2022-01-31

Sublinear-time computation in the presence of online erasures

Sublinear-Time Computation in the Presence of Online Erasures

Iden Kalemaj
Department of Computer Science, Boston University, MA, USA

Sofya Raskhodnikova
Department of Computer Science, Boston University, MA, USA

Nithin Varma
Chennai Mathematical Institute, India

Abstract

We initiate the study of sublinear-time algorithms that access their input via an online adversarial erasure oracle. After answering each query to the input object, such an oracle can erase t input values. Our goal is to understand the complexity of basic computational tasks in extremely adversarial situations, where the algorithm's access to data is blocked during the execution of the algorithm in response to its actions. Specifically, we focus on property testing in the model with online erasures. We show that two fundamental properties of functions, linearity and quadraticity, can be tested for constant t with asymptotically the same complexity as in the standard property testing model. For linearity testing, we prove tight bounds in terms of t, showing that the query complexity is $\Theta(\log t)$. In contrast to linearity and quadraticity, some other properties, including sortedness and the Lipschitz property of sequences, cannot be tested at all, even for $t = 1$. Our investigation leads to a deeper understanding of the structure of violations of linearity and other widely studied properties.

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

Keywords and phrases Randomized algorithms, property testing, Fourier analysis, linear functions, quadratic functions, Lipschitz and monotone functions, sorted sequences

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.90

Related Version *Full Version*: `https://arxiv.org/abs/2109.08745` [39]

Funding *Iden Kalemaj*: This work was supported by NSF award CCF-1909612 and Boston University's Dean's Fellowship.
Sofya Raskhodnikova: This work was supported by NSF award CCF-1909612.
Nithin Varma: This work was done in part while the author was a postdoctoral researcher at the University of Haifa, Israel, where he was supported by the Israel Science Foundation, grant number 497/17 and the PBC Fellowship for Postdoctoral Fellows by the Israeli Council of Higher Education.

Acknowledgements We are thankful to Kobbi Nissim for suggesting investigating settings with online adversarial erasures. We thank Noga Ron-Zewi for pointing out relevant references and the anonymous ITCS reviewers for helpful comments.

1 Introduction

We initiate the study of sublinear-time algorithms that compute in the presence of an online adversary that blocks access to some data points in response to the algorithm's queries. A motivating scenario is when a user wishes to remove their data from a dataset due to privacy concerns, as enabled by right to be forgotten laws such as the EU General Data Protection Regulation [44]. The online aspect of our model suitably captures the case of individuals who are prompted to restrict access to their data after noticing an inquiry into their or others' data. We choose to model such user actions as adversarial in order to perform worst-case

analysis. Two other motivating scenarios are naturally adversarial. In one, an algorithm is trying to detect some fraud (e.g., tax fraud) and the adversary wants to obstruct access to data in order to make it hard to uncover any evidence. In the other scenario, an algorithm's goal is to determine an optimal course of action (e.g., whether to invest in a stock or to buy an item), whereas the adversary leads the algorithm astray by adaptively blocking access to pertinent information.

In our model, after answering each query to the input object, the adversary can hide a small number of input values. Our goal is to understand the complexity of basic computational tasks in extremely adversarial situations, where the algorithm's access to data is blocked during the execution of the algorithm in response to its actions. Specifically, we represent the input object as a function f on an arbitrary finite domain[1], which the algorithm can access by querying a point x from the domain and receiving the answer $\mathcal{O}(x)$ from an oracle. At the beginning of computation, $\mathcal{O}(x) = f(x)$ for all points x in the domain of the function. We parameterize our model by a natural number t that controls the number of function values the adversary can erase *after* the oracle answers each query[2]. Mathematically, we represent the oracle and the adversary as one entity. However, it might be helpful to think of the oracle as the data holder and of the adversary as the obstructionist. A *t-online-erasure* oracle can replace values $\mathcal{O}(x)$ on up to t points x with a special symbol $\perp$, thus erasing them. The new values will be used by the oracle to answer future queries to the corresponding points. The locations of erasures are unknown to the algorithm. 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.

We focus on investigating property testing in the presence of online erasures. In the property testing model, introduced by [59, 33] with the goal of formally studying sublinear-time algorithms, a property is represented by a set $\mathcal{P}$ (of functions satisfying the desired property). A function f is ε-far from $\mathcal{P}$ if f differs from each function $g \in \mathcal{P}$ on at least an ε fraction of domain points. The goal is to distinguish, with constant probability, functions $f \in \mathcal{P}$ from functions that are ε-far from $\mathcal{P}$. We call an algorithm a *t-online-erasure-resilient ε-tester* for property $\mathcal{P}$ if, given parameters $t \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, and access to an input function f via a t-online-erasure oracle, the algorithm accepts with probability at least $2/3$ if $f \in \mathcal{P}$ and rejects with probability at least $2/3$ if f is ε-far from $\mathcal{P}$.

We study the query complexity of online-erasure-resilient testing of several fundamental properties. We show that for linearity and quadraticity of functions $f : \{0,1\}^d \to \{0,1\}$, the query complexity of t-online-erasure-resilient testing for constant t is asymptotically the same as in the standard model. For linearity, we also prove tight bounds in terms of t, showing that the query complexity is $\Theta(\log t)$. A function $f(x)$ is *linear* if it can be represented as a sum of monomials of the form $x[i]$, where $x = (x[1], \ldots, x[d])$ is a vector of d bits; the function is *quadratic* if it can be represented as a sum of monomials of the form $x[i]$ or $x[i]x[j]$.

To understand the difficulty of testing in the presence of online erasures, consider the case of linearity and $t = 1$. The celebrated tester for linearity in the standard property testing model was proposed by Blum, Luby, and Rubinfeld [18]. It looks for witnesses of

[1] Input objects such as strings, sequences, images, matrices, and graphs can all be represented as functions.

[2] If the adversary were allowed to erase the query of the algorithm before answering it, the algorithm would only see erased values. We give several motivating scenarios for the adversarial behavior in our model. The first example is a situation where the adversary reacts by deleting additional data after some bank records are pulled by authorities as part of an investigation. In the GDPR example mentioned previously, we argued that individuals could be prompted to restrict access to their data only after noticing an inquiry into their or others' data. Finally, in a legal setting, if the adversary is served a subpoena, they are legally bound to answer the query, but could nonetheless destroy related evidence that is not included in the subpoena.

non-linearity that consist of three points x, y, and $x \oplus y$ satisfying $f(x) + f(y) \neq f(x \oplus y)$, where addition is mod 2, and $\oplus$ denotes bitwise XOR. Bellare et al. [7] show that if f is ε-far from linear, then a triple $(x, y, x \oplus y)$ is a witness to non-linearity with probability at least ε when $x, y \in \{0,1\}^d$ are chosen uniformly and independently at random. In our model, after x and y are queried, the oracle can erase the value of $x \oplus y$. To overcome this, our tester considers witnesses with more points, namely, of the form $\sum_{x \in T} f(x) \neq f(\bigoplus_{x \in T} x)$ for sets $T \subset \{0,1\}^d$ of even size.

Witnesses of non-quadraticity are even more complicated. The tester of Alon et al. [2] looks for witnesses consisting of points x, y, z, and all four of their linear combinations. We describe a two-player game that models the interaction between the tester and the adversary and give a winning strategy for the tester-player. We also consider *witness structures* in which all specified tuples are witnesses of non-quadraticity (to allow for the possibility of the adversary erasing some points from the structure). We analyze the probability of getting a witness structure under uniform sampling when the input function is ε-far from quadratic. Our investigation leads to a deeper understanding of the structure of witnesses for both properties, linearity and quadraticity.

In contrast to linearity and quadraticity, we show that several other properties, specifically, sortedness and the Lipschitz property of sequences, and the Lipschitz property of functions $f : \{0,1\}^d \to \{0,1,2\}$, cannot be tested in the presence of an online-erasure oracle, even with $t = 1$, no matter how many queries the algorithm makes. Interestingly, witnesses for these properties have a much simpler structure than witnesses for linearity and quadraticity. Consider the case of sortedness of integer sequences, represented by functions $f : [n] \to \mathbb{N}$. A sequence is *sorted* (or the corresponding function is *monotone*) if $f(x) \leq f(y)$ for all $x < y$ in $[n]$. A witness of non-sortedness consists of two points $x < y$, such that $f(x) > f(y)$. In the standard model, sortedness can be ε-tested with an algorithm that queries an $O(\sqrt{n/\varepsilon})$ uniform and independent points [29]. (The fastest testers for this property have $O(\frac{\log \varepsilon n}{\varepsilon})$ query complexity [26, 25, 15, 20, 11], but they make correlated queries that follow a more complicated distribution.) Our impossibility result demonstrates that even the simplest testing strategy of querying independent points can be thwarted by an online adversary. To prove this result, we use sequences that are far from being sorted, but where each point is involved in only one witness, allowing the oracle to erase the second point of the witness as soon as the first one is queried. Using a version of Yao's principle that is suitable for our model, we turn these examples into a general impossibility result for testing sortedness with a 1-online-erasure oracle.

Our impossibility result for testing sortedness uses sequences with many (specifically, n) distinct integers. We show that this is not a coincidence by designing a t-online-erasure-resilient sortedness tester that works for sequences that have $O(\frac{\varepsilon^2 n}{t})$ distinct values. However, the number of distinct values does not have to be large to preclude testing the Lipschitz property in our model. A function $f : [n] \to \mathbb{N}$, representing an n-integer sequence, is *Lipschitz* if $|f(x) - f(y)| \leq |x - y|$ for all $x, y \in [n]$. Similarly, a function $f\colon \{0,1\}^d \to \mathbb{R}$ is *Lipschitz* if $|f(x) - f(y)| \leq \|x - y\|_1$ for all $x, y \in \{0,1\}^d$. We show that the Lipschitz property of sequences, as well as d-variate functions, cannot be tested even when the range has size 3, even with $t = 1$, no matter how many queries the algorithm makes.

Comparison to related models. Our model is closely related to (offline) erasure-resilient testing of Dixit et al. [24]. In the model of Dixit et al., also investigated in [56, 54, 12, 50, 43, 47], the adversary performs all erasures to the function before the execution of the algorithm. An (offline) erasure-resilient tester is given a parameter $\alpha \in (0,1)$, an upper bound on the

fraction of the values that are erased. The adversary we consider is more powerful in the sense that it can perform erasures online, during the execution of the tester. However, in some parameter regimes, our oracle cannot perform as many erasures. Importantly, all three properties that we show are impossible to test in our model, are testable in the model of Dixit et al. with essentially the same query complexity as in the standard model [24]. It is open if there are properties that have lower query complexity in the online model than in the offline model. The models are not directly comparable because the erasures are budgeted differently.

Another widely studied model in property testing is that of tolerant testing [52]. As explained by Dixit et al., every tolerant tester is also (offline) erasure-resilient with corresponding parameters. As pointed out in [52], the BLR tester is a tolerant tester of linearity for α significantly smaller than ε. Tolerant testing of linearity with distributional assumptions was studied in [42] and tolerant testing of low-degree polynomials over large alphabets was studied in [34]. Tolerant testing of sortedness is closely related to approximating the distance to monotonicity and estimating the longest increasing subsequence. These tasks can be performed with polylogorithmic in n number of queries [52, 1, 60]. As we showed, sortedness is impossible to test in the presence of online erasures.

1.1 Our Results

We design t-online-erasure-resilient testers for linearity and quadraticity, two properties widely studied because of their connection to probabilistically checkable proofs, hardness of approximating NP-hard problems, and coding theory. Our testers have *1-sided error*, that is, they always accept functions with the property. They are also *nonadaptive*, that is, their queries do not depend on answers to previous queries.

Linearity. Starting from the pioneering work of [18], linearity testing has been investigated, e.g., in [9, 10, 27, 7, 8, 66, 65, 62, 36, 13, 61, 63, 64, 40] (see [55] for a survey). Linearity can be ε-tested in the standard property testing model with $O(1/\varepsilon)$ queries by the BLR tester. We say that a pair (x, y) *violates* linearity if $f(x) + f(y) \neq f(x \oplus y)$. The BLR tester repeatedly selects a uniformly random pair of domain points and rejects if it violates linearity. A tight lower bound on the probability that a uniformly random pair violates linearity was proven by Bellare et al. [7] and Kaufman et al. [40].

We show that linearity can be ε-tested with $\widetilde{O}(\log t/\varepsilon)$ queries with a t-online-erasure oracle.

▶ **Theorem 1.1.** *There exist a constant $c_0 \in (0,1)$ and a 1-sided error, nonadaptive, t-online-erasure-resilient ε-tester for linearity of functions $f\colon \{0,1\}^d \to \{0,1\}$ that works for $t \leq c_0 \cdot \varepsilon^{5/4} \cdot 2^{d/4}$ and makes $O\big(\min\big(\frac{1}{\varepsilon}\log\frac{t}{\varepsilon}, \frac{t}{\varepsilon}\big)\big)$ queries.*

Our linearity tester has query complexity $O(1/\varepsilon)$ for constant t, which is optimal even in the standard property testing model, with no erasures. The tester looks for more general witnesses of non-linearity than the BLR tester, namely, it looks for tuples T of elements from $\{0,1\}^d$ such that $\sum_{x\in T} f(x) \neq f(\bigoplus_{x\in T} x)$ and $|T|$ is even. We call such tuples *violating*. The analysis of our linearity tester crucially depends on the following structural theorem.

▶ **Theorem 1.2.** *Let T be a tuple of a fixed even size, where each element of T is sampled uniformly and independently at random from $\{0,1\}^d$. If a function $f\colon \{0,1\}^d \to \{0,1\}$ is ε-far from linear, then*

$$\Pr_T\Big[\sum_{x\in T} f(x) \neq f(\bigoplus_{x\in T} x)\Big] \geq \varepsilon.$$

Our theorem generalizes the result of [7], which dealt with the case $|T| = 2$. We remark that the assertion in Theorem 1.2 does not hold for odd $|T|$. Consider the function $f(x) = x[1] + 1 \pmod 2$, where $x[1]$ is the first bit of x. Function f is $\frac{1}{2}$-far from linear, but has no violating tuples of odd size.

The core procedure of our linearity tester queries $\Theta(\log(t/\varepsilon))$ uniformly random points from $\{0,1\}^d$ to build a reserve and then queries sums of the form $\bigoplus_{x \in T} x$, where T is a uniformly random tuple of reserve elements such that $|T|$ is even. The *quality* of the reserve is the probability that T is violating. The likelihood that the procedure catches a violating tuple depends on the quality of the reserve (which is a priori unknown to the tester) and the number of sums queried. Instead of querying the same number of sums in each iteration of this core procedure, one can obtain a better query complexity by guessing different reserve qualities for each iteration and querying the number of sums that is inversely proportional to the reserve quality. We decide on the number of sums to query based on the *work investment* strategy by Berman, Raskhodnikova, and Yaroslavtsev [14], which builds on an idea proposed by Levin and popularized by Goldreich [32].

Next, we show that our tester has optimal query complexity in terms of the erasure budget t.

▶ **Theorem 1.3.** *For all $\varepsilon \in (0, \frac{1}{4}]$, every t-online-erasure-resilient ε-tester for linearity of functions $f\colon \{0,1\}^d \to \{0,1\}$ must make more than $\log_2 t$ queries.*

The main idea in the proof of Theorem 1.3 is that when a tester makes $\lfloor \log_2 t \rfloor$ queries, the adversary has the budget to erase all linear combinations of the previous queries after every step. As a result, the tester cannot distinguish a random linear function from a random function.

Quadraticity. Quadraticity and, more generally, low-degree testing have been studied, e.g., in [6, 5, 31, 27, 30, 59, 57, 2, 3, 46, 45, 41, 61, 63, 38, 16, 35, 58, 22]. Low-degree testing is closely related to local testing of Reed-Muller codes. The Reed-Muller code $\mathcal{C}(k, d)$ consists of codewords, each of which corresponds to all evaluations of a polynomial $f\colon \{0,1\}^d \to \{0,1\}$ of degree at most k. A local tester for a code queries a few locations of a codeword; it accepts if the codeword is in the code; otherwise, it rejects with probability proportional to the distance of the codeword from the code.

In the standard property testing model, quadraticity can be ε-tested with $O(1/\varepsilon)$ queries by the tester of Alon et al. [2] that repeatedly selects $x, y, z \sim \{0,1\}^d$ and queries f on all of their linear combinations – the points themselves, the double sums $x \oplus y, x \oplus z, y \oplus z$, and the triple sum $x \oplus y \oplus z$. The tester rejects if the values of the function on all seven queried points sum to 1, since this cannot happen for a quadratic function. A tight lower bound on the probability that the resulting 7-tuple is a witness of non-quadraticity was proved by Alon et al. [2] and Bhattacharyya et al. [16].

We prove that quadraticity can be ε-tested with $O(1/\varepsilon)$ queries with a t-online-erasure-oracle for constant t. Our tester can be easily modified to give a local tester for the Reed-Muller code $\mathcal{C}(2, d)$ that works with a t-online-erasure oracle.

▶ **Theorem 1.4.** *There exists a 1-sided error, nonadaptive, t-online-erasure-resilient ε-tester for quadraticity of functions $f\colon \{0,1\}^d \to \{0,1\}$ that makes $O(\frac{1}{\varepsilon})$ queries for constant t.*

The dependence on t in the query complexity of our quadraticity tester is at least doubly exponential, and it is an open question whether it can be improved. The main ideas behind our quadraticity tester are explained in Subsection 1.2.

Sortedness. Sortedness testing (see [53] for a survey) was introduced by Ergun et al. [26]. Its query complexity has been pinned down to $\Theta(\frac{\log(\varepsilon n)}{\varepsilon})$ by [26, 28, 21, 11].

We show that online-erasure-resilient testing of integer sequences is, in general, impossible.

▶ **Theorem 1.5.** *For all $\varepsilon \in (0, \frac{1}{12}]$, there is no 1-online-erasure-resilient ε-tester for sortedness of integer sequences.*

In the case without erasures, sortedness can be tested with $O(\sqrt{n/\varepsilon})$ uniform and independent queries [29]. Theorem 1.5 implies that a uniform tester for a property does not translate into the existence of an online-erasure-resilient tester, counter to the intuition that testers that make only uniform and independent queries should be less prone to adversarial attacks. Our lower bound construction demonstrates that the structure of violations to a property plays an important role in determining whether the property is testable.

The hard sequences from the proof of Theorem 1.5 have n distinct values. Pallavoor et al. [49, 51] considered the setting when the tester is given an additional parameter r, the number of distinct elements in the sequence, and obtained an $O(\frac{\log r}{\varepsilon})$-query tester. Two lower bounds apply to this setting: $\Omega(\log r)$ for nonadaptive testers [17] and $\Omega(\frac{\log r}{\log\log r})$ for all testers for the case when $r = n^{1/3}$ [11]. Pallavoor et al. also showed that sortedness can be tested with $O(\sqrt{r}/\varepsilon)$ uniform and independent queries. We extend the result of Pallavoor et al. to the setting with online erasures for the case when r is small.

▶ **Theorem 1.6.** *Let $c_0 > 0$ be a constant. There exists a 1-sided error, nonadaptive, t-online-erasure-resilient ε-tester for sortedness of n-element sequences with at most r distinct values. The tester makes $O(\frac{\sqrt{r}}{\varepsilon})$ uniform and independent queries and works when $r < \frac{\varepsilon^2 n}{c_0 t}$.*

Thus, sortedness is not testable with online erasures when r is large and is testable in the setting when r is small. For example, for Boolean sequences, it is testable with $O(1/\varepsilon)$ queries. The proofs of our results on sortedness appear in the full version of this work [39].

The Lipschitz property. Lipschitz testing, introduced by [37], was subsequently studied in [20, 23, 14, 4, 19]. Lipschitz testing of functions $f : [n] \to \{0, 1, 2\}$ can be performed with $O(\frac{1}{\varepsilon})$ queries [37]. For functions $f : \{0,1\}^d \to \mathbb{R}$, it can be done with $O(\frac{d}{\varepsilon})$ queries [37, 20].

We show that the Lipschitz property is impossible to test in the online-erasures model even when the range of the function has only 3 distinct values. This applies to both domains, $[n]$ and $\{0,1\}^d$.

▶ **Theorem 1.7.** *For all $\varepsilon \in (0, \frac{1}{8}]$, there is no 1-online-erasure-resilient ε-tester for the Lipschitz property of functions $f\colon [n] \to \{0, 1, 2\}$. The same statement holds when the domain is $\{0,1\}^d$ instead of $[n]$.*

The proof of this theorem appears in the full version of this work [39].

Yao's minimax principle. All our lower bounds use Yao's minimax principle. A formulation of Yao's principle suitable for our online-erasures model appears in the full version [39].

1.2 The Ideas Behind Our Quadraticity Tester

One challenge in generalizing the tester of [2] to work with an online-erasure oracle is that its queries are correlated. First, we want to ensure that the tester can obtain function values on tuples of the form $(x, y, z, x \oplus y, x \oplus z, y \oplus z, x \oplus y \oplus z)$. Then we want to ensure that, if

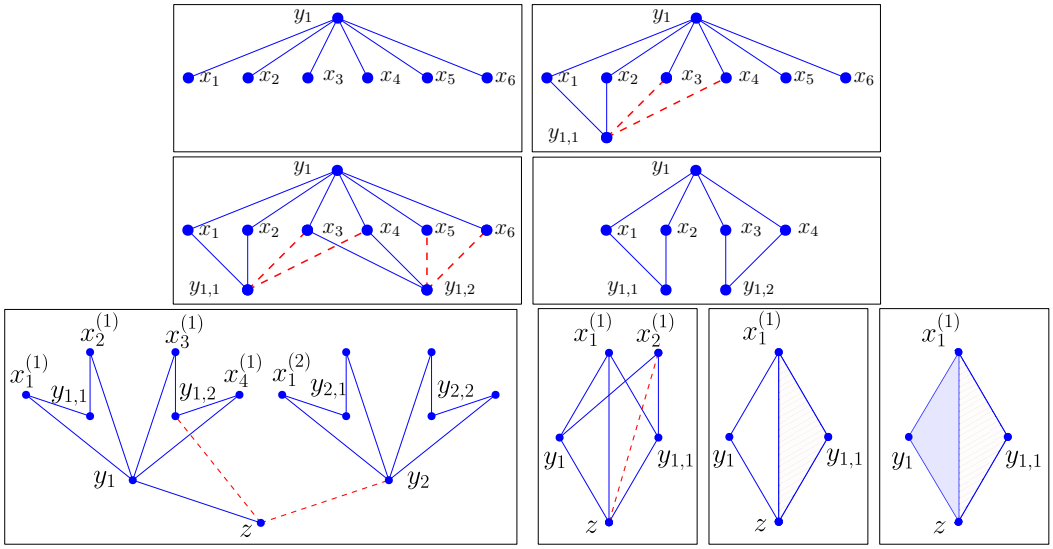

Figure 1 Stages in the quadraticity game for $t = 1$, played according to the winning strategy for Player 1: connecting the y-decoys from the first tree to x-decoys (frames 1-4); drawing z and connecting it to y-decoys and an x-decoy (frames 5-6), and creation of a blue triangle (frames 7–8). Frame 5 contains edges from z to two structures, each replicating frame 4. We depict only points and edges relevant for subsequent frames.

the original function is far from the property, the tester is likely to catch such a tuple that is also a witness to not satisfying the property. Next, we formulate a two-player game[3] that abstracts the first task. In the game, the tester-player sees what erasures are made by the oracle-player. This assumption is made to abstract out the most basic challenge and is not used in the algorithms' analyses.

Quadraticity testing as a two-player game. Player 1 represents the tester and Player 2 represents the adversary. The players take turns drawing points, connecting points with edges, and coloring triangles specified by drawn points, each in their own color. Player 1 wins the game if it draws in blue all the vertices and edges of a triangle and colors the triangle blue. The vertices represent the points $x, y, z \in \{0,1\}^d$, the edges are the sums $x \oplus y, x \oplus z, y \oplus z$, and the triangle is the sum $x \oplus y \oplus z$. A move of Player 1 consists of drawing a point or an edge between two existing non-adjacent points or coloring an uncolored triangle between three existing points (in blue). A move of Player 2 consists of at most t steps; in each step, it can draw a red edge between existing points or color a triangle between three existing points (in red).

Our online-erasure-resilient quadraticity tester is based on a winning strategy for Player 1 with $t^{O(t)}$ moves. At a high level, Player 1 first draws many decoys for x. The y-decoys are organized in $t+1$ full $(t+1)$-ary trees of depth t. The root for tree i is y_i, its children are $y_{i,j}$, where $j \in [t+1]$, etc. We jot the rest of the winning strategy for $t = 1$ and depict it in Figure 1. In this case, Player 1 does the following for each of two trees: it draws points $x_1^{(i)}, \ldots, x_{12}^{(i)}, y_i$; connects y_i to half of x-decoys (w.l.o.g., $x_1^{(i)}, \ldots, x_6^{(i)}$); draws point $y_{i,1}$, connects it to two of the x-decoys adjacent to y_1 (w.l.o.g., $x_1^{(i)}$ and $x_2^{(i)}$); draws point $y_{i,2}$,

[3] This game has been tested on real children, and they spent hours playing it.

connects it to two of $x_3^{(i)}, \ldots, x_6^{(i)}$ (w.l.o.g., $x_3^{(i)}$ and $x_4^{(i)}$); draws z and connects it to one of the roots (w.l.o.g., y_1), connects z to one of $y_{1,1}$ and $y_{1,2}$ (w.l.o.g., $y_{1,1}$), connects z to one of $x_1^{(1)}$ and $x_2^{(1)}$ (w.l.o.g., $x_1^{(1)}$), and finally colors one of the triangles $x_1^{(1)}y_1z$ and $x_1^{(1)}y_{1,1}z$, thus winning the game. The decoys are arranged to guarantee that Player 1 always has at least one available move in each step of the strategy.

For general t, the winning strategy is described in full detail in Algorithm 2. Recall that the y-decoys are organized in $t+1$ full $(t+1)$-ary trees of depth t. For every root-to-leaf path in every tree, Player 1 draws edges from all the nodes in that path to a separate set of $t+1$ decoys for x. After z is drawn, the tester "walks" along a root-to-leaf path in one of the trees, drawing edges between z and the y-decoys on the path. The goal of this walk is to avoid the parts of the tree spoiled by Player 2. Finally, Player 1 connects z to an x-decoy that is adjacent to all vertices in the path, and then colors a triangle involving this x-decoy, a y-decoy from the chosen path, and z. The structure of decoys guarantees that Player 1 always has $t+1$ options for its next move, only t of which can be spoiled by Player 2.

From the game to a tester. There are two important aspects of designing a tester that are abstracted away in the game: First, the tester does not actually know which values are erased until it queries them. Second, the tester needs to catch a witness demonstrating a violation of the property, not merely a tuple of the right form with no erasures. Here, we briefly describe how we overcome these challenges.

Our quadraticity tester is based directly on the game. It converts the moves of the winning strategy of Player 1 into a corresponding procedure, making a uniformly random guess at each step about the choices that remain nonerased. There are three core technical lemmas used in the analysis of the algorithm. Lemma 4.4 lower bounds the probability that the tester makes correct guesses at each step about which edges (double sums) and triangles (triple sums) remain nonerased, thus addressing part of the first challenge. This probability depends only on the erasure budget t. To address the second challenge, Lemma 4.3 gives a lower bound on the probability that uniformly random sampled points (the x- and y- decoys together with z) form a large violation structure, where all triangles that Player 1 might eventually complete violate quadraticity. Building on a result of Alon et al., we show that even though the number of triangles involved in the violation structure is large, namely $t^{O(t)}$, the probability of sampling such a structure is $\alpha = \Omega(\min(\varepsilon, c_t))$, where $c_t \in (0,1)$ depends only on t. Finally, Lemma 4.2 shows that despite the online adversarial erasures, the tester has a probability of $\alpha/2$ of sampling the points of such a large violation structure and obtaining their values from the oracle. The three lemmas combined show that quadraticity can be tested with $O(1/\varepsilon)$ queries for constant t.

1.3 Conclusions and Open Questions

We initiate a study of sublinear-time algorithms in the presence of online adversarial erasures. We design efficient online-erasure-resilient testers for several important properties (linearity, quadraticity, and – for the case of small number of distinct values – sortedness). For linearity, we prove tight upper and lower bounds in terms of t. We also show that several basic properties, specifically, sortedness of integer sequences and the Lipschitz properties, cannot be tested in our model. We now list several open problems.

- Sortedness is an example of a property that is impossible to test with online erasures, but is easy to test with offline erasures, as well as tolerantly. Is there a property that has smaller query complexity in the online-erasure-resilient model than in the (offline) erasure-resilient model of [24]?

- We design a t-online-erasure-resilient quadraticity tester that makes $O(1/\varepsilon)$ queries for constant t. What is the query complexity of t-online-erasure-resilient quadraticity testing in terms of t and ε? Specifically, the dependence on t in the query complexity of our quadraticity tester is at least doubly exponential, and it is open whether it can be improved.
- The query complexity of ε-testing if a function is a polynomial of degree at most k is $\Theta(\frac{1}{\varepsilon} + 2^k)$ [2, 16]. Is there a low-degree test for $k \geq 3$ that works in the presence of online erasures?

2 An Online-Erasure-Resilient Linearity Tester

To prove Theorem 1.1, we present and analyze two testers. Our main online-erasure-resilient linearity tester (Algorithm 1) is presented in this section. Its query complexity has optimal dependence on t and nearly optimal dependence on ε. Its performance is summarized in Theorem 2.1. To complete the proof of Theorem 1.1, we give a $O(t/\varepsilon)$-query linearity tester in the full version of our work [39].

▶ **Theorem 2.1.** *There exists $c_0 \in (0,1)$ and a 1-sided error, nonadaptive, t-online-erasure-resilient ε-tester for linearity of functions $f\colon \{0,1\}^d \to \{0,1\}$ that works for $t \leq c_0 \cdot \varepsilon^{5/4} \cdot 2^{d/4}$ and makes $O(\frac{1}{\varepsilon} \log \frac{t}{\varepsilon})$ queries.*

The t-online-erasure-resilient tester guaranteed by Theorem 2.1 is presented in Algorithm 1.

Algorithm 1 An Online-Erasure-Resilient Linearity Tester.

Require: $\varepsilon \in (0, \frac{1}{2})$, erasure budget $t \in \mathbb{N}$, access to function $f : \{0,1\}^d \to \{0,1\}$ via a t-online-erasure oracle.

Let $q = 2 \log \frac{50t}{\varepsilon}$.
for all $j \in [\log \frac{8}{\varepsilon}]$:
  repeat $\frac{8 \ln 5}{2^j \varepsilon}$ times:
    for all $i \in [q]$:
      Sample $x_i \sim \{0,1\}^d$ and query f at x_i.
    repeat $4 \cdot 2^j$ times:
      Sample a uniform nonempty subset I of $[q]$ of even size.
      Query f at $\bigoplus_{i \in I} x_i$.
      Reject if $\sum_{i \in I} f(x_i) \neq f(\bigoplus_{i \in I} x_i)$ and all points are nonerased.
Accept.

2.1 Proof of Theorem 1.2

In this section, we prove Theorem 1.2, the main structural result needed for Theorem 2.1. Recall that a k-tuple $(x_1, \ldots, x_k) \in (\{0,1\}^d)^k$ violates linearity if $f(x_1) + \cdots + f(x_k) \neq f(x_1 \oplus \cdots \oplus x_k)$. (Addition is mod 2 when adding values of Boolean functions.) Theorem 1.2 states that if f is ε-far from linear, then for all even k, with probability at least ε, independently and uniformly sampled points $x_1, \ldots, x_k \sim \{0,1\}^d$ form a violating k-tuple. Our proof of Theorem 1.2 builds on the proof of [7, Theorem 1.2], which is a special case of Theorem 1.2 for $k = 2$. The proof is via Fourier analysis. Next, we state some standard facts and definitions related to Fourier analysis. See, e.g., [48] for proofs of these facts.

Consider the space of all real-valued functions on $\{0,1\}^d$ equipped with the inner-product

$$\langle g, h\rangle = \mathop{\mathbb{E}}_{x\sim\{0,1\}^d}[g(x)h(x)],$$

where $g, h\colon \{0,1\}^d \to \mathbb{R}$. The *character* functions $\chi_S\colon \{0,1\}^d \to \{-1,1\}$, defined as $\chi_S = (-1)^{\sum_{i\in S} x[i]}$ for $S \subseteq [d]$, form an orthonormal basis for the space of functions under consideration. Hence, every function $g : \{0,1\}^d \to \mathbb{R}$ can be uniquely expressed as a linear combination of the functions χ_S, where $S \subseteq [d]$. The Fourier coefficients of g are the coefficients on the functions χ_S in this linear representation of g.

▶ **Definition 2.2** (Fourier coefficient). *For $g\colon \{0,1\}^d \to \mathbb{R}$ and $S \subseteq [d]$, the Fourier coefficient of g on S is $\widehat{g}(S) = \langle g, \chi_S\rangle = \mathop{\mathbb{E}}_{x\sim\{0,1\}^d}[g(x)\chi_S(x)]$.*

We will need the following facts about Fourier coefficients.

▶ **Theorem 2.3** (Parseval's Theorem). *For all functions $g\colon \{0,1\}^d \to \mathbb{R}$, it holds that $\langle g, g\rangle = \sum_{S\subseteq[d]} \widehat{g}(S)^2$. In particular, if $g\colon \{0,1\}^d \to \{-1,1\}$ then $\sum_{S\subseteq[d]} \widehat{g}(S)^2 = 1$.*

▶ **Theorem 2.4** (Plancherel's Theorem). *For all functions $g, h\colon \{0,1\}^d \to \mathbb{R}$, it holds that $\langle g, h\rangle = \sum_{S\subseteq[d]} \widehat{g}(S)\widehat{h}(S)$.*

A function $g\colon \{0,1\}^d \to \{-1,1\}$ is linear if $g(x)g(y) = g(x \oplus y)$ for all $x, y \in \{0,1\}^d$.

▶ **Lemma 2.5.** *The distance of $g\colon \{0,1\}^d \to \{-1,1\}$ to linearity is $\frac{1}{2} - \frac{1}{2}\max_{S\subseteq[d]} \widehat{g}(S)$.*

Finally, we also use the convolution operation, defined below, and one of its key properties.

▶ **Definition 2.6** (Convolution). *Let $g, h\colon \{0,1\}^d \to \mathbb{R}$. Their convolution is the function $g * h : \{0,1\}^d \to \mathbb{R}$ defined by $(g * h)(x) = \mathop{\mathbb{E}}_{y\sim\{0,1\}^d}[g(y)h(x \oplus y)]$.*

▶ **Theorem 2.7.** *Let $g, h\colon \{0,1\}^d \to \mathbb{R}$. Then, for all $S \subseteq [d]$, it holds $\widehat{g * h}(S) = \widehat{g}(S)\widehat{h}(S)$.*

Proof of Theorem 1.2. Define $g\colon \{0,1\}^d \to \{-1,1\}$ so that $g(x) = (-1)^{f(x)}$. That is, g is obtained from the function f by encoding its output with ± 1. Note that the distance to linearity of g is the same as the distance to linearity of f. We have that the expression $\frac{1}{2} - \frac{1}{2}g(x_1)\dots g(x_k)g(x_1 \oplus \dots \oplus x_k)$ is an indicator for the event that points $x_1, \dots, x_k \sim \{0,1\}^d$ violate linearity for g. Define g^{*k} to be the convolution of g with itself k times, i.e., $g^{*k} = g * \dots * g$, where the $*$ operator appears $k-1$ times. We obtain

$$\begin{aligned}
&\Pr_{x_1,\dots,x_k\sim\{0,1\}^d}[(x_1,\dots,x_k) \text{ violates linearity}]\\
&= \mathop{\mathbb{E}}_{x_1,\dots,x_k\sim\{0,1\}^d}\Big[\frac{1}{2} - \frac{1}{2}g(x_1)\dots g(x_k)g(x_1 \oplus \dots \oplus x_k)\Big]\\
&= \frac{1}{2} - \frac{1}{2}\mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}\sim\{0,1\}^d}[g(x_1)\dots g(x_{k-1}) \cdot \mathop{\mathbb{E}}_{x_k\sim\{0,1\}^d}[g(x_k)g(x_1 \oplus \dots \oplus x_k)]]\\
&= \frac{1}{2} - \frac{1}{2}\mathop{\mathbb{E}}_{x_1,\dots,x_{k-1}\sim\{0,1\}^d}[g(x_1)\dots g(x_{k-1})(g * g)(x_1 \oplus \dots \oplus x_{k-1})] &(1)\\
&= \frac{1}{2} - \frac{1}{2}\mathop{\mathbb{E}}_{x_1\sim\{0,1\}^d}[g(x_1) \cdot (g^{*k})(x_1)] &(2)\\
&= \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[d]} \widehat{g}(S)\widehat{g^{*k}}(S) &(3)\\
&= \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[d]} \widehat{g}(S)^{k+1}, &(4)
\end{aligned}$$

where Equation 1 holds by the definition of convolution, Equation 2 follows by repeated application of the steps used to obtain Equation 1, Equation 3 follows from Plancherel's Theorem (Theorem 2.4), and Equation 4 follows from Theorem 2.7.

Note that $|\widehat{g}(S)| \leq 1$ for all $S \subseteq [d]$, because $\widehat{g}(S)$ is the inner product of two functions with range in $\{-1, 1\}$. In addition, $\widehat{g}(S) \geq 0$ for S such that χ_S is the closest linear function to g. Then, for even k,

$$\sum_{S\subseteq[d]} \widehat{g}(S)^{k+1} \leq \max_{S\subseteq[d]}(\widehat{g}(S)^{k-1}) \sum_{S\subseteq[d]} \widehat{g}(S)^2 = \max_{S\subseteq[d]}(\widehat{g}(S)^{k-1}) \leq \max_{S\subseteq[d]} \widehat{g}(S),$$

where the equality follows from Parseval's Theorem (Theorem 2.3). By Lemma 2.5, the distance of g to linearity is $\frac{1}{2} - \frac{1}{2}\max_{S\subseteq[d]} \widehat{g}(S)$, which is at least ε, since g is ε-far from linear. This concludes the proof. ◀

2.2 Proof of Theorem 2.1

In this section, we prove Theorem 2.1 using Theorem 1.2. In Lemma 2.8, we analyze the probability of good events that capture, roughly, that the queries made in the beginning of each iteration haven't already been "spoiled" by the previous erasures. Then we use the work investment strategy of [14], stated in Lemma 2.9, together with Theorem 1.2 and Lemma 2.8 to prove Theorem 2.1.

Each iteration of the outer **repeat** loop in Steps 3-9 of Algorithm 1 is called a *round*. We say a query x is *successfully obtained* if it is nonerased when queried, i.e., the tester obtains $f(x)$ as opposed to $\perp$.

▶ **Lemma 2.8** (Good events). *Fix one round of Algorithm 1. Consider the points $x_1, \ldots, x_q$ queried in Step 5 of this round, where $q = 2\log(50t/\varepsilon)$, and the set S of all sums $\bigoplus_{i\in I} x_i$, where I is a nonempty subset of $[q]$ of even size. Let G_1 be the (good) event that all points in S are distinct. Let G_2 be the (good) event that all points $x_1, \ldots, x_q$ are successfully obtained and all points in S are nonerased at the beginning of the round. Finally, let $G = G_1 \cap G_2$. Then $\Pr[\overline{G}] \leq \frac{\varepsilon}{2}$ for all adversarial strategies.*

Proof. First, we analyze event G_1. Consider points $x_{i_1}, \ldots, x_{i_k}, x_{i'_1}, \ldots, x_{i'_\ell} \sim \{0,1\}^d$, where $\{i_1, \ldots, i_k\} \neq \{i'_1, \ldots, i'_\ell\}$, $k, \ell \in [q]$ and k, ℓ are even. Since the points are distributed uniformly and independently, so are the sums $x_{i_1} \oplus \cdots \oplus x_{i_k}$ and $x_{i'_1} \oplus \cdots \oplus x_{i'_\ell}$. The probability that two uniformly and independently sampled points $x, y \sim \{0,1\}^d$ are identical is $\frac{1}{2^d}$. The number of sets $I \subseteq [q]$ of even size is 2^{q-1} because every subset of $[q-1]$ can be uniquely completed to such a set I. By a union bound over all pairs of sums, $\Pr[\overline{G_1}] \leq \frac{2^{2q}}{4\cdot 2^d}$.

To analyze G_2, fix any adversarial strategy. The number of queries made by Algorithm 1 is at most

$$\sum_{j=1}^{\log(8/\varepsilon)} \frac{8\ln 5}{2^j\varepsilon}\left(q + 4\cdot 2^j\right) \leq \frac{8\ln 5}{\varepsilon}q + \frac{32\ln 5}{\varepsilon}\log\frac{8}{\varepsilon} \leq \frac{8\ln 5}{\varepsilon}q + \frac{16\ln 5}{\varepsilon}q \leq \frac{40q}{\varepsilon}. \tag{5}$$

Hence, the oracle erases at most $\frac{40qt}{\varepsilon}$ points. Since each point x_i is sampled uniformly from $\{0,1\}^d$,

$$\Pr_{x_i\sim\{0,1\}^d}[x_i \text{ is erased when queried}] \leq \frac{40qt}{\varepsilon\cdot 2^d}.$$

Additionally, before the queries x_i are revealed to the oracle, each sum $\bigoplus_{i\in I} x_i$ is distributed uniformly at random. Therefore, for every $\{i_1, \ldots, i_k\} \subseteq [q]$,

$$\Pr_{x_{i_1},\ldots,x_{i_k}\sim\{0,1\}^d}[x_{i_1} \oplus \cdots \oplus x_{i_k} \text{ is erased at the beginning of the round}] \leq \frac{40qt}{\varepsilon\cdot 2^d}.$$

By a union bound over the q points sampled in Step 5 and at most 2^{q-1} sums, we get

$$\Pr[\overline{G_2}] \leq \frac{40qt}{\varepsilon 2^d}(q + 2^{q-1}) \leq \frac{40qt \cdot 2^q}{\varepsilon \cdot 2^d}.$$

Since $q = 2\log\frac{50t}{\varepsilon}$, we get $\frac{40qt}{\varepsilon} \leq \frac{3}{4} \cdot 2^q$ and, consequently,

$$\Pr[\overline{G}] \leq \frac{2^{2q}}{4 \cdot 2^d} + \frac{40qt \cdot 2^q}{\varepsilon \cdot 2^d} \leq \frac{2^{2q}}{2^d} \leq \frac{50^4 \cdot t^4}{\varepsilon^4 2^d} \leq \frac{50^4 \cdot c_0^4 \cdot \varepsilon^5}{\varepsilon^4} \leq \frac{\varepsilon}{2},$$

since $t \leq c_0 \cdot \varepsilon^{5/4} \cdot 2^{d/4}$, as stated in Theorem 2.1, and assuming c_0 is sufficiently small. ◀

Next, we state the work investment lemma.

▶ **Lemma 2.9** (Lemma 2.5 of [14]). *Let X be a random variable taking values in $[0,1]$. Suppose $\mathbb{E}[X] \geq \alpha$. Let $s = \lceil \log(\frac{4}{\alpha}) \rceil$ and $\delta \in (0,1)$ be the desired probability of error. For all $j \in [s]$, let $p_j = \Pr[X \geq 2^{-j}]$ and $k_j = \frac{4\ln(1/\delta)}{2^j \alpha}$. Then $\prod_{j=1}^{s}(1-p_j)^{k_j} \leq \delta$.*

Proof of Theorem 2.1. By (5), the query complexity of Algorithm 1 is $O(\frac{q}{\varepsilon}) = O(\frac{\log(t/\varepsilon)}{\varepsilon})$. Algorithm 1 is nonadaptive and always accepts if f is linear. Suppose now that f is ε-far from linear and fix any adversarial strategy. We show that Algorithm 1 rejects with probability at least $2/3$.

Consider the *last* round of Algorithm 1. For points $x_1, \ldots, x_q \sim \{0,1\}^d$ sampled in Step 5 of this last round, let Y denote the fraction of nonempty sets $\{i_1, \ldots, i_k\} \subseteq [q]$ such that k is even and $(x_{i_1}, \ldots, x_{i_k})$ violates linearity. Recall the event G defined in Lemma 2.8. Let $\mathbf{1}_G$ be the indicator random variable for the event G for the last round.

▷ Claim 2.10. Let $X = Y \cdot \mathbf{1}_G$, where Y is as defined above. Then $\mathbb{E}[X] \geq \frac{\varepsilon}{2}$.

Proof. For all nonempty $\{i_1, \ldots, i_k\} \subseteq [q]$, such that k is even, let $Y_{i_1,\ldots,i_k}$ be the indicator for the event that $(x_{i_1}, \ldots, x_{i_k})$ violates linearity. By Theorem 1.2 and the fact that k is even,

$$\mathbb{E}[Y_{i_1,\ldots,i_k}] = \Pr_{x_{i_1},\ldots,x_{i_k} \sim \{0,1\}^d}[(x_{i_1}, \ldots, x_{i_k}) \text{ violates linearity}] \geq \varepsilon.$$

We obtain a lower bound on $\mathbb{E}[Y]$ by linearity of expectation.

$$\mathbb{E}[Y] = \frac{1}{2^{q-1}-1} \sum_{\{i_1,\ldots,i_k\} \subseteq [q], k \text{ even}} \mathbb{E}[Y_{i_1,\ldots,i_k}] \geq \varepsilon. \tag{6}$$

Observe that $X = Y$ when G occurs, and $X = 0$ otherwise. By the law of total expectation,

$$\begin{aligned}\mathbb{E}[X] &= \mathbb{E}[X|G] \cdot \Pr[G] + \mathbb{E}[X|\overline{G}] \cdot \Pr[\overline{G}] = \mathbb{E}[X|G] \cdot \Pr[G] = \mathbb{E}[Y|G] \cdot \Pr[G] \\ &= \mathbb{E}[Y] - \mathbb{E}[Y|\overline{G}] \cdot \Pr[\overline{G}] \geq \varepsilon - 1 \cdot (\varepsilon/2) = \varepsilon/2,\end{aligned}$$

where the inequality follows from Equation 6, the fact that $Y \leq 1$, and Lemma 2.8. ◁

Fix any round of Algorithm 1 and the value of j used in this round (as defined in Step 2). Let X' be defined as X, but for this round instead of the last one. The round is *special* if $X' \geq 2^{-j}$. Let $p'_j = \Pr[X' \geq 2^{-j}]$ and $p_j = \Pr[X \geq 2^{-j}]$. Then $p'_j \geq p_j$, since the number of erasures only increases with each round. For each j, there are $k_j = \frac{8\ln 5}{2^j \varepsilon}$ rounds of

Algorithm 1 that are run with this particular value of j. Since Algorithm 1 uses independent random coins for each round, the probability that no round is special is at most

$$\prod_{j=1}^{\log \frac{8}{\varepsilon}} (1-p'_j)^{k_j} \leq \prod_{j=1}^{\log \frac{8}{\varepsilon}} (1-p_j)^{k_j} \leq \frac{1}{5},$$

where the last inequality follows by Lemma 2.9 applied with $\delta = 1/5$ and $\alpha = \varepsilon/2$ and Claim 2.10. Therefore, with probability at least $\frac{4}{5}$, Algorithm 1 has a special round.

Consider a special round of Algorithm 1 and fix the value of j for this round. We show that Algorithm 1 rejects in the special round with probability at least 5/6. We call a sum $\bigoplus_{i\in I} x_i$ *violating* if the tuple $(x_i \colon i \in I)$ violates linearity. Since G occurred, all q points queried in Step 5 of Algorithm 1 were successfully obtained. So, the algorithm will reject as soon as it successfully obtains a violating sum. Since G occurred, there are at least $2^{q-1}-1$ distinct sums that can be queried in Step 8, all of them nonerased at the beginning of the round. Algorithm 1 makes at most $q + 4 \cdot 2^j$ queries in this round, and thus the fraction of these sums erased during the round is at most

$$\begin{aligned} t \cdot \frac{q+4\cdot 2^j}{2^{q-1}-1} &\leq t \cdot \left(\frac{1}{2^{q/2}} + \frac{3\cdot 2^j}{2^{q-2}}\right) \leq \frac{t}{t\cdot\frac{50}{\varepsilon}} + \frac{12t\cdot 2^j}{t^2\cdot(\frac{50}{\varepsilon})^2} \\ &= \frac{8}{50\cdot\frac{8}{\varepsilon}} + \frac{12\cdot 8^2\cdot 2^j}{50^2 t\cdot(\frac{8}{\varepsilon})^2} \leq \frac{0.16}{2^j} + \frac{0.3072}{2^j} \leq \frac{1}{2\cdot 2^j}, \end{aligned}$$

where in the first inequality we used that $\frac{q}{2^{q-1}-1} \leq \frac{1}{2^{q/2}}$ for $q \geq 9$ and that $2^{q-1}-1 \geq (4/3)\cdot 2^{q-2}$ for $q \geq 3$ (note that $q \geq 2\log(50t/\varepsilon) \geq 2\log 100 > 13$), in the second inequality we used $q \geq 2\log(50t/\varepsilon)$, and in the third inequality we used $2^j \leq \frac{8}{\varepsilon}$ and $t \geq 1$.

Since the round is special, at least a 2^{-j} fraction of the sums that can be queried in Step 8 are violating. Thus, the fraction of the sums that are violating and nonerased before each iteration of Steps 8-9 in this round is at least $2^{-j} - 2^{-j-1} = 2^{-j-1}$.

Then, each iteration of Steps 8-9 rejects with probability at least 2^{-j-1}. Since there are $4\cdot 2^j$ iterations with independently chosen sums, the probability that the special round accepts is at most

$$(1-2^{-j-1})^{4\cdot 2^j} \leq e^{-2} \leq 1/6.$$

That is, the probability that Algorithm 1 rejects in the special round is at least 5/6. Since the special round exists with probability at least $\frac{4}{5}$, Algorithm 1 rejects with probability at least $\frac{4}{5}\cdot\frac{5}{6} = \frac{2}{3}$. ◀

3 A Lower Bound for Online-Erasure-Resilient Linearity Testing

In this section, we prove Theorem 1.3 that shows that every t-online-erasure-resilient ε-tester for linearity of functions $f\colon \{0,1\}^d \to \{0,1\}$ must make more than $\log_2 t$ queries.

Proof of Theorem 1.3. Let $\mathcal{D}^+$ be the uniform distribution over all linear Boolean functions on $\{0,1\}^d$ and $\mathcal{D}^-$ be the uniform distribution over all Boolean functions functions on $\{0,1\}^d$.

We show that a function $f \sim \mathcal{D}^-$ is $\frac{1}{4}$-far from linear with probability at least 6/7. Let g be a linear function, $f \sim \mathcal{D}^-$, and $\mathrm{dist}(f,g)$ be the fraction of domain points on which f and g differ. Then, $\mathbb{E}[\mathrm{dist}(f,g)] = \frac{1}{2}$. By the Hoeffding bound, $\Pr_{f\sim\mathcal{D}^-}[\mathrm{dist}(f,g) \leq \frac{1}{4}] \leq e^{-\frac{2^d}{8}}$. By a union bound over the 2^d linear functions, $\Pr_{f\sim\mathcal{D}^-}[f \text{ is } \frac{1}{4}\text{-far from linear}] \geq 1 - 2^d\cdot e^{-\frac{2^d}{8}}$. For d large enough, this probability is at least 6/7.

We fix the following strategy for a t-online-erasure oracle $\mathcal{O}$: after responding to each query, erase t sums of the form $\bigoplus_{x\in T} x$, where T is a subset of the queries made so far, choosing the subsets T in some fixed order. If at most $\log_2 t$ queries are made, the adversary erases all the sums of queried points.

Let A be a deterministic algorithm that makes $q \leq \log_2 t$ queries to the oracle $\mathcal{O}$. Assume w.l.o.g. that A does not repeat queries. We describe two random processes $\mathcal{P}^+$ and $\mathcal{P}^-$ that interact with algorithm A in lieu of oracle $\mathcal{O}$ and provide query answers consistent with a random function from $\mathcal{D}^+$ and $\mathcal{D}^-$, respectively. For each query of A, both processes $\mathcal{P}^+$ and $\mathcal{P}^-$ return $\perp$ if the value has been previously erased by $\mathcal{O}$; otherwise, they return 0 or 1 with equal probability. Thus, the distribution over query-answer histories when A interacts with $\mathcal{P}^+$ is the same as when A interacts with $\mathcal{P}^-$.

Next, we describe how the processes $\mathcal{P}^+$ and $\mathcal{P}^-$ assign values to the locations of f that were either not queried by A or erased when queried, and show that they generate $\mathcal{D}^+$ and $\mathcal{D}^-$, respectively. After A finishes its queries, $\mathcal{P}^-$ sets the remaining unassigned locations (including the erased locations) of the function to be 0 or 1 with equal probability. Clearly, $\mathcal{P}^-$ generates a function from the distribution $\mathcal{D}^-$.

To describe $\mathcal{P}^+$ fully, first let $Q \subseteq \{0,1\}^d$ denote the queries of A that are answered with a value other than $\perp$. Since $q \leq \log_2 t$, by our choice of the oracle $\mathcal{O}$, the sum of any subset of vectors in Q is not contained in Q. Hence, the vectors in Q are linearly independent. Then, $\mathcal{P}^+$ completes Q to a basis B for $\{0,1\}^d$ and sets the value of f on all vectors in $B \setminus Q$ independently to 0 or 1 with equal probability.

Since B is a basis, each vector $y \in \{0,1\}^d$ can be expressed as a linear combination of vectors in B (with coefficients in $\{0,1\}$), that is, $y = \bigoplus_{x\in T} x$ for some $T \subseteq B$. The process $\mathcal{P}^+$ sets $f(y) = \sum_{x\in T} f(x)$, where addition is mod 2. The function f is linear and agrees with all values previously assigned by $\mathcal{P}^+$ to the vectors in Q. Moreover, f is distributed according to $\mathcal{D}^+$, since one can obtain a uniformly random linear function by first specifying a basis for $\{0,1\}^d$, and then setting the value of f to be 0 or 1 with equal probability for each basis vector.

Thus, $\mathcal{P}^+$ generates linear functions, $\mathcal{P}^-$ generates functions that are $\frac{1}{4}$-far from linear with probability at least $\frac{6}{7}$, and the query-answer histories for any deterministic algorithm A that makes at most $\log_2 t$ queries and runs against our t-online-erasure oracle $\mathcal{O}$ are identical under $\mathcal{P}^+$ and $\mathcal{P}^-$. Consequently, the lower bound follows from Yao's Minimax Principle. ◀

4 An Online-Erasure-Resilient Quadraticity Tester

In this section, we state our online-erasure-resilient quadraticity tester (Algorithm 2) and prove Theorem 1.4.

The main idea behind Algorithm 2 and its representation as a two-player game appear in Subsection 1.2, accompanied by explanatory figures for the case when $t = 1$. We now give a high level overview of Algorithm 2. For a function $f\colon \{0,1\}^d \to \{0,1\}$ and $x, y, z \in \{0,1\}^d$, let

$$T_f(x,y,z) = \sum_{\emptyset \neq S \subseteq \{x,y,z\}} f\Big(\bigoplus_{u\in S} u\Big),$$

where the first sum is mod 2. We say a triple (x, y, z) *violates* quadraticity if $T_f(x, y, z) = 1$. The tester of Alon et al. samples three vectors $x, y, z \in \{0,1\}^d$ uniformly and independently at random and rejects if (x, y, z) violates quadraticity. Our tester looks for the same kind of violations as the tester of Alon et al.

The main challenge in the design of our online-erasure-resilient tester is to ensure it can query all three such points and their sums in the presence of a t-online-erasure adversary. In each iteration of the **repeat** loop of Algorithm 2, the following steps are performed. For each $\ell \in [t+1]$, we first query a *reserve* of uniformly and independently sampled points $x_i^{(\ell)}$ for $i \in [(t+1)^2(2t+1)^t]$. Next, for each $\ell \in [t+1]$, we query a set of points that we visualize as being the nodes of a $(t+1)$-ary tree of depth t. There is a one-to-one correspondence between the nodes of such a tree and vectors of length up to $t+1$ over the alphabet $[t+1]$. For $m \in [t]$, we represent by $y_{(\ell,j_1,\ldots,j_m)}$, the sampled point visualized as a node at depth m in the ℓ-th tree, where the j_i's specify the unique path from the root to that node in the tree. Now, for $\ell \in [t+1]$, and for each node $y_{\boldsymbol{j}}$ in the ℓ-th tree, where $\boldsymbol{j}$ is shorthand for $(\ell, j_1, \ldots, j_m)$, we associate with that node a subset $S_{\boldsymbol{j}}$ of points from the reserve, and query the points $y_{\boldsymbol{j}} \oplus x$ for each x in $S_{\boldsymbol{j}}$. The set $S_{\boldsymbol{j}}$ is a subset of a specified cardinality of the set $S_{\boldsymbol{j}^{(-1)}}$, where $S_{\boldsymbol{j}^{(-1)}}$ is the set associated with the parent node of $y_{\boldsymbol{j}}$ in the ℓ-th tree. Finally, the algorithm queries a point z sampled uniformly and independently at random from $\{0,1\}^d$, and samples a uniformly random leaf $y_{\boldsymbol{j}}$ of a uniformly random tree $\ell \in [t+1]$. The set $S_{\boldsymbol{j}}$ associated with the leaf $y_{\boldsymbol{j}}$ has, by construction, $t+1$ points in it. All the points in $S_{\boldsymbol{j}}$, again, by construction, also belong to the sets $S_{\boldsymbol{j}'}$ associated with the nodes $y_{\boldsymbol{j}'}$ on the path from the root to the leaf $y_{\boldsymbol{j}}$ of the ℓ-th tree. Our algorithm queries $y_{\boldsymbol{j}'} \oplus z$ for all such nodes $y_{\boldsymbol{j}'}$. It then samples x uniformly at random from $S_{\boldsymbol{j}}$ and queries $x \oplus z$. Finally, it samples a uniformly random node $y_{\boldsymbol{j}'}$ on the path from the root to the leaf $y_{\boldsymbol{j}}$ and queries $x \oplus y_{\boldsymbol{j}'} \oplus z$. Observe that, by design, the point $x \oplus y_{\boldsymbol{j}'}$ has already been queried in an earlier step. The algorithm rejects if all the points involved in the sum $T_f(x, y_{\boldsymbol{j}'}, z)$ are nonerased and the triple $(x, y_{\boldsymbol{j}'}, z)$ violates quadraticity.

Algorithm 2 uses the following notation. For a vector $\boldsymbol{j} = (\ell, j_1, \ldots, j_m)$, where $m \in [0,t]$ and $\ell, j_1, \ldots, j_t \in [t+1]$, we use the convention that $\boldsymbol{j} = (\ell)$ for $m = 0$. For $k \in [0,m]$, let $\boldsymbol{j}^{(k)} = (\ell, j_1, \ldots, j_k)$ be the vector containing the first $k+1$ entries of $\boldsymbol{j}$. Finally, let $\boldsymbol{j}^{(-1)}$ be the vector $\boldsymbol{j}$ with its last entry removed. If $\boldsymbol{j} = (\ell)$, then $\boldsymbol{j}^{(-1)}$ is the empty vector $\emptyset$.

Algorithm 2 A t-Online-Erasure-Resilient Quadraticity Tester.

Require: $\varepsilon \in (0,1)$, access to function $f : \{0,1\}^d \to \{0,1\}$ via a t-online-erasure oracle.

Let $I = (t+1)^2(2t+1)^t$, $J = \frac{(t+1)^{(t+1)}-1}{t}$, and $\alpha = \min\left(\frac{\varepsilon}{2}, \frac{7}{(18 \cdot IJ(t+1))^{IJ(t+1)}}\right)$.
repeat $4c_t/\alpha$ times: ▷ c_t is a constant from Lemma 4.4 that depends only on t.
  for all $\ell \in [t+1]$:
    Query f at independent $x_1^{(\ell)}, \ldots, x_I^{(\ell)} \sim \{0,1\}^d$, and let $S_\emptyset = \{x_1^{(\ell)}, \ldots, x_I^{(\ell)}\}$.
    for all integer $m \in [0,t]$:
      for all $(j_1, j_2, \ldots, j_m) \in [t+1]^m$: ▷ When $m = 0$, the loop is run once.
        Let $\boldsymbol{j} = (\ell, j_1, \ldots, j_m)$ and query f at $y_{\boldsymbol{j}} \sim \{0,1\}^d$.
        ▷ When $m = 0$, $\boldsymbol{j} = (\ell)$.
        Let $S_{\boldsymbol{j}}$ be a uniformly random subset of $S_{\boldsymbol{j}^{(-1)}}$ of size $(t+1)(2t+1)^{t-m}$.
        Query $x \oplus y_{\boldsymbol{j}}$ for all $x \in S_{\boldsymbol{j}}$.
        Remove $S_{\boldsymbol{j}}$ from $S_{\boldsymbol{j}^{(-1)}}$.
  Sample $z \sim \{0,1\}^d$ and query f at z.
  Sample $\boldsymbol{j} = (\ell, j_1, \ldots, j_t) \sim [t+1]^{(t+1)}$ and query f at $y_{\boldsymbol{j}^{(m)}} \oplus z$ for all $m \in [0,t]$.
  Suppose $S_{\boldsymbol{j}} = \{x_1^{(\ell)}, x_2^{(\ell)}, \ldots, x_{t+1}^{(\ell)}\}$. Sample $i \sim [t+1]$ and query f at $x_i^{(\ell)} \oplus z$.
  Sample integer $m \sim [0,t]$ and query f at $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$.
  Reject if $T_f(x_i^{(\ell)}, y_{\boldsymbol{j}^{(m)}}, z) = 1$. ▷ All points needed for computing T_f are nonerased.
Accept.

Proof of Theorem 1.4. If f is quadratic, then Algorithm 2 always accepts. Suppose that f is ε-far from quadratic. Fix an adversarial strategy and a round of Algorithm 2. We show that Algorithm 2 rejects with probability $\Omega(\varepsilon)$ in this round.

Observe that Algorithm 2 makes queries of three types: *singletons* (of the form $x_i^{(\ell)}, y_{\boldsymbol{j}^{(m)}}$, and z), *doubles* (of the form $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}}$,$x_i^{(\ell)} \oplus z$, and $y_{\boldsymbol{j}^{(m)}} \oplus z$), and *triples* (of the form $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$), where $i \in [I], \boldsymbol{j} = (\ell, j_1, \ldots, j_t) \in [t+1]^{(t+1)}, m \in [0,t]$. We call points of the form $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}}$, $x_i^{(\ell)} \oplus z$, and $y_{\boldsymbol{j}^{(m)}} \oplus z$ *double decoys*, and points of the form $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$ *triple decoys*. We refer to double and triple decoys simply as decoys. Only some of the decoys become actual queries.

Let $\mathcal{G}$ denote the *good* event that, for the fixed round, all of the following hold:

- all singleton queries are successfully obtained;
- all double decoys of the form $x_i^{(\ell)} \oplus y_{\boldsymbol{j}^{(m)}}$ are nonerased right before $y_{\boldsymbol{j}^{(m)}}$ is queried;
- all double and triple decoys involving z (as well as $x_i^{(\ell)}$ and/or $y_{\boldsymbol{j}^{(m)}}$) are nonerased right before z is queried.

To lower bound the rejection probability of the algorithm, we consider the event that all triples of the form $(x_i^{(\ell)}, y_{\boldsymbol{j}^{(m)}}, z)$, where $i \in [I], \boldsymbol{j} = (\ell, j_1, \ldots, j_t) \in [t+1]^{(t+1)}, m \in [0,t]$, violate quadraticity, and all queries in the round are successfully obtained.

▶ **Definition 4.1** (Witness). *The singleton queries form a* witness *if* $T_f(x_i^{(\ell)}, y_{\boldsymbol{j}^{(m)}}, z) = 1$ *for all* $i \in [I], \boldsymbol{j} = (\ell, j_1, \ldots, j_t) \in [t+1]^{(t+1)}, m \in [0,t]$*, and, in addition, all singletons and all decoys are distinct.*

Let W be the event that the singleton queries form a witness. Let α be as defined in Step 1.

▶ **Lemma 4.2** (Probability of Successful Singleton Queries). *If* $f\colon \{0,1\}^d \to \{0,1\}$ *is* ε*-far from being quadratic, then* $\Pr[W \cap \mathcal{G}] \geq \alpha/2$.

In other words, Lemma 4.2 shows that for every adversarial strategy, with probability at least $\frac{\alpha}{2}$, the tester successfully obtains singleton queries that form a witness and, in addition, right before each singleton is queried, the decoys involving that singleton are nonerased. The proof of Lemma 4.2 (in Subsection 4.1) relies on the following key structural result about the fraction of large structures where all triples of a certain form violate quadraticity.

▶ **Lemma 4.3** (Probability of Singletons Forming a Violation Structure). *Let* $I, J, t \in \mathbb{N}$. *Suppose* $f\colon \{0,1\}^d \to \{0,1\}$ *is* ε*-far from being quadratic. For points* $x_i^{(\ell)}, y_j^{(\ell)}, z \sim \{0,1\}^d$*, where* $(i,j,\ell) \in [I] \times [J] \times [t+1]$*,*

$$\Pr\left[\bigcap_{i\in[I], j\in[J], \ell\in[t+1]} [T_f(x_i^{(\ell)}, y_j^{(\ell)}, z) = 1]\right] \geq \alpha, \tag{7}$$

where $\alpha = \min\left(\frac{\varepsilon}{2}, \frac{7}{(18\cdot IJ(t+1))^{IJ(t+1)}}\right)$.

We prove Lemma 4.3 in Subsection 4.2, building on a result of [2]. To prove Lemma 4.2, we use Lemma 4.3 with $I = (t+1)^2(2t+1)^t$ and $J = \sum_{m=0}^{t}(t+1)^m = \frac{(t+1)^{(t+1)}-1}{t}$, which is the number of nodes in each tree.

Next, in Lemma 4.4, we show that the probability of successfully obtaining all double and triple queries in the round, given the event $W \cap \mathcal{G}$, depends only on t. The proof of Lemma 4.4 appears in Subsection 4.3.

▶ **Lemma 4.4** (Probability of All Double and Triple Queries Being Nonerased). *The probability that all queries made in one round of Algorithm 2 are successfully obtained, conditioned on the event $W \cap \mathcal{G}$, is at least $1/c_t$, where c_t depends only on t.*

Let H denote the event that all queries in the round are successfully obtained, and recall that W is the event that the singleton queries form a witness. The probability that Algorithm 2 rejects in the given round is at least $\Pr[W \cap H]$. Note that

$$\Pr[W \cap H] \geq \Pr[W \cap H \cap \mathcal{G}] = \Pr[H \mid W \cap \mathcal{G}] \Pr[W \cap \mathcal{G}].$$

By Lemmas 4.2 and 4.4, we have that Algorithm 2 rejects with probability at least $\frac{1}{c_t} \cdot \frac{\alpha}{2}$ for a fixed round. Thus, after $\frac{4c_t}{\alpha}$ rounds, Algorithm 2 rejects with probability at least $\frac{2}{3}$. ◀

4.1 Proof of Lemma 4.2

In this section, we prove Lemma 4.2 that bounds from below the probability that the singleton queries are successfully obtained and form a witness, and, in addition, right before each singleton is queried, the decoys involving that singleton are nonerased. We show that if the fraction of large violation structures is at least α (Lemma 4.3), then, despite the adversarial erasures, the probability of successful singleton queries, as described above, is at least $\alpha/2$.

Proof of Lemma 4.2. The key idea of the proof is to keep track of *active* witnesses during the execution of the round. To simplify notation, let $K = (I + J)(t + 1) + 1$ and denote by $u_1, \ldots, u_K$ the singleton queries made by Algorithm 2 in the given round.

▶ **Definition 4.5** (Active witness). *For $i \in [K]$, let $u_1, \ldots, u_i$ denote the singleton queries made by Algorithm 2 up until a given timepoint of the fixed round. We say a witness $(v_1, \ldots, v_K) \in (\{0,1\}^d)^K$ is* i-active *if*

1. *The first i entries of the tuple are equal to $u_1, u_2, \ldots, u_i$.*
2. *All decoys involving u_j, where $j \leq i$, were nonerased right before u_j was queried.*

Furthermore, let A_i be a random variable denoting the number of active witnesses right after the i-th singleton query, and let B_i denote the number of active witnesses right before the i-th singleton query. Let B_1 denote the number of witnesses at the beginning of the round, such that all singletons and decoys for the witness are nonerased. Note that $\Pr[W \cap \mathcal{G}] = \Pr[A_K = 1] = \mathbb{E}[A_K]$. To lower bound $\mathbb{E}[A_K]$, we first show a lower bound on B_1, obtained in turn by a lower bound on the total fraction of witnesses. We then bound the difference between A_i and B_{i+1} and show a relationship between $\mathbb{E}[A_i]$ and $\mathbb{E}[B_i]$ for general i. All expectations in this proof are over the choice of singletons $u_1, \ldots, u_K \sim \{0,1\}^d$. The proofs of Claims 4.6-4.8 appear in the full version of our work [39].

▷ Claim 4.6. Let α be as defined in Lemma 4.3. If f is ε-far from being quadratic then

$$\Pr_{u_1,\ldots,u_K \sim \{0,1\}^d}[(u_1, \ldots, u_K) \text{ is a witness}] \geq \frac{7\alpha}{8}.$$

▷ Claim 4.7. For all adversarial strategies, $B_1 \geq \frac{3\alpha}{4} \cdot 2^{Kd}$.

▷ Claim 4.8. For all $i \in [K-1]$ and all adversarial strategies,

$$A_i - B_{i+1} \leq (t+1)^2(2t+1)^t \cdot |S| \cdot 2^{(K-1-i)d}.$$

▷ Claim 4.9. For all $i \in [K]$, it holds that $\mathbb{E}[A_i] = \frac{1}{2^d}\mathbb{E}[B_i]$.

Proof. For $v \in \{0,1\}^d$, let $B_{i,v}$ denote the number of witnesses that are active right before the i-th singleton query and whose i-th entry is equal to v. Then, $B_i = \sum_{v\in\{0,1\}^d} B_{i,v}$. Let $\mathbf{1}(v)$ be the indicator random variable for the event that the i-th singleton query is equal to v. Then,

$$\mathbb{E}[A_i] = \mathbb{E}\Big[\sum_{v\in\{0,1\}^d} B_{i,v}\mathbf{1}(v)\Big] = \mathbb{E}[\mathbf{1}(v)]\mathbb{E}\Big[\sum_{v\in\{0,1\}^d} B_{i,v}\Big] = \frac{1}{2^d}\mathbb{E}[B_i]. \qquad \lhd$$

We combine the claims above to complete the proof of the lemma. By Claim 4.9,

$$\begin{aligned}\mathbb{E}[A_K] &= \frac{1}{2^d}\mathbb{E}[B_K] = \frac{1}{2^d}\mathbb{E}[A_{K-1} + B_K - A_{K-1}] = \frac{1}{2^d}\mathbb{E}[A_{K-1}] - \frac{1}{2^d}\mathbb{E}[A_{K-1} - B_K] \\ &= \frac{1}{2^{2d}}\mathbb{E}[B_{K-1}] - \frac{1}{2^d}\mathbb{E}[A_{K-1} - B_K] = \cdots = \frac{1}{2^{Kd}}\mathbb{E}[B_1] - \sum_{i=1}^{K-1}\frac{\mathbb{E}[A_i - B_{i+1}]}{2^{(K-i)d}}.\end{aligned}$$

By Claim 4.7,

$$\frac{1}{2^{Kd}}\mathbb{E}[B_1] \geq \frac{3\alpha}{4}.$$

In addition, Claim 4.8 yields that

$$\begin{aligned}\sum_{i=1}^{K-1}\frac{\mathbb{E}[A_i - B_{i+1}]}{2^{(K-i)d}} &\leq \sum_{i=1}^{K-1}\frac{(t+1)^2(2t+1)^t\cdot|S|\cdot 2^{(K-1-i)d}}{2^{(K-i)d}} \\ &\leq \frac{K\cdot(t+1)^2(2t+1)^t\cdot|S|}{2^d} \leq \frac{\alpha}{4},\end{aligned}$$

where the last inequality holds for large enough d. We obtain that $\Pr[W\cap\mathcal{G}] = \mathbb{E}[A_K] \geq \frac{3\alpha}{4} - \frac{\alpha}{4} = \frac{\alpha}{2}$. ◀

4.2 Proof of Lemma 4.3

In this section, we prove Lemma 4.3 on the fraction of large violation structures for which all triples $(x_i^{(\ell)}, y_j^{(\ell)}, z)$, where $(i,j,\ell) \in [I]\times[J]\times[t+1]$, violate quadraticity. Our proof builds on a result of [2].

Proof of Lemma 4.3. Let η denote the fraction of violating triples for f, i.e.,

$$\eta := \Pr_{x,y,z\sim\{0,1\}^d}[T_f(x,y,z) = 1].$$

The distance of f to quadraticity, denoted by ε_f, is the minimum of $\Pr_x[f(x) \neq g(x)]$ over all quadratic functions g over the same domain as f. Using this notation, we state a result from [2] for the special case of quadraticity.

▷ Claim 4.10 (Theorem 1 of [2]). For all f, we have $\eta \geq \min(\frac{7}{3}\varepsilon_f, \frac{1}{40})$.

Let η' denote the left-hand side of Equation 7, that is, the probability that for $x_i^{(\ell)}, y_j^{(\ell)}, z \sim \{0,1\}^d$, all triples $(x_i^{(\ell)}, y_j^{(\ell)}, z)$ are violating, where $(i,j,\ell) \in [I]\times[J]\times[t+1]$. Claim 4.11 lower bounds η' in terms of η for all values of ε_f. Claim 4.12 lower bounds η' for small values of ε_f. We combine these results and use Claim 4.10 to conclude the proof of the lemma.

▷ Claim 4.11. For all f and points $x_i^{(\ell)}, y_j^{(\ell)}, z \sim \{0,1\}^d$, where $i\in[I], j\in[J], \ell\in[t+1]$, it holds that

$$\Pr\Big[\bigcap_{i\in[I],j\in[J],\ell\in[t+1]}[T_f(x_i^{(\ell)}, y_j^{(\ell)}, z) = 1]\Big] \geq \eta^{IJ(t+1)}.$$

Proof. The proof uses the Hölder's inequality as its key ingredient.

For $x_i^{(\ell)}, y_j^{(\ell)}, z \sim \{0,1\}^d$, let $T_{ij}^{(\ell)}$ be the event $T_f(x_i^{(\ell)}, y_j^{(\ell)}, z) = 1$. Then

$$\begin{aligned}
\Pr\Big[\bigcap_{i\in[I],j\in[J],\ell\in[t+1]} T_{ij}^{(\ell)}\Big] &= \sum_{u\in\{0,1\}^d} \Pr\Big[\bigcap_{i\in[I],j\in[J],\ell\in[t+1]} T_{ij}^{(\ell)} \mid z=u\Big]\Pr[z=u] \\
&= \frac{1}{2^d}\sum_{u\in\{0,1\}^d} \Pr\Big[\bigcap_{i\in[I],j\in[J],\ell\in[t+1]} T_{ij}^{(\ell)} \mid z=u\Big] \\
&= \frac{1}{2^d}\sum_{u\in\{0,1\}^d} \Pr\Big[\bigcap_{i\in[I],j\in[J]} T_{ij}^{(1)} \mid z=u\Big]^{t+1} \\
&\geq \Big(\frac{1}{2^d}\sum_{u\in\{0,1\}^d} \Pr\Big[\bigcap_{i\in[I],j\in[J]} T_{ij}^{(1)} \mid z=u\Big]\Big)^{t+1} \\
&= \Pr\Big[\bigcap_{i\in[I],j\in[J]} T_{ij}^{(1)}\Big]^{t+1},
\end{aligned}$$

where the first equality holds by the law of total probability; the third equality holds because, conditioned on z taking a specific value, the events $\bigcap_{i\in[I],j\in[J]} T_{ij}^{(\ell)}$ for $\ell \in [t+1]$ are independent and have the same probability; the inequality follows from the Hölder's inequality.

We use a similar argument to obtain

$$\Pr\Big[\bigcap_{i\in[I],j\in[J]} T_{ij}^{(1)}\Big] \geq \Big(\Pr\Big[\bigcap_{j\in[J]} T_{1j}^{(1)}\Big]\Big)^I,$$

where we condition on the values of $y_1^{(1)}, \ldots, y_J^{(1)}$, and z. Similarly, by conditioning on the values of $x_1^{(1)}$ and z, we obtain

$$\Pr\Big[\bigcap_{j\in[J]} T_{1j}^{(1)}\Big] \geq (\Pr[T_{11}^{(1)}])^J.$$

Since $\Pr[T_{11}^{(1)}] = \eta$, the claim follows. ◁

Next, we consider the case when ε_f is small. For $u_1, u_2, u_3 \in \{0,1\}^d$, let $\mathrm{span}(u_1, u_2, u_3)$ be the set of points $\bigoplus_{i\in T} u_i$ for $\emptyset \neq T \subseteq [3]$ and

$$S = \bigcup_{i\in[I],j\in[J],\ell\in[t+1]} \mathrm{span}(x_i^{(\ell)}, y_j^{(\ell)}, z).$$

The set S has $(I+J)(t+1)+1$ singletons, at most $I(J+1)(t+1)$ double sums, and at most $IJ(t+1)$ triple sums. Therefore, $|S| \leq 3IJ(t+1)$.

▷ Claim 4.12. Suppose $\varepsilon_f \leq \frac{1}{2|S|}$. Then $\eta' \geq \frac{\varepsilon_f}{2}$.

Proof. Let g be a closest quadratic function to f. Any two elements of S are uniformly and independently distributed in $\{0,1\}^d$. Then, for $x_i^{(\ell)}, y_j^{(\ell)}, z \sim \{0,1\}^d$, we have

$$\begin{aligned}
\eta' &\geq \Pr[f(z) \neq g(z) \text{ and } f(u) = g(u)\ \forall\, u \in S \setminus \{z\}] \\
&\geq \Pr[f(z) \neq g(z)] - \sum_{u\in S\setminus\{z\}} [f(z) \neq g(z) \text{ and } f(u) \neq g(u)] \\
&\geq \varepsilon_f - (|S|-1)\varepsilon_f^2 = \varepsilon_f(1 - (|S|-1)\varepsilon_f).
\end{aligned}$$

If $\varepsilon_f \leq \frac{1}{2|S|}$, then $1 - (|S|-1)\varepsilon_f \geq 1 - |S| \cdot \frac{1}{2|S|} \geq \frac{1}{2}$, which concludes the proof. ◁

Suppose $\frac{1}{2|S|} \leq \varepsilon_f \leq \frac{3}{7 \cdot 40}$. In this case, by Claims 4.10 and 4.11 and using the fact that $|S| \leq 3IJ(t+1)$, we have

$$\eta' \geq \Big(\frac{7}{3}\varepsilon_f\Big)^{IJ(t+1)} \geq \Big(\frac{7}{6 \cdot |S|}\Big)^{IJ(t+1)} \geq \frac{7}{(18 \cdot IJ(t+1))^{IJ(t+1)}}.$$

Finally, if $\varepsilon_f \geq \frac{1}{40}$, then again by Claims 4.10 and 4.11, we have $\eta' \geq \frac{1}{40^{(IJ(t+1))}}$. We have obtained that $\eta' \geq \min\big(\frac{\varepsilon}{2}, \frac{7}{(18 \cdot IJ(t+1))^{IJ(t+1)}}\big)$. ◀

4.3 Proof of Lemma 4.4

In this section, we prove Lemma 4.4, which shows that conditioned on the good event $W \cap \mathcal{G}$, all queries in one round of Algorithm 2 are successfully obtained. In particular, this probability only depends on the per-query erasure budget t.

Proof of Lemma 4.4. We first prove a lower bound on the probability that the queries made in Step 10 are successfully obtained.

▷ Claim 4.13. Conditioned on the event $W \cap \mathcal{G}$, the probability that in one execution of Step 10 at level $m \in [0, t]$, all queries in the step are successfully obtained is at least

$$\Big(\frac{1}{(t^2+t)(2t+1)^{t-m}+1}\Big)^{(t+1)(2t+1)^{t-m}}.$$

Proof. Fix the values of $\ell, m, j_1, \ldots, j_m$ for the given execution of Step 10 and let $\boldsymbol{j} = (\ell, j_1, \ldots, j_m)$. We can assume, without loss of generality, that Step 6 considers the tuples $(j_1, \ldots, j_m)$ in the lexicographic order. When $j_m = 1$, then right before Step 10 is executed, we have $|S_{\boldsymbol{j}^{(-1)}}| = (t+1)(2t+1)^{t-m+1}$. The size of the set $S_{\boldsymbol{j}^{(-1)}}$ decreases as the value of j_m increases, so that for $j_m = t+1$, we have

$$\begin{aligned}|S_{\boldsymbol{j}^{(-1)}}| &= (t+1)(2t+1)^{t-m+1} - t(t+1)(2t+1)^{t-m} \\ &= (t+1)(2t+1)^{t-m}(2t+1-t) = (t+1)^2(2t+1)^{t-m}.\end{aligned}$$

Thus, right before Step 10 is executed, the size of $S_{\boldsymbol{j}^{(-1)}}$ is at least $(t+1)$ times the size of the subset $S_{\boldsymbol{j}}$.

Let s denote the size of $S_{\boldsymbol{j}^{(-1)}}$ right before the execution of Step 9. In Step 9 we sample a uniformly random subset $S_{\boldsymbol{j}}$ of $S_{\boldsymbol{j}^{(-1)}}$ of size $s' = (t+1)(2t+1)^{t-m}$ and in Step 10 we query $x \oplus y_{\boldsymbol{j}}$ for all $x \in S_{\boldsymbol{j}}$. Conditioned on the event $W \cap \mathcal{G}$, all sums $x \oplus y_{\boldsymbol{j}}$, where $x \in S_{\boldsymbol{j}}$, are distinct and nonerased right before $y_{\boldsymbol{j}}$ is queried. On the i-th query of Step 10, the tester selects a uniformly random x out of $s-(i-1)$ elements. Right before the i-th query of the tester, the oracle can have erased at most ti of the sums $x \oplus y_{\boldsymbol{j}}$, for $x \in S_{\boldsymbol{j}^{(-1)}}$, since the adversary is not aware of the points belonging to $S_{\boldsymbol{j}}$ until their sums with $y_{\boldsymbol{j}}$ are queried in Step 10. Therefore, the probability that the tester successfully obtains the sum on its i-th query is at least $1 - \frac{ti}{s-i+1}$, where $i \in [s']$. We argued that, for all values of j_m, right before the execution of Step 10, it always holds that $s \geq (t+1)s'$. Therefore, the probability that the tester successfully obtains a sum is always positive. In particular, using the bound $s \geq (t+1)s'$, the probability that all queries in Step 10 are successfully obtained is at least

$$\prod_{i=1}^{s'}\Big(1 - \frac{ti}{s-i+1}\Big) \geq \Big(1 - \frac{ts'}{s-s'+1}\Big)^{s'} \geq \Big(\frac{1}{(t+1)s'-s'+1}\Big)^{s'} = \Big(\frac{1}{s't+1}\Big)^{s'}.$$

Substituting $s' = (t+1)(2t+1)^{t-m}$ concludes the proof. ◁

In Steps 4-12, the tester makes only singleton and double queries. Conditioned on $W \cap \mathcal{G}$, all singleton queries are successfully obtained. All double queries are made in Step 10. By Claim 4.13, the probability that all double queries in Step 10 are successfully obtained depends only on t.

Before z is queried in Step 12, conditioned on the event $W \cap \mathcal{G}$, all double and triple sums of the form $y_{\boldsymbol{j}^{(m)}} \oplus z$, $x \oplus z$, and $x \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$, where $m \in [0,t]$, $\boldsymbol{j} = (\ell, j_1 \ldots, j_t) \in [t+1]^{(t+1)}$, and $x \in S_{\boldsymbol{j}}$, are distinct and nonerased. After z is queried, the oracle can perform at most t erasures. Therefore, there exists $\ell \in [t+1]$, say $\ell = 1$, such that all double and triple sums $y_{\boldsymbol{j}^{(m)}} \oplus z$, $x \oplus z$, and $x \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$, where $m \in [0,t]$, $\boldsymbol{j} = (1, j_1 \ldots, j_t) \in [t+1]^{(t+1)}$, and $x \in S_{\boldsymbol{j}}$, are nonerased after z is queried. With probability $\frac{1}{t+1}$, the tester samples $\ell = 1$ in Step 13. By a similar reasoning, with probability $\frac{1}{(t+1)^t}$ the tester samples values of $j_1, \ldots, j_t$ in Step 13, say $j_1 = \cdots = j_t = 1$, such that all queries $y_{\boldsymbol{j}^{(m)}} \oplus z$, where $m \in [0,t]$, are successfully obtained. In addition, right before $y_{\boldsymbol{j}^{(t)}} \oplus z$ is queried, all sums of the form $x \oplus z$ and $x \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$, where $\boldsymbol{j} = (1, \ldots, 1, j_t) \in [t+1]^{(t+1)}$, $m \in [0,t]$, and $x \in S_{\boldsymbol{j}}$, are nonerased.

After $y_{\boldsymbol{j}^{(t)}} \oplus z$ is queried, the oracle performs at most t erasures, so there exists $x \in S_{\boldsymbol{j}}$ such that $x \oplus z$ and $x \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$, where $\boldsymbol{j} = (1, 1, \ldots, 1)$ and $m \in [0,t]$, are nonerased. With probability $\frac{1}{t+1}$, the tester samples this x in Step 14, and with probability $\frac{1}{t+1}$, it samples $m \in [0,t]$ such that the triple sum $x \oplus y_{\boldsymbol{j}^{(m)}} \oplus z$ is successfully obtained in Step 15. Thus, with probability $\frac{1}{(t+1)^{t+3}}$, all queries in Steps 13-15 are successfully obtained. Therefore the probability that all queries in one round of Algorithm 2 are successfully obtained, conditioned on $W \cap \mathcal{G}$, is $1/c_t$, where c_t is a constant depending only on t. ◀

References

1. Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007. doi:10.1002/rsa.20167.
2. Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005. doi:10.1109/TIT.2005.856958.
3. Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. doi:10.1007/s00493-003-0025-0.
4. Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016. doi:10.1007/s00453-015-9984-y.
5. László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 21–31, 1991. doi:10.1145/103418.103428.
6. László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. doi:10.1007/BF01200056.
7. Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos A. Kiwi, and Madhu Sudan. Linearity testing in characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996. doi:10.1109/18.556674.
8. Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM Journal on Computing (SICOMP)*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.
9. Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 294–304, 1993. doi:10.1145/167088.167174.

10 Mihir Bellare and Madhu Sudan. Improved non-approximability results. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 184–193, 1994. doi:10.1145/195058.195129.

11 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.

12 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.

13 Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 612–621, 2003. doi:10.1145/780542.780631.

14 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014. doi:10.1145/2591796.2591887.

15 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM Journal on Computing (SICOMP)*, 41(6):1380–1425, 2012. doi:10.1137/110826655.

16 Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of Reed-Muller codes. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 488–497, 2010. doi:10.1109/FOCS.2010.54.

17 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings, IEEE Conference on Computational Complexity (CCC)*, pages 309–320, 2014. doi:10.1109/CCC.2014.38.

18 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

19 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. *ACM Transactions on Algorithms (TALG)*, 13(2):20:1–20:30, 2017. doi:10.1145/3039241.

20 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013. doi:10.1145/2488608.2488661.

21 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.

22 Irit Dinur and Venkatesan Guruswami. PCPs via low-degree long code and hardness for constrained hypergraph coloring. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 340–349, 2013. doi:10.1109/FOCS.2013.44.

23 Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta. Testing the Lipschitz property over product distributions with applications to data privacy. In *Theory of Cryptography Conference (TCC)*, pages 418–436, 2013. doi:10.1007/978-3-642-36594-2_24.

24 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin Varma. Erasure-resilient property testing. *SIAM Journal on Computing (SICOMP)*, 47(2):295–329, 2018. doi:10.1137/16M1075661.

25 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.

26 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000. doi:10.1006/jcss.1999.1692.

27 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. doi:10.1145/226643.226652.

28 Eldar Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004. doi:10.1016/j.ic.2003.09.003.

29 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002. doi:10.1145/509907.509977.

30 Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Third Israel Symposium on Theory of Computing and Systems (ISTCS)*, pages 190–198, 1995. doi:10.1109/ISTCS.1995.377032.

31 Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 32–42, 1991. doi:10.1145/103418.103429.

32 Oded Goldreich. On multiple input problems in property testing. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 704–720, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.704.

33 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

34 Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 306–317, 2005. doi:10.1007/11538462_26.

35 Elad Haramaty, Amir Shpilka, and Madhu Sudan. Optimal testing of multivariate polynomials over small prime fields. *SIAM Journal on Computing (SICOMP)*, 42(2):536–562, 2013. doi:10.1137/120879257.

36 Johan Håstad and Avi Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Structures and Algorithms*, 22(2):139–160, 2003. doi:10.1002/rsa.10068.

37 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM Journal on Computing (SICOMP)*, 42(2):700–731, 2013. doi:10.1137/110840741.

38 Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. *Random Structures and Algorithms*, 35(2):163–193, 2009. doi:10.1002/rsa.20262.

39 Iden Kalemaj, Sofya Raskhodnikova, and Nithin Varma. Sublinear-time computation in the presence of online erasures. *CoRR*, abs/2109.08745, 2021.

40 Tali Kaufman, Simon Litsyn, and Ning Xie. Breaking the epsilon-soundness bound of the linearity test over GF(2). *SIAM Journal on Computing (SICOMP)*, 39(5):1988–2003, 2010. doi:10.1137/080715548.

41 Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing (SICOMP)*, 36(3):779–802, 2006. doi:10.1137/S0097539704445615.

42 Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 601–614, 2009. doi:10.1007/978-3-642-03685-9_45.

43 Amit Levi, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Erasure-resilient sublinear-time graph algorithms. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, pages 80:1–80:20, 2021. doi:10.4230/LIPIcs.ITCS.2021.80.

44 Alessandro Mantelero. The EU proposal for a general data protection regulation and the roots of the 'right to be forgotten'. *Computer Law and Security Review*, 29(3):229–235, 2013. doi:10.1016/j.clsr.2013.03.010.

45 Dana Moshkovitz. Low-degree test with polynomially small error. *Computational Complexity*, 26(3):531–582, 2017. doi:10.1007/s00037-016-0149-4.

46 Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. *SIAM Journal on Computing (SICOMP)*, 38(1):140–180, 2008. doi:10.1137/060656838.

47 Ilan Newman and Nithin Varma. New sublinear algorithms and lower bounds for LIS estimation. In *Proceedings, International Colloquium on Automata, Languages and Programming (ICALP)*, pages 100:1–100:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.100.

48 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions.

49 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Parameterized property testing of functions. *ACM Transactions on Computation Theory*, 9(4):17:1–17:19, 2018. doi:10.1145/3155296.

50 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. *Random Structures and Algorithms*, 2021. Preliminary version appeared in SODA 2020. doi:10.1137/1.9781611975994.123.

51 Ramesh Krishnan Pallavoor Suresh. *Improved Algorithms and New Models in Property Testing*. PhD thesis, Boston University, 2020.

52 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 6(72):1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.

53 Sofya Raskhodnikova. Testing if an array is sorted. *Encyclopedia of Algorithms*, pages 2219–2222, 2016. doi:10.1007/978-1-4939-2864-4_700.

54 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.

55 Sofya Raskhodnikova and Ronitt Rubinfeld. Linearity testing/testing Hadamard codes. In *Encyclopedia of Algorithms*, pages 1107–1110. Springer, 2016. doi:10.1007/978-1-4939-2864-4_202.

56 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.

57 Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997. doi:10.1145/258533.258641.

58 Noga Ron-Zewi and Madhu Sudan. A new upper bound on the query complexity of testing generalized Reed-Muller codes. *Theory of Computing*, 9:783–807, 2013. doi:10.4086/toc.2013.v009a025.

59 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing (SICOMP)*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

60 Michael E. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. *SIAM Journal on Computing (SICOMP)*, 46(2):774–823, 2017. doi:10.1137/130942152.

61 Alex Samorodnitsky. Low-degree tests at large distances. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 506–515, 2007. doi:10.1145/1250790.1250864.

62 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 191–199, 2000. doi:10.1145/335305.335329.

63 Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and PCPs. *SIAM Journal on Computing (SICOMP)*, 39(1):323–360, 2009. doi:10.1137/070681612.

64 Amir Shpilka and Avi Wigderson. Derandomizing homomorphism testing in general groups. *SIAM Journal on Computing (SICOMP)*, 36(4):1215–1230, 2006. doi:10.1137/S009753970444658X.

65 Madhu Sudan and Luca Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 18–27, 1998. doi:10.1109/SFCS.1998.743425.

66 Luca Trevisan. Recycling queries in PCPs and in linearity tests (extended abstract). In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 299–308, 1998. doi:10.1145/276698.276769.