Received: 12 July 2019 | Accepted: 14 October 2020 | Published on: 7 July 2021

DOI: 10.1002/rsa.21031

RESEARCH ARTICLE

WILEY

Erasures versus errors in local decoding and property testing

Sofya Raskhodnikova[1] | **Noga Ron-Zewi**[2] | **Nithin Varma**[2]

[1]Department of Computer Science, Boston University, Boston, Massachusetts, USA
[2]Department of Computer Science, University of Haifa, Haifa, Israel

Correspondence
Sofya Raskhodnikova, Department of Computer Science, Boston University, Boston, Massachusetts, USA.
Email: sofya@bu.edu

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. We first investigate 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, arbitrarily close to 1, with list sizes and query complexity better than in the Goldreich–Levin theorem. We further study *approximate* locally erasure list-decodable codes and use them to construct a property that is erasure-resiliently testable with query complexity independent of the input length, n, but requires $n^{\Omega(1)}$ queries for tolerant testing. We also investigate the general relationship between local decoding in the presence of errors and in the presence of erasures.

KEYWORDS

erasures versus errors, Goldreich–Levin theorem, Hadamard code, local decoding, property testing

1 | INTRODUCTION

The contributions of this work are twofold: 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.

A preliminary version [54] of this work has appeared in the proceedings of ITCS 2019.

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 [13, 24, 25, 49], probabilistically checkable proofs [2, 3, 5, 53], derandomization [6, 59, 60], and private information retrieval [16]. LDCs that work in the presence of errors have been extensively studied [5, 7, 8, 13, 20, 21, 24, 25, 53, 63]. The related notion of locally list-decodable codes (LLDCs) has also received a lot of attention [8, 30, 32, 34, 39, 43, 45, 59] and found applications in cryptography [30], learning theory [46], average-to-worst-case reductions [15, 31, 48], and hardness amplification and derandomization [6, 59]. 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 [35] and Guruswami and Indyk [36]. In particular, Guruswami [35] 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, local (unique and list) decoding from erasures has only been studied from the perspective of hardness amplification where the interest is in proving lower bounds on query complexity [4, 14, 33, 62].[1]

Motivated by applications in property testing [29, 58], we begin our investigation of effects of erasures with local *list*-decoding. Our first result is a local *erasure list-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 [30]. 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

[1]There is a related line of work on local list recovery [32, 40], where codeword positions are associated with sets of symbols. The goal, given oracle access to such a codeword, is to output a list of codewords such that for each codeword in the list, the symbol at each position is equal to one of the symbols from the set associated with that position. In these terms, an erased codeword position corresponds to its associated set being equal to the alphabet.

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 [26–28, 55, 57], to (error) tolerant testing [52] and erasure-resilient testing [19].

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 [19]. Fischer and Fortnow [23] separated standard and tolerant testing by describing a property that is *easy* to test in the standard model and *hard* to test tolerantly. Dixit, Raskhodnikova, Thakurta, and Varma [19] showed that the property defined by Fischer and Fortnow separates standard property testing from erasure-resilient testing in the same sense. Dixit, Raskhodnikova, Thakurta, and Varma [19] 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 erasure list-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, decoding tasks, and property testing models we study, and also statements and discussion of our main results.

1.1.1 | Local erasure list-decoding and the Hadamard code

In this article, 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 erasure list-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 ith bit of x.

Definition 1.2 (Locally erasure list-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 erasure list-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.

Item 2 in the above definition can be used to obtain a high probability worst-case bound on the query complexity of the algorithms, by incurring a constant factor loss in the query complexity expression. 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 [30] 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: $H_a(x) = (\sum_{i\in[n]} a_i \cdot x_i)$ *mod* 2 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 erasure list-decodability of the Hadamard code. It is an analogue of the Goldreich–Levin theorem [30] for corruptions in the form of erasures. We first state the Goldreich–Levin theorem and then state our result.

Theorem 1.4 (Goldreich–Levin theorem [30])**.** *There is a* $\left(\alpha, O(\frac{1}{(1/2-\alpha)^2}), O(\frac{1}{(1/2-\alpha)^2})\right)$*-local list-decoder for the Hadamard code that works for every* $\alpha \in [0, 1/2)$.

Theorem 1.5 (Local erasure list-decoder for Hadamard)**.** *There is a* $\left(\alpha, O(\frac{1}{1-\alpha}), O(\frac{1}{1-\alpha})\right)$*-local erasure list-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 erasure list-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 1.5 as compared to Goldreich–Levin is in the list size and the query complexity: from $\Theta(\frac{1}{(1/2-\alpha)^2})$ to $O(\frac{1}{1-\alpha})$. Such an improvement is impossible if we are decoding against errors as opposed to erasures. Specifically, for the list size, Blinovsky [12] and Guruswami and Vadhan [38] 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 $\Omega(\frac{1}{(1/2-\alpha)^2})$. For the query complexity, Theorem 1.4 is also optimal, as shown by Ron-Zewi, Shaltiel, and Varma [56] in a work subsequent to ours. Together with Theorem 1.5, these works give a separation between errors and erasures in the context of local list-decoding. Moreover, it follows from the works of Guruswami [35] and Ron-Zewi, Shaltiel, and Varma [56] that Theorem 1.5 is also optimal for both the list size and query complexity.

Finally, Observation 5.4 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 erasure list-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 quadratically better list size and query complexity in Theorem 1.5.

1.1.2 | 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 a string $x \in \{0, 1\}^n$ and a property $\mathcal{P} \subseteq \{0, 1\}^n$, the Hamming distance of x from $\mathcal{P}$ is equal to the minimum, over $y \in \mathcal{P}$, of the Hamming distance between x and y. A string $x \in \{0, 1\}^n$ is ε-far (α-close) from (to, respectively) a property $\mathcal{P} \subseteq \{0, 1\}^n$, if the Hamming distance of x from $\mathcal{P}$ is at least εn (at most αn, respectively).

Definition 1.6 (α-Erased strings and completions)**.** 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.

Definition 1.7 (Erasure-resilient tester)**.** An *α-erasure-resilient ε-tester* [19] for a property $\mathcal{P}$ is a randomized algorithm that, given parameters $\alpha \in [0, 1), \varepsilon \in (0, 1)$ such that $\alpha+\varepsilon < 1$ 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 every completion of x is ε-far from $\mathcal{P}$.[2] 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 the input length n).

For the special case with no erasures, that is, when $\alpha = 0$, we refer to the algorithm above as an ε-tester.

[2]The rejection condition in this definition of erasure-resilient testing is differently parameterized than that in the definition due to Dixit, Raskhodnikova, Thakurta, and Varma [19]. We use the current definition as it gives cleaner query complexity expressions and is consistent with the definition of erasure-resilient graph property testing defined by Levi, Pallavoor, Raskhodnikova, and Varma [47]. We refer the interested reader to Appendix A for a comparison of the two definitions.

Definition 1.8 (Tolerant tester)**.** An (α, ε')-*tolerant tester* [52] 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 on the input length n).

Comparison of parameters

We remark that, while comparing the two models, one possibility is to compare $(\alpha, \alpha + \varepsilon)$-tolerant testing of a property $\mathcal{P}$ with α-erasure-resilient ε-testing of $\mathcal{P}$ for the same values of $\alpha \in [0,1)$ and $\varepsilon \in (0,1)$ such that $\alpha + \varepsilon < 1$. 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 every completion of an input string, one needs to change at least an ε fraction of the completion to make it satisfy $\mathcal{P}$. Similarly, an $(\alpha, \alpha + \varepsilon)$-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 additional ε 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 1.9 (Separation)**.** *There exists a property $\mathcal{P}$ and constants $\alpha, \varepsilon \in (0,1)$ such that*

- *$\mathcal{P}$ is α-erasure-resiliently ε-testable;*
- *$\mathcal{P}$ is not $(\alpha, \alpha + \varepsilon)$-tolerantly testable.*

Approximate local erasure list-decoding and strengthened separation

We obtain a separation better than in Theorem 1.9 with the help of a variant of LLEDCs, called approximate locally erasure list-decodable codes (ALLEDC). An approximate local erasure list-decoder is identical to a local erasure list-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)-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 5.2) 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 [9, 41] to obtain efficient ALLEDCs. We use them and get our strengthened separation.

Theorem 1.10 (Strengthened separation)**.** *There exists a property $\mathcal{P}'$ and constants $\alpha, \varepsilon \in (0,1)$ such that*

- *$\mathcal{P}'$ is α-erasure-resiliently ε-testable;*
- *every $(\alpha, \alpha + \varepsilon)$-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 1.11 (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 an α fraction of erasures, makes at most q queries to w and outputs x_i with probability at least $\frac{2}{3}$.

A (α, q)-LDC is defined similarly to an (α, q)-LEDC except that the input word w contains at most an α fraction of errors instead of erasures. We observe (Observation 7.4) 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 1.12. *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 $(\frac{\alpha}{O(3^q)}, O(3^q))$-locally decodable.*

To prove Theorem 1.12, we start with a local erasure-decoder for $\{C_n\}_{n\in\mathbb{N}}$ and transform it to be a nonadaptive and smooth local erasure-decoder, where this transformation uses ideas developed by Katz and Trevisan [42]. An algorithm is nonadaptive if its queries do not depend on the answers to the previous queries. A decoding algorithm is smooth if it decodes uncorrupted codewords by querying nearly uniformly distributed codeword indices. We first make the local erasure-decoder for $\{C_n\}_{n\in\mathbb{N}}$ nonadaptive. We then show that every nonadaptive decoding algorithm for an LEDC can be transformed into a smooth decoding algorithm. We then use this "smoothness" feature to show that the code family is locally decodable from a smaller fraction of errors than erasures.

The technique outlined above cannot be directly used to obtain an analog of Theorem 1.12 for the case of local list-decoding since the notion of smoothness (the way we define it for use in our transformation) does not make sense in the local list-decoding setting. Smooth local decoding assumes oracle access to an uncorrupted codeword and the goal is to decode the message by making nearly uniformly distributed queries. Local list-decoding, however, is relevant in the setting that a codeword has a higher number of corrupt bits than the unique decoding radius.

We remark 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 [44, 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 [44, 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 erasure

list-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, would 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 LDC that works in the presence of erasures implies the existence of a LDC 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). This raises the question about the power of such an assumption.

In our work, we show the existence of a property $\mathcal{P}$ and parameters $\alpha, \varepsilon \in (0,1)$ satisfying $\alpha < \varepsilon$ and $\alpha + \varepsilon < 1$ such that $\mathcal{P}$ has an efficient α-erasure-resilient ε-tester but no efficient $(\alpha, \alpha+\varepsilon)$-tolerant tester. In the work that introduced the erasure-resilient testing model, Dixit, Raskhodnikova, Thakurta, and Varma [19] prove that for some range of parameters, tolerant testing is at least as hard as erasure-resilient testing.

Observation 1.13 (Dixit, Raskhodnikova, Thakurta, and Varma [19])**.** Let $\alpha, \varepsilon \in (0,1)$ be such that $\alpha < \varepsilon$. If there is an (α, ε)-tolerant tester with query complexity q for a property $\mathcal{P}$, then there is an α-erasure-resilient ε-tester for $\mathcal{P}$ with query complexity q.

Observation 1.13 does not rule out the existence of an (α, ε)-tolerantly testable property that is not α-erasure-resiliently ε'-testable for $\varepsilon' < \varepsilon$. It would be an interesting direction to explore the exact relationship between the two models for the above range of parameters.

Organization

The article is organized as follows. Section 2 defines some of the notation that will be used throughout the article. Our local erasure list-decoder for the Hadamard code is presented in Section 3. Next, in Section 4, we show our separation result (Theorem 1.9) based on the Hadamard code. Section 5 contains our transformation from approximate local list-decoding to approximate local erasure list-decoding. In Section 6, we show our strengthened separation result (Theorem 1.10) implied by the resulting approximate local erasure list-decoding algorithm. Finally, in Section 7, we detail our transformation from local erasure (unique) decoding to local (unique) decoding. Appendix A contains a comparison of the erasure-resilient model that we adopt in this article with that of the original definition proposed by Dixit, Raskhodnikova, Thakurta, and Varma [19].

2 | PRELIMINARIES

In this section, we define some of the notation used in the article. We use $\mathbb{F}_2$ to denote the finite field of characteristic 2 that contains the elements 0 and 1. Given $a, b \in \mathbb{F}_2$, we use $a + b$ to denote the addition of a and b modulo 2. Let $n \in \mathbb{N}$. For $x \in \mathbb{F}_2^n$ and $i \in [n]$, we use x_i to denote the ith coordinate of x. Given $x, y \in \mathbb{F}_2^n$, we use $x \oplus y$ to denote the element of $\mathbb{F}_2^n$ whose ith entry is $x_i + y_i$. Let $e_k \in \mathbb{F}_2^n$ for $k \in [n]$ denote the kth standard basis vector, and let $\vec{0} \in \mathbb{F}_2^n$ denote the zero vector. Since a function can be represented by a string of evaluations over points in its domain, we often view a codeword of the Hadamard code $\mathcal{H}_n$ (see Definition 1.3) as the string of all evaluations of a linear function mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$. A function f is α-erased, if f evaluates to $\perp$ on at most an α fraction of its domain.

An α-erased string $x \in \{0,1,\perp\}^n$ is ε-far from a property $\mathcal{P} \subseteq \{0,1\}^n$ if every completion (see Definition 1.6) of x is ε-far from $\mathcal{P}$. In other words, there is no way to complete x to a string that satisfies $\mathcal{P}$ without changing at least $\varepsilon \cdot |x|$ nonerased values in x. For strings $x \in \{0,1,\perp\}^n$ and $y \in \{0,1\}^n$, the Hamming distance between x and y is defined to be the minimum number of nonerased values in x that need to be changed in order for it to be completable to y.

3 | LOCAL ERASURE LIST-DECODING OF THE HADAMARD CODE

In this section, we describe a local erasure list-decoder for the Hadamard code and prove Theorem 1.5. We follow the style of the proof of the Goldreich–Levin theorem given in a tutorial by Trevisan [61] on the applications of coding theory to complexity.

Proof of Theorem 1.5. Our local erasure list-decoder, described in Algorithm 1, gets a parameter $\alpha \in [0,1)$ as its input and has oracle access to an 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 the Hadamard code $\mathcal{H}_n$).

We now analyze Algorithm 1. Recall that for a string $a \in \mathbb{F}_2^n$, the function $H_a : \mathbb{F}_2^n \to \mathbb{F}_2$ denotes the Hadamard encoding of a (see Definition 1.3). We will show that, with probability at least $2/3$, for every $a \in \mathbb{F}_2^n$ such that the functions H_a and f agree with each other on all the nonerased points, one of the local decoders output by Algorithm 1 implicitly computes a (see Definition 1.1).

There exists some iteration of Step 2 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 8 and 3 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 $2/3$. Consider the first set $S' \subseteq [t]$ (in the order that A considers sets) such that

Algorithm 1. Local erasure list-decoder for the Hadamard code

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

▷ Let $t \leftarrow \lceil \log_2(\frac{12}{1-\alpha}) \rceil$.

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

$f(x \oplus z_{S'}) \neq \perp$. According to the description of A,

$$\begin{aligned} A(x) &= \left(\bigoplus_{j \in S' \cap B} b_j \right) + \left(\bigoplus_{j \in S' \cap ([t] \setminus B)} f(z_j) \right) + f(x \oplus z_{S'}) \\ &= \left(\bigoplus_{j \in S' \cap B} H_a(z_j) \right) + \left(\bigoplus_{j \in S' \cap ([t] \setminus B)} H_a(z_j) \right) + H_a(x \oplus z_{S'}) \\ &= \left(\bigoplus_{j \in S'} H_a(z_j) \right) + H_a(x) + \left(\bigoplus_{j \in S'} H_a(z_j) \right) = H_a(x). \end{aligned}$$

The second equality above holds as $b_i = H_a(z_i)$ for all $i \in B$, and $H_a(y) = f(y)$ for all nonerased $y \in \mathbb{F}_2^n$. The third equality holds because H_a, being a linear function, satisfies $H_a(y \oplus y') = H_a(y) + H_a(y')$ for all $y, y' \in \mathbb{F}_2^n$.

It remains to show that, with probability at least $2/3$, there exists some set $S \subseteq [t]$ such that $f(x \oplus z_S) \neq \perp$. Let $\alpha^\star \leq \alpha$ denote the fraction of erasures in f. For each $S \subseteq [t]$, we have that $f(x \oplus z_S) \neq \perp$ with probability $1 - \alpha^\star$, since x (and therefore, $x \oplus z_S$) is uniformly distributed in $\mathbb{F}_2^n$. Define indicator random variables $Z_S = \mathbb{1}(f(x \oplus z_S) \neq \perp)$ for $S \subseteq [t]$ and let $Z = \sum_{S \subseteq [t]} Z_S$. The random variable Z is equal to the number of nonerased values among $f(x \oplus z_S)$ for $S \subseteq [t]$. The event that $\forall S \subseteq [t], f(x \oplus z_S) = \perp$ is equivalent to the event that $Z < 1$.

For each $S \subseteq [t]$, we have $\mathbb{E}[Z_S] = 1 - \alpha^\star$. Therefore, by the linearity of expectation,

$$\mathbb{E}[Z] = \sum_{S \subseteq [t]} \mathbb{E}[Z_S] = 2^t (1 - \alpha^\star).$$

For every 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 $\mathbb{F}_2^n$. Thus, the collection $\{x \oplus z_S | S \subseteq [t]\}$ is pairwise independent, and hence the random variables Z_S for $S \subseteq [t]$ are also pairwise independent. Now, for each $S \subseteq [t]$, we have $\mathrm{Var}(Z_S) = (1 - \alpha^\star) \cdot \alpha^\star$, and by the pairwise independence,

$$\mathrm{Var}[Z] = \sum_{S \subseteq [t]} \mathrm{Var}[Z_S] = 2^t \cdot \alpha^\star (1 - \alpha^\star).$$

Applying the Chebyshev's inequality,

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

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 $\frac{2}{3}$.

Finally, we prove that T implicitly computes $a \in \mathbb{F}_2^n$ and that the expected number of queries that T makes to f is $O(\frac{1}{1-\alpha})$. It is clear that the output of T on input $k \in [n]$ is always $a[k] = H_a(y \oplus e_k) + H_a(y) = H_a(e_k)$. The number of queries made by T to A is a geometric random variable with success probability at least $1/3$. 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 at most $3 \cdot 2^t$, which is at most $\frac{72}{1-\alpha}$. The number of algorithms whose descriptions are generated is also at most 2^t, which is at most $\frac{24}{1-\alpha}$. ∎

4 | 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 1.9. In fact, we prove the following (more general) statement and show that it implies Theorem 1.9.

Theorem 4.1. *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 $\varepsilon \in (\frac{3\varepsilon^\star}{4}, 1)$ such that $\alpha + \varepsilon < 1$, the property $\mathcal{P}$ can be α-erasure-resiliently ε-tested using $O(\frac{1}{\varepsilon})$ 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 $\tilde{\Omega}(\log N)$.*

4.1 | 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 [29, 58], a probabilistically checkable proof system (PCP of proximity [10, 18, 22]) for the problem of testing $\mathcal{R}$, and the Hadamard code. We discuss them below. The idea of using PCPs of proximity in separating the two property testing models comes from the work of Fischer and Fortnow [23]. 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 [11] 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 4.2 ([11])**.** *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 $\Omega(n)$.*

An important ingredient in the description of the separating property $\mathcal{P}$ is a probabilistically checkable proof system for property testing problems. The notion of proof assisted property testing was introduced by Ergün, Kumar, and Rubinfeld [22]. Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [10], and Dinur and Reingold [18] defined and studied a special case of proof-assisted property testers called PCPs of proximity (or alternatively, assignment testers). PCPs of proximity were further studied by Dinur [17] and Meir [50, 51].

Definition 4.3 (PCP of proximity [10, 18, 22])**.** Given 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 $\varepsilon \in (0,1]$, 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 [17, Corollary 8.4] implies that there are *efficient* PCPPs (over a small constant alphabet Σ) for testing properties (over Σ) that are decidable using polynomial-sized circuits. The following restatement of this result is obtained by representing the symbols in Σ using the binary alphabet.

Lemma 4.4 ([17]). *If $P_n \subseteq \{0,1\}^n$ is a property decidable by a circuit of size $s(n)$, then there exists a randomized algorithm V' that gets oracle access to a string $y \circ \pi \in \{0,1\}^*$, where $y \in \{0,1\}^n$ is the input and π is a proof of length at most $s(n) \cdot \text{polylog } s(n)$, and satisfies the following:*

- *if $y \in P_n$, then for some proof π, the algorithm V' always accepts $y \circ \pi$;*
- *if $y \notin P_n$, then for every π, the algorithm V' rejects $y \circ \pi$ with probability proportional to the relative Hamming distance of y from P_n.*

Moreover, V' makes a constant number of nonadaptive queries.

An algorithm guaranteed by Lemma 4.4 for a property P can be converted to an efficient PCPP for P by simply repeating the former algorithm sufficiently many times.

Lemma 4.5 ([17]). *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 4.6 uses Lemma 4.5 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 4.6. *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 $cn \cdot \text{polylog } n$, 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 4.2) 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 4.5. ∎

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 $\mathcal{R}_{\phi_n}$, 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 4.6 accept.

Definition 4.7 (Separating property $\mathcal{P}$). Let $\varepsilon^\star \in (0,1)$ and $\aleph \subseteq \mathbb{N}$ be as in Lemma 4.2. For $n \in \aleph$, let $p(n) \leq cn \cdot \text{polylog } n$ denote the length of proof that the algorithm V in Claim 4.6 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 4.2.
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 4.6) accepts when given oracle access to y and $\pi(y)$.

4.2 | Proof of Theorem 4.1

In this section, we prove Theorem 4.1, which in turn implies Theorem 1.9. Lemmas 4.8 and 4.12 prove the first and second parts of Theorem 4.1, 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 4.7) of an input string $x \in \{0,1\}^N$ using the local erasure list-decoder guaranteed by Theorem 1.5. 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 4.7) consists of repetitions of y, and $\pi(y)$ is a proof string such that the algorithm V (from Claim 4.6) 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 erasure list-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 4.6 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 a 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 $\mathcal{P}$. Recall that 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 $\varepsilon \cdot |x|$ nonerased values in x.

Lemma 4.8. *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 4.2. For every $\alpha \in \left[0, \frac{3\varepsilon^\star}{16}\right)$ and $\varepsilon \in \left(\frac{3\varepsilon^\star}{4}, 1\right)$ such that $\alpha + \varepsilon < 1$, the property $\mathcal{P}$ can be α-erasure-resiliently ε-tested using $O\left(\frac{1}{\varepsilon}\right)$ queries.*

Proof. The erasure-resilient tester for $\mathcal{P}$ is described in Algorithm 2. The query complexity of the tester is $O(1/\varepsilon)$ as 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 4.2. Fix $n \in \aleph$ and let $p(n)$ and N be as in Definition 4.7. 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 4.7, we refer to the substring $x[1 \ldots sn]$ as the plain part of x and the substring $x[sn+1 \ldots N]$ as the encoded part of x.

Assume that $x \in \mathcal{P}$. By this assumption, we can see that there exists a string $y \circ \pi \in \{0,1\}^{n+p(n)}$ such that (1) $y \in \mathcal{R}_{\phi_n}$ and the plain part of x can be completed to a repetition of y, (2) π is a proof such that the algorithm V (from Claim 4.6) accepts when given oracle access to $y \circ \pi$, and (3) the encoded part of x can be completed to the Hadamard encoding of $y \circ \pi$. Since $\alpha < 3\varepsilon^\star/16$ and the length of the encoded part is equal to $N - sn = N \cdot \varepsilon^\star/4$, the fraction of erasures in the encoded part of x is less than $(3\varepsilon^\star/16)/(\varepsilon^\star/4)$, which is equal to $3/4$. Hence, by Theorem 1.5, with probability at least $2/3$, there exists an algorithm T_k computed in Step 2 of Algorithm 2 such that T_k implicitly computes the string $y \circ \pi \in \{0,1\}^{n+p(n)}$. Therefore, k is not discarded in either Step 7 or Step 10. 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 first show that, conditioned on E, the tester rejects with probability at least $4/5$.

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 $x \in \{0,1,\perp\}^N$

▷ Set $s \leftarrow (\frac{4}{\varepsilon^\star} - 1) \cdot \frac{2^{(n+p(n))}}{n}$, $\varepsilon' \leftarrow \frac{\varepsilon}{3}$, $q \leftarrow 288$, $L \leftarrow 96$.

▷ Set $Q \leftarrow 10q + 10qL \cdot \left(\left\lceil \frac{9 \log L}{\varepsilon} \right\rceil + \lceil 4 \log L \rceil \cdot \frac{3C}{\varepsilon} \right)$, where C is the constant in the O notation of Claim 4.6.

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

2: **Run** a $(\frac{3}{4}, q, L)$-local erasure list-decoder for the Hadamard code (Algorithm 1) with oracle access to $x[sn+1..N]$, the encoded part of x. ▷ Note that q and L are constants for local list-decoding from at most a $3/4$ fraction of erasures, and the specific values given here follow from the proof of Theorem 1.5.

▷ Let $T_1, T_2, \ldots, T_L$ be the list of algorithms returned in the above step.

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

▷ 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.

4: **repeat** $\left\lceil \frac{9 \log L}{\varepsilon} \right\rceil$ times:

5: Pick $a \in_R [n], i \in_R [s]$.

6: **if** $x[(i-1)n + a] \neq \perp$ and $T_k(a) \neq x[(i-1)n + a]$ **then**

7: **Discard** the current k

▷ 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.

8: **repeat** $\lceil 4 \log L \rceil$ times:

9: Run V, from Claim 4.6, with input ε' and oracle access to T_k.

10: **Discard** the current k if V rejects.

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

Claim 4.9. *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. Since x is ε-far from satisfying $\mathcal{P}$, at least εN nonerased values in x need to be changed in order to complete it to a string satisfying $\mathcal{P}$. The length $\frac{\varepsilon^\star}{4} \cdot N$ of the encoded part of x is an upper bound on the number of nonerased values in the encoded part, and therefore, it is at most $\varepsilon N/3$ since $\varepsilon \in (\frac{3\varepsilon^\star}{4}, 1)$. Thus, the plain part of x needs to be changed in at least $2\varepsilon N/3$ nonerased values in order for it to be s repetitions of a string $y \in \mathcal{R}_{\phi_n}$. The claim follows. ∎

From Claim 4.9, it follows that at least $\frac{2\varepsilon \cdot sn}{3}$ 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 4.10. *For any $y \in \{0,1\}^n$, if the plain part of x can be changed to s repetitions of y by modifying less than $\frac{\varepsilon \cdot sn}{3}$ nonerased values, then y is $\frac{\varepsilon}{3}$-far from $\mathcal{R}_{\phi_n}$.*

Proof. Consider $y \in \{0,1\}^n$ such that we can change less than $\varepsilon \cdot sn/3$ 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 $\varepsilon \cdot n/3$. 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 $\varepsilon \cdot sn/3$ nonerased points) and then modifying at most $s \cdot \varepsilon \cdot n/3$ nonerased points to make it s repetitions of y'. In other words, $x[1 \ \ldots \ sn]$

can be modified in less than $2\varepsilon \cdot sn/3$ nonerased points to make it s repetitions of a string y' in $\mathcal{R}_{\phi_n}$. This contradicts Claim 4.9. ∎

Fix $k \in [L]$, where L is the number of algorithms returned by the local erasure list-decoder. 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 $\frac{\varepsilon \cdot sn}{3}$ nonerased points in the plain part of x for it to become s repetitions of y'. We show that in this case, Steps 4–7 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 $\varepsilon \cdot sn/3$ nonerased points in the plain part of x for it to become s repetitions of y', there are at least $\varepsilon \cdot sn/3$ witnesses in the plain part of x. In each iteration of Steps 4–7, the point selected is a witness with probability at least $\frac{\varepsilon \cdot sn}{3sn} = \frac{\varepsilon}{3}$. Thus, the probability that Algorithm 2 does not find a witness (and does not discard k) in $\lceil \frac{9 \log L}{\varepsilon} \rceil$ iterations is at most

$$\left(1 - \frac{\varepsilon}{3}\right)^{\frac{9 \log L}{\varepsilon}} \leq \left(1 - \frac{\varepsilon}{3}\right)^{\frac{3 \log(10) \cdot \log(L)}{\varepsilon}} \leq \frac{1}{10L},$$

where we have used the inequality $3 \log(10) \leq 9$.

Case II: In this case, we assume that we can change less than $\varepsilon \cdot sn/3$ nonerased points in the plain part of x and make it s repetitions of y'. Then, by Claim 4.10, y' is $\varepsilon/3$-far from $\mathcal{R}_{\phi_n}$. Let $\varepsilon' = \frac{\varepsilon}{3}$. By Claim 4.6, for every proof $\pi \in \{0,1\}^{p(n)}$, the algorithm V (from Claim 4.6), 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 8–10 is at most

$$\left(1 - \frac{2}{3}\right)^{4 \log L} \leq \left(1 - \frac{2}{3}\right)^{(3/2) \cdot \log(10) \cdot \log(L)} \leq \frac{1}{10L}.$$

Therefore, the probability that the tester fails to discard k is at most $\frac{1}{10L} + \frac{1}{10L} \leq \frac{1}{5L}$. 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. The number of queries made in Step 2 is at most q. For all $k \in [L]$, the expected number of queries that each invocation of the algorithm T_k makes is at most q. Hence, the expected number of queries made in Steps 4–7 is at most $L \cdot \left(\left\lceil \frac{9 \log L}{\varepsilon} \right\rceil \cdot q \right)$.

By Claim 4.6, the number of queries made by the algorithm V (from Claim 4.6) on input $\varepsilon' = \frac{\varepsilon}{3}$ and oracle access to T_k, is at most $\frac{3C}{\varepsilon}$, where C is the constant in the O notation of Claim 4.6. Thus, the expected number of queries made in Steps 8–10 by Algorithm 2 is at most $L \cdot \left(\lceil 4 \log L \rceil \cdot q \cdot \frac{3C}{\varepsilon} \right)$.

Therefore, the expected total number of queries made by the tester is at most

$$q + qL \cdot \left(\left\lceil \frac{9 \log L}{\varepsilon} \right\rceil + \lceil 4 \log L \rceil \cdot \frac{3C}{\varepsilon} \right).$$

Hence, the probability that the number of queries exceed Q (as defined in Algorithm 2) is at most $1/10$ by the Markov's inequality. Thus, the probability that the tester accepts x that is ε-far from $\mathcal{P}$ is at most $1/10 + 1/5 \leq 1/3$.

Remark 4.11. We point out that the local erasure list-decoder (Algorithm 1) used in Algorithm 2 can be replaced by the local erasure list-decoder obtained by applying Observation 5.4 to the Goldreich–Levin theorem by incurring only a constant factor loss in the query complexity of Algorithm 2.

Lemma 4.12. *Let* $\varepsilon^\star \in (0,1)$ *be as in Lemma 4.2. For every* $\alpha \in (\frac{\varepsilon^\star}{8}, 1)$ *and* $\varepsilon' \in \left(\alpha, \varepsilon^\star - \frac{(\varepsilon^\star)^2}{4}\right)$, *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 4.2. 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 4.7. Let s denote $\left(\frac{4}{\varepsilon^\star} - 1\right) \cdot \frac{2^{n+p(n)}}{n}$.

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. Recall that we refer to the substring $x[1 \ldots sn]$ as the plain part of x and the substring $x[sn+1 \ldots N]$ 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.

We will show that 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 a $1/2$ 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 $\mathcal{P}$. Since $sn/N = (1 - \frac{\varepsilon^\star}{4})$ as observed above, the relative Hamming distance of x from $\mathcal{P}$ is at least $\frac{\varepsilon^\star \cdot sn}{N} = \varepsilon^\star - \frac{(\varepsilon^\star)^2}{4}$. That is, 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 1.9. Theorem 4.1 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 1.9, 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 $(\alpha, \alpha+\varepsilon)$-tolerantly testable. In other words, the constraints imposed on $\alpha, \varepsilon, \varepsilon'$ must have a solution for the setting of $\varepsilon' = \varepsilon + \alpha$.

$$\frac{\varepsilon^\star}{8} < \alpha < \frac{3\varepsilon^\star}{16}; \qquad \frac{3\varepsilon^\star}{4} < \varepsilon < 1;$$

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

For every $0 < \varepsilon^\star < 1/100$, the value $\varepsilon^\star - (\varepsilon^\star)^2/4$ is strictly greater than $\varepsilon^\star - \varepsilon^\star/400 = 399\varepsilon^\star/400$. For $\alpha = \varepsilon^\star/6$ and $\varepsilon = 4\varepsilon^\star/5$, which satisfy the first two inequalities, we can see that $\alpha + \varepsilon =$

$29\varepsilon^\star/30 < 399\varepsilon^\star/400 < \varepsilon^\star - (\varepsilon^\star)^2/4$. Thus there exists $\alpha, \varepsilon \in (0,1)$ satisfying $\alpha + \varepsilon < 1$ such that $\mathcal{P}$ is α-erasure-resiliently ε-testable, but not $(\alpha, \alpha+\varepsilon)$-tolerantly testable. Theorem 1.9 follows. ∎

5 | APPROXIMATE LOCAL ERASURE LIST-DECODING

In this section, we prove the existence of an approximate locally erasure list-decodable code (ALLEDC) with inverse polynomial rate. Our starting point is an ALLDC due to Impagliazzo, Jaiswal, Kabanets, and Wigderson [41]. 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 5.1 ([41] as restated by [9]). *For every $\gamma, \beta > 0$, there exist 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 5.2. 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 erasure list-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 $5/6$. This can be ensured by running A twice and outputting the concatenation of lists obtained in both iterations (the resulting algorithm succeeds if one of the iterations succeed). The approximate local erasure list-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. In other words, for $b \in \{0,1\}$, if b is the value that w' takes in least half the erased points in w, then w' and w_b disagree on at most an α 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 $\Pr[\overline{E_1} \cup \overline{E_2}] \le \frac{1}{3}$. Hence, A' is a $(2\alpha, \beta, 4q, 4L)$-approximate local erasure list-decoder for C_k. ∎

Applying Observation 5.2 to Theorem 5.1, we get the ALLEDCs that we need.

Lemma 5.3. *Let $c_3 > 0$ be a constant. For every $\gamma, \beta > 0$, there exist 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 erasure list-decodable.*

The following is a corollary of Observation 5.2.

Observation 5.4. 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 erasure list-decodable.

6 | 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 1.10. The following theorem implies Theorem 1.10.

Theorem 6.1. *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 $\alpha \in (0, \frac{\varepsilon^\star}{57\ 600 \cdot c_2})$ such that $\alpha + \varepsilon < 1$, property $\mathcal{P}'$ can be α-erasure-resiliently ε-tested using $O(\frac{1}{\varepsilon})$ queries.*
- *For every $\alpha \in (\frac{\varepsilon^\star}{57\ 600 \cdot c_2 + 2\varepsilon^\star}, 1)$ and $\varepsilon' \in \left(\alpha, \frac{28\ 800 \cdot c_2 \cdot \varepsilon^\star}{28\ 800 \cdot c_2 + \varepsilon^\star}\right)$, every (α, ε')-tolerant tester for $\mathcal{P}'$ on inputs of length N has query complexity $N^{\Omega(1)}$.*

6.1 | 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 4.7). 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 4.6 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 "smooth" PCPP accept. In addition, the encoding uses an ALLEDC (from Section 5) instead of the Hadamard code.

We first describe the "smooth" PCPP used in our construction. The following lemma by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [10] and Guruswami and Rudra [37, Lemma 5] states that algorithms making nonadaptive queries can be transformed into algorithms that make nearly uniform queries.

Lemma 6.2 ([10, 37])**.** *Let $n \in \mathbb{N}$. Consider a nonadaptive algorithm T that gets oracle access to strings from $\{0,1\}^n$. There exists a mapping $\varphi_T : \{0,1\}^n \to \{0,1\}^{n'}$ and an algorithm T' satisfying the following:*

- *For every $x \in \{0,1\}^n$, the distribution on outcomes of T with oracle access to x is identical to the distribution on outcomes of T' with oracle access to $\varphi_T(x)$. Moreover, $3n < n' \leq 4n$, and the number of queries that T' makes to $\varphi_T(x)$ is at most twice the number of queries that T makes to x.*
- *Upon 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 4.4 with Lemma 6.2 (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 "smooth" PCPP for $\mathcal{R}$.

Lemma 6.3 (Smooth PCPP)**.** *Let $c_1 > 0, c_2 > 1$ be fixed constants. Let $n \in \mathbb{N}$. 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 $c_1 n \cdot \operatorname{poly}\log n$, 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 $1/n$;*

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

Proof. Let $c > 0$ be the constant from Claim 4.6. Consider the algorithm V' guaranteed by Lemma 4.4 for the property $\mathcal{R}_{\phi_n}$. The algorithm V' gets oracle access to the concatenation of an input $y \in \{0,1\}^n$ and a proof $\pi\prime \in \{0,1\}^{p'(n)}$, where $p'(n) \leq cn \cdot \operatorname{poly}\log n$.

We now describe an algorithm V'' that, on oracle access to a string $y \circ \pi''$, where $y \in \{0,1\}^n$ and $\pi'' \in \{0,1\}^{n+p\prime(n)}$, and does the following:

1. Sample a uniformly random $i \in [n]$ and **reject** if $y[i] \neq \pi''[i]$.
2. Simulate V' with oracle access to π'' and **reject** if V' rejects.
3. **Accept** if neither of the above events happen.

We prove the following claim about the algorithm V''.

Claim 6.4. *V'' is an algorithm satisfying:*

- *if $y \in \mathcal{R}_{\phi_n}$, then for some proof π'', the algorithm V' always accepts $y \circ \pi''$;*
- *if $y \notin \mathcal{R}_{\phi_n}$, then for every π'', the algorithm V' rejects $y \circ \pi''$ with probability proportional to the relative Hamming distance of y from $\mathcal{R}_{\phi_n}$.*

Proof. Assume $y \in \mathcal{R}_{\phi_n}$. There exists a proof π' of length at most $cn \cdot \operatorname{poly}\log n$ such that the algorithm V' accepts when given oracle access to $y \circ \pi'$. Therefore, algorithm V'' accepts if given oracle access to $y \circ \pi''$, where $\pi'' = y \circ \pi'$.

Next, assume that $y \notin \mathcal{R}_{\phi_n}$. Let δ be the relative Hamming distance of y from $\mathcal{R}_{\phi_n}$. Fix $\pi\prime\prime \in \{0,1\}^{n+p\prime(n)}$. Let δ' be the relative Hamming distance of y from the string y' obtained by considering the first n bits of π''. Step 1 of the algorithm V'' rejects with probability δ', since, for a uniformly random index $i \in [n]$, we have that $y[i] \neq y'[i]$ with probability δ'. If $\delta' \geq \delta/2$, then Step 1 of algorithm V'' rejects with probability at least $\delta/2$. If $\delta' < \delta/2$, then the relative Hamming distance of y' from $\mathcal{R}_{\phi_n}$ has to be greater than $\delta/2$; otherwise, the distance of y from $\mathcal{R}_{\phi_n}$ is less than δ, which is a contradiction. If y' has distance at least $\delta/2$ from $\mathcal{R}_{\phi_n}$, for every string $z \in \{0,1\}^{p'(n)}$ that forms the last $p'(n)$ bits of π'', the algorithm V' with oracle access to $\pi'' = y' \circ z$ rejects with probability $\Omega(\delta)$. That is, Step 2 of V'' rejects with probability $\Omega(\delta)$. ∎

We can think of V'' as running two algorithms V_1 and V_2, where V_1 makes the input queries of V'' and V_2 makes the proof queries of V''. We observe that the query distribution of V_1 is uniform over the input part. By applying Lemma 6.2 to V_2 we obtain a mapping $\varphi : \{0,1\}^* \to \{0,1\}^*$ and an algorithm V_2' such that each query of V_2' is to a particular location in the string $\varphi(\pi'')$ with probability at most $2/|\varphi(\pi'')|$. By Lemma 6.2, we also have: $|\varphi(\pi'')| \leq 4|\pi''|$.

Let $p(n)$ denote $|\varphi(\pi'')|$, where $\pi\prime\prime \in \{0,1\}^{n+p\prime(n)}$. Consider the algorithm V''' that runs V_1 and V_2' using a common random string with oracle access to a string $y \circ z$, where $y \in \{0,1\}^n$ and $z \in \{0,1\}^{p(n)}$, and rejecting whenever V'' rejects based on the query answers. In addition, V''' also rejects if the answers to its queries to z are not consistent with any string in the image of φ. Observe that V'' can check this condition, since it completely knows the mapping φ, which is fully determined by V_2 (by Lemma 6.2).

If $y \in \mathcal{R}_{\phi_n}$, then there exists a proof π'' such that V'' accepts $y \circ \pi''$, implying that for the same π'', the algorithm V''' accepts $y \circ \pi$, where $\pi = \varphi(\pi'')$. If $y \notin \mathcal{R}_{\phi_n}$, then for every proof π'', the algorithm V'' rejects $y \circ \pi''$ with probability proportional to the relative Hamming distance of y from $\mathcal{R}_{\phi_n}$. This implies that for every proof π, the algorithm V''' rejects $y \circ \pi$ with probability proportional to the relative Hamming distance of y from $\mathcal{R}_{\phi_n}$.

On input $\varepsilon \in (0,1)$, the algorithm V guaranteed by the statement of the lemma repeats for $\Theta(1/\varepsilon)$ time, the algorithm V'''. The acceptance and rejection guarantees of V are immediate. Note also that the distribution of a single input or proof query does not change by repetition. The lemma follows. ∎

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 6.5 (Separating property $\mathcal{P}'$)**.** Let $\aleph$, $\{\mathcal{R}_{\phi_n}\}_{n\in\aleph}$ and $\varepsilon^\star \in (0,1)$ be as in Lemma 4.2. Let $c_1 > 0$, $c_2 > 1$ be as in Lemma 6.3. Let $c_3 > 0$ be as in Lemma 5.3. Let $m = \frac{28\ 800\cdot c_2}{\varepsilon^\star}$, $\gamma = \frac{1}{2} + \frac{\varepsilon^\star}{57\ 600\cdot c_2}$, 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 $p(n) \le c_1 \cdot n \cdot \text{polylog}\, n$ denote the length of a valid proof that makes the algorithm V from Lemma 6.3 accept. Let $f(\cdot,\cdot)$ be as in Lemma 5.3. 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 5.3.

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 $y \in \{0,1\}^n$, where $y \in \mathcal{R}_{\phi_n}$ of length n.
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 6.3 accepts when given oracle access to $y\circ\pi$. The second half of the encoded part is the complement of its first half.

6.2 | Proof of strengthened separation

In this section, we prove Theorem 6.1. Lemmas 6.6 and 6.10 together imply the first and second parts of Theorem 6.1, respectively. The high level idea of the proof of Lemma 6.6 is very similar to that of Lemma 4.8. 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 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 smooth PCPP verifier rejects.

Lemma 6.6. *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 4.2 and $c_2 > 1$ be as in Lemma 6.3. For every $\varepsilon \in \left(\frac{\varepsilon^\star}{8}, 1\right)$ and $\alpha \in \left(0, \frac{\varepsilon^\star}{57\ 600\cdot c_2}\right)$ such that $\alpha + \varepsilon < 1$, the property $\mathcal{P}'$ is α-erasure-resiliently ε-testable using $O\left(\frac{1}{\varepsilon}\right)$ queries.*

Proof. We first show that Algorithm 3 accepts, with probability at least $3/5$, strings satisfying $\mathcal{P}'$ and rejects, with probability at least $3/5$, strings that are ε-far from $\mathcal{P}'$. The success probability can be

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 $x \in \{0,1,\perp\}^N$

▷ 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 $L \leftarrow \frac{c_3}{(1-\gamma)^2}$.

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

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

▷ Steps 2–7 check that the plain part of x is the repetition of a string $y \in \{0,1\}^n$.

2: **repeat** $\lceil \frac{432}{\varepsilon} \rceil$ times:
3: Sample a uniformly random point u from the plain part.
4: **if** $x[u] \neq \perp$ **then**
5: Let $i \in [s]$, $a \in [n]$ be such that $u = (i-1) \cdot n + a$.
6: Repeatedly sample $j \in [s]$ uniformly at random until $x[(j-1)n + a] \neq \perp$.
7: **Reject** if $x[u] \neq x[(j-1)n + a]$.

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

8: **Run** the decoder for the (γ, β, q, L)-ALLED code (from Lemma 5.3) with oracle access to the encoded part of x.
▷ Let $A_1, A_2, \ldots, A_L$ be the list of algorithms returned in the above step.
▷ Steps 9–14 check that $y \in \mathcal{R}_{\phi_n}$ using the smooth PCPP V (from Lemma 6.3) on decoded proofs.
9: **for** each $k \in [L]$ **do**
10: **repeat** $\lceil 6 \ln 6L \rceil$ times:
11: Sample $i \in [s]$ uniformly at random.
12: Run the smooth PCPP V with proximity parameter $\frac{24\varepsilon}{75}$, and oracle access to the concatenation of $x[(i-1) \cdot n + 1, \ldots, (i-1) \cdot n + n]$ and the string decoded by T_k.
13: **Discard** the current k if all query answers to V are nonerased and V rejects.
14: **Reject** if every $k \in [L]$ is **discarded**; otherwise, **accept**.

amplified by to $2/3$ by repeating Algorithm 3 a constant number of times and returning the majority decision.

The erasure-resilient tester is presented in Algorithm 3. Let m denote $\frac{28\ 800 \cdot c_2}{\varepsilon^\star}$. Let $\gamma = \frac{1}{2} + \frac{\varepsilon^\star}{57\ 600 \cdot c_2}$, $\beta = \frac{\varepsilon^\star}{9000 c_2 \cdot \left\lceil \ln \frac{6c_3}{(1-\gamma)^2} \right\rceil}$, $q = \frac{c_3 \log(1/\beta)}{(1-\gamma)^3}$, 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 y for some $y \in \mathcal{R}_{\phi_n}$. Therefore, Steps 2–7 do not reject. By the 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 5.3) of a string $\pi(y) \in \{0,1\}^{p(n)}$ such that the smoothed PCPP V with oracle access to $y \circ \pi(y)$ always accepts. The second half of the encoding is completable to 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 is at most $(m+1)\cdot\alpha \le \frac{1}{2}+\frac{1}{2m} = \gamma$ in either the first half or the second half of the encoded part.

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 a 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 10–13. Since each proof query of V (in Step 12) 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 in some iteration of Steps 10–13 is at most $6\cdot\lceil\ln 6L\rceil\cdot 2\beta\cdot\frac{c_2\cdot 75}{24\epsilon}$, where we used the fact that $\lceil 6\ln 6L\rceil \le 6\cdot\lceil\ln 6L\rceil$. Now, the tester makes a wrong decision only if either (1) the approximate local list-decoder fails (which happens with probability at most $1/3$), or (2) if the approximate local list-decoder succeeds but Steps 10–13 discard k. Hence, by the union bound over the two events, the probability that the tester makes a wrong decision is at most $\frac{1}{3}+2\cdot 6\cdot\lceil\ln 6L\rceil\cdot\frac{c_2\cdot 75}{24\epsilon}\cdot\beta \le \frac{2}{5}$, where the inequality follows from our setting of β. Hence, Step 14 rejects with probability at most $2/5$. 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 $s\cdot n$, the length of the plain part) 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. In what follows, we upper bound the probability that Algorithm 3 accepts, conditioned on E. We prove later, in Claim 6.9, that $\Pr[\overline{E}] \le 1/30$.

Let ϵ_{pl} denote the fraction of points (with respect to $s\cdot n$, the length of the plain part) in the plain part 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 ath 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 ath segment.

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

For $a \in [n]$, let ϵ_a denote the smallest fraction of points in S_a whose values need to be changed in order to satisfy $x[u] = x[v]$ for all $u, v \in \mathcal{N}_a$. For every $a \in [n]$ and $u \in \mathcal{N}_a$, the number of $v \in \mathcal{N}_a$ such that $x[u] \ne x[v]$, is at least $\epsilon_a \cdot s$. It is immediate that $\epsilon_{\text{pl}}\cdot s\cdot n = \sum_{a\in[n]}\epsilon_a\cdot s$.

Let F denote the event that the tester rejects in a single iteration of the loop in Steps 2–7. Let G_a for all $a \in [n]$ denote the event that the tester samples a nonerased point u from S_a in Step 3. Conditioned on G_a, the number of nonerased points in S_a that make the tester reject is at least $\epsilon_a \cdot s$. Putting all this together, we have,

$$\Pr[F|E] = \sum_{a\in[n]}\Pr[G_a|E]\cdot\Pr[F|G_a,E] = \sum_{a\in[n]}\frac{|\mathcal{N}_a|}{sn}\cdot\frac{\epsilon_a\cdot s}{|\mathcal{N}_a|} = \sum_{a\in[n]}\frac{1}{n}\cdot\epsilon_a = \epsilon_{\text{pl}} \ge \frac{\epsilon}{144}.$$

Therefore, conditioned on E, in at least $432/\epsilon$ iterations, the tester will reject with probability at least $19/20$. Hence, in Case I, the algorithm accepts with probability at most $\frac{1}{20}+\Pr[\overline{E}] \le \frac{1}{20}+\frac{1}{30} \le \frac{2}{5}$, where we prove later (in Claim 6.9) $\Pr[\overline{E}] \le 1/30$. Thus, the algorithm rejects with probability at least $3/5$.

Case II: The plain part of x is $\epsilon/144$-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 6.7. *The string y^* is $\epsilon/2$-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 $sn \cdot \frac{\varepsilon}{144} \leq N \cdot \frac{\varepsilon}{144}$ 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 $m = \frac{28\ 800 \cdot c_2}{\varepsilon^\star} > \frac{144}{71\varepsilon}$. 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 ith 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 $\mathcal{N}_i \subseteq B_i$ denote the set of nonerased points in the ith block. Let ε_i for all $i \in [s]$ denote the fraction of points in B_i whose values need to be changed in order to satisfy $x[(i-1)n + a] = y^*[a]$ for all $a \in [n]$. In other words, $\varepsilon_i n$ is the smallest number of points in $\mathcal{N}_i$ that need to be changed in order for the ith block to be completable to y^*.

Fix $k \in [L]$. We show that Algorithm 3 discards k with high probability. Consider a single iteration of the repeat-loop in Steps 11–13. Let y' denote the (partially erased) string represented by the block that Algorithm 3 samples in Step 11. 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 11–13 conditioned on G_1 and G_2.

Claim 6.8. *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 $48\alpha n + \frac{\varepsilon n}{6} + d$ bits 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 smooth 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 6.3, V makes $\frac{c_2 \cdot 75}{24\varepsilon}$ queries and each query of V to the input part is made to each of the n input indices with probability $1/n$. Hence $\Pr[B_1 | E, G_1, G_2]$, the probability that some input query is made to an erased point, is at most $\frac{c_2 \cdot 75}{24\varepsilon} \cdot 48\alpha$.

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 4.3, at most $1/3$. Thus, the probability that the smooth PCPP accepts, conditioned on E, G_1, and G_2, is by the union bound, at most

$$\frac{c_2 \cdot 75}{24\varepsilon} \cdot 48\alpha + \frac{1}{3} \leq \frac{1}{24} + \frac{1}{3},$$

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 11–13, 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,

$$\mathbb{E}[X] = \sum_{i \in [s]} \frac{1}{s} \cdot \varepsilon_i = \varepsilon_{\text{pl}} \leq \frac{\varepsilon}{144}.$$

By Markov's inequality,

$$\Pr[\overline{G_1}] = \Pr\left[X \geq \frac{\varepsilon}{6}\right] \leq \mathbb{E}[X]/(\varepsilon/6) \leq 1/24.$$

To bound $\Pr[\overline{G_2}]$, let the random variable Y denote the fraction of erasures in y'. We have that

$$\mathbb{E}[Y] = \sum_{i\in[s]} \frac{\alpha_i}{s} = \alpha_{\text{pl}}.$$

Even if all the erasures were in the plain part, $\alpha_{\text{pl}} \leq \frac{\alpha N}{sn} \leq \alpha \cdot (1 + \frac{1}{m})$. Again, by an application of Markov's inequality, we get

$$\Pr[\overline{G_2}] = \Pr[Y > 48\alpha] \leq \frac{\mathbb{E}[Y]}{48\alpha} \leq \frac{1 + \frac{1}{m}}{48} \leq 1/24.$$

Therefore, conditioned on E, the probability that the PCPP accepts in one iteration of Steps 11–13 is at most

$$\Pr[B_1|E, G_1, G_2] + \Pr[B_2|E, G_1, G_2] + \Pr[\overline{G_2}] + \Pr[\overline{G_1}] \leq \frac{1}{24} + \frac{1}{3} + \frac{1}{24} + \frac{1}{24} \leq \frac{2}{3}.$$

That is, conditioned on E, for a fixed $k \in [L]$, in $\lceil 6 \ln 6L \rceil$ independent repetitions of Steps 11–13, the probability that the PCPP does not discard k is at most $\left(1 - \frac{1}{3}\right)^{\lceil 6 \ln 6L \rceil} \leq \frac{1}{36L^2}$. Hence, conditioned on E, the probability that for some $k \in [L]$, Steps 10–13 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, $\frac{1}{36L} + \Pr[\overline{E}] \leq \frac{1}{36L} + \frac{1}{30} \leq \frac{2}{5}$, where Claim 6.9 shows that $\Pr[\overline{E}]$ is at most $1/30$, which then completes the proof of Lemma 6.6.

Claim 6.9. *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. Lemma 6.3 implies that the verifier V, when run with parameter $\frac{24\epsilon}{75}$, makes at most $\frac{c_2 \cdot 75}{24\epsilon}$ queries. Hence, the number of queries made in Steps 9–13 is at most $L\lceil 6 \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\epsilon} \cdot q$, 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 3–7. Let Y denote the number of queries made in a particular iteration of Steps 3–7. The variable Y is nonzero only if the sampled point u is nonerased. To calculate $\mathbb{E}[Y]$:

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

Hence, the expected number of queries made by the tester in Steps 2–7 is $\lceil \frac{432}{\varepsilon} \rceil$. Hence, setting Q to $30 \cdot \left(\lceil \frac{432}{\varepsilon} \rceil + L\lceil 6 \ln 6L \rceil \cdot \frac{c_2 \cdot 75}{24\epsilon} \cdot q \right)$, and applying Markov's inequality, one can see that $\Pr[\overline{E}] \leq 1/30$. ∎

Next, we show that it is hard to tolerant test $\mathcal{P}'$. The proof of Lemma 6.10 is identical to the proof of Lemma 4.12 up to change in parameters.

Lemma 6.10. *Let $\varepsilon^\star \in (0,1)$ be as in Lemma 4.2 and $c_2 > 1$ be as in Lemma 6.3. For every $\alpha \in (\frac{\varepsilon^\star}{57\,600\cdot c_2+2\varepsilon^\star}, 1)$, and $\varepsilon' \in \left(\alpha, \frac{28\,800\cdot c_2\cdot\varepsilon^\star}{28\,800\cdot c_2+\varepsilon^\star}\right)$, every (α,ε')-tolerant tester for $\mathcal{P}'$ requires $\tilde{\Omega}(N^{0.2})$ queries.*

Proof. Let $\aleph$ be as in Lemma 4.2 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 6.5. Let s denote $m \cdot 2f(\gamma,\beta)\cdot(p(n))^5/n$.

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 $x[1 \ldots sn]$ as the plain part of x and the substring $x[sn+1 \ldots N]$ 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.

We will show that 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 a $\frac{1}{2}$ fraction of its positions in order to make it the encoding of a string π, where π is a proof that makes a smooth PCPP for testing $\mathcal{R}_{\phi_n}$ (as guaranteed by Lemma 6.3) 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 $\frac{1}{2}\cdot\frac{N-sn}{N} = \frac{1}{2(m+1)} = \frac{\varepsilon^\star}{57\,600\cdot c_2+2\varepsilon^\star}$, 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 $\tilde{\Omega}(N^{0.2})$. ∎

Remark 6.11. We would like to point out that the lower bound on the query complexity of tolerant testing (from Lemma 6.10) can be improved if there exist approximate local erasure list-decodable codes with larger rate. In other words, constant-query approximate local erasure list-decodable codes with larger rate, when used in our above construction, directly imply an even stronger separation between the query complexity of erasure-resilient and tolerant testing models.

Proof of Theorem 1.10. From Theorem 6.1, we get the following constraints on $\alpha,\varepsilon,\varepsilon'$:

$$\frac{\varepsilon^\star}{57\,600\cdot c_2+2\varepsilon^\star} < \alpha < \frac{\varepsilon^\star}{57\,600\cdot c_2};\quad \varepsilon > \frac{\varepsilon^\star}{8};\quad \varepsilon' < \frac{28\,800\cdot c_2\varepsilon^\star}{28\,800\cdot c_2+\varepsilon^\star}.$$

To complete the proof of Theorem 1.10, it is enough to find values of ε,α that satisfy the above constraints, where we set $\varepsilon' = \varepsilon+\alpha$. For sufficiently small $\varepsilon^\star$, the upper bound on $\varepsilon+\alpha$ is strictly greater than $\varepsilon^\star/2$. So, it is enough to find $\varepsilon < \varepsilon^\star/4$ and $\alpha < \varepsilon^\star/4$ that also satisfy the first two conditions. The existence of such ε and α is clear from the bounds imposed on them by the first two constraints. ∎

7 | LOCAL ERASURE-DECODING VERSUS LOCAL DECODING

In this section, we prove Theorem 1.12 and an observation that if a code is locally decodable, it is also locally erasure-decodable up to (nearly) twice as many erasures. A part of our proof (Claim 7.2) uses ideas developed Katz and Trevisan [42].

Definition 7.1 (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 nonadaptive $(0, q)$-local erasure-decoder A (see Definition 1.11) that, given oracle access to an uncorrupted codeword $w \in \mathbb{F}_2^N$, and an 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 1.12.

Claim 7.2. *For every $\alpha \in [0, 1)$, if a code family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is (α, q)-locally erasure-decodable, then $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is (q', η)-smooth locally decodable, where $q' = 3^q$, and $\eta = \frac{q'}{\alpha N}$.*

Claim 7.3. *For every $\alpha \in [0, 1)$, if a code family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is $(q, \frac{q}{\alpha\cdot N})$-smooth locally decodable, then $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is $(\frac{\alpha}{O(q)}, O(q))$-locally decodable.*

Proof of Claim 7.2. Let A be an (α, q)-local erasure-decoder for C_n. Since A could be adaptive, for every choice of random coins, the execution of A can be represented as a ternary tree, where each node represents a query. The root represents the first query made by A. The three children of a non-leaf node u represent the next points that A will query for the cases that the answers to the query u are 0, 1, or $\perp$. The size of this tree is at most 3^q. Consider an algorithm A_1 that, after having generated its random string $r \in \{0, 1\}^*$, queries all the points in the tree of execution of A on the string r. After obtaining the answers to its queries, A_1 outputs the value at the end of the root-to-leaf path that matches with the actual query answers. Note that there is exactly one such path. Therefore, A_1 is a nonadaptive local erasure-decoder for C_n that makes at most $q' = 3^q$ queries and has the same success probability as A.

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 $q_i \le q'$ denote the number of queries made by A_1 on input i and let S_i denote the set consisting of indices in $[N]$ that get queried by A_1 (on input i) with probability more than $\frac{q'}{\alpha N}$. For $i \in [n]$, $k \in [q_i]$, it is clear that

$$\sum_{j\in[N]} \Pr[k\text{th query of } A_1^{(\cdot)}(i) \text{ is to position } j] = 1.$$

Hence, for each $i \in [n]$,

$$\sum_{j\in[N]}\sum_{k\in[q_i]} \Pr[k\text{th query of } A_1^{(\cdot)}(i) \text{ is to position } j] = q_i \le q'.$$

From this, we have $|S_i| \le \alpha \cdot N$. 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 7.3. Consider a $(q, \frac{q}{\alpha N})$-smooth local decoder A for C_n. We will construct an $(\frac{\alpha}{12q}, 72q)$-local decoder A' for C_n. Algorithm A', on input $i \in [n]$ and oracle access to a word w with

at most an $\frac{\alpha}{12q}$ 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 a $\frac{2}{3}$ 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 an $\frac{\alpha}{12q} \cdot N \cdot \frac{q}{\alpha N} = \frac{1}{12}$ 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, by a Chernoff bound, the probability that A', which is obtained by running 72 independent iterations of A and outputting the majority answer, outputs x_i correctly is at least $2/3$. The query complexity of A' is $72q$. ∎

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

Observation 7.4. Every (α, q)-LDC family $\{C_n : \mathbb{F}_2^n \to \mathbb{F}_2^N\}_{n\in\mathbb{N}}$ is also $(2\alpha - \rho, O(q))$-locally erasure-decodable, where $\rho = O(\sqrt{\frac{\alpha}{N}})$.

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. ∎

ACKNOWLEDGMENTS

The authors express their gratitude to anonymous reviewers whose comments helped improve the presentation of this article. The authors are thankful to Venkatesan Guruswami for helping to tighten the analysis of the local erasure list-decoder for the Hadamard code and also for making a suggestion that led to Observation 5.2. 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 thank the sponsors and organizers of the Workshop on Local Algorithms 2018 for making this collaboration possible. The first author was supported by National Science Foundation (NSF) grants CCF-142297, CCF-1832228, and CCF-1909612. The second author was supported in part by the Israel Science Foundation (ISF) grant 735/20. Most of this work was done when the third author was a student at the Boston University, where he was supported by NSF grants CCF-142297, and CCF-1832228. The third author was also supported by the ISF grant 497/17, and by the PBC Fellowship for Postdoctoral Fellows by the Israeli Council of Higher Education.

REFERENCES

1. N. Alon, J. Edmonds, and M. Luby, *Linear time erasure codes with nearly optimal recovery (extended abstract)*, Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1995), 1995, pp. 512–519.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and intractability of approximation problems*, J. ACM **45**(3) (1998), 501–555.

3. S. Arora and S. Safra, *Probabilistic checkable proofs: A new characterization of NP*, J. ACM **45**(1) (1998), 70–122.
4. S. Artemenko and R. Shaltiel, *Lower bounds on the query complexity of non-uniform and adaptive reductions showing hardness amplification*, Comput. Complex. **23**(1) (2014), 43–83.
5. L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy, *Checking computations in polylogarithmic time*, in *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991*, C. Koutsougeras and J. S. Vitter, Eds., ACM, New Orleans, Louisiana, 1991, 21–31.
6. L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, *BPP has subexponential time simulations unless EXPTIME has publishable proofs*, Comput. Complex. **3**(4) (1993), 307–318.
7. A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond, *Breaking the* $o\left(n^{1/(2k-1)}\right)$ *barrier for information-theoretic private information retrieval*, in *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002*, IEEE Computer Society, Vancouver, BC, Canada, 2002, 261–270.
8. A. Ben-Aroya, K. Efremenko, and A. Ta-Shma, *Local list decoding with a constant number of queries*, in *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010*, IEEE Computer Society, Las Vegas, Nevada, 2010, 715–722.
9. A. Ben-Aroya, K. Efremenko, and A. Ta-Shma, A note on amplifying the error-tolerance of locally decodable codes, *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 17, 2010, p. 134.
10. E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. P. Vadhan, *Robust PCPs of proximity, shorter PCPs, and applications to coding*, SIAM J. Comput. **36**(4) (2006), 889–974.
11. E. Ben-Sasson, P. Harsha, and S. Raskhodnikova, *Some 3CNF properties are hard to test*, SIAM J. Comput. **35**(1) (2005), 1–21.
12. V. M. Blinovsky, *Bounds for codes in the case of list decoding of finite volume*, Probl. Inf. Transm. **22**(1) (1986), 7–19.
13. M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. Syst. Sci. **47**(3) (1993), 549–595.
14. A. Bogdanov and M. Safra, *Hardness amplification for errorless heuristics*, Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20–23, 2007, IEEE Computer Society, Providence, RI, 2007, pp. 418–426.
15. J.-Y. Cai, A. Pavan, and D. Sivakumar, *On the hardness of permanent*, in *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science STACS 99, March 4-6, 1999*, Lecture Notes in Computer Science, Vol **1563**, C. Meinel and S. Tison, Eds., Springer, Trier, Germany, 1999, 90–99.
16. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, *Private information retrieval*, J. ACM **45**(6) (1998), 965–981.
17. I. Dinur, *The PCP theorem by gap amplification*, J. ACM **54**(3) (2007), 12.
18. I. Dinur and O. Reingold, *Assignment testers: Towards a combinatorial proof of the PCP theorem*, SIAM J. Comput **36**(4) (2006), 975–1024.
19. K. Dixit, S. Raskhodnikova, A. Thakurta, and N. M. Varma, *Erasure-resilient property testing*, SIAM J. Comput. **47**(2) (2018), 295–329.
20. Z. Dvir, P. Gopalan, and S. Yekhanin, *Matching vector codes*, SIAM J. Comput. **40**(4) (2011), 1154–1178.
21. K. Efremenko, *3-query locally decodable codes of subexponential length*, SIAM J. Comput. **41**(6) (2012), 1694–1703.
22. F. Ergün, R. Kumar, and R. Rubinfeld, *Fast approximate probabilistically checkable proofs*, Inf. Comput. **189**(2) (2004), 135–159.
23. E. Fischer and L. Fortnow, *Tolerant versus intolerant testing for Boolean properties*, Theory Comput. **2**(9) (2006), 173–183.
24. P. Gemmell, R. J. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self-testing/correcting for polynomials and for approximate functions*, Proc. STOC **1991** (1991), 32–42.
25. P. Gemmell and M. Sudan, *Highly resilient correctors for polynomials*, Inf. Process. Lett. **43**(4) (1992), 169–174.
26. O. 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*, *Lecture Notes in Computer Science*, Vol **6650**, O. Goldreich, Ed., Springer, New York, NY, 2011, 465–469.
27. O. 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*, *Lecture Notes in Computer Science*, Vol **6650**, O. Goldreich, Ed., Springer, New York, NY, 2011, 470–506.
28. O. Goldreich, *Introduction to property testing*, Cambridge University Press, Cambridge, MA, 2017.

29. O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM **45**(4) (1998), 653–750.
30. O. Goldreich and L. 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*, D. S. Johnson, Ed., ACM, Seattle, Washington, 1989, 25–32.
31. O. Goldreich, R. Rubinfeld, and M. Sudan, *Learning polynomials with queries: The highly noisy case*, SIAM J. Discret. Math. **13**(4) (2000), 535–570.
32. S. Gopi, S. Kopparty, R. Oliveira, N. Ron-Zewi, and S. Saraf, *Locally testable and locally correctable codes approaching the gilbert-varshamov bound*, IEEE Trans. Inf. Theory **64**(8) (2018), 5813–5831.
33. A. Grinberg, R. Shaltiel, and E. Viola, *Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs*, in *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, October 7-9, 2018*, M. Thorup, Ed., IEEE Computer Society, Paris, France, 2018, 956–966.
34. A. Guo and S. Kopparty, *List-decoding algorithms for lifted codes*, IEEE Trans. Inf. Theory **62**(5) (2016), 2719–2725.
35. V. Guruswami, *List decoding from erasures: Bounds and code constructions*, IEEE Trans. Inf. Theory **49**(11) (2003), 2826–2833.
36. V. Guruswami and P. Indyk, *Linear-time encodable/decodable codes with near-optimal rate*, IEEE Trans. Inf. Theory **51**(10) (2005), 3393–3400.
37. V. Guruswami and A. Rudra, *Tolerant locally testable codes*, in *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, August 22-24, 2005*, *Lecture Notes in Computer Science*, Vol **3624**, C. Chekuri, K. Jansen, J. D. P. Rolim, and L. Trevisan, Eds., Springer, Berkeley, CA, 2005, 306–317.
38. V. Guruswami and S. P. Vadhan, *A lower bound on list size for list decoding*, IEEE Trans. Inf. Theory **56**(11) (2010), 5681–5688.
39. D. Gutfreund and G. N. Rothblum, *The complexity of local list decoding*, in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, Proceedings of the 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, August 25-27, 2008*, *Lecture Notes in Computer Science*, A. Goel, K. Jansen, J. D. P. Rolim, and R. Rubinfeld, Eds., Springer, 5171, Boston, MA, 2008, 455–468.
40. B. Hemenway, N. Ron-Zewi, and M. Wootters, *Local list recovery of high-rate tensor codes & applications*, Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS). IEEE Computer Society, Washington, DC, 2017.
41. R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. Wigderson, *Uniform direct product theorems: Simplified, optimized, and derandomized*, SIAM J. Comput. **39**(4) (2010), 1637–1665.
42. J. Katz and L. Trevisan, *On the efficiency of local decoding procedures for error-correcting codes*, in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, May 21-23, 2000*, F. F. Yao and E. M. Luks, Eds., ACM, Portland, OR, 2000, 80–86.
43. S. Kopparty, *List-decoding multiplicity codes*, Theory Comput. **11** (2015), 149–182.
44. S. Kopparty, O. Meir, N. Ron-Zewi, and S. Saraf, *High-rate locally correctable and locally testable codes with sub-polynomial query complexity*, J. ACM **64**(2) (2017), 11:1–11:42.
45. S. Kopparty and S. Saraf, *Local list-decoding and testing of random linear codes from high error*, SIAM J. Comput. **42**(3) (2013), 1302–1326.
46. E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM J. Comput. **22**(6) (1993), 1331–1348.
47. A. Levi, R. K. S. Pallavoor, S. Raskhodnikova, and N. Varma, *Erasure-resilient sublinear-time graph algorithms*, in *Proceedings of the 12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, *LIPIcs*, Vol **185**, J. R. Lee, Ed., Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany, 2021, 80:1–80:20.
48. R. J. Lipton, *New directions in testing*, in *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, October 4-6, 1989*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol **2**, J. Feigenbaum and M. Merritt, Eds., DIMACS/AMS, Princeton, NJ, 1989, 191–202.
49. R. J. Lipton, *Efficient checking of computations,* Proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS), San Jose, CA, 1990, pp. 207–215.
50. O. Meir, *Combinatorial PCPs with efficient verifiers*, Comput. Complex. **23**(3) (2014), 355–478.
51. O. Meir, *Combinatorial PCPs with short proofs*, Comput. Complex. **25**(1) (2016), 1–102.
52. M. Parnas, D. Ron, and R. Rubinfeld, *Tolerant property testing and distance approximation*, J. Comput. Syst. Sci. **72**(6) (2006), 1012–1042.
53. A. Polishchuk and D. A. Spielman, *Nearly-linear size holographic proofs*, in *Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 23-25 May 1994*, F. T. Leighton and M. T. Goodrich, Eds., ACM, Montréal, Québec, Canada, 1994, 194–203.

54. S. Raskhodnikova, N. Ron-Zewi, and N. M. Varma, *Erasures vs. errors in local decoding and property testing*, Proceedings of the 10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10–12, 2019, San Diego, CA, 2019, pp. 63:1–63:21.
55. D. Ron, *Algorithmic and analysis techniques in property testing*, Found. Trends Theor. Comput. Sci. **5**(2) (2009), 73–205.
56. N. Ron-Zewi, R. Shaltiel, and N. Varma, *Query complexity lower bounds for local list-decoding and hard-core predicates (even for small rate and huge lists)*, in *Proceedings of the 12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, *LIPIcs*, Vol **185**, J. R. Lee, Ed., Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany, 2021, 33:1–33:18.
57. R. Rubinfeld and E. Blais, *Something for (almost) nothing: New advances in sublinear-time algorithms*, in *Handbook of big data*, Chapman & Hall/CRC Press, Boca Raton, FL, 2016, 155–167.
58. R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput. **25**(2) (1996), 252–271.
59. M. Sudan, L. Trevisan, and S. P. Vadhan, *Pseudorandom generators without the XOR lemma*, J. Comput. Syst. Sci. **62**(2) (2001), 236–266.
60. L. Trevisan, *List-decoding using the XOR lemma*, Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Cambridge, MA, 2003, pp. 126–135.
61. L. Trevisan, *Some applications of coding theory in computational complexity*, CoRR (2004), cs.CC/0409044.
62. T. Watson, *Query complexity in errorless hardness amplification*, Comput. Complex. **24**(4) (2015), 823–850.
63. S. Yekhanin, *Towards 3-query locally decodable codes of subexponential length*, J. ACM **55**(1) (2008), 1:1–1:16.

How to cite this article: S. Raskhodnikova, N. Ron-Zewi, and N. Varma, *Erasures versus errors in local decoding and property testing*, Random Struct. Algorithms. **59** (2021), 640–670. https://doi.org/10.1002/rsa.21031

APPENDIX A: TWO DEFINITIONS OF ERASURE-RESILIENT TESTING

In this section, we show that for constant $\alpha, \varepsilon \in (0,1)$, the definition of α-erasure-resilient ε-testing model used in this article is equivalent to that defined by Dixit et al. [19]. For convenience, we refer to the former and latter definitions as the new and old definitions, respectively. We first describe the rejection condition of an erasure-resilient tester according to the old definition, which is the only difference between the two definitions. For $\alpha \in [0,1)$ and $\varepsilon \in (0,1)$, an α-erasure-resilient ε-tester for a property $\mathcal{P}$ (of strings of length n) rejects, with probability at least $2/3$, an α-erased string $x \in \{0,1,\perp\}^n$ if every completion of x has to be changed in at least $\varepsilon \cdot |\mathcal{N}|$ nonerased values in order for it to satisfy $\mathcal{P}$, where $\mathcal{N}$ denotes the set of nonerased points in x.

Claim A.1. *Let $\alpha, \varepsilon \in (0,1)$ such that $\alpha + \varepsilon < 1$. Let $\mathcal{P}$ be a property over strings of length n. If T is an α-erasure-resilient ε-tester for a property $\mathcal{P}$ with query complexity $q(\varepsilon, \alpha, n)$ w.r.t. the old definition, then T is also an α-erasure-resilient ε-tester for $\mathcal{P}$ with query complexity $q(\varepsilon, \alpha, n)$ w.r.t. the new definition.*

Proof. Consider an α-erased string $x \in \{0,1,\perp\}^n$. If x satisfies $\mathcal{P}$, then T accepts x with probability at least $2/3$. If x is ε-far from $\mathcal{P}$ w.r.t. the new definition, then $\mathcal{P}$ is $\frac{\varepsilon \cdot n}{|\mathcal{N}|}$-far from $\mathcal{P}$ w.r.t. the old definition. Since $\frac{\varepsilon \cdot n}{|\mathcal{N}|} \geq \varepsilon$, the tester T, when run with parameters α and ε, rejects x with probability at least $2/3$. Moreover, the query complexity of T remains the same. ∎

Claim A.2. *Let $\alpha, \varepsilon \in (0,1)$ and $\varepsilon' = \varepsilon(1-\alpha)$. Let $\mathcal{P}$ be a property over strings of length n. If T is an α-erasure-resilient ε'-tester with query complexity $q(\varepsilon', \alpha, n)$ for $\mathcal{P}$ w.r.t. the new definition, then T is an α-erasure-resilient ε-tester for $\mathcal{P}$ with query complexity $q(\varepsilon(1-\alpha), \alpha, n)$ w.r.t. the old definition.*

Proof. Consider an α-erased string $x \in \{0,1,\perp\}^n$. If x satisfies $\mathcal{P}$, then T accepts x with probability at least $2/3$. If x is ε-far from $\mathcal{P}$ w.r.t. the old definition, then $\mathcal{P}$ is $\frac{\varepsilon \cdot |\mathcal{N}|}{n}$-far from $\mathcal{P}$ w.r.t. the new definition. Since $\frac{\varepsilon \cdot |\mathcal{N}|}{n} \geq \varepsilon(1-\alpha)$, the tester T, when run with parameters $\alpha' = \alpha$ and $\varepsilon' = \varepsilon(1-\alpha)$, rejects x with probability at least $2/3$. ∎