18 accepts is, by the union bound, at most $1/36L$. Thus, if x is in Case II, the probability that the tester accepts is at most, $\frac{1}{36L} + \Pr[\overline{E}] \le \frac{1}{36L} + \frac{1}{30} \le \frac{2}{5}$, where Claim 5.8 shows that $\Pr[\overline{E}]$ is at most $1/30$.

▶ Claim 5.8. The probability that Algorithm 3 exceeds its query budget is at most $1/30$.

Proof. We first compute the expected number of queries that the tester makes. The number of queries made in Steps 13-18 is at most $L\lceil 6 \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\varepsilon} \cdot q$, where q and L are the query complexity and list size of the approximate local list decoder, respectively.

We now calculate the expected number of queries made from Steps 6-9. Let U be the random variable for the number of queries to get the first nonerased point. Let V denote the number of queries needed to get the second nonerased point. Clearly, $\mathbb{E}[U] = \frac{1}{1-\alpha_{\text{pl}}}$. To calculate $\mathbb{E}[V]$:

$$\mathbb{E}[V] = \sum_{a\in[n]} \frac{|\mathcal{N}_a|}{|\mathcal{N}_{\text{pl}}|} \cdot \frac{1}{1-\alpha_a} = \sum_{a\in[n]} \frac{(1-\alpha_a)s}{sn(1-\alpha_{\text{pl}})} \cdot \frac{1}{1-\alpha_a} = \frac{1}{1-\alpha_{\text{pl}}}.$$

Hence, the expected number of queries made by the tester in Steps 5-9 is $\lceil \frac{432}{\varepsilon} \rceil \cdot \frac{2}{1-\alpha_{\text{pl}}}$. Since $\alpha_{\text{pl}} \le \alpha \cdot (m+1)$, this expectation is at most $\lceil \frac{432}{\varepsilon} \rceil \cdot \frac{2}{1-(m+1)\alpha}$. Note that $(m+1) \cdot \alpha < 1$ by the setting of α and m. Hence, setting Q to $30 \cdot \left(\lceil \frac{432}{\varepsilon} \rceil \cdot \frac{2}{1-(m+1)\alpha} + L\lceil 6 \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\varepsilon} \cdot q \right)$, and applying Markov's inequality, one can see that $\Pr[\overline{E}] \le 1/30$. ◀

This completes the proof of Lemma 5.5. ◀

Next, we show that it is hard to tolerant test $\mathcal{P}'$. The proof of Lemma 5.9 is identical to the proof of Lemma 3.10 up to change in parameters.

▶ **Lemma 5.9.** *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 3.2 and $c_2 > 1$ be as in Lemma 5.3. For every $\alpha \in (\frac{\varepsilon^\star}{57600 \cdot c_2 + 2\varepsilon^\star}, 1)$, and $\varepsilon' \in \left(\alpha, \frac{28800 \cdot c_2 \cdot \varepsilon^\star}{28800 \cdot c_2 + \varepsilon^\star}\right)$, every (α, ε')-tolerant tester for $\mathcal{P}'$ requires $\tilde{\Omega}(N^{0.2})$ queries.*

Proof. Let $\aleph$ be as in Lemma 3.2 and let $n \in \aleph$. We will prove the lemma by showing a reduction from $\varepsilon^\star$-testing of $\mathcal{R}_{\phi_n}$. Let N and $p(n)$ be as in Definition 5.4. Let s denote $m \cdot \frac{2f(\gamma,\beta)\cdot(p(n))^5}{n}$.

Consider a string $y \in \{0,1\}^n$ that we want to $\varepsilon^\star$-test for $\mathcal{R}_{\phi_n}$. Let $x \in \{0,1\}^N$ be the string where the first sn bits of x are s repetitions of y and the remaining bits are all 0s. We refer to the substring $x[1 \dots sn]$ as the plain part of x and the substring $x[sn+1 \dots N]$ as the encoded part of x.

Assume that A is an (α, ε')-tolerant tester for $\mathcal{P}'$. We now describe an $\varepsilon^\star$-tester A' for $\mathcal{R}_{\phi_n}$ that has the same query complexity as A. Given oracle access to $y \in \{0,1\}^n$, the tester A' runs the tester A on the string $x \in \{0,1\}^N$ (as constructed from y above) and accepts iff A accepts. Observe that one can simulate a query to x by making at most one query to y.

If $y \in \mathcal{R}_{\phi_n}$, then x is α-close to $\mathcal{P}'$. Observe that the encoded part of x needs to be changed in at most $\frac{1}{2}$ fraction of its positions in order to make it the encoding of a string π, where π is a proof that makes a PCPP for testing $\mathcal{R}_{\phi_n}$ (as guaranteed by Lemma 5.3) accept. This follows from the fact that the encoded part of every string that satisfies the property contains an equal number of 0s and 1s. Thus, the fraction of bits in x that needs to be changed in order to make it satisfy $\mathcal{P}'$ is at most $\frac{1}{2} \cdot \frac{N-sn}{N} = \frac{1}{2(m+1)} = \frac{\varepsilon^\star}{57600\cdot c_2 + 2\varepsilon^\star}$, which is less than α. Therefore, by definition, A' will accept x with probability at least $\frac{2}{3}$.

Assume now that y is $\varepsilon^\star$-far from $\mathcal{R}_{\phi_n}$. Then x needs to be changed in at least $\varepsilon^\star \cdot sn$ positions to make it satisfy $\mathcal{P}'$. From this, one can observe that x is $\varepsilon^\star \cdot \frac{m}{m+1}$-far from $\mathcal{P}'$. Hence, for all $\varepsilon' < \varepsilon^\star \cdot \frac{m}{m+1}$, we have that A will reject x with probability at least $2/3$, and therefore A' will reject y with probability at least $2/3$.

Thus, we have shown that the query complexity of (α, ε')-tolerant testing $\mathcal{P}'$ is at least the query complexity of $\varepsilon^\star$-testing $\mathcal{R}_{\phi_n}$. Hence, the query complexity of (α, ε')-tolerant testing $\mathcal{P}'$ is $\Omega(n)$, which is equal to $\tilde{\Omega}(N^{0.2})$. ◀

Proof of Theorem 1.6. From Theorem 5.1, we get the following constraints on ε and α:

$$\frac{\varepsilon^\star}{57600 \cdot c_2 + 2\varepsilon^\star} < \alpha < \frac{\varepsilon^\star}{57600 \cdot c_2};\ \varepsilon > \frac{\varepsilon^\star}{8};\ \varepsilon' = \alpha + \varepsilon(1-\alpha) < \frac{28800 \cdot c_2 \varepsilon^\star}{28800 \cdot c_2 + \varepsilon^\star}$$

For sufficiently small $\varepsilon^\star$, setting $\alpha = \frac{\varepsilon^\star}{57600\cdot c_2 + \varepsilon^\star}$ and $\varepsilon = \frac{57599}{57600\cdot c_2 + 2\varepsilon^\star}$ satisfies all the above constraints. ◀

6 Local Erasure-Decoding Versus Local Decoding

In this section, we prove Theorem 1.8 and an observation that if a code is locally decodable, it is also locally erasure-decodable up to (nearly) twice as many erasures. To prove Theorem 1.8, we first make the local erasure-decoder for $\{C_n\}_{n\in\mathbb{N}}$ nonadaptive. We then show that every LEDC with a nonadaptive decoding algorithm is such that uncorrupted codewords can be locally decoded using an algorithm that queries nearly uniformly distributed codeword indices. We then use this 'smoothness' property to show that the code family is locally decodable from a smaller fraction of errors than erasures. We formalize the 'smoothness' property in Definition 6.1.

▶ **Definition 6.1** (Smooth Locally Decodable Codes). A code family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is (q,η)-smooth locally decodable if there exists a $(0,q)$-local erasure-decoder A (see Definition 1.7) that, given oracle access to an uncorrupted codeword $w \in \mathbb{F}_2^N$, and input $i \in [n]$, is such that for all $j \in [N]$, the probability that A queries j is at most η.

It is easy to see that the following two claims imply Theorem 1.8.

▶ Claim 6.2. For every $\alpha \in [0,1)$, if a code $C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N$ is (α, q)-locally erasure-decodable, then C_n is (q', η)-smooth locally decodable, where $q' = 18q \cdot 9^{q-1}$, and $\eta = \frac{q'}{\alpha N}$.

▶ Claim 6.3. For every $\alpha \in [0,1)$, if a code $C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N$ is $(q, \frac{q}{\alpha \cdot N})$-smooth locally erasure-decodable, then C_n is $(\frac{\alpha}{12q^2}, 72q)$-locally decodable.

Proof of Claim 6.2. Let A be an (α, q)-local erasure-decoder for C_n. We first design a nonadaptive local erasure-decoder A_1 for C_n that makes a higher number of queries than A.

Consider a (partially erased) codeword $w \in (\{\perp\} \cup \mathbb{F}_2)^N$ that has at most α fraction of erasures. The algorithm A_1, on oracle access to w and an input $i \in [n]$, has $18 \cdot 9^{q-1}$ independent iterations. In each iteration, A_1 simulates A, guesses the answers to the first $q-1$ queries made by A, and decides the q-th query of A based on these guesses. A' then queries w on the q queries made by A in the simulation. If all the query answers agree with the guesses, the decoder A_1 stores the output of A as the output of the current iteration. Otherwise, it stores a uniformly random bit as the output of the current iteration. Finally, A_1 outputs the majority value output among all iterations.

In any particular iteration, the probability that A_1 outputs the correct answer in that iteration, is at least $\frac{3^{q-1}-1}{3^{q-1}} \cdot \frac{1}{2} + \frac{1}{3^{q-1}} \cdot \frac{2}{3}$, which is equal to $\frac{1}{2} + \frac{1}{6 \cdot 3^{q-1}}$. Hence, using standard arguments (see [51], for example), the probability that A_1 outputs the correct answer after $18 \cdot 9^{q-1}$ independent iterations, is at least $2/3$. The query complexity of A_1 is $18q \cdot 9^{q-1}$, which, for the rest of the proof, we denote by q'.

We now use A_1 to construct A_2, a $(q', \frac{q'}{\alpha N})$-smooth local decoder for C_n. Consider an uncorrupted codeword $w = C_n(x)$ for $x \in \mathbb{F}_2^n$. For each $i \in [n]$, let S_i denote the set consisting of indices in $[N]$ that get queried by A_1 (on input i) with probability more than $\frac{q'}{\alpha N}$. Since $\sum_{j \in [N]} \Pr[A_1^{C_n(x)}(i) \text{ queries } j] = q'$, we have $|S_i| \leq \alpha \cdot N$. On input $i \in [n]$ and oracle access to $w = C_n(x)$, the algorithm A_2 simulates A_1 in the following way. If A_1 queries $j' \in S_i$, the algorithm A_2 does not query j' and assumes that $w[j'] = \perp$. Thus, A_2 is a $(q', \frac{q'}{\alpha N})$-smooth local decoder for C_n. ◀

Proof of Claim 6.3. Consider a $(q, \frac{q}{\alpha N})$-smooth local decoder A for C_n. We will construct an $(\frac{\alpha}{12q^2}, 72q)$-local decoder A' for C_n. Algorithm A', on input $i \in [n]$ and oracle access to a word w with at most $\frac{\alpha}{12q^2}$ fraction of errors, performs 72 independent repetitions of A and outputs the majority value output among all the iterations.

Let $x \in \mathbb{F}_2^n$ be such that $y = C_n(x)$ is the codeword closest to w. If A is run on input i with oracle access to y, then for at least $\frac{2}{3}$ fraction of the sequences of its random coin tosses, A returns x_i correctly. When A is run on input i with oracle access to w, by the union bound and the smoothness of A, at most $q \cdot \frac{\alpha}{12q^2} \cdot N \cdot \frac{q}{\alpha N} = \frac{1}{12}$ fraction of sequences of its random coin tosses result in an erroneous position being queried. Hence, the probability that A, on input i and oracle access to w, returns x_i correctly is at least $\frac{2}{3} - \frac{1}{12}$. Hence, the probability that A' outputs x_i correctly is at least $2/3$. The query complexity of A' is $72q$. ◀

We remark that the above technique cannot be directly used to obtain an analog of Theorem 1.8 for the case of local list decoding. In particular, the first transformation in Claim 6.2 will decrease the success probability of the decoder exponentially (in n) and blow up the query complexity of the final decoder. However, by making a soundness assumption on the list returned by a local list decoder, we can get an analog of Theorem 1.8 for the case of local list decoding. In particular, assume that an LLEDC $\mathcal{C}$ has a decoder such that every algorithm in the output list corresponds to a valid message (whose encoding is close to the codeword). If the fraction of erasures is so *small* that there is only one codeword that agrees with the input word on all nonerased bits, we can use the sound decoder for local unique

decoding. This would imply the existence of a local unique decoder for $\mathcal{C}$ that decodes from an *even smaller* fraction of errors (by Theorem 1.8). One can then apply a transformation by Ben-Aroya et al. [8] and show that there is another code $\mathcal{C}'$ (with a slightly worse rate) that is an LLDC.

The following observation is based on an idea suggested to us by Venkatesan Guruswami.

▶ Observation 6.4. Every (α, q)-locally decodable code $C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N$ is also $(2\alpha - \rho, 72q)$-locally erasure-decodable, where $\rho = 2 \cdot \sqrt{\frac{\alpha}{N} \cdot \ln(12)}$.

Proof. Consider an (α, q)-local decoder A for $C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N$. Let $w \in (\mathbb{F}_2 \cup \{\perp\})^N$ be a codeword with at most $(2\alpha - \rho)N$ erasures. Consider algorithm A' that, on input $i \in [n]$ and oracle access to w, runs A on input $i \in [n]$ and oracle access to w', where w' is generated on the fly by filling in the erased bits of w with 0 or 1 u.a.r. The expected Hamming distance of w' to the code is at most $\alpha N - \frac{\rho}{2}N$. By a Chernoff bound, the probability that the Hamming distance of w' to the code is more than αN is at most $\frac{1}{12}$. The probability of failure of A' is at most $\frac{5}{12}$. One can amplify the success probability to 2/3 by performing 72 independent repetitions of A' and outputting the majority answer. ◀

References

1. Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519. IEEE Computer Society, 1995.
2. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and intractability of approximation problems. *Journal of ACM*, 45(3):501–555, 1998.
3. Sanjeev Arora and Shmuel Safra. Probabilistic checkable proofs: A new characterization of NP. *Journal of ACM*, 45(1):70–122, 1998.
4. László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–31. ACM Press, 1991.
5. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.
6. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-Francois Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 261–270. IEEE Computer Society, 2002.
7. Avraham Ben-Aroya, Klim Efremenko, and Amnon Ta-Shma. Local list decoding with a constant number of queries. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 715–722, 2010.
8. Avraham Ben-Aroya, Klim Efremenko, and Amnon Ta-Shma. A note on amplifying the error-tolerance of locally decodable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:134, 2010.
9. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.
10. Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3CNF properties are hard to test. *SIAM J. Comput.*, 35(1):1–21, 2005.
11. Volodia M. Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):7–19, 1986.

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

13 Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the hardness of permanent. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 1999.

14 Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of ACM*, 45(6):965–981, 1998.

15 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

16 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.

17 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin M. Varma. Erasure-resilient property testing. *SIAM J. Comput.*, 47(2):295–329, 2018.

18 Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM J. Comput*, 40(4):1154–1178, 2011.

19 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

20 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2(9):173–183, 2006.

21 Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 32–42. ACM Press, 1991.

22 Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, 28 September 1992.

23 Goldreich, Rubinfeld, and Sudan. Learning polynomials with queries: The highly noisy case. *SIJDM: SIAM Journal on Discrete Mathematics*, 13, 2000.

24 Oded Goldreich. A brief introduction to property testing. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 465–469. 2011.

25 Oded Goldreich. Introduction to testing graph properties. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 470–506. 2011.

26 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

27 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

28 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

29 Sivakanth Gopi, Swastik Kopparty, Rafael Mendes de Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the gilbert-varshamov bound. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2073–2091, 2017.

30 Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *FOCS*, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/061`.

31 Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Trans. Information Theory*, 62(5):2719–2725, 2016.

32 Venkatesan Guruswami. List decoding from erasures: bounds and code constructions. *IEEE Trans. Information Theory*, 49(11):2826–2833, 2003.

33 Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. Information Theory*, 51(10):3393–3400, 2005.

34 Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 306–317, 2005.

35 Venkatesan Guruswami and Salil P. Vadhan. A lower bound on list size for list decoding. *IEEE Trans. Information Theory*, 56(11):5681–5688, 2010.

36 Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 455–468, 2008.

37 Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

38 Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11:149–182, 2015.

39 Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *J. ACM*, 64(2):11:1–11:42, 2017.

40 Swastik Kopparty and Shubhangi Saraf. Local list-decoding and testing of random linear codes from high error. *SIAM J. Comput.*, 42(3):1302–1326, 2013.

41 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

42 R. J. Lipton. New directions in testing. In *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. American Mathematics Society, 1991.

43 Richard J. Lipton. Efficient checking of computations. In *proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 207–215. Springer, 1990.

44 Or Meir. Combinatorial PCPs with efficient verifiers. *Computational Complexity*, 23(3):355–478, 2014.

45 Or Meir. Combinatorial PCPs with short proofs. *Computational Complexity*, 25(1):1–102, 2016.

46 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.

47 Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–203. ACM Press, 1994.

48 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2009.

49 Ronitt Rubinfeld and Eric Blais. Something for (almost) nothing: New advances in sublinear-time algorithms. In *Handbook of Big Data.*, pages 155–167. 2016.

50 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

51 Jayalal Sarma and Princy Lunawat. Amplification lemma, 2012. URL: `https://www.cse.iitm.ac.in/~jayalal/teaching/CS6840/2012/lecture11.pdf`.

52 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

53 Luca Trevisan. List-decoding using the XOR lemma. In *proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 126–135. IEEE Computer Society, 2003.

54 Luca Trevisan. Some applications of coding theory in computational complexity. *CoRR*, cs.CC/0409044, 2004.

55 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM*, 55(1):1:1–1:16, 2008.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il