Erasure-Resilient Property Testing*

Kashyap Dixit[1], Sofya Raskhodnikova[1], Abhradeep Thakurta[2], and Nithin Varma[1]

1 **Pennsylvania State University**
kashyap@cse.psu.edu, sofya@cse.psu.edu, nzm154@psu.edu

2 **Work done while at Yahoo Labs**
guhathakurta.abhradeep@gmail.com

Abstract

Property testers form an important class of sublinear algorithms. In the standard property testing model, an algorithm accesses the input function $f : \mathcal{D} \mapsto \mathcal{R}$ via an oracle. With very few exceptions, all property testers studied in this model rely on the oracle to provide function values at all queried domain points. However, in many realistic situations, the oracle may be unable to reveal the function values at some domain points due to privacy concerns, or when some of the values get erased by mistake or by an adversary. The testers do not learn anything useful about the property by querying those *erased* points. Moreover, the knowledge of a tester may enable an adversary to erase some of the values so as to increase the query complexity of the tester arbitrarily or, in some cases, make the tester entirely useless.

In this work, we initiate a study of property testers that are resilient to the presence of *adversarially erased* function values. An α-erasure-resilient ε-tester is given parameters $\alpha, \varepsilon \in (0,1)$, along with oracle access to a function f such that at most an α fraction of function values have been erased. The tester does not know whether a value is erased until it queries the corresponding domain point. The tester has to accept with high probability if there is a way to assign values to the erased points such that the resulting function satisfies the desired property $\mathcal{P}$. It has to reject with high probability if, for every assignment of values to the erased points, the resulting function has to be changed in at least an ε-fraction of the non-erased domain points to satisfy $\mathcal{P}$.

We design erasure-resilient property testers for a large class of properties. For some properties, it is possible to obtain erasure-resilient testers by simply using standard testers as a black box. However, there are more challenging properties for which all known testers rely on querying a specific point. If this point is erased, all these testers break. We give efficient erasure-resilient testers for several important classes of such properties of functions including monotonicity, the Lipschitz property, and convexity. Finally, we show a separation between the standard testing and erasure-resilient testing. Specifically, we describe a property that can be ε-tested with $O(\frac{1}{\varepsilon})$ queries in the standard model, whereas testing it in the erasure-resilient model requires number of queries polynomial in the input size.

1998 ACM Subject Classification F.2.2

Keywords and phrases Randomized algorithms, property testing, error correction, monotone and Lipschitz functions

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

* This work was supported in part by NSF award CCF-1320814 , NSF CAREER award CCF-0845701, NSF award CCF-1422975, Pennsylvania State University College of Engineering Fellowship and Pennsylvania State University Graduate Fellowship.

1 Introduction

In this paper, we revisit the question of how sublinear-time algorithms access their input. With very few exceptions, all algorithms studied in the literature on sublinear-time algorithms have *oracle* access to their input[1]. However, in many applications, this assumption is unrealistic. The oracle may be unable to reveal parts of the data due to privacy concerns, or when some of the values get erased by mistake or by an adversary. Motivated by these scenarios, we propose to study sublinear algorithms that work with partially erased data.

Formally, we view a dataset as a function over some discrete domain $\mathcal{D}$, such as $[n] = \{1, \dots, n\}$ or $[n]^d$. For example, the classical problem of testing whether a list of n numbers is sorted in nondecreasing order can be viewed as a problem of testing whether a function $f : [n] \to \mathbb{R}$ is monotone (nondecreasing). Given a parameter $\alpha \in (0, 1)$, we say that a function is α*-erased* if at most an α fraction of its domain points are marked as "erased" or protected (that is, an algorithm is denied access to these values). An algorithm that takes an α-erased function as its input does not know which values are erased until it queries the corresponding domain points. For each queried point x, the algorithm either learns $f(x)$ or, if x is an erased point, gets back a special symbol $\perp$. We study algorithms that work in the presence of *adversarial erasures*. In other words, the query complexity of an algorithm is the number of queries it makes in the *worst case* over all α-erased input functions.

In this work, we initiate a systematic study of property testers that are resilient to the presence of adversarial erasures. An α-erasure-resilient ε-tester is given parameters $\alpha, \varepsilon \in (0, 1)$, along with oracle access to an α-erased function f. The tester has to accept with high probability if f can be restored to a function on the whole domain that satisfies the desired property $\mathcal{P}$ and reject with high probability if every restoration of f is ε-far from $\mathcal{P}$ on the nonerased part of the domain.

Generic transformations. Our first goal is to understand which existing algorithms in the standard property testing model [35, 24] can be easily made erasure-resilient. We show (in Section 2) how to obtain erasure-resilient testers for some properties by using standard testers for these properties as black box. Our transformations apply to testers that query uniformly and independently sampled points, with some additional restrictions. More specifically, our transformations work for uniform *proximity oblivious testers* (POTs) [25, 27] and uniform testers for *extendable properties*. As a result, we are able to obtain erasure-resilient testers for being a low-degree polynomial [35], monotonicity over general poset domains [22], convexity of black and white images [7], and having k runs of 0s and 1s.

Erasure-resilient testers for more challenging properties. One challenge in designing erasure-resilient testers by using existing algorithms in the standard model as a starting point is that many existing algorithms are more likely to query certain points in the domain. Therefore, if these points are erased, the algorithms break. Specifically, the optimal algorithms for testing whether a list of numbers is sorted (and there are at least three different algorithms for this problem [17, 8, 13]) have this feature. Moreover, it is known that an algorithm that makes uniformly random queries is far from optimal: it needs $\Theta(\sqrt{n})$ queries instead of $\Theta(\log n)$ for n-element lists [17, 20].

[1] Sublinear-time algorithms with various distributional assumptions on the positions of the input the algorithms access have been investigated, for example, in [24, 4, 26]. There is also a line of work, initiated by [6], that studies sublinear algorithms that access distributions, as opposed to fixed datasets. In this work, we focus on fixed datasets.

There is a number of well studied properties for which all known optimal algorithms heavily rely on querying specific points. Most prominent examples include monotonicity, the Lipschitz properties and, more generally, bounded-derivative properties of real-valued functions on $[n]$ and $[n]^d$, as well as convexity of real-valued functions on $[n]$. It is especially challenging to deal with real-valued functions in our model, because there are many possibilities for erased values. We give efficient erasure-resilient testers for all aforementioned properties of real-valued functions in Sections 3-5.

Relationships with other models. In the full version of our paper, we provide a separation between our erasure-resilient model and the standard model. Specifically, we prove the existence of a property that can be tested with $O(1/\varepsilon)$ queries in the standard model, but requires polynomially many queries in the length of the input in the erasure-resilient model. This result builds on the ideas of Fischer and Fortnow [21] that separate tolerant testing, defined by Parnas et al. [32], from standard testing.

A tolerant tester for a property $\mathcal{P}$, given two parameters $\varepsilon_1, \varepsilon_2 \in (0,1)$, where $\varepsilon_1 < \varepsilon_2$, is required to, with probability at least 2/3, accept inputs that are ε_1-close to $\mathcal{P}$ and reject inputs that are ε_2-far from $\mathcal{P}$. Intuitively, the relationship of our erasure-resilient model to tolerant testing is akin to the relationship between error-correcting codes that withstand erasures and error-correcting codes that withstand general errors. In the full version, we prove that the existence of tolerant testers implies the existence of erasure-resilient testers with related parameters. Using this implication and existing tolerant testers for sortedness [36], monotonicity [19], and convexity [18], we get erasure-resilient testers for these properties as corollaries. However, we obtain erasure-resilient testers for these properties with much better parameters in the technical sections of this article. We conjecture that erasure-resilient testing can be separated from tolerant testing in the same strong sense as in our separation of standard testing from erasure-resilient testing.

1.1 The Erasure-Resilient Testing Model

We formalize our erasure-resilient model for the case of property testing. Erasure-resilient versions of other computational models, such as tolerant testing, can be defined analogously.

▶ **Definition 1.1** (α-erased function)**.** Let $\mathcal{D}$ be a domain, $\mathcal{R}$ be a range, and $\alpha \in (0,1)$. A function[2] $f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$ is *α-erased* if f evaluates to $\perp$ on at most an α fraction of domain points. The points on which f evaluates to $\perp$ are called *erased*. The set of remaining (nonerased) points is denoted by $\mathcal{N}$.

A function f is ε-far from a property (set) $\mathcal{P}$ if it needs to be changed on at least an ε fraction of domain points to obtain a function in $\mathcal{P}$. A function $f' : \mathcal{D} \to \mathcal{R}$ that differs from a function f only on points erased in f is called a *restoration* of f.

▶ **Definition 1.2** (Erasure-resilient tester)**.** An *α-erasure-resilient ε-tester* of property $\mathcal{P}$ gets input parameters $\alpha, \varepsilon \in (0,1)$ and oracle access to an α-erased function $f : \mathcal{D} \to \mathcal{R} \cup \{\perp\}$. It outputs, with probability[3] at least 2/3,

[2] Any object can be viewed as a function. E.g., an n-element array of real numbers can be viewed as a function $f : [n] \to \mathbb{R}$, an image—as a map from the plane to the set of colors, and a graph—as a map from the set of vertex pairs to $\{0,1\}$.

[3] In general, the error probability can be any $\delta \in (0,1)$. For simplicity, we formulate our model and the results with $\delta = 1/3$. To get results for general δ, by standard arguments, it is enough to multiply the complexity of an algorithm by $\log 1/\delta$.

- **accept** if there is a restoration $f' : \mathcal{D} \to \mathcal{R}$ of f that satisfies $\mathcal{P}$;
- **reject** if every restoration $f' : \mathcal{D} \to \mathcal{R}$ of f needs to be changed on at least an ε fraction of $\mathcal{N}$, the nonerased portion of f's domain, to satisfy $\mathcal{P}$ (that is, f' is $\varepsilon \cdot \frac{|\mathcal{N}|}{|\mathcal{D}|}$-far from $\mathcal{P}$).

The tester has *1-sided error* if the first item holds with probability 1.

Let $f_{|\mathcal{N}}$ denote the function f restricted to the set $\mathcal{N}$ of nonerased points. In the full version, we show that if property $\mathcal{P}$ is extendable, we can define a property $\mathcal{P}_{\mathcal{N}}$ such that the erasure-resilient tester is simply required to distinguish the case that $f_{|\mathcal{N}}$ satisfies $\mathcal{P}_{\mathcal{N}}$ from the case that it is ε-far from satisfying it. For example, if $\mathcal{P}$ is monotonicity of functions on a partially-ordered domain $\mathcal{D}$ then $\mathcal{P}_{\mathcal{N}}$ is monotonicity of functions on $\mathcal{N}$. (Most of the properties we consider in this article, including monotonicity, Lipschitz properties and convexity, are extendable properties.) Note that, even for the case of extendable properties, our problem is different from the standard property testing problem because the tester does not know in advance which points are erased.

1.2 Properties We Consider

Next we define properties of real-valued functions considered in this article and summarize previous work on testing them. Most properties of real-valued functions studied in the property testing framework are for functions over the *line* domain $[n]$ and, more generally, the *hypergrid* domain $[n]^d$.

▶ **Definition 1.3** (Hypergrid, line). Given $n, d \in \mathbb{N}$, the hypergrid of size n and dimension d is the set $[n]^d$ associated with an order relation $\preceq$, such that $x \preceq y$ for all $x, y \in [n]^d$ iff $x_i \leq y_i$ for all $i \in [d]$, where x_i (respectively y_i) denotes the i^{th} coordinate of x (respectively, y). The special case $[n]$ is called a *line*.

We consider domains that are subsets of $[n]^d$ to be able to handle arbitrary erasures on $[n]^d$.

Monotonicity. Monotonicity of functions, first studied in the context of property testing in [23], is one of the most widely investigated properties in this model [17, 16, 31, 22, 1, 20, 28, 5, 32, 2, 8, 11, 9, 13, 14, 10, 12]. A function $f : \mathcal{D} \mapsto \mathbb{R}$, defined on a partially ordered domain $\mathcal{D}$ with order $\preceq$, is monotone if $x \preceq y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{D}$. The query complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [17, 20]; for functions $f : [n]^d \mapsto \mathbb{R}$, it is $\Theta(d \log n/\varepsilon)$ [13, 14], and for functions over arbitrary partially ordered domains $\mathcal{D}$, it is $O\left(\sqrt{|\mathcal{D}|/\varepsilon}\right)$ [22].

Lipschitz properties. Lipschitz continuity is defined for functions between arbitrary metric spaces, but was specifically studied for real-valued functions on hypergrid domains [29, 3, 13, 15, 10, 12] because of applications to privacy [29, 15]. For $\mathcal{D} \subseteq [n]^d$ and $c \in \mathbb{R}$, a function $f : \mathcal{D} \mapsto \mathbb{R}$ is c-Lipschitz if $|f(x) - f(y)| \leq c \cdot ||x - y||_1$ for all $x, y \in \mathcal{D}$, where $||x - y||_1$ is the L_1 distance between x and y. More generally, f is (α, β)-Lipschitz, where $\alpha < \beta$, if $\alpha \cdot ||x - y||_1 \leq |f(x) - f(y)| \leq \beta \cdot ||x - y||_1$ for all $x, y \in [n]^d$. All (α, β)-Lipschitz properties can be tested with $O(d \log n/\varepsilon)$ queries [13].

Bounded derivative properties. The class of bounded derivative properties (BDPs), defined by Chakrabarty et al. [12], is a natural generalization of monotonicity and the (α, β)-Lipschitz properties. An ordered set $\mathbf{B}$ of $2d$ functions $l_1, u_1, l_2, u_2, \ldots, l_d, u_d : [n-1] \mapsto \mathbb{R} \cup \{\pm\infty\}$ is a *bounding family* if for all $r \in [d]$ and $y \in [n-1]$, $l_r(y) < u_r(y)$. Let $\mathbf{B}$ be a bounding family of functions and let $\mathbf{e}_r$ be the unit vector along dimension r. The property $\mathcal{P}(\mathbf{B})$

of being $\mathbf{B}$-*derivative bounded* is the set of functions $f : [n]^d \mapsto \mathbb{R}$ such that $l_r(x_r) \le f(x + \mathbf{e}_r) - f(x) \le u_r(x_r)$ for all $r \in [d]$ and $x \in [n]^d$ with $x_r \ne n$, where x_r is the r^{th} coordinate of x. Chakrabarty et al. [12] showed that the complexity of testing BDPs of functions $f : [n]^d \mapsto \mathbb{R}$ is $\Theta(d \log n/\varepsilon)$.

Convexity of functions. A function $f : \mathcal{D} \mapsto \mathbb{R}$ is convex if $f(t\mathbf{x} + (1-t)\mathbf{y}) \le tf(\mathbf{x}) + (1-t)f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ and $t \in [0,1]$. If $\mathcal{D} \subseteq [n]$, equivalently, f is convex if $\frac{f(y)-f(x)}{y-x} \le \frac{f(z)-f(y)}{z-y}$ for all $x < y < z$. Parnas et al. [33] gave a convexity tester for functions $f : [n] \mapsto \mathbb{R}$ with query complexity $O(\log n/\varepsilon)$. Blais et al. [10] gave an $\Omega(\log n)$ bound for nonadaptive testers for this problem.

1.3 Our Results

We give efficient erasure-resilient testers for all properties discussed in Section 1.2. All our testers have optimal complexity for the case with no erasures and have an additional benefit of not relying too heavily on the value of the input function at any specific point.

Monotonicity on the line. We start by giving (in Section 3) an erasure-resilient monotonicity tester on $[n]$.

▶ **Theorem 1.4** (Monotonicity tester on the line). *There exists a one-sided error α-erasure-resilient ε-tester for monotonicity of real-valued functions on the line $[n]$ that works for all $\alpha, \varepsilon \in (0,1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Without erasure resilience, the complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [17, 20]. Thus, the query complexity of our erasure-resilient tester has optimal dependence on the domain size and on ε.

The starting point of our algorithm is the tester for sortedness from [17]. This tester picks a random element of the input array and performs a binary search for that element. It rejects if the binary search does not lead to the right position. The first challenge is that the tester always queries the middle element of the array and is very likely to query other elements that are close to the root in the binary search tree. So, it will break if these elements are erased. To make it resilient to erasures, we randomize the binary tree with respect to which it performs the binary search. The second challenge is that the tester does not know which points are erased. To counteract that, our tester samples points from appropriate intervals until it encounters a nonerased point.

To analyze the tester, we bound the expected number of queries required to traverse a uniformly random search path in an arbitrary binary search tree built over the nonerased points in an α-erased n-element array (Claim 3.2). This expectation depends only on the depth of the tree and α. This is the most interesting part of our analysis and captures the intuition that a randomized binary search for a uniformly random search point is biased towards visiting intervals containing a larger fraction of nonerased points.

BDPs on the hypergrid. In Section 4, we generalize our monotonicity tester in two ways: (1) to work over general hypergrid domains, and (2) to apply to all BDPs. We achieve it by giving (1) a reduction from testing BDPs on the line to testing monotonicity on the line that applies to erasure-resilient testers and (2) an erasure-resilient version of the dimension reduction from [12]. We obtain the following result.

▶ **Theorem 1.5** (BDP tester on the hypergrid). *For every BDP $\mathcal{P}$, there exists a one-sided error α-erasure-resilient ε-tester for $\mathcal{P}$ of real-valued functions on the hypergrid $[n]^d$ that works for all $\alpha, \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/970d$, with query complexity $O\left(\frac{d \log n}{\varepsilon(1-\alpha)}\right)$.*

Every known tester of a BDP for real-valued functions over general hypergrid domains work by sampling an *axis-parallel line* uniformly at random and checking for violations on the sampled line. Our erasure-resilient testers also follow this paradigm. To check for violations on the sampled line, we use one iteration of our BDP tester for the line. In the full version, we show the existence of α-erased functions $f : \{0,1\}^d \mapsto \mathbb{R}$ that are ε-far from monotone for $\alpha = \Theta(\varepsilon/\sqrt{d})$ but do not have violations to monotonicity along any of the axis parallel lines (which are the edges of the hypercube, in this case). It implies that every tester for monotonicity that follows the paradigm above will fail when $\alpha = \Omega(\varepsilon/\sqrt{d})$. Thus, some restriction on α in terms of d and ε is necessary for such testers.

Convexity on the line. Finally, in Section 5, we develop additional techniques to design a tester for convexity (which is not a BDP) on the line. The query complexity of our tester has the same dependence on n and ε as in the standard convexity tester of Parnas et al. [33]. The dependence on n is known to be optimal for nonadaptive testers [10], and the tester from [33] is conjectured to be optimal in the standard model.

▶ **Theorem 1.6** (Convexity tester on the line). *There exists a one-sided error α-erasure-resilient ε-tester for convexity of real-valued functions on the line $[n]$ that works for all $\alpha, \varepsilon \in (0,1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Our algorithm for testing convexity combines ideas on testing convexity from [33], testing sortedness from [17], and our idea of randomizing the search. The tester of [33] traverses a uniformly random path in a binary tree on the array $[n]$ by selecting one of the half-intervals of an interval uniformly at random at each step. Instead of doing this, our tester samples a uniformly random nonerased search point and traverses the path to that point in a uniformly random binary search tree just as in our modification of the tester of [17]. This is done to bias our algorithm to traverse paths containing intervals that have a larger fraction of nonerased points. However, instead of checking whether the selected point can be found, as in our monotonicity tester, the convexity tester checks a more complicated "goodness condition" in each visited interval of the binary search tree. It boils down to checking that the slope of the functions between pairs of carefully selected points satisfies the convexity condition. In addition to spending queries on erased points due to sampling, like in the monotonicity tester, our tester also performs "walking queries" to find the nearest nonerased points for the pivots in our random binary search tree. We show that the overhead in the query complexity due to querying erased points is at most a factor of $O(1/(1-\alpha))$.

2 Generic transformations

In this section, we state our transformation theorems that can be applied to some classes of testers that use only uniform samples. We show how to make two classes of testers erasure-resilient: (1) uniform proximity oblivious testers (POTs) defined in [25, 27] (Theorem 2.2), and (2) uniform testers for extendable properties (Theorem 2.4). All omitted details appear in the full version. We first define POTs.

▶ **Definition 2.1** (POT [25, 27]). Let $\mathcal{P}$ be a property of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$. Let $\rho : (0,1] \mapsto (0,1]$ be a monotone function and $c \in (0,1]$ be a constant. A tester T is a

(ρ, c)*-POT* for $\mathcal{P}$ if it decides to accept or reject based only on the answers to the queries that it makes, such that

- for every function $f \in \mathcal{P}$, the probability that T accepts is at least c, and
- for every function $f \notin \mathcal{P}$, the probability that T accepts is at most $c - \rho(\varepsilon_f)$, where ε_f denotes the relative Hamming distance of f from $\mathcal{P}$.

A POT that queries points sampled uniformly and independently at random from $\mathcal{D}$ is called a uniform POT. Next, we state our first generic transformation.

▶ **Theorem 2.2.** *If T is a uniform (ρ,c)-POT for a property $\mathcal{P}$ of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$ making q queries, then there exists a uniform α-erasure-resilient (ρ',c)-POT T' for $\mathcal{P}$ that makes q queries for all $\alpha < \rho(\varepsilon_f \cdot (1-\alpha))/q$, where $\rho'(x) = \rho(x \cdot (1-\alpha)) - \alpha \cdot q$ for $x \in (0, 1]$.*

In the full version, we apply Theorem 2.2 to a tester for the property of being a polynomial of degree at most d due to Rubinfeld and Sudan [35] and get an erasure-resilient tester.

Next, we define extendable properties and state our second transformation. Given $\mathcal{S} \subseteq \mathcal{T}$, the *extension* of a function $f : \mathcal{S} \mapsto \mathcal{R}$ to $\mathcal{T}$ is a function $g : \mathcal{T} \mapsto \mathcal{R}$ that agrees with f on every point in $\mathcal{S}$.

▶ **Definition 2.3** (Extendable properties)**.** For a domain $\mathcal{D}$ and all $\mathcal{S} \subseteq \mathcal{D}$, let $\mathcal{P}_\mathcal{S}$ denote a property of functions $f : \mathcal{S} \mapsto \mathcal{R}$. The class of properties $\{\mathcal{P}_\mathcal{S} : \mathcal{S} \subseteq \mathcal{D}\}$ is extendable if, for all $\mathcal{S}, \mathcal{T} : \mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{D}$,

- for every function $f : \mathcal{S} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{S}$, there is an extension $f' : \mathcal{T} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{T}$, and
- for every function $f : \mathcal{S} \mapsto \mathcal{R}$ that is ε-far from $\mathcal{P}_\mathcal{S}$, every function $f' : \mathcal{T} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{T}$ differs from f on at least an ε fraction of points in $\mathcal{S}$.

▶ **Theorem 2.4.** *Let $q(\cdot, \cdot)$ be a function that is monotone non-decreasing in the first argument and monotone non-increasing in the second argument. Let $\{\mathcal{P}_\mathcal{S} : \mathcal{S} \subseteq \mathcal{D}\}$ be a class of extendable properties, where $\mathcal{P}_\mathcal{S}$ is a property of functions of the form $f : \mathcal{S} \mapsto \mathcal{R}$. Suppose $T_\mathcal{S}$ is a uniform one-sided error ε-tester of $\mathcal{P}_\mathcal{S}$ that makes $q(|\mathcal{S}|, \varepsilon)$ queries from $\mathcal{S}$, for every $\mathcal{S} \subseteq \mathcal{D}$. Assume also that for each $\mathcal{S} \subseteq \mathcal{D}$, the probability that $T_\mathcal{S}$ tests correctly does not decrease when it makes more queries. Then, there exists a uniform one-sided error α-erasure-resilient ε-tester T' of $\mathcal{P}_\mathcal{D}$ that makes $O\left(q(|\mathcal{D}|, \varepsilon)/(1-\alpha)\right)$ queries for all $\alpha \in [0, 1)$.*

Using Theorem 2.4, we can make the uniform tester for convexity of black and white images [7] and the uniform tester for monotonicity of real-valued functions over arbitrary partially ordered domains [22] erasure-resilient.

We also develop a uniform tester for the property of being a k-run Boolean function and make it erasure-resilient by applying Theorem 2.4. A function $f : [n] \mapsto \{0, 1\}$ has k *runs* if the list $f(1), f(2), \ldots, f(n)$ has at most $k-1$ alternations of values. The problem at hand is to test whether a given function $f : [n] \mapsto \{0, 1\}$ has k runs or is ε-far from this property. Kearns and Ron [30] studied a relaxation of this problem. Specifically, they showed that $O(1/\varepsilon^2)$ queries suffice to test whether a Boolean function has k runs or is ε-far from being a k/ε-run function. They also developed a uniform $O(\sqrt{k}/\varepsilon^{2.5})$-query tester for this relaxation and proved that every uniform ε-tester for the k-run property requires $\Omega(\sqrt{k})$ queries. Balcan et al. [4] obtained a $O(1/\varepsilon^4)$-query tester for this property in the active testing model. They also developed a uniform $O(\sqrt{k}/\varepsilon^5)$-query tester. We prove the following theorem.

▶ **Theorem 2.5.** *There is an ε-tester for the property of having k runs for functions of the form $f : [n] \mapsto \{0, 1\}$ that makes $O\left(\frac{k \log k}{\varepsilon}\right)$ uniform independent queries, where $\varepsilon > k^2/n$.*

3 Erasure-Resilient Monotonicity Tester for the Line

In this section, we prove Theorem 1.4. Recall that, for a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, the set of nonerased points (the ones that map to $\mathbb{R}$) is denoted by $\mathcal{N}$. The function f is monotone if $x < y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{N}$. The tester does not know $\mathcal{N}$ in advance.

We present our tester in Algorithm 1. It has oracle access to f and takes α and ε as inputs. In each iteration, it performs a randomized binary search for a nonerased index sampled uniformly at random (u.a.r.) from $\mathcal{N}$ and rejects if it finds violations to monotonicity. In the description of our tester, we use $I[i, j]$ to denote the set of natural numbers from i until and including j. We alternatively refer to it as the interval from i to j.

Algorithm 1 Erasure-Resilient Monotonicity Tester for the Line

```
1:  Set $Q = \left\lceil \frac{60 \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2:  Accept at any point if the number of queries exceeds $Q$.
3:  loop $2/\varepsilon$ times:
4:      Sample points u.a.r. from $I[1, n]$ and query them until we get a point $s \in \mathcal{N}$.
5:      Set $\ell \leftarrow 1$, $r \leftarrow n$.
6:      while $\ell \leq r$ do
7:          Sample points u.a.r. from $I[\ell, r]$ and query them until we get a point $m \in \mathcal{N}$.
8:          if $s < m$ then set $r \leftarrow m - 1$ and Reject if $f(s) \geq f(m)$.
9:          if $s > m$ then set $\ell \leftarrow m + 1$ and Reject if $f(s) \leq f(m)$.
10:         if $s = m$ then Go to Step 3.                    ▷ Search completed.
11: Accept.
```

Every iteration of Algorithm 1 can be viewed as a traversal of a uniformly random search path in a uniformly random binary search tree defined on the set $\mathcal{N}$ of nonerased points. Given a binary search tree T over $\mathcal{N}$, we associate every node of T with a unique sub-interval I of $I[1, n]$ as follows. The root of T is associated with $I[1, n]$. Suppose the interval associated with a node Γ in T that contains $s \in \mathcal{N}$ is $I[i, j]$. Then the interval associated with the left child of Γ is $I[i, s - 1]$ and the interval associated with the right child of Γ is $I[s + 1, j]$. A search path is a path from the root to some node Γ of T.

If f is ε-far from monotone, we prove that, with high probability, the tester finds a violation. It is easy to prove this, using a generalization of an argument from [17], for the case when Algorithm 1 manages to complete all iterations of Step 3 before it runs out of queries. The challenge is that the algorithm might get stuck pursuing long paths in a random search tree and waste many queries on erased points. To resolve the issue of many possible queries to erased points, we prove an upper bound on the expected number of queries made while traversing a uniformly random search path in a binary search tree on $\mathcal{N}$. We combine this with the fact that the expected depth of a random binary search tree is $O(\log n)$ to obtain the final bound on the probability that the algorithm exceeds its query budget.

3.1 Analysis

We analyze the tester in this section. The query complexity of the tester is clear from its description. The main statement of Theorem 1.4 follows from Lemma 3.1, proved next.

▶ **Lemma 3.1.** *Algorithm 1 accepts if f is monotone, and rejects with probability at least $2/3$ if f is ε-far from monotone.*

Proof. The tester accepts whenever f is monotone. To prove the other part of the lemma, assume that f is ε-far from monotone. Let A be the event that the tester accepts f. Let q denote the total number of queries made. We prove that $\Pr[A] \leq 1/3$. The event A occurs if either $q > Q$ or the tester does not find a violation in any of the $2/\varepsilon$ iterations of Step 3. Thus, $\Pr[A] \leq \Pr[A|q \leq Q] + \Pr[q > Q]$.

First we bound the probability that the tester does not find a violation in one iteration of Step 3, conditioned on the event that $q \leq Q$. Consider an arbitrary binary search tree T defined over points in $\mathcal{N}$. A point $s \in \mathcal{N}$ is called *searchable* with respect to T if Algorithm 1 does not detect a violation to monotonicity while traversing the search path to s in T. Consider two indices $i, j \in \mathcal{N}$, where $i < j$, both searchable with respect to T. Let $a \in \mathcal{N}$ be the pivot corresponding to the lowest common ancestor of the leaves containing i and j. Since i and j are both *searchable*, it must be the case that $f(i) < f(a)$ and $f(a) < f(j)$ and hence, $f(i) < f(j)$. Thus, for every tree T, the function restricted to the domain points that are *searchable* with respect to T is monotone. Therefore, if f is ε-far from monotone, for every binary search tree T, at least an ε-fraction of the points in $\mathcal{N}$ are not *searchable*. Thus, the tester detects a violation with probability ε in each iteration. Consequently, $\Pr[A|q \leq Q] \leq (1-\varepsilon)^{\frac{2}{\varepsilon}} < 1/4$.

In the rest of the proof, we will bound $\Pr[q > Q]$. We will first state and prove a claim that bounds the expected number of queries to traverse a search path, for every binary search tree. Recall that a search path in a search tree T is a path from the root to some node in T. Let I be an interval associated with a node v of T and let α_I denote the fraction of erased points in I. The number of queries to be made to sample a nonerased point from I with uniform sampling is a geometric random variable with expectation $1/(1-\alpha_I)$. We define the *query-weight* of node v to be this expectation. The query-weight of a search path is the sum of query-weights of the nodes on the path (which is the expected number of queries that the algorithm makes while traversing that path).

▶ **Claim 3.2.** *Consider an arbitrary binary search tree T on $\mathcal{N}$ of height h. The expected query-weight of a uniformly random search path in T is at most $h/(1-\alpha)$.*

Proof. There are exactly $|\mathcal{N}|$ search paths in T. Let S denote the sum of query-weights of all the search paths. The expected query-weight is equal to $S/|\mathcal{N}|$.

Consider a node v in T associated with an interval I. There are $|I|(1-\alpha_I)$ nonerased points in I. The search paths from the root of T to all these nonerased points pass through v, and hence, the query-weight of v gets added to the query-weights of all of those paths. Therefore, the total contribution of v towards S is $|I|$, since the query-weight of v is $1/(1-\alpha_I)$. Note that the intervals associated with nodes at the same level of T are disjoint from each other. Therefore, the total contribution to S from all nodes on the same level of T is at most n. Hence the value of S is at most $n \cdot h$. Observe that this quantity is independent of the fraction of erasures α. Therefore, the expected query-weight of a search path is at most $n \cdot h/|\mathcal{N}|$, which is at most $h/(1-\alpha)$, since $|\mathcal{N}| \geq n \cdot (1-\alpha)$. ◀

We will next see a fact on the expected depth of a uniformly random binary search tree and combine it with the above claim to prove the required bound on the expected query-weight of a uniformly random search path in a uniformly random binary search tree.

▶ **Claim 3.3** ([34]). *If H_n is the random variable denoting the height of a random binary search tree on n nodes, then $\mathbf{E}[H_n] \leq 5 \log n$.*

▶ **Corollary 3.4.** *The expected number of queries made by Algorithm 1 to traverse a uniformly random search path in a uniformly random binary search tree on $\mathcal{N}$ is at most $5 \log n/(1-\alpha)$.*

By linearity of expectation, the expected number of queries made by the tester over all its iterations is at most $10\log n/(\varepsilon \cdot (1-\alpha))$. Applying Markov's inequality to q, we can then see that $\Pr[q > Q] \leq 1/6$. Therefore, the probability of the tester not finding a violation is at most $1/3$. This completes the proof of the lemma. ◀

4 Erasure-Resilient Monotonicity Testers for the Hypergrid

In this section, we present our erasure-resilient tester for monotonicity over hypergrid domains and prove the following theorem, which is a special case of Theorem 1.5. We defer the discussion of erasure-resilient testers for all BDPs to the full version.

▶ **Theorem 4.1.** *There exists a one-sided error α-erasure-resilient ε-tester for monotonicity of real-valued functions on the hypergrid $[n]^d$ that works for all $\alpha, \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/250d$, with query complexity $O(\frac{d\log n}{\varepsilon(1-\alpha)})$.*

Let $\mathcal{L}$ denote the set of all *axis-parallel lines* in the hypergrid. Our monotonicity tester, which is described in Algorithm 2, samples an *axis-parallel line* uniformly at random in each iteration and does a randomized binary search for a uniformly randomly sampled nonerased point on that line. It rejects if and only if a violation to monotonicity is found within its query budget. To analyze the tester, we first state two important properties of a uniformly random axis-parallel line in Lemma 4.2 and Lemma 4.3, which we jointly call the erasure-resilient dimension reduction. The statements and proofs of more general versions of these lemmas, applicable to all BDPs, are given in the full version.

▶ **Lemma 4.2** (Dimension reduction: distance)**.** *Let ε_f be the relative Hamming distance of an α-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$ from monotonicity. Given an axis-parallel line $\ell \in \mathcal{L}$, let $f_\ell : [n] \mapsto \mathbb{R} \cup \{\perp\}$ denote the restriction of f to ℓ and let ε_ℓ denote the relative Hamming distance of f_ℓ from monotonicity. Then $\mathbf{E}_{\ell\sim\mathcal{L}}[\varepsilon_\ell] \geq (((1-\alpha)\cdot\varepsilon_f)/4d) - \alpha$.*

▶ **Lemma 4.3** (Dimension reduction: fraction of erasures)**.** *Consider an α-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$. Given $\ell \in \mathcal{L}$, let α_ℓ denote the fraction of erased points in ℓ. Then, for every $\eta \in (0,1)$, we have, $\Pr_{\ell\sim\mathcal{L}}[\alpha_\ell > \alpha/\eta] \leq \eta$.*

Algorithm 2 Erasure-Resilient Monotonicity Tester for $[n]^d$

```
Require: parameters $\varepsilon \in (0,1), \alpha \in [0, \varepsilon/250d]$; oracle access to $f : [n]^d \to \mathbb{R}$
1: Set $Q = \lceil \frac{1200d\cdot\log n}{\varepsilon(1-\alpha)} \rceil$.
2: loop $\frac{12d}{\varepsilon(1-\alpha)-4d\alpha}$ times:
3:     Sample a line $\ell \in \mathcal{L}$ uniformly at random.
4:     Sample and query points u.a.r. from $\ell$ and query them until we get a point $s \in \mathcal{N}$.
5:     Perform a randomized binary search for $s$ on $\ell$ as in Algorithm 1.
6:     Reject if any violation to monotonicity is found.
7: Accept at any point if the number of queries exceed $Q$.
```

The query complexity of the tester is evident from its description. We will now prove its correctness in the following lemma, which will then imply Theorem 4.1.

▶ **Lemma 4.4.** *Algorithm 2 accepts if f is monotone, and rejects with probability at least $2/3$ if f is ε-far from monotone.*

Proof. The tester accepts if f is monotone. So, assume f is ε-far from being monotone. Let A denote the event that no iteration of the tester finds a violation to monotonicity. If q denotes the total number of queries made by the tester, we have, $\Pr[A] \le \Pr[A|q \le Q] + \Pr[q > Q]$.

Let t denote the number of iterations of the tester. Let A_i denote the event that the tester does not find a violation in its i-th iteration. For $\ell \in \mathcal{L}$, let f_ℓ denote f restricted to the line ℓ. Let ε_ℓ denote the relative Hamming distance of f_ℓ from monotonicity. We have, $\Pr[A_i|q \le Q] = \sum_{\ell\in\mathcal{L}}(1-\varepsilon_\ell)\Pr[\ell] = 1 - \mathbf{E}_{\ell\sim\mathcal{L}}[\varepsilon_\ell]$. By Lemma 4.2 and the fact that $\varepsilon_f \ge \varepsilon$, we have, $\mathbf{E}_{\ell\sim\mathcal{L}}[\varepsilon_\ell] \ge \frac{(1-\alpha)\cdot\varepsilon_f}{4d} - \alpha \ge \frac{(1-\alpha)\cdot\varepsilon}{4d} - \alpha$. Therefore,

$$\Pr[A|q \le Q] = \prod_{i=1}^{t}\Pr[A_i|q \le Q] \le \left(1 - \frac{(1-\alpha)\cdot\varepsilon - 4d\alpha}{4d}\right)^t < \frac{1}{10}.$$

It remains to bound $\Pr[q > Q]$. Let $\eta = 1/10t$. Let α_i be the fraction of erasures in the line sampled during iteration i and let q_i be the number of queries the tester makes in iteration i. Let G be the (good) event that $\alpha_i \le \alpha/\eta$ for all iterations $i \in [t]$. By Corollary 3.4, $\mathbf{E}[q_i|G] \le 5\eta \cdot \log n/(\eta - \alpha)$. By the linearity of expectation, $\mathbf{E}[q|G] \le \log n/(2(\eta-\alpha)) \le 120d\log n/(\varepsilon(1-\alpha))$, where the last inequality follows from our assumption that $\alpha \le \varepsilon/250d$. Using Markov's inequality, $\Pr[q > Q|G] \le 1/10$. Also, by combining Lemma 4.3 with a union bound, we can see that $\Pr[\overline{G}] \le 1/10$. Therefore, $\Pr[q > Q] \le \Pr[q > Q|G] + \Pr[\overline{G}] \le 1/5$. ◀

5 Erasure-Resilient Convexity Tester for the Line

In this section, we prove Theorem 1.6. Given an α-erased function $f : [n] \mapsto \mathbb{R}\cup\{\perp\}$, let ν_i be the i-th nonerased domain point in $[n]$. The derivative of f at a point $\nu_i \in \mathcal{N}$, denoted by $\Delta f(\nu_i)$, is $\frac{f(\nu_{i+1})-f(\nu_i)}{\nu_{i+1}-\nu_i}$, whenever $\nu_{i+1} \le n$. The function f is convex iff $\Delta f(\nu_i) \le \Delta f(\nu_{i+1})$ for all $i \in [|\mathcal{N}|-2]$. Our tester builds upon the ideas in the convexity tester from [33].

A high level idea of the tester is as follows. Our tester (Algorithm 3) has several iterations. Every iteration of the tester can be thought of as a traversal of a uniformly random *search* path of a uniformly random binary search tree on $\mathcal{N}$, just as Algorithm 1. For each interval on such a path, we check a set of conditions computed based on the values at some nonerased points in the interval, called *anchor points*, and two real numbers, called the left and right slopes. More specifically, we verify that the function restricted to the sampled nonerased points in the interval is convex, by comparing the slopes across consecutive points. The algorithm accepts if all the intervals it sees pass these checks. The main steps in the analysis

Algorithm 3 Erasure-Resilient Convexity Tester

```
Require: parameters $\varepsilon, \alpha \in (0,1)$; oracle access to $f : [n] \mapsto \mathbb{R}\cup\{\perp\}$.
1: Set $Q = \lceil \frac{180\log n}{\varepsilon(1-\alpha)} \rceil$.
2: Accept at any point if the number of queries exceeds $Q$.
3: loop $2/\varepsilon$ times
4:     Sample points in $I[1,n]$ u.a.r and query them until we get a point $s \in \mathcal{N}$.
5:     Test-Interval($I[1,n], \emptyset, -\infty, +\infty, s$) and Reject if it rejects.
6: Accept.
```

of the tester follows that of the analysis of Algorithm 1. To prove that, with high probability, the algorithm does not run out of its budget of queries Q, we classify the queries that the tester makes into two kinds and analyze them separately. The queries where the tester repeatedly samples and queries from an interval until it finds a nonerased domain point are called *sampling queries.* The queries where the tester keeps querying consecutive points, starting from a nonerased point, until it gets the next nonerased point are called *walking queries.* We show that the expected number of walking queries is at most twice the number of the expected number of the sampling queries and use Corollary 3.4 to bound the expected number of sampling queries. We then prove that, conditioned on the tester not running out of its queries, in every iteration, with probability ε, the tester will detect a violation while testing for a function that is ε-far from being convex. This part draws ideas from the proof of correctness of the tester in [33]. The analysis of the tester is deferred to the full version.

Procedure 4 TEST-INTERVAL($I[i,j], \mathcal{A} = \{a_1, a_2, \dots, a_k\}, m_\ell, m_r, s$)

```
Require: interval $I[i,j]$; a set of nonerased points $\mathcal{A}$; left slope $m_\ell \in \mathbb{R}$; right slope $m_r \in \mathbb{R}$;
    search point $s \in \mathcal{N}$.
 1: Sample points u.a.r. from $I[i,j]$ and query them until we get a point $x \in \mathcal{N}$.
 2: Sequentially query points $x+1, x+2 \dots$ until we get a nonerased point $y$.
 3: Sequentially query points $x-1, x-2 \dots$ until we get the nonerased point $z$.
 4: Let $(a_1, a_2, \dots, a_k)$ denote the sorted list of points in the set $\mathcal{A} \cup \{x, y, z\}$.
 5: Let $m_i = (f(a_{i+1}) - f(a_i))/(a_{i+1} - a_i)$ for all $i \in [k-1]$.
 6: Reject if $m_\ell \le m_1 \le m_2 \le \dots \le m_{k-1} \le m_r$ is not true.
 7: Let $\mathcal{A}'_\ell$ and $\mathcal{A}'_r$ be the sets of points in $\mathcal{A}$ that are smaller and larger than $x$, respectively.
 8: if $s < x$ then
 9:     Reject if TEST-INTERVAL($I[i,z], \mathcal{A}'_\ell, m_\ell, \Delta f(z), s$) rejects.
10: if $s > x$ then
11:     Reject if TEST-INTERVAL($I[y,j], \mathcal{A}'_r, \Delta f(x), m_r, s$) rejects.
12: Accept.
```

6 Conclusions and Open Problems

In this paper, we initiate a study of property testing in the presence of adversarial erasures. We design efficient erasure-resilient testers for several important properties such as monotonicity, the Lipschitz properties and convexity over different domains. All our testers for properties of functions on the line domain work for an arbitrary fraction of erasures. All our testers have only a small additional overhead of $O(1/(1-\alpha))$ in their query complexity in comparison to the query complexity of the currently best, and, in some cases, optimal, standard testers for the same properties. We also show that not all properties are easy to test in the erasure-resilient testing model by proving the existence of a property that is easy to test in the standard model but hard to test in the erasure-resilient model even for a small fraction of erasures. We now list some open problems.

- We show that tolerant testing is at least as hard as erasure-resilient testing. Determining if tolerant testing is strictly harder than erasure-resilient testing is an interesting direction.
- The fraction of erasures that our monotonicity tester for hypergrid domains ($[n]^d$) can tolerate decreases inversely with d. We also show that an inverse dependence on $\sqrt{d}$ is necessary for testers that work by sampling axis-parallel lines uniformly at random and then test for the property on them. It is an interesting combinatorial question to determine the exact tradeoff between the fraction of erasures and the fraction of axis parallel lines that are far from monotone.

Acknowledgements: We thank Jalaj Upadhyay for comments on a draft of this article.

References

1 N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Inform. and Comput.*, 204(11):1704–1717, 2006.

2 N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Structures Algorithms*, 31(3):371–383, 2007.

3 Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016.

4 Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 21–30, 2012.

5 T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. *Inform. and Comput.*, 196(1):42–56, 2005.

6 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *J. ACM*, 60(1):4, 2013.

7 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. *Testing Convexity of Figures Under the Uniform Distribution*, 2015. To appear in *SoCG 2016*.

8 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.

9 E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Comp. Complexity*, 21(2):311–358, 2012.

10 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 309–320, 2014.

11 J. Briët, S. Chakraborty, D. García-Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

12 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. *Property Testing on Product Distributions: Optimal Testers for Bounded Derivative Properties*, 2015. To appear in ACM Transactions on Algorithms.

13 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *STOC*, pages 419–428, 2013.

14 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. In *APPROX-RANDOM*, pages 425–435, 2013.

15 Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta. Testing the Lipschitz property over product distributions with applications to data privacy. In *TCC*, pages 418–436, 2013.

16 Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings, International Workshop on Randomization and Computation (RANDOM)*, 1999.

17 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

18 Shahar Fattal and Dana Ron. Approximating the distance to convexity. Unpublished manuscript.

19 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3), 2010.

20 E. Fischer. On the strength of comparisons in property testing. *Inform. and Comput.*, 189(1):107–116, 2004.

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

22 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 474–483, New York, NY, USA, 2002. ACM.

23 O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.

24 O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

25 Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM J. Comput.*, 40(2):534–566, 2011.

26 Oded Goldreich and Dana Ron. On sample-based testers. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 337–345, 2015.

27 Oded Goldreich and Igor Shinkar. Two-sided error proximity oblivious testing. *Random Struct. Algorithms*, 48(2):341–383, 2016.

28 S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. *Random Structures Algorithms*, 33(1):44–67, 2008.

29 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013.

30 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000.

31 E. Lehman and D. Ron. On disjoint chains of subsets. *J. Combin. Theory Ser. A*, 94(2):399–404, 2001.

32 M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 6(72):1012–1042, 2006.

33 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5):1158–1184, 2003.

34 Bruce A. Reed. The height of a random binary search tree. *J. ACM*, 50(3):306–332, 2003.

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

36 Michael E. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 458–467, 2010.