

Online Versus Offline Adversaries in Property Testing


Esty Kelman ✉ 

Boston University, MA, USA

Massachusetts Institute of Technology, Cambridge, MA, USA

Ephraim Linder ✉ 

Boston University, MA, USA

Sofya Raskhodnikova ✉ 

Boston University, MA, USA

Abstract

We study property testing with incomplete or noisy inputs. The models we consider allow for *adversarial* manipulation of the input, but differ in whether the manipulation can be done only *offline*, i.e., before the execution of the algorithm, or *online*, i.e., as the algorithm runs. The manipulations by an adversary can come in the form of erasures or corruptions. We compare the query complexity and the randomness complexity of property testing in the offline and online models. Kalemaj, Raskhodnikova, and Varma (Theory Comput. '23) provide properties that can be tested with a small number of queries with offline erasures, but cannot be tested at all with online erasures. We demonstrate that the two models are incomparable in terms of query complexity: we construct properties that can be tested with a constant number of queries in the online corruption model, but require querying a significant fraction of the input in the offline erasure model. We also construct properties that exhibit a strong separation between the randomness complexity of testing in the presence of offline and online adversaries: testing these properties in the online model requires exponentially more random bits than in the offline model, even when they are tested with nearly the same number of queries in both models. Our randomness separation relies on a novel reduction from randomness-efficient testers in the adversarial online model to query-efficient testers in the standard model.

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

Keywords and phrases Property Testing, Online Adversary, Offline Adversary, Query Complexity, Randomness Complexity, Separations

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.65

Related Version *Full Version*: <https://arxiv.org/abs/2411.18617> [22]

Funding *Esty Kelman*: Supported in part by the National Science Foundation under Grant No. 2022446 and in part by NSF TRIPODS program (award DMS-2022448).

Acknowledgements The authors thank Uri Meir for fruitful discussions regarding the online model.

1 Introduction

Property testing [31, 15] aims to lay algorithmic foundations for processing big data. It is a formal study of fast algorithms that accept objects with a given property and reject objects that are far. The goal of this work is to compare models of property testing that address the situations when the data that needs to be analyzed is not only large, but also incomplete or noisy. The models we consider allow for adversarial manipulation of the input, but differ in whether the manipulation can be done only *offline*, i.e., before the execution of the algorithm, or *online*, i.e., as the algorithm runs. The manipulations by an adversary can come in the



© Esty Kelman, Ephraim Linder, and Sofya Raskhodnikova;
licensed under Creative Commons License CC-BY 4.0

16th Innovations in Theoretical Computer Science Conference (ITCS 2025).

Editor: Raghu Meka; Article No. 65; pp. 65:1–65:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

form of erasures or corruptions. The offline erasure-resilient property testing model was proposed by Dixit et al. [9]. The model that captures offline corruptions is called *tolerant testing* and was introduced by Parnas et al. [25]. The online analogues of these models were recently defined by Kalemaj et al. [21].

We compare the query complexity and the randomness complexity of property testing in the offline and online models. In the offline-erasure model, a constant fraction of the input is erased adversarially before the execution of the algorithm, whereas in the online-erasure model, the adversary is allowed to make a fixed number of erasures after each query. It may seem that the online adversary has strictly more power, and thus testing in the online models should require more queries. Indeed, [21] justify this intuition and show that simple properties – sortedness and the Lipschitz property of sequences – cannot be tested at all with online erasures¹, even though they are testable using a small number of queries when erasures are offline. Intuitively, [21] show that testing in the presence of one online erasure per query (performed by an adversary with the knowledge of previous queries) can be harder than testing when a constant fraction of the input is adversarially erased offline (in advance). The following natural open question is stated in [21]: “Is there a property that has smaller query complexity in the online-erasure-resilient model than in the (offline) erasure-resilient model of [9]?” We answer this question in the affirmative: specifically, we construct a property that can be tested with a constant number of queries in the online-erasure model (and even in the (harder) online-corruption model), but requires a nearly linear number of queries in the offline-erasure model. We conclude that the two adversarial erasure models are incomparable.

Our second result concerns the randomness complexity of testing in the online and offline manipulation models. Randomness is an important tool in the design of a wide variety of algorithms, and extensive research has been conducted to understand the power of randomness in computation. The investigation of the randomness complexity of property testing algorithms was initiated by Goldreich and Sheffet [18]. They showed that all property testers in the standard model can be derandomized to a large extent: they can be converted to testers that use the number of random bits that is logarithmic in the size of the input, while incurring at most a constant factor blowup in query complexity. The result of [18] easily extends to all offline testing models, but does not apply in the online setting. Moreover, the existing testers in the online model [21, 24, 6, 33, 3] use lots of randomness to fool the adversary. A natural question that arises is how much derandomization is possible in the online setting. We show that a lot of randomness is sometimes indeed necessary to test in the presence of an online adversary. In particular, we exhibit a property that requires exponentially more random bits to test in the online manipulation model than in the offline manipulation model. This result provides a natural converse to the statement of [24] that “the (online) adversary cannot foil our plan if there is no plan” – i.e., random queries are robust to online manipulation of the input.

1.1 Our results

We represent the input to our property testing algorithm as a string. The algorithm is given access to its input string via queries (specifically, of the form “what is the character at index i ?”). For each proximity parameter $\varepsilon \in (0, 1)$, the algorithm has to distinguish between strings that have a specified property and strings that are ε -far from having the property – i.e., strings that need to be changed on at least an ε -fraction of indices in order to satisfy

¹ In contrast, in the offline models, testing is always possible when the entire input is read.

the property. The query complexity (the number of queries to the input) of the tester is measured in terms of ε and the input length n . In this section, for simplicity, ε is considered to be a constant and omitted from the statements.

1.1.1 Query complexity: Model Incomparability

We show that the online and offline adversarial models of property testing are incomparable in terms of the query complexity. Recall that [21] prove that sortedness and Lipschitzness of sequences are efficiently testable with offline erasures, but cannot be tested at all even with one online erasure per query². We complement this result with the following theorems, which show that testing in the presence of a small fraction of offline erasures can be harder than testing with online erasures. The first of this theorems (Theorem 1.1) considers the regime with many online erasures per query; the second (Theorem 1.2) considers the regime with one erasure per query. The former exhibits a logarithmic gap in query complexity, where as the latter exhibits a nearly linear gap.

► **Theorem 1.1** (Informal version of Corollary 3.2). *There exists a property that can be tested using a constant number of queries in the presence of an online adversary that makes $O(\frac{n}{\log n})$ erasures per query; but requires $\tilde{\Omega}(\log n)$ queries³ to test when the characters at $\Theta(\frac{n}{\log n})$ indices are erased offline (in advance).*

We cannot expect to prove an analogue of Theorem 1.1 for the case when the number of online erasures per query is the same as the total number of offline erasures. This is because, after the first query, an online adversary can erase the same part of the input that the offline adversary would have erased in advance.

► **Remark 1.2.** Although the online erasure rate in Theorem 1.1 is very large, the total number of erasures available to the online adversary is smaller than the number of offline erasures we considered. It is an intriguing open question whether there exists a property for which every tester in the online model accumulates at least as many erasures as the best tester in the offline model, but also requires fewer queries to test in the online model than in the offline model.

At the other end of the spectrum, we show that if the number of erasures is much smaller in the online model than in the offline model, then there is an exponentially large gap – i.e., there are properties that can be tested with a constant number of queries when the erasures are online but require a nearly linear number of queries in the offline erasure model.

► **Theorem 1.3** (Informal version of Corollary 3.3). *There exists a property that can be tested with a constant number of queries in the presence of an online adversary that makes one erasure per query, but requires $\tilde{\Omega}(n)$ queries to test when the characters at $\Theta(\frac{n}{\log n})$ indices are erased offline (in advance).*

² We note that there is an even simpler property than sortedness and Lipschitzness of sequences that is easily testable with offline erasures or corruptions, but is not testable at all with online erasures. Moreover, this property is over the alphabet $\Sigma = \{0, 1\}$. Specifically, $\mathcal{P} = \{ww : w \in \{0, 1\}^*\}$. We can estimate the distance to this property within additive error ε using $O(\frac{1}{\varepsilon^2})$ queries (and therefore it is easily tolerantly testable). However, we cannot test it with online erasures because, whenever the tester queries some position p in an input string of length $2n$, the adversary can erase the corresponding position $(n + p \text{ if } p \leq n \text{ and } p - n \text{ if } p > n)$.

³ The notation $\tilde{\Omega}(f(n))$ hides $\text{polylog } f(n)$ factors, e.g., $\tilde{\Omega}(\log n) = \Omega(\log n / \text{polylog } \log n)$.

In fact, the query complexity separations in Theorems 1.1 and 1.3 hold even if the online adversary can make *corruptions* instead of erasures.

To prove Theorems 1.1 and 1.3, we show that any property that separates the offline model from the standard one, i.e., a property that is easy to test in the standard model but hard in the offline model, can be “lifted” to separate the offline and online models. We can then leverage the state of the art separation (between the standard and offline-manipulation testing models) of [5] to obtain strong separations between the online and offline manipulation models.

Our “lifting” operation is simply encoding the property with a repetition code. Intuitively, repetition won’t make the property easier to test in the presence of offline erasures since the same erasures can be made on all copies. Our main technical proof for this part is to show that repetition code is robust against online adversaries. We prove the following general result.

► **Lemma 1.4** (Informal version of Lemma 3.5). *If a property \mathcal{P} is testable when no adversary is present then the repeated version \mathcal{P}^r , given by concatenating each string in \mathcal{P} with itself r times, is testable in the presence of an online adversary (as long as the number of erasures per query is not too large).*

1.1.2 Randomness complexity separations

As discussed earlier, the investigation of the randomness complexity of property testing algorithms was initiated by Goldreich and Sheffet [18] who showed that every property that is testable in the standard model using q queries can be tested using $O(q)$ queries and $O(\log n)$ random bits. We build on their investigation by studying the randomness complexity of the erasure models. Though their result easily extends to the offline models, the same extension does not work in the online model. Indeed, we show that the derandomization of [18] cannot be extended to the online testing model – that is, we construct a property that requires $\omega(\log n)$ random bits to test in the online model.

Note that a randomness separation trivially holds for properties which are not testable with online erasures, or when the offline-erasure-resilient tester is allowed to query the entire input⁴. Prior to our work it was not clear if there is a randomness separation that applies to properties that are testable in the online model, and still holds if we require that the testers in both models to have a sublinear (in n) query complexity.

We demonstrate such a separation. In particular, we show that there exists a property that is testable in both models using a sublinear number of queries, and, moreover, testing this property in the online erasure model requires exponentially more random bits than testing it in the offline erasure model.

► **Theorem 1.5** (Informal version of Corollary 4.2). *There exists a property that can be tested using $O(\sqrt{n})$ queries in the presence of either an online or an offline adversary. The tester against the offline adversary uses $O(\log n)$ random bits, whereas every tester in the presence of an online adversary that makes one erasure per query requires $\Omega(\sqrt{n})$ random bits to succeed with constant probability.*

Theorem 1.5 follows easily from a more general result, Theorem 4.1, which we prove in Section 4 and which yields meaningful separations for larger online erasure rates as well. The main technique we develop to prove a lower bound on the randomness complexity of testing

⁴ In this case, the tester in the offline model is deterministic. Note that it is impossible to deterministically test any nontrivial property in the online model, since the adversary can erase all but the first query made by a deterministic tester.

with an online adversary is a transformation from randomness-efficient testers in the online model to query efficient testers in the standard model. A special case of the guarantees of our transformation is stated in Lemma 1.6. (The general version appears in Lemma 4.3.)

To prove Theorem 4.1, we combine our transformation with existing results regarding the property of strings called τ -Distinct-Elements, which is parametrized by $\tau \in \mathbb{N}$, and consists of strings that have at most τ distinct characters. Testing τ -Distinct-Elements was recently investigated in [17, 2, 1, 20], and it is a natural testing version of the distinct elements approximation problem studied in [28, 32]. For a more detailed exposition regarding the history of the τ -Distinct-Elements property and its variants see [20].

► **Lemma 1.6** (Special case of Lemma 4.3 for one online erasure per query). *Fix $r \in \mathbb{N}$. If a property \mathcal{P} can be tested using r random bits in the presence of an online adversary that makes one erasure per query, then \mathcal{P} can be tested in the standard model (with the same proximity parameter ε) using at most r queries.*

The technique from Lemma 1.6, combined with the derandomization results of [18], immediately yields a separation for $\varepsilon = o(\frac{1}{\log n})$: Consider the property $\mathcal{P} \subseteq \{0, 1\}^n$ of being the zero string. By the folklore property testing bound, $\Theta(\frac{1}{\varepsilon})$ queries are necessary and sufficient to test \mathcal{P} . By [18], there exists a tester in the offline erasure model that uses $O(\log n)$ random bits. However, by Lemma 1.6, every tester in the online model must use $\Omega(\frac{1}{\varepsilon}) = \omega(\log n)$ random bits. While this works for small ε , the separation we show in Theorem 1.5 (and the more general Theorem 4.1) holds for constant ε as well.

1.2 Related work

Relationship between offline and standard models

In terms of query complexity, testing with offline erasures lies between standard and tolerant testing. By definition, an offline-erasure-resilient tester is also a property tester in the standard model, because it has to work on the inputs with no erasures. Not surprisingly, offline testing with erasures is no harder than offline testing with corruptions. This is formalized in [9, Theorem 1.4] that shows that the existence of a tolerant tester for a property implies the existence of an offline-erasure-resilient tester for that property for a comparable setting of parameters. There are also separations that show that standard testing is strictly easier than offline-erasure-resilient testing, which in turn is strictly easier than tolerant testing. Specifically, Dixit et al. [9] showed that a property defined by Fischer and Fortnow [13] can be tested in the standard model with a constant number of queries, but requires $n^{\Omega(1)}$ queries in the offline-erasure-resilient model. This separation was strengthened by Ben-Eliezer et al. [5], improving $n^{\Omega(1)}$ to $\tilde{\Omega}(n)$. Finally, Raskhodnikova et al. [29] prove a separation similar to that of [9] between offline-erasure-resilient testing and tolerant testing. Thus, at a high level, we have a complete picture in terms of the relative difficulty of testing in the three offline models.

Relationship between offline and online testing

Kalemaj et al. [21] exhibit two properties of strings for which testing with online erasures is impossible for every proximity parameter $\varepsilon \leq \frac{1}{2}$, whereas testing in the offline models is easy. The first property, *sortedness*, consists of sorted arrays, i.e., strings x of length n such that $x_i \leq x_{i+1}$ for all $i \in [n-1]$. The query complexity of testing sortedness (for constant ε) is $\Theta(\log n)$ in both the standard and the offline erasures model (when the erased part is at most a constant fraction of the input) [11, 10, 26, 12, 7, 8, 27, 4, 9]. The second property, *Lipschitzness*, consists of strings $x \in \{0, 1, 2\}^n$ satisfying $|x_i - x_{i+1}| \leq 1$ for all $i \in [n-1]$. It can be tested with $\Theta(\frac{1}{\varepsilon})$ queries in both the standard and the offline models [19, 9].

2 Preliminaries

Notation

We use $[n]$ to denote the set of integers $\{1, 2, \dots, n\}$, and \mathbb{N} to denote the set of positive integers.

Property testing

We start by stating standard property testing definitions. We represent an input to a property testing algorithm as a string of length n over some alphabet Σ_n that might depend on n . For example, Σ_n might be $\{0, 1\}$ or $[n]$.

► **Definition 2.1** (Relative Hamming distance, property, ε -far). *The relative Hamming distance between two strings x, y of length n is $\delta_H(x, y) = \Pr_{i \sim [n]}[x_i \neq y_i]$ where i is uniformly random index from $[n]$. For a set \mathcal{S} of strings of length n , define $\delta_H(x, \mathcal{S}) = \min_{y \in \mathcal{S}} \delta_H(x, y)$. A property \mathcal{P} is a subset of Σ^* given by $\bigcup_{n \in \mathbb{N}} \mathcal{P}_n$, where each \mathcal{P}_n consists of strings of length n over some alphabet Σ_n . A string x of length n is ε -far from \mathcal{P} if $\delta_H(x, \mathcal{P}_n) \geq \varepsilon$.*

► **Definition 2.2** (ε -tester in the standard model [31, 15]). *For every property $\mathcal{P} \subseteq \Sigma^*$ and proximity parameter $\varepsilon \in (0, 1)$, an algorithm \mathcal{T} is an ε -tester for \mathcal{P} if, given a parameter $n \in \mathbb{N}$ and oracle access to input $x \in \Sigma^n$, the algorithm \mathcal{T} accepts with probability at least $\frac{2}{3}$ whenever $x \in \mathcal{P}$ and rejects with probability at least $\frac{2}{3}$ whenever x is ε -far from \mathcal{P} . A tester has a one-sided error if it always accepts inputs $x \in \mathcal{P}$.*

Offline manipulations

The offline model with erasures was introduced by [9] and subsequently studied in [30, 29, 5, 23]. The definition we use here is adapted from [23] and only differs from the original definition in how the parameter ε is interpreted. We use \perp to denote an erased symbol in the input.

► **Definition 2.3** (α -erased string, completion). *For each $\alpha \in (0, 1)$, a string $x \in (\Sigma \cup \{\perp\})^n$ is α -erased if at most an α fraction of symbols in it are \perp . The indices of the \perp symbols in the string are called erased. A string $y \in \Sigma^n$ that differs from a string $x \in (\Sigma \cup \{\perp\})^n$ only on indices erased in x is called a completion of x .*

► **Definition 2.4** (Offline-erasure-resilient tester [9]). *For every property $\mathcal{P} \subseteq \Sigma^*$ and parameters $\alpha \in [0, 1)$ and $\varepsilon \in (0, 1)$, an algorithm \mathcal{T} is an α -offline-erasure-resilient ε -tester for \mathcal{P} if, given a parameter $n \in \mathbb{N}$ and oracle access to an α -erased input $x \in \Sigma^n$, the algorithm \mathcal{T} accepts with probability at least $\frac{2}{3}$ whenever there exists a completion $y \in \mathcal{P}$ of x and rejects with probability at least $\frac{2}{3}$ whenever every completion y of x is ε -far from \mathcal{P} .*

When $\alpha = 0$, the α -offline-erasure-resilient model is the same as the standard model. Another generalization of property testing is tolerant testing, introduced by [25] and extensively studied over the last decades. It can be viewed as guaranteeing resilience to an α fraction of the input being corrupted by an offline adversary.

► **Definition 2.5** ((α, ε) -tolerant tester [25]). *For every property $\mathcal{P} \subseteq \Sigma^*$ and parameters $\alpha, \varepsilon \in (0, 1)$, an algorithm \mathcal{T} is an (α, ε) -tolerant tester for \mathcal{P} if, given a parameter $n \in \mathbb{N}$ and oracle access to input $x \in \Sigma^n$, the algorithm \mathcal{T} accepts with probability at least $\frac{2}{3}$ whenever x is α -close to \mathcal{P} and rejects with probability at least $\frac{2}{3}$ whenever x is ε -far from \mathcal{P} .*

When we want to stress comparison to other models we study, we call an (α, ε) -tolerant tester an α -offline-corruption-resilient ε -tester.

Online manipulations

In this model, the input string $x \in \Sigma^n$ is accessed via an adversarial oracle \mathcal{O} . After answering each query made by the algorithm, the adversary can erase or corrupt a small number of data points. At the beginning of the execution of the algorithm, $\mathcal{O}(i) = x_i$ for all $i \in [n]$. So, the first query is always answered correctly. The number of queries the adversary can manipulate *after* answering each query is parameterized by $t \in \mathbb{N}$. The manipulated values are used by the oracle to answer future queries to the corresponding indices. The algorithm does not know which input locations have been tempered with. As stated in [21], “the actions of the oracle can depend on the input, the queries made so far, and even on the publicly known code that the algorithm is running, but *not* on future coin tosses of the algorithm.”

► **Definition 2.6** (Adversarial oracle, online testers [21]). *Fix $t \in \mathbb{N}$. A t -online-erasure oracle can replace values $\mathcal{O}(i)$ on up to t indices $i \in [n]$ with the erasure symbol \perp after answering each query. A t -online-corruption oracle is defined analogously, except that it replaces each symbol with an arbitrary symbol from Σ instead of erasing it. For each $t \in \mathbb{N}$, a t -online-erasure-resilient ε -tester \mathcal{T} is defined as in the standard model (Definition 2.2), except that it accesses its input via a t -online-erasure oracle as opposed to querying the input directly. The t -online-corruption-resilient ε -tester is defined analogously.*

The standard property testing model is a special case of this enhanced model and corresponds to the case when $t = 0$. Moreover, it is clear from the definition that testing with online erasures is no harder than testing with online corruptions. Specifically, every t -online-corruption-resilient tester can be simulated by a tester that accesses its input via a t -online-erasure oracle – the tester can simply replace each \perp with an arbitrary value from Σ .

3 Query complexity separation

In this section, we prove that testing with an offline adversary can require more queries than testing with an online adversary.

► **Theorem 3.1** (Query complexity separation). *Fix $\ell \in \mathbb{N}$ and let $r = r(n) < n$ be a function of the input length n . There exists a property $\mathcal{P} \subseteq \{0, 1\}^*$ and a constant $\varepsilon_1 = \varepsilon_1(\ell) \in (0, 1)$ such that:⁵*

1. *For all $\varepsilon \in (0, 1)$ and $t = (\frac{\varepsilon}{2})^{O(\ell)} \cdot r$, the property \mathcal{P} is t -online-corruption-resiliently ε -testable using $(\frac{2}{\varepsilon})^{O(\ell)}$ queries.*
2. *For all $n \in \mathbb{N}$, $\varepsilon \in (0, \varepsilon_1)$, and $\alpha = \Omega\left(\frac{1}{\log^{(\ell)}(n/r)}\right)$ such that $\varepsilon + \alpha < 1$, every α -offline-erasure-resilient ε -tester for \mathcal{P} must make $\Omega\left(\frac{n}{r \cdot \text{polylog}^{(\ell)}(n/r)}\right)$ queries on inputs of length n .*

For $\ell = 1$ and $r = \frac{n}{\log n}$, we obtain the following corollary.

► **Corollary 3.2.** *There exist a property \mathcal{P} such that for all $\varepsilon \in (0, 1)$ and $t \leq \text{poly}(\varepsilon) \cdot \frac{n}{\log n}$, the property \mathcal{P} is ε -testable using $\text{poly}(\frac{1}{\varepsilon})$ queries in the t -online-corruption model. However, for all $\alpha = \Omega\left(\frac{1}{\log \log n}\right)$ such that $\alpha + \varepsilon < 1$, every α -offline-erasure-resilient ε -tester requires $\tilde{\Omega}(\log n)$ queries.*

⁵ We use $\log^{(\ell)}$ to denote the log function applied ℓ times.

While Corollary 3.2 shows that testing in the α -offline-erasure model can be harder than testing in the t -online-model for large t and small α , the difference in query complexity is quite mild. To obtain a large gap in query complexity from Theorem 3.1, we fix $\ell \in \mathbb{N}$ and constant proximity parameter $\varepsilon \leq \varepsilon_1(\ell)$, and $r = r(\varepsilon, \ell)$ such that $t = 1$. This yields the following corollary.

► **Corollary 3.3.** *Fix $\ell \in \mathbb{N}$ and a constant $\varepsilon = \varepsilon(\ell)$. There exists a property \mathcal{P} such that \mathcal{P} is ε -testable in the 1-online-corruption model using constantly many queries. But, for all $\alpha = \Omega\left(\frac{1}{\log^{(\ell)} n}\right)$ such that $\alpha + \varepsilon < 1$, every α -offline-erasure-resilient ε -tester requires $\Omega(n / \text{polylog}^{(\ell)} n)$ queries.*

One feature of the results in Theorem 3.1 (as well as Corollaries 3.2 and 3.3) is that they hold even when the adversary in the online model is allowed to make corruptions, while the adversary in the offline model is restricted to erasures.

To prove Theorem 3.1, we introduce the following “lifting” result (Lemma 3.5). Informally, the lemma states that every property \mathcal{P} that is testable in the standard model has an encoding \mathcal{P}^r that is testable with the same query complexity even in the presence of an online-corruption adversary. This result allows us to transfer existing separations between the standard model and the offline-erasure model to a separation between the online-corruption model and the offline-erasure model.

To “lift” a property \mathcal{P} , we simply encoded it with a repetition code, that is, repeat the corresponding input string.

► **Definition 3.4** (r -concatenated string x^r and property \mathcal{P}^r). *For all $r \in \mathbb{N}$ and strings x , let x^r denote the concatenation of r copies of x , written $x^r[1] \circ x^r[2] \circ \dots \circ x^r[r]$. Additionally, for a property \mathcal{P} of strings, let \mathcal{P}^r denote the property $\{x^r : x \in \mathcal{P}\}$.*

Our lifting lemma holds for properties of strings over any alphabet.

► **Lemma 3.5** (Lifting lemma). *Let Σ be an alphabet and $\delta \in (0, 1)$ be a sufficiently small constant. Let $\mathcal{P} \subseteq \Sigma^*$ be a property of strings that is ε -testable in the standard model using $q(m, \varepsilon)$ queries on inputs of length m , where $q(m, \varepsilon) = \Omega\left(\frac{1}{\varepsilon}\right)$. Then, for all $r \in \mathbb{N}$ and $t \leq \frac{\delta \cdot r}{\lceil q(m, \frac{\varepsilon}{2}) \log q(m, \frac{\varepsilon}{2}) \rceil^2}$, the property \mathcal{P}^r is t -online-corruption-resiliently ε -testable using $\tilde{O}(q(m, \frac{\varepsilon}{2}))$ queries on inputs of length $n = m \cdot r$.*

Next we use Lemma 3.5 to prove Theorem 3.1, deferring the proof of Lemma 3.5 to Section 3.1. We leverage the following result of Ben-Eliezer, Fisher, Levi, and Rothblum [5], which separates the offline-erasure-resilient model from the standard model.

► **Theorem 3.6** ([5, Theorem 6.2]). *For all constant $\ell \in \mathbb{N}$, there exist a property $\mathcal{Q}^{(\ell)} \subseteq \{0, 1\}^*$ and a constant $\varepsilon_1 = \varepsilon_1(\ell) \in (0, 1)$ such that the following hold:*

1. *For every $\varepsilon \in (0, 1)$, the property $\mathcal{Q}^{(\ell)}$ can be ε -tested using $\left(\frac{2}{\varepsilon}\right)^{O(\ell)}$ queries.*
2. *For all $m \in \mathbb{N}$, $\varepsilon \in (0, \varepsilon_1)$, and $\alpha = \Omega(1 / \log^{(\ell)} m)$ satisfying $\varepsilon + \alpha < 1$, every α -erasure resilient ε -tester for $\mathcal{Q}^{(\ell)}$ must make $\Omega(m / (10^\ell \cdot \text{polylog}^{(\ell)} m))$ queries on inputs of length m .*

Proof of Theorem 3.1. Fix $\ell \in \mathbb{N}$ and let $m, r \in \mathbb{N}$ be sufficiently large. Let \mathcal{P} denote the property $\mathcal{Q}^{(\ell)}$ of strings of length m given by Theorem 3.6. By Theorem 3.6, the property \mathcal{P} can be ε -tested using $\left(\frac{2}{\varepsilon}\right)^{O(\ell)}$ queries. Thus, Lemma 3.5 guarantees that for all $t = \left(\frac{\varepsilon}{2}\right)^{O(\ell)} \cdot r$, the property \mathcal{P}^r is t -online-corruption-resiliently ε -testable using $\left(\frac{2}{\varepsilon}\right)^{O(\ell)}$ queries on inputs of length $n = mr$.

Now we prove that \mathcal{P}^r requires $\Omega\left(\frac{n}{r \cdot \text{polylog}^{(\ell)}(n/r)}\right)$ queries to ε -test in the α -offline-erasure-resilient model. To do that, we show how to use any α -offline-erasure-resilient ε -tester \mathcal{T}_r for \mathcal{P}^r to construct an α -offline-erasure-resilient ε -tester \mathcal{T} for \mathcal{P} with the same query complexity as \mathcal{T}_r . Given a string x^r , let $x^r[i]_j$ denote index j of the i -th copy of x . Observe that for every α -erased input x , the string x^r is also α -erased and the distance of x from \mathcal{P} is the same as the distance of x^r from \mathcal{P}^r ; moreover, for all $i \in [r]$ and $j \in [m]$, we have $x^r[i]_j = x_j$. Thus, the tester \mathcal{T} , given query access to x , can simulate \mathcal{T}_r executed with query access to x^r and output the result given by \mathcal{T}_r . By Theorem 3.6, the tester \mathcal{T}_r has query complexity $\Omega\left(\frac{m}{\text{polylog}^{(\ell)} m}\right) = \Omega\left(\frac{n}{r \cdot \text{polylog}^{(\ell)}(n/r)}\right)$ whenever $\alpha = \Omega\left(\frac{1}{\log^{(\ell)}(n/r)}\right)$. ◀

3.1 Proof of the lifting lemma (Lemma 3.5)

Recall that Lemma 3.5 states that if a property \mathcal{P} is testable in the standard model, then the property \mathcal{P}^r is testable in the online-corruption model. Let \mathcal{T} be an ε -tester for \mathcal{P} that has failure probability $\frac{1}{12}$, and query complexity $c_0 \cdot q(m, \varepsilon)$ for some constant $c_0 > 0$ (we can obtain such a tester via standard amplification techniques). We construct a tester \mathcal{T}' for \mathcal{P}^r (Algorithm 1) and show that it is t -online-corruption-resilient. The tester \mathcal{T}' consists of two phases: first, $\mathcal{T}'(s)$ calls REPETITION-TEST (Algorithm 2) to test that s is a repetition of some string, w , and second, $\mathcal{T}'(s)$ simulates \mathcal{T} with query access to w .

For a string s of length $n = r \cdot m$ composed of r substrings each of length m , we use the notation $s[i]_j$ to denote index j , in the i -th substring of s .

■ **Algorithm 1** t -online-corruption-resilient ε -tester \mathcal{T}' .

Parameters: length parameter m , repetition parameter r , proximity parameter $\varepsilon \in (0, 1)$

Input: query access to string $s = s[1] \circ \dots \circ s[r]$ such that $|s[i]| = m$ for each $i \in [r]$

Subroutines: REPETITION-TEST (Algorithm 2) and $c_0 \cdot q(m, \varepsilon)$ -query tester \mathcal{T} for \mathcal{P}

- 1: $c_1 \leftarrow 24 \cdot c_0$
- 2: **if** REPETITION-TEST($s, \frac{\varepsilon}{2}$) **rejects** **then reject**
- 3: simulate \mathcal{T} with proximity parameter $\frac{\varepsilon}{2}$, answering each query j made by \mathcal{T} as follows:
 - sample a set of $d = \log(c_1 \cdot q(m, \frac{\varepsilon}{2}))$ uniform indices $i_1, \dots, i_d \in [r]$
 - if** there exists a pair of indices $k, k' \in [d]$ such that $s[i_k]_j \neq s[i_{k'}]_j$ **then reject**
 - else** provide the answer $s[i_1]_j$ to \mathcal{T}
- 4: **if** the simulation of \mathcal{T} **rejects** **then reject**
- 5: **else accept**

■ **Algorithm 2** REPETITION-TEST.

Parameters: length parameter m , repetition parameter r , proximity parameter $\varepsilon \in (0, 1)$

Input: query access to string $s = s[1] \circ \dots \circ s[r]$ such that $|s[i]| = m$ for each $i \in [r]$

- 1: **repeat** $\frac{2}{\varepsilon}$ **times**:
 - sample $i_1, i_2 \sim [r]$ and $j \sim [m]$ uniformly and independently at random
- 2: **if** $s[i_1]_j \neq s[i_2]_j$ **then reject**
- 3: **else accept**

We start by analyzing REPETITION-TEST (Algorithm 2).

▷ **Claim 3.7.** For all alphabets Σ and parameters $\varepsilon \in (0, 1)$ and $r, m \in \mathbb{N}$, REPETITION-TEST is a one-sided error ε -tester for the repetition code defined as $\mathcal{C}_{m,r} = \{s^r : s \in \Sigma^m\}$. It makes $O(\frac{1}{\varepsilon})$ queries and has error probability at most $\frac{1}{6}$.

Proof. Let $s = s[1] \circ \dots \circ s[r]$, where $s[i] = s[i]_1 \dots s[i]_m$ for all $i \in [r]$, be the input string over the alphabet Σ . If $s \in \mathcal{C}_{m,r}$, then REPETITION-TEST always accepts s .

Suppose that string s is ε -far from $\mathcal{C}_{m,r}$. Let $\text{plurality}_{i \in [r]}(a_i)$ denote a function that takes inputs $a_1, \dots, a_r \in \Sigma$ and outputs the most frequent among them, resolving ties arbitrarily. Let \hat{w} be the string of length m defined by $\hat{w}_j = \text{plurality}_{i \in [r]} s[i]_j$ for all $j \in [m]$. Since s is ε -far from $\mathcal{C}_{m,r}$, we can bound the distance between the input string and \hat{w} repeated r times:

$$\delta_H(s, \hat{w}^r) \geq \varepsilon. \quad (1)$$

For all $j \in [m]$ and $a \in \Sigma$, let $p_j(a)$ denote $\Pr_{i \in [r]}[s[i]_j = a]$, the probability that a randomly chosen repetition of the j -th character in s is equal to a . The probability that one iteration of the loop in REPETITION-TEST accepts s is

$$\begin{aligned} \Pr_{\substack{i_1, i_2 \in [r] \\ j \in [m]}} [s[i_1]_j = s[i_2]_j] &= \frac{1}{m} \sum_{j \in [m]} \sum_{a \in \Sigma} [p_j(a)]^2 && \text{by law of total probability} \\ &\leq \frac{1}{m} \sum_{j \in [m]} p_j(\hat{w}_j) \sum_{a \in \Sigma} p_j(a) && \text{as } p_j(a) \leq p_j(\hat{w}_j) \ \forall j \in [m], a \in \Sigma \\ &= \frac{1}{m} \sum_{j \in [m]} p_j(\hat{w}_j) && \text{since } \sum_{a \in \Sigma} p_j(a) = 1 \ \forall j \in [m] \\ &= \Pr_{\substack{i \in [r] \\ j \in [m]}} [s[i]_j = \hat{w}_j] && \text{by law of total probability} \\ &= 1 - \delta_H(s, \hat{w}^r) && \text{by definition of } \delta_H \\ &\leq 1 - \varepsilon && \text{by (1).} \end{aligned}$$

Hence, each iteration of the loop in REPETITION-TEST rejects with probability at least ε . Therefore, REPETITION-TEST accepts s with probability at most $(1 - \varepsilon)^{2/\varepsilon} \leq e^{-\varepsilon \cdot 2/\varepsilon} \leq \frac{1}{6}$. \triangleleft

Next, we analyze the tester \mathcal{T}' .

▷ **Claim 3.8** (\mathcal{T}' is a tester in the standard model). For all $\varepsilon \in (0, 1)$, $r \in \mathbb{N}$, alphabets Σ , and properties $\mathcal{P} \in \Sigma^*$, Algorithm 1 is an ε -tester for \mathcal{P}^r in the standard model (without any adversary). Moreover, it accepts ε -far inputs with probability at most $\frac{1}{6}$, and has one-sided error whenever \mathcal{T} has one-sided error.

Proof. If the input s is in \mathcal{P}^r then $s = w^r$ for some string $w \in \mathcal{P}$. In this case, REPETITION-TEST accepts with probability 1, and \mathcal{T}' accepts with the same probability as \mathcal{T} . Next, suppose s is ε -far from \mathcal{P}^r . Consider the string $\hat{w} \in \Sigma^m$, where $\hat{w}_j = \text{plurality}_{i \in [r]} s[i]_j$ for all $j \in [m]$. Recall that the repetition code is defined as $\mathcal{C}_{m,r} = \{s^r : s \in \Sigma^m\}$. Observe that the distance from s to $\mathcal{C}_{m,r}$ is equal to $\delta_H(s, \hat{w}^r)$. If this distance is at least $\frac{\varepsilon}{2}$ then, by Claim 3.7, REPETITION-TEST accepts with probability at most $\frac{1}{6}$, completing the proof.

Now assume $\delta_H(s, \hat{w}^r) < \frac{\varepsilon}{2}$. We will show that, in this case, the simulation of \mathcal{T} in Algorithm 1 accepts with probability at most $\frac{1}{6}$. Specifically, we consider two failure events: let E_1 be the event that the simulation of \mathcal{T} feeds \mathcal{T} at least one query answer inconsistent with \hat{w} , and E_2 be the event that the simulation of \mathcal{T} accepts string \hat{w} . We will show that $\Pr[E_1] \leq \frac{1}{12}$ and $\Pr[E_2] \leq \frac{1}{12}$. Then a union bound over E_1 and E_2 completes the proof.

Since s is ε -far from \mathcal{P}^r and $\delta_H(s, \hat{w}^r) < \frac{\varepsilon}{2}$, we get

$$\varepsilon \leq \delta_H(s, \mathcal{P}^r) \leq \delta_H(s, \hat{w}^r) + \delta_H(\hat{w}^r, \mathcal{P}^r) \leq \frac{\varepsilon}{2} + \delta_H(\hat{w}, \mathcal{P}),$$

implying that $\delta_H(\hat{w}, \mathcal{P}) \geq \frac{\varepsilon}{2}$. Since \mathcal{T} is a tester for \mathcal{P} with amplified success probability, and \hat{w} is $\frac{\varepsilon}{2}$ -far from \mathcal{P} , algorithm \mathcal{T} accepts with probability at most $\frac{1}{12}$ when run on input \hat{w} and with proximity parameter $\frac{\varepsilon}{2}$; that is, $\Pr[E_2] \leq \frac{1}{12}$.

Next we analyze $\Pr[E_1]$. Fix $j \in [m]$, and let a_j denote the answer provided by \mathcal{T}' to query j in the simulation. If \mathcal{T}' does not reject while simulating this query, then

$$\begin{aligned} \Pr[a_j \neq \hat{w}_j] &= \Pr_{i_1, \dots, i_d \in [r]}[s[i_k]_j = s[i_{k'}]_j \wedge s[i_k]_j \neq \hat{w}_j \ \forall k, k' \in [d]] \\ &= \Pr_{i_1, \dots, i_d \in [r]}[s[i_k]_j = s[i_1]_j \ \forall k \in \{2, \dots, d\} \mid s[i_1]_j \neq \hat{w}_j] \cdot \Pr_{i_1 \in [r]}[s[i_1]_j \neq \hat{w}_j] \\ &\leq 2^{-d+1} \\ &= 2(c_1 \cdot q(m, \varepsilon/2))^{-1}. \end{aligned} \tag{2}$$

The inequality in (2) follows from the following simple observation: once $s[i_1]_j$ has been sampled, if it is not the plurality, then for each $k \in \{2, \dots, d\}$, every subsequent $s[i_k]_j$ is equal to $s[i_1]_j$ with probability at most $\frac{1}{2}$. By our choice of c_1 , and a union bound over all the $c_0 \cdot q(m, \varepsilon/2)$ queries made by \mathcal{T} , the probability of E_1 is at most $\frac{1}{12}$.

Since \mathcal{T}' can accept only if $E_1 \cup E_2$ occurs, a union bound over E_1 and E_2 implies that \mathcal{T}' is indeed a tester for \mathcal{P}^r , and has error probability at most $\frac{1}{6}$ when no adversary is present. \triangleleft

Finally, we prove the lifting lemma.

Proof of Lemma 3.5. First, we argue that the probability \mathcal{T}' (Algorithm 1) queries a corrupted index is small. Recall that $n = m \cdot r$ is the length of the input to \mathcal{T}' and that $c_0 \cdot q(m, \varepsilon)$ is the query complexity of \mathcal{T} with proximity parameter ε . Let $q' = q'(n, \varepsilon) = \frac{8}{\varepsilon} + c_0 \cdot q(m, \frac{\varepsilon}{2}) \log(c_1 \cdot q(m, \frac{\varepsilon}{2}))$ be the number of queries made by \mathcal{T}' . Then, there are at most $q't$ corruptions at any point during the execution. Moreover, each query made by \mathcal{T}' is an index that is uniform over the set of corresponding indices in the r different segments of s . It follows that each query made by \mathcal{T}' is corrupted with probability at most $\frac{q't}{r}$. By a union bound over the q' queries,

$$\Pr[\mathcal{T}' \text{ queries a corrupted index}] \leq \frac{(q')^2 t}{r} \leq \frac{1}{6}. \tag{3}$$

The upper bound follows from the hypothesis that $t \leq \frac{\delta \cdot r}{[q(m, \varepsilon/2) \log q(m, \varepsilon/2)]^2}$, that $q(m, \varepsilon) = \Omega(\frac{1}{\varepsilon})$, and that δ is a sufficiently small constant. By a union bound, the probability that \mathcal{T}' errs in the presence of the adversary is at most the probability \mathcal{T}' errs when no adversary is present plus the probability \mathcal{T}' queries a corrupted index. By Claim 3.8 and (3), this probability is at most $\frac{1}{3}$, which completes the proof of Lemma 3.5. \blacktriangleleft

4 Randomness complexity separation

In this section, we prove our result separating the randomness complexity of testing in the online and offline models.

► **Theorem 4.1** (Randomness separation). *For all $\tau = \tau(n)$, there exist a property $\mathcal{P}^{(\tau)} = \bigcup_{n \in \mathbb{N}} \mathcal{P}_n^{(\tau)}$, where $\mathcal{P}_n^{(\tau)} \subseteq [n]^n$, and an algorithm \mathcal{T} that takes $\varepsilon \in (0, 1)$ as an input and, for all $\varepsilon \in (0, 1)$, is a one-sided error ε -tester for $\mathcal{P}^{(\tau)}$ that make $O(\frac{\tau}{\varepsilon})$ queries. Additionally, the following hold:*

1. *For all $\varepsilon, \alpha \in (0, 1)$, tester \mathcal{T} is an α -offline-erasure-resilient ε -tester for $\mathcal{P}^{(\tau)}$. Moreover, \mathcal{T} can be simulated using $O(\log n)$ random bits and no additional queries.*
2. *For all $\varepsilon \in (0, 1)$ and $t \leq \left(\frac{0.01\varepsilon\sqrt{n}}{\tau}\right)^2$, tester \mathcal{T} is a t -online-erasure-resilient ε -tester for $\mathcal{P}^{(\tau)}$.*
3. *There exists a constant $\varepsilon > 0$ such that for all $t \in \mathbb{N}$ and $\tau \leq \frac{0.01\sqrt{n}}{\log n}$, every t -online-erasure-resilient ε -tester for $\mathcal{P}^{(\tau)}$ uses $\Omega\left(\frac{\tau \log(t+1)}{\log \tau}\right)$ random bits. Moreover, if the tester has one-sided error, then it uses $\Omega(\tau \log(t+1))$ random bits.*

Theorem 4.1 yields meaningful separations for all settings of τ between $\Omega(\log n)$ and $O\left(\frac{\sqrt{n}}{\log n}\right)$. As an example, the following corollary is derived by setting $\tau = \frac{0.01\varepsilon\sqrt{n}}{\log n}$ and $t = 1$.

► **Corollary 4.2.** *There exists a property that can be ε -tested (for every constant ε) using $O\left(\frac{\sqrt{n}}{\log n}\right)$ queries in the presence of either an online or offline adversary. The offline-erasure-resilient tester uses $O(\log n)$ random bits, but there exists a constant ε such that every 1-online-erasure-resilient ε -tester, requires $\Omega\left(\frac{\sqrt{n}}{\text{polylog } n}\right)$ random bits to succeed with constant probability. Moreover, if the 1-online-erasure-resilient tester has one-sided error then it requires $\Omega(\sqrt{n})$ random bits.*

To prove our randomness lower bound, we present a general reduction from randomness-efficient testers in the online-erasure model to query-efficient testers in the standard model.

► **Lemma 4.3** (Generalization of Lemma 1.6). *For all $r, t \in \mathbb{N}$ and properties \mathcal{P} , if \mathcal{P} is ε -testable in the presence of a t -online-erasure adversary using only r bits of randomness then \mathcal{P} is ε -testable in the standard model using at most $\frac{r}{\log(t+1)}$ queries.*

Proof. Let \mathcal{T} be a t -online-erasure-resilient ε -tester for \mathcal{P} that uses at most r random bits. We construct an adversary that allows \mathcal{T} to make at most $\frac{r}{\log(t+1)}$ non-erased queries. Consider the following random seed elimination adversary \mathcal{A} . Given an input x , the description of algorithm \mathcal{T} , and query-answer history of \mathcal{T} on input x , the adversary simulates the next query of \mathcal{T} on every random seed $s \in \{0, 1\}^r$ that is consistent with the query-answer history of \mathcal{T} thus far and erases the t most queried indices of x . Intuitively, these are the indices that are most likely to be queried by the tester at this step, based on the random seeds that are consistent with the current query-answer history. Each index that the oracle decides not to erase at this step can appear in at most $\frac{1}{t+1}$ fraction of currently relevant random seeds. If \mathcal{T} queries such an index, all random seeds that would have lead to a different query can be eliminated. Thus, each time \mathcal{T} makes a non-erased query, at most a $\frac{1}{t+1}$ fraction of the relevant random seeds remain. Consequently, \mathcal{T} can make at most $\frac{r}{\log(t+1)}$ non-erased queries before \mathcal{A} can exactly determine the random seed being used. Once \mathcal{A} determines the random seed, it can exactly predict the queries of \mathcal{T} and erase all of them.

Next, we use the adversary strategy \mathcal{A} to construct an $\frac{r}{\log(t+1)}$ -query tester for \mathcal{P} . Let \mathcal{T}^* be defined as follows. Draw a random seed $s^* \in \{0, 1\}^r$ and simulate \mathcal{T} with random seed s^* . After making each query, \mathcal{T}^* simulates \mathcal{A} with the query-answer history of \mathcal{T} . If \mathcal{A} erases the next query of \mathcal{T} , then \mathcal{T}^* answers the query with \perp and continues the simulation (the algorithm \mathcal{T}^* need not make any queries in this case). Otherwise, if \mathcal{A} does not erase

the next query of \mathcal{T} , then \mathcal{T}^* queries x , provides \mathcal{T} with the answer, and continues the simulation. Since \mathcal{T} makes at most $\frac{\tau}{\log(t+1)}$ non-erased queries and successfully tests \mathcal{P} , the algorithm \mathcal{T}^* makes at most $\frac{\tau}{\log(t+1)}$ queries and also successfully tests \mathcal{P} . \blacktriangleleft

4.1 The τ -Distinct-Elements property

The property testing problem we use to prove Theorem 4.1 is a version of support size approximation. In support size approximation, one is given sample access to a distribution \mathcal{D} over the domain $[n]$ and asked to approximate $|\text{supp}(\mathcal{D})|$ up to additive error ε . We study the analogous testing problem over strings: Given parameters $\tau \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, and a string x of length n , we wish to ε -test whether $|\{x_1, x_2, \dots, x_n\}| \leq \tau$, that is, to determine whether x has at most τ distinct elements or x is ε -far from every x' (of length n) with at most τ distinct elements.

► **Definition 4.4** (τ -Distinct-Elements). *For all $n \in \mathbb{N}$ and $\tau = \tau(n)$, let $\mathcal{P}_n^{(\tau)}$ be the set of strings in $[n]^n$ with at most τ distinct symbols. The property τ -Distinct-Elements is defined as $\mathcal{P}^{(\tau)} = \bigcup_{n \in \mathbb{N}} \mathcal{P}_n^{(\tau)}$.*

The following tester for τ -Distinct-Elements appeared in [17]. We include the analysis of its guarantees here for completeness.

► **Fact 4.5** (A tester for τ -Distinct-Elements). *For all $\varepsilon \in (0, 1)$, there exists a one-sided error tester for $\mathcal{P}^{(\tau)}$ that uses $O(\frac{\tau}{\varepsilon})$ samples and fails with probability at most $\frac{1}{10}$.*

Proof. We first make the following observation: if x is ε -far from $\mathcal{P}^{(\tau)}$, then every collection of τ distinct elements occupies at most a $1 - \varepsilon$ fraction of the indices in x . Suppose we sample uniformly random elements from x until we have seen $\tau + 1$ distinct elements. Notice that while we have seen at most τ distinct elements, our next sample is a new distinct element with probability at least ε (the at most τ distinct elements we have seen can have probability mass at most $1 - \varepsilon$). Let X_i be the number of samples to get the i -th distinct element given we have seen $i - 1$ distinct elements, and set $X = \sum_{i=1}^{\tau+1} X_i$. Then $\mathbb{E}[X] \leq (\tau + 1)/\varepsilon$, and hence, by Markov's inequality, the probability we need more than $3(\tau + 1)/\varepsilon$ samples is at most $\frac{1}{3}$. This analysis inspires the following simple tester:

Sample $s = 3(\tau + 1)/\varepsilon$ elements from x and accept if and only if there are at most τ distinct symbols from $[n]$ in the sample (not counting the erasure symbol \perp).

Repeating the tester constantly many times suffices to make the failure probability smaller than $\frac{1}{10}$. \blacktriangleleft

Recall that Lemma 4.3 states that to prove a lower bound on the randomness complexity of testing in the online model, it suffices to prove a lower bound on the query complexity of testing in the standard model. In Lemma 4.6, we show a lower bound on the query complexity of testing τ -Distinct-Elements. Our lower bound is a direct corollary of the lower bound of [32].

► **Lemma 4.6** (Query lower bound for testing τ -Distinct-Elements). *There exists $\varepsilon \in (0, 1)$ such that for all input lengths n , if $\tau \leq \frac{\sqrt{n}}{\log n}$, then the query complexity of ε -testing $\mathcal{P}^{(\tau)}$ is $\Omega(\frac{\tau}{\log \tau})$. Moreover, the query complexity of ε -testing $\mathcal{P}^{(\tau)}$ with one-sided error is $\Omega(\tau)$.*

Proof. We use the following fact to argue that we can assume the tester is sample-based with no loss of generality.

▷ **Fact 4.7** (Theorem 6.1 [16]). A property of strings is *symmetric* if it is closed under permutations of the string indices (that is, under reordering of the characters in the string). Let \mathcal{P} be a symmetric property of strings that is ε -testable with query complexity q . Then there exists a sample-based tester for \mathcal{P} with sample complexity $O(q)$.

Since $\mathcal{P}^{(\tau)}$ is symmetric, we can without loss of generality consider sample-based testers. First, we prove the lower bound for two-sided error testers. The essence of our proof is a reduction from the hard instances used in [32] for estimating the support size of a distribution. While we cannot use the lower bound of [32] as a black box, we can nonetheless leverage their hard instances. Indeed, combining Theorem 1, Fact 5, and the remarks regarding the support size of their hard distributions (following Definition 12) from [32], we obtain the following immediate corollary.

▷ **Fact 4.8** (Support size approximation lower bound [32]). Let $\alpha > 0$ be a sufficiently small constant. There exist distributions \mathcal{D}^+ and \mathcal{D}^- over $[m]$ such that

1. Every element in the support of \mathcal{D}^+ or \mathcal{D}^- has probability mass at least $\frac{1}{m}$.
2. $|\text{supp}(\mathcal{D}^+)| \leq m(\frac{1}{2} + \alpha)$, and \mathcal{D}^- is $(\frac{1}{2} - 2\alpha)$ -far from every distribution with support size at most $m(\frac{1}{2} + \alpha)$.

Moreover, every algorithm that successfully distinguishes \mathcal{D}^+ from \mathcal{D}^- with constant probability requires $\Omega\left(\frac{m}{\log m}\right)$ samples.

While the distributions in [32] are defined over \mathbb{R} , they all have support size at most m . Thus, we can, without loss of generality, consider distributions over $[m]$. Indeed, given a distribution \mathcal{D} over \mathbb{R} , the distribution \mathcal{D}^π obtained by choosing a uniformly random permutation π of $[m]$, and then mapping the i -th element in the support of \mathcal{D} to $\pi(i)$, has domain $[m]$ and the same support size as \mathcal{D} . Moreover, if two distributions \mathcal{D}_0 and \mathcal{D}_1 each have support size at most m , then for $b \in \{0, 1\}$, the distribution \mathcal{D}_b^π can be generated by first, sampling a random permutation π of $[m]$, second, sampling from \mathcal{D}_b , and third, sending each distinct sample to the first available element of π . Thus, any algorithm which distinguishes \mathcal{D}_1^π and \mathcal{D}_0^π can be used to distinguish \mathcal{D}_0 and \mathcal{D}_1 with the same sample complexity.

To apply Fact 4.8 in our setting, we note that whenever $m \leq \frac{\sqrt{n}}{\log n}$, there exists $x \in [m]^n$ such that the distribution obtained by sampling a uniform index in x has total variation distance at most $\frac{m}{n} = \frac{1}{\sqrt{n} \log n}$ from \mathcal{D} . Thus, for sufficiently large n , there exist strings $x^+, x^- \in [m]^n$ such that x^+ has at most $m(\frac{1}{2} + \alpha)$ distinct symbols, and the distributions generated by sampling from x^+ and x^- have total variation distance $\frac{1}{\sqrt{n} \log n}$ from \mathcal{D}^+ and \mathcal{D}^- , respectively. Let $\tau = m(\frac{1}{2} + \alpha)$ and $\varepsilon = \frac{1}{2} - 2\alpha - o(1)$, then $x^+ \in \mathcal{P}^{(\tau)}$ and x^- is ε -far from $\mathcal{P}^{(\tau)}$. By Fact 4.8, every algorithm for distinguishing x^+ from x^- uses $\Omega\left(\frac{m}{\log m}\right) = \Omega\left(\frac{\tau}{\log \tau}\right)$ samples. By Fact 4.7, every tester for $\mathcal{P}^{(\tau)}$ has query complexity $\Omega\left(\frac{\tau}{\log \tau}\right)$.

To prove the one-sided error lower bound, it suffices to note that every one-sided error algorithm must accept if its sample contains at most τ symbols. Thus, every one-sided error tester requires $\Omega(\tau)$ samples. This completes the proof of Lemma 4.6. ◀

4.2 Proof of the randomness separation (Theorem 4.1)

To complete the proof of the theorem, we argue that the tester given by Fact 4.5 is a valid tester for τ -Distinct-Elements in both the online and offline erasure models. To prove the lower bound on the randomness complexity of testing τ -Distinct-Elements in the online model, we combine the query lower bound of Lemma 4.6 with the reduction stated in Lemma 4.3. To

prove the upper bound on the randomness complexity of testing τ -Distinct-Elements in the offline model, we argue that the offline tester can be simulated with few bits of randomness at a small cost in error probability.

We first prove the part of the theorem regarding testing τ -Distinct-Elements in the offline erasure model.

Proof of Theorem 4.1, Item 1. First, for all $\tau \in \mathbb{N}$, we construct an offline-erasure-resilient tester for τ -Distinct-Elements, and second, we argue that the tester can be simulated with $O(\log n)$ random bits and no additional queries. Let \mathcal{T} be the tester given by Fact 4.5. Recall that \mathcal{T} accepts if and only if the number of nonerased symbols in the sample is at most τ . Below, we argue that \mathcal{T} is an α -offline-erasure-resilient ε -tester.

Let x be an α -erased string in $[n]^n$. If some completion of x is in $\mathcal{P}^{(\tau)}$, then the number of nonerased symbols in the sample drawn by \mathcal{T} is at most τ , so the algorithm accepts. Next we argue that \mathcal{T} accepts with probability at most $\frac{1}{4}$ when every completion of x is ε -far from $\mathcal{P}^{(\tau)}$. Let x^\perp denote the erased portion of x , and x^\top denote the nonerased portion of x . Notice that $|x^\top| = (1 - \alpha)n$, and thus if x^\top is $\frac{\varepsilon}{1 - \alpha}$ -close to $\mathcal{P}^{(\tau)}$, then there is a completion of x that is ε -close to $\mathcal{P}^{(\tau)}$ (change εn symbols in x^\top and fill out x^\perp with an arbitrary symbol from x^\top). Thus, if every completion of x is ε -far from $\mathcal{P}^{(\tau)}$, then x^\top is $\frac{\varepsilon}{1 - \alpha}$ -far from $\mathcal{P}^{(\tau)}$. By the proof of Fact 4.5, a set of $O(\frac{\tau(1 - \alpha)}{\varepsilon})$ uniform and independent samples from x^\top contains at least $\tau + 1$ distinct symbols with probability at least $\frac{9}{10}$. Moreover, a sample of $O(\frac{\tau}{\varepsilon})$ elements from x contains $\Omega(\frac{\tau(1 - \alpha)}{\varepsilon})$ such samples from x^\top with probability at least $\frac{9}{10}$. By the union bound, \mathcal{T} accepts x with probability at most $\frac{1}{4}$.

It remains to show that \mathcal{T} can be simulated by a tester \mathcal{T}' that uses $O(\log n)$ random bits and has the same query complexity as \mathcal{T} . Fact 4.9 states that any randomized oracle machine that solves a promise problem can be simulated using $\log n + \log \log |\Sigma| + O(1)$ random bits, where $|\Sigma|$ is the size of the alphabet. This fact was originally proven by Goldreich and Sheffet [18] and is stated as an exercise in [14]. We use the statement from the exercise since it is more convenient for our setting.

▷ **Fact 4.9 (Exercise 1.21 [14]).** Fix $n \in \mathbb{N}$ and let Π be a promise problem regarding strings in Σ^n . Suppose that \mathcal{M} is a randomized q -query oracle machine that solves Π with error probability at most $\frac{1}{4}$. Then there exists a randomized q -query oracle machine \mathcal{M}' that solves Π with error probability at most $\frac{1}{3}$ and $\log n + \log \log |\Sigma| + O(1)$ random bits.

Let Π be the promise problem defined by α -offline-erasure-resilient ε -testing $\mathcal{P}^{(\tau)}$. By the first part of the proof, there exists a randomized oracle machine \mathcal{T} that solves Π with error probability at most $\frac{1}{4}$. Applying Fact 4.9 directly, we see that \mathcal{T} can be simulated by a tester \mathcal{T}' using $\log n + \log \log |\Sigma| + O(1)$ random bits; since $\Sigma = [n]$, tester \mathcal{T}' uses $O(\log n)$ random bits, and has error probability at most $\frac{1}{3}$. To complete the proof, it suffices to note that \mathcal{T}' has the same query complexity as \mathcal{T} . ◀

Next, we prove the part of the theorem regarding testing τ -Distinct-Elements in the online erasure model.

Proof of Theorem 4.1, Items 2 and 3. We will show that for all $\tau \in \mathbb{N}$, the tester given by Fact 4.5 is online-erasure-resilient. Let \mathcal{T} be the tester for τ -Distinct-Elements given by Fact 4.5. Recall that \mathcal{T} accepts if and only if the number of nonerased symbols in its sample is at most τ . We show that \mathcal{T} can test in the presence of a t -online-erasure-oracle, by arguing that with high constant probability, none of the queries made by \mathcal{T} are erased. Notice that with proximity parameter ε , the tester makes $O(\frac{\tau}{\varepsilon})$ queries. Thus, there are

$O\left(\frac{t\tau}{\varepsilon}\right)$ erasures at every point during the execution of \mathcal{T} . Since \mathcal{T} makes uniform and independent queries, the probability that any particular query is erased is $O\left(\frac{t\tau}{n\varepsilon}\right)$. Since \mathcal{T} makes $O\left(\frac{\tau}{\varepsilon}\right)$ queries, we can apply the union bound to see that some query is erased with probability at most $O\left(\frac{t\tau^2}{n\varepsilon^2}\right)$. By the hypothesis that $t \leq \left(\frac{.01\varepsilon\sqrt{n}}{\tau}\right)^2$, we have $O\left(\frac{t\tau^2}{n\varepsilon^2}\right) \leq .01$ – that is, \mathcal{T} sees an erasure with probability at most .01. By Fact 4.5, the algorithm \mathcal{T} is a t -online-erasure-resilient ε -tester for $\mathcal{P}^{(\tau)}$. Moreover, since an erasure cannot increase number of distinct nonerased symbols in x , tester \mathcal{T} has one-sided error. This completes the proof of Item 2.

To see why Item 3 holds, recall that by Lemma 4.6, there exists $\varepsilon \in (0, 1)$ such that for all $n, \tau \in \mathbb{N}$ with $\tau \leq \frac{\sqrt{n}}{\log n}$, every ε -tester for $\mathcal{P}^{(\tau)}$ has query complexity $\Omega\left(\frac{\tau}{\log \tau}\right)$. Similarly every tester with one-sided error has query complexity $\Omega(\tau)$. Thus, applying Lemma 4.3, the reduction from query complexity to randomness complexity, suffices to complete the proof. ◀

References

- 1 Tomer Adar and Eldar Fischer. Refining the adaptivity notion in the huge object model. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPICs*, pages 45:1–45:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.APPROX/RANDOM.2024.45.
- 2 Tomer Adar, Eldar Fischer, and Amit Levi. Support testing in the huge object model. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPICs*, pages 46:1–46:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.APPROX/RANDOM.2024.46.
- 3 Vipul Arora, Esty Kelman, and Uri Meir. On optimal testing of linearity. In *Symposium on Simplicity in Algorithms (SOSA)*. SIAM, 2025. To appear.
- 4 Aleksandrs Belovs. Adaptive lower bound for testing monotonicity on the line. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 31:1–31:10, 2018. doi:10.4230/LIPICs.APPROX-RANDOM.2018.31.
- 5 Omri Ben-Eliezer, Eldar Fischer, Amit Levi, and Ron D. Rothblum. Hard properties with (very) short PCPPs and their applications. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, pages 9:1–9:27, 2020. doi:10.4230/LIPICs.ITCS.2020.9.
- 6 Omri Ben-Eliezer, Esty Kelman, Uri Meir, and Sofya Raskhodnikova. Property testing with online adversaries. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPICs*, pages 11:1–11:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ITCS.2024.11.
- 7 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012. doi:10.1137/110826655.
- 8 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014. doi:10.4086/toc.2014.v010a017.
- 9 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin Varma. Erasure-resilient property testing. *SIAM Journal on Computing*, 47(2):295–329, 2018. doi:10.1137/16M1075661.

- 10 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 97–108, 1999. doi:10.1007/978-3-540-48413-4_10.
- 11 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000. doi:10.1006/JCSS.1999.1692.
- 12 Eldar Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004. doi:10.1016/j.ic.2003.09.003.
- 13 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory Comput.*, 2(9):173–183, 2006. doi:10.4086/TOC.2006.V002A009.
- 14 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.
- 15 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.
- 16 Oded Goldreich and Dana Ron. On sample-based testers. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 337–345. ACM, 2015. doi:10.1145/2688073.2688080.
- 17 Oded Goldreich and Dana Ron. Testing distributions of huge objects. *CoRR*, abs/2212.12802, 2022. doi:10.48550/arXiv.2212.12802.
- 18 Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Computational Complexity*, 19:99–133, 2010. doi:10.1007/S00037-009-0282-4.
- 19 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42(2):700–731, 2013. doi:10.1137/110840741.
- 20 Renato Ferreira Pinto Jr. and Nathaniel Harms. Testing support size more efficiently than learning histograms. *CoRR*, 2024. arXiv:2410.18915, doi:10.48550/arXiv.2410.18915.
- 21 Iden Kalemaj, Sofya Raskhodnikova, and Nithin Varma. Sublinear-time computation in the presence of online erasures. *Theory Comput.*, 19 (1):1–48, 2023. URL: <https://theoryofcomputing.org/articles/v019a001/v019a001.pdf>, doi:10.4086/TOC.2023.V019A001.
- 22 Esty Kelman, Ephraim Linder, and Sofya Raskhodnikova. Online versus offline adversaries in property testing. *CoRR*, 2024. arXiv:2411.18617.
- 23 Amit Levi, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Erasure-resilient sublinear-time graph algorithms. *ACM Trans. Comput. Theory*, 14(1):1:1–1:22, 2022. doi:10.1145/3488250.
- 24 Dor Minzer and Kai Zhe Zheng. Adversarial low degree testing. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4395–4409. SIAM, 2024. doi:10.1137/1.9781611977912.154.
- 25 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.
- 26 Sofya Raskhodnikova. Monotonicity testing. *Masters Thesis, MIT*, 1999.
- 27 Sofya Raskhodnikova. Testing if an array is sorted. *Encyclopedia of Algorithms*, pages 2219–2222, 2016. doi:10.1007/978-1-4939-2864-4_700.
- 28 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS '07*, pages 559–569. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.67.

- 29 Sofya Raskhodnikova, Noga Ron-Zewi, and Nithin Varma. Erasures versus errors in local decoding and property testing. *Random Structures and Algorithms*, 59(4):640–670, 2021. doi:10.1002/rsa.21031.
- 30 Sofya Raskhodnikova and Nithin Varma. Brief announcement: Erasure-resilience versus tolerance to errors. In *Proceedings, International Colloquium on Automata, Languages and Programming (ICALP)*, pages 111:1–111:3, 2018. doi:10.4230/LIPIcs.ICALP.2018.111.
- 31 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.
- 32 Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. *Electron. Colloquium Comput. Complex.*, TR10-179, 2010. URL: <https://eccc.weizmann.ac.il/report/2010/179>.
- 33 Alek Westover, Edward Yu, and Kai Zheng. New direct sum tests. *CoRR*, abs/2409.10464, 2024. doi:10.48550/arXiv.2409.10464.