Testing the Lipschitz Property over Product Distributions with Applications to Data Privacy[*]

Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta[**]

Pennsylvania State University, USA
{mxj201,kashyap,sofya,azg161}@cse.psu.edu

Abstract. In the past few years, the focus of research in the area of statistical data privacy has been in designing algorithms for various problems which satisfy some rigorous notions of privacy. However, not much effort has gone into designing techniques to computationally verify if a given algorithm satisfies some predefined notion of privacy. In this work, we address the following question: *Can we design algorithms which tests if a given algorithm satisfies some specific rigorous notion of privacy (e.g., differential privacy)?*

We design algorithms to test privacy guarantees of a given algorithm $\mathcal{A}$ when run on a dataset x containing potentially sensitive information about the individuals. More formally, we design a computationally efficient algorithm $\mathcal{T}_{priv}$ that verifies whether $\mathcal{A}$ satisfies *differential privacy on typical datasets* (DPTD) guarantee in time sublinear in the size of the domain of the datasets. DPTD, a similar notion to *generalized differential privacy* first proposed by [3], is a *distributional* relaxation of the popular notion of *differential privacy* [14].

To design algorithm $\mathcal{T}_{priv}$, we show a formal connection between the testing of privacy guarantee for an algorithm and the testing of the Lipschitz property of a related function. More specifically, we show that an efficient algorithm for testing of Lipschitz property can be used as a subroutine in $\mathcal{T}_{priv}$ that tests if an algorithm satisfies *differential privacy on typical datasets*.

Apart from formalizing the connection between the testing of privacy guarantee and testing of the Lipschitz property, we generalize the work of [21] to the setting of property testing under product distribution. More precisely, we design an efficient Lipschitz tester for the case where the domain points are drawn from hypercube according to some fixed but unknown product distribution instead of the uniform distribution.

1 Introduction

The trend towards data driven decision making has resulted in many commercial data sharing platforms like *BlueKai*, *TellApart* or *Criteo*. These platforms extensively collect and share user data with third-parties (e.g., advertisers) to enhance specific user experience (e.g., better behavioral targeting). Since the data which gets shared is extremely

[*] All omitted proofs appear in the full version [8].

[**] K.D. is supported in part by NSF Grant CCF-0964655. M.J. and S.R. are supported by NSF CAREER grant CCF-0845701 and NSF grant CDI-0941553. A.T. is supported in part by NSF Awards CCF-0747294, CDI-0941553 and US National Institutes of Health Clinical and Translational Science Award.

rich in user information, it poses serious privacy concerns about the users' information contained in the data [23,6]. A more cautious approach would be to require third-party clients to submit their algorithms (e.g. as binary executables or as programs) and run it "in-house" (i.e., within the data sharing platform itself) and only release the outputs of their algorithms. But what if the output of the algorithm itself reveals some private information? Fortunately, there are notions of privacy (e.g. *differential privacy* [14]) which impose strict *privacy requirements on the algorithm* computing on the data and guarantee that the output of the algorithm does not disclose too much information (provided the algorithm satisfies these requirements). There still remains the nagging question that these algorithms come from third-parties. How does one ensure that they have implemented their algorithms in a way which meet the specifications of the privacy-requirements?

One approach (e.g. [25,30,29]) that has been taken to address the above problem is to require clients to write their programs using a specific set of *trusted* built-in functions provided by the platform. The platform ensures (either statically or at run time) that the implementation complies by the rules of using only the built-in functions while operating on the private data. Another approach [21] based on *property reconstruction* allows *arbitrary* programs and uses the *global sensitivity* framework of [14] as the underlying privacy mechanism. However, this approach *provably* [1] requires prohibitively huge running time. Another approach [26] uses the algorithmic framework of [32,27] to allow arbitrary programs but the utility guarantees are limited by the guarantees of the framework.

In this work, we propose a new approach to the above problem which we call *privacy testing*. We do this by formulating the above problem in the well-studied framework of property testing [31,17]. Property testing is concerned with *approximately* deciding whether an input *object* (e.g. a graph or a function) satisfies a given property (e.g. connectivity or monotonicity). In the same spirit, we treat algorithm $\mathcal{A}$ as an object (e.g. as a family of functions) which is required to satisfy some fixed property, specifically, the property of being private under some well-defined notion of privacy. The goal is to design efficient algorithms which can test if $\mathcal{A}$ satisfies the privacy definition under consideration.

In this work, we design an algorithm $\mathcal{T}_{priv}$ to test if an untrusted algorithm $\mathcal{A}$ satisfies a *distributional relaxation* of the popular notion of *differential privacy* [10,14,11,12]. Roughly speaking, differential privacy guarantees that the output of an algorithm $\mathcal{A}$ does not depend "too much" on any particular record of the underlying dataset x. The distributional relaxation we adhere to in our work is called *differential privacy on typical datasets* (DPTD). DPTD ensures a similar guarantee as differential privacy, except that the guarantee is now only over *typical* data sets, namely, datasets with sufficiently high probability mass under a fixed data generating distribution. DPTD is a special case of *generalized differential privacy* from [3].

To test for DPTD, we show a new connection between differential privacy and the problem of testing the *Lipschitz property* of functions first studied by [21]. Informally, a function $f(x_1, \ldots, x_d)$ is Lipschitz if changing at most one input of f arbitrarily while keeping the other inputs fixed does not change the value of f drastically. For testing algorithm $\mathcal{A}$ for the property of being DPTD, we view $\mathcal{A}$ as a family of functions.

We show that testing DPTD reduces to testing the property that every function in the family is *simultaneously* Lipschitz. We allow $\mathcal{A}$ to be an arbitrary randomized algorithm (indeed, most privacy preserving algorithms are randomized) but, in this work, restrict it to have finite domain and range. While property testing algorithms usually only require black-box access to the object, in this exploratory work, we assume oracle access to values $\Pr[\mathcal{A}(x) = r]$ given arbitrary domain point x and range point r. (We discuss these assumptions in more detail in Section 7.)

Going beyond privacy testing, we show how to convert an arbitrary algorithm $\mathcal{A}$ into an algorithm which always satisfies DPTD. We test algorithm $\mathcal{A}$ for DPTD and only if the tester accepts, we allow it to be run on the private data. Details appear in Section 4.2.

Property testing of functions deals with algorithms which can distinguish between functions which satisfy a given property $\mathcal{P}$ (in our case, the Lipschitz property) from those which are *far* from the property. A function f is far from property $\mathcal{P}$ if the distance between f and every member of $\mathcal{P}$ (where we view $\mathcal{P}$ as the set of functions satisfying $\mathcal{P}$) is large under a suitable definition of distance between functions. In the standard property testing, the distance between functions f and g is given by $\Pr[f(x) \neq g(x)]$ where x is chosen uniformly from the domain. We refer to this as property testing under uniform distribution. While previous works [21,2,7] studied Lipschitz testing under uniform distribution, in this work we focus on the setting when the distribution on the underlying data set is an *unknown product* distribution. This is important to test the notion of DPTD for a large class of distributions (and not merely the uniform distribution). Property testing under unknown distribution is a well-studied area under the name of *distribution free testing* [19] with many positive and negative results [19,20]. In this work, we give the first Lipschitz tester which works for arbitrary unknown product distribution with nearly the same running time as the Lipschitz tester for the uniform distribution from [21].

1.1 Summary of Our Contributions

Formulate testing of data privacy property as Lipschitz property testing. In this paper we initiate the study of testing privacy properties of a given candidate algorithm $\mathcal{A}$. The specific privacy property that we test is *differential privacy on typical datasets* (DPTD) (see Definition 3.2). In order to design a tester for DPTD property, we cast the problem of testing DPTD property as a problem of testing the Lipschitz property. (See Theorem 4.1.) The problem of testing the Lipschitz property was initially proposed by [21].

Design a generic transformation to convert an algorithm $\mathcal{A}$ to its DPTD variant. We design a generic transformation to convert a candidate algorithm $\mathcal{A}$ to its DPTD variant. (See Theorem 4.2.)

Lipschitz testers over product distributions. In order to allow our privacy tester to be effective for a large class of data generating distributions, we extend the existing Lipschitz testers to work with product distributions. We give the first efficient tester for the Lipschitz property on the hypercube domain which works for arbitrary product distribution. (See Theorem 5.1.) Previous works [21,2,7] on Lipschitz testing focuses on the case of uniform distribution.

Concrete instantiation of privacy testers based on old and new Lipschitz testers. We instantiate privacy tester with the Lipschitz tester described in the previous item to get a concrete instantiation of the privacy tester. This also leads to a concrete instantiation of Item 2 mentioned above. We also instantiate privacy testers based on the state-of-the-art Lipschitz tester from [7] for the uniform distribution. This is summarized in Section 6.

1.2 Related Work

Recently, various notions of data privacy have been proposed such as k-anonymity [33], ℓ-diversity [24], differential privacy [14], noiseless privacy [4], generalized differential privacy [3] and natural differential privacy [5]. With known attacks (e.g. [15]) on k-anonymity and ℓ-diversity, privacy community has pretty much converged to theoretically sound notions of privacy like differential privacy. In this paper, we work with the definition of differential privacy on typical datasets (DPTD) (Definition 3.2). DPTD is a special case of generalized differential privacy (GDP), where we assume that the auxiliary information $\mathcal{A}ux$ in the GDP definition is all but one entry in the underlying dataset. The primary difference between GDP and the other related definitions is that it incorporates both the randomness in the underlying dataset x and the randomness of the algorithm $\mathcal{A}$, where as other notions (like noiseless privacy and differential privacy) consider either the randomness of the data or the algorithm.

[21] initiated the study of testing (and reconstruction) of the Lipschitz property. Subsequently, [2,7] gave Lipschitz testers with [7] being the current state-of-the-art for the Boolean hypercube domain. All these testers work for the uniform distribution on the domain. In our work, we allow arbitrary product distribution on the underlying domain. Thus, our work is closely related to the work done in the area of distribution free testing introduced by Goldreich et al [17]. (See also [19].) They noted that many graph properties have testers with query complexity independent of the input size when the points are drawn from the uniform distribution (e.g. bipartiteness, k-colorability etc.), but the distribution free testers for the same properties do not have query complexity independent of the input size. In contrast, Halevi and Kushilevitz gave a series of positive results for distribution-free testing in [20] and [19]. In particular, they proved that there are testers with time complexity independent of the domain size for the properties like sparse graph connectivity [19] and juntas, parities, low-degree polynomials and Boolean literals [20].

1.3 Organization of the Paper

In Section 2, we review the concepts of general property testing and provide the definition of the Lipschitz property testers that we use in our work. In Section 3, we show the connection between testing of differential privacy on typical datasets (DPTD) and Lipschitz property testing. Section 4 is dedicated to our main result giving the privacy tester (Section 4.1) and application of privacy tester in obtaining DPTD-algorithms (Section 4.2). In Section 5, we present our new Lipschitz property testers over product distributions on the hypercube domain. In Section 6, we instantiate privacy testers with Lipschitz testers. Lastly, in Section 7, we conclude with discussion about the limitations of our current approach and some open problems.

2 Preliminaries for Lipschitz Property Testing

Property testing [17,31] is concerned with distinguishing objects which satisfy a given property $\mathcal{P}$ from those which are *far* from satisfying it. When $\mathcal{P}$ is a property of functions $f : \mathcal{X}^d \to \mathbb{R}$ over a finite domain $\mathcal{X}^d$, distance to the property is usually measured in terms of the fraction of points in the domain $\mathcal{X}^d$ on which f must be modified in order to satisfy the property. A more general notion of distance is defined with respect to a probability distribution on the domain $\mathcal{X}^d$.

Definition 2.1 (Distance to a property). *Let $\mathcal{P}$ be a property (i.e., a set) of functions $f : \mathcal{X}^d \to \mathbb{R}$. Let Π be a distribution on $\mathcal{X}^d$. The* distance $dist_\Pi(f, g)$ *between functions $f, g : \mathcal{X}^d \to \mathbb{R}$ (with respect to the distribution Π) is* $\Pr_{x \sim \Pi}[f(x) \neq g(x)]$. *The* distance $dist_\Pi(f, \mathcal{P})$ *of a function f from the property $\mathcal{P}$ is* $\min_{g \in \mathcal{P}} dist_\Pi(f, g)$. *We say that f is* ϵ-far *from the property $\mathcal{P}$ if $dist_\Pi(f, \mathcal{P}) \geq \epsilon$.*

In this work, we study the Lipschitz property of functions, first considered in the context of property testing by [21].

Definition 2.2. *A real-valued function $f : \mathcal{X}^d \to \mathbb{R}$ is* c-Lipschitz *if $|f(x) - f(y)| \leq c \cdot d_H(x, y)$ where $d_H(x, y)$ is the Hamming distance between x and y, that is, the number of coordinates in which x an y differ. We say f is* Lipschitz *if f is 1-Lipschitz.*

Next we define a Lipschitz tester. Our definition differs from the standard definition of a property tester (e.g., as used in [21]) in two aspects: (i) we only require a Lipschitz tester to distinguish Lipschitz functions from functions which are ϵ-far from $(1 + \delta)$-Lipschitz for some fixed $\delta \geq 0$ and (ii) we measure the distance between functions with respect to a fixed probability distribution Π on the domain. The relaxation of Condition (i) has been considered earlier e.g. in [28] for the property of having small diameter and in [21] for the Lipschitz property. The generalization of Condition (ii) is well-studied in property testing under the name *distribution free testing*. See e.g. [19]. We remark that setting $(1 + \delta) = 1$, $\rho = 1/3$ and Π to be the uniform distribution on $\mathcal{X}^d$ in the definition below recovers the standard definition of property tester (in our case, the Lipschitz tester as defined in [21]).

Definition 2.3 (Approximate Lipschitz Tester). *A $(1+\delta)$-approximate Lipschitz tester $\mathcal{T}_{Lip}(\epsilon, \rho, \delta, d)$ is a randomized algorithm that gets as input: (i) oracle access to function $f : \mathcal{X}^d \to \mathbb{R}$; (ii) oracle access to independent samples from distribution Π on $\mathcal{X}^d$ and (iii) a few parameters, namely, the* proximity parameter ϵ, *the* error probability parameter ρ, *the* approximation parameter δ *and the parameter d. The tester provides the following guarantee. If f is Lipschitz, then the algorithm accepts. If f is ϵ-far from the $(1 + \delta)$-Lipschitz property with respect to distribution Π, then with probability at least $1 - \rho$, the algorithm rejects.*

We say the Lipschitz property can be tested with *one-sided* error if there exists a Lipschitz tester as defined above. Further, we say it can be tested *nonadaptively* if all the queries made by $\mathcal{T}_{Lip}$ to its oracles are made in advance without the knowledge of answers to the previous queries.

3 Differential Privacy and Its Connections to Testing the Lipschitz Property

Intuitively, the output of a differentially private algorithm is almost the same whether or not a specific person's data is present in the dataset. Datasets are modeled as fixed-length vectors from an arbitrary domain $\mathcal{X}^d$, where each coordinate represents one person's data. (For example, when $\mathcal{X}^d$ is $\{0,1\}^d$, datasets consist of d-bit vectors and each person's data is a Boolean value.) An algorithm is differentially private if it has similar distribution on outputs when run on datasets which are close in Hamming distance. The Hamming distance between x and x', denoted $d_H(x, x')$, is the number of coordinates on which x and x' differ.

Definition 3.1 ((α,β)-Differential Privacy [14,13]). *A randomized algorithm $\mathcal{A}$ is (α,β)-differentially private if for any two datasets x and x' in $\mathcal{X}^d$, and for all measurable sets $\mathcal{Z} \subseteq Range(\mathcal{A})$, the following holds:*

$$\Pr[\mathcal{A}(x) \in \mathcal{Z}] \leq e^{\alpha \cdot d_H(x,x')} \Pr[\mathcal{A}(x') \in \mathcal{Z}] + \beta. \tag{1}$$

If $\beta = 0$, algorithm $\mathcal{A}$ is called α-differentially private.

In this work, we focus on differentially private algorithms which output values in a *finite* range space $\mathcal{Z}$. Such algorithms have a clean characterization in terms of the Lipschitz property. Specifically, define functions $f_z : x \to \mathbb{R}$ for every $z \in \mathcal{Z}$ by setting $f_z(x) = \log \Pr[\mathcal{A}(x) = z]$. We make the following simple but important observation.

Observation 3.1 (Differential Privacy as a Lipschitz Condition). Algorithm $\mathcal{A}$ is α-differentially private if and only if for every $z \in \mathcal{Z}$, function f_z is α-Lipschitz.

Therefore, one could check if an algorithm specified by functions f_z is differentially private, given oracle access to these functions, if one could design a procedure that decides if an input function is Lipschitz. However, as noted in [21], deciding if a given function is Lipschitz is NP-hard. Can we still efficiently check for some relaxation of differential privacy? Towards answering this question, we take motivation from relaxations of differential privacy considered in the literature based on *distributional* assumptions. Specifically, we adapt a particular relaxation from [3] and show that it can indeed be tested using a connection to testing the Lipschitz property. The relaxation we consider assumes that datasets come from some fixed distribution Π on the set of all datasets. The notion of privacy is relaxed from the worst case guarantee over all pairs of datasets (i.e., differential privacy) to a notion where the differential privacy condition is required to hold only on datasets which are more likely to occur (i.e., have high-probability mass under distribution Π). We refer to this notion as *differential privacy on typical datasets* (DPTD) (Definition 3.2). As mentioned earlier, DPTD is an adaptation of more general definition introduced in [3] under the name of *generalized differential privacy* (GDP). GDP was defined in the context of a related distributional notion of privacy called *noiseless privacy*, first introduced by [4]. A related notion called *natural differential privacy* has also been recently proposed by [5]. Since we focus on DPTD in this work, we do not discuss noiseless privacy (and its variants) further. Next we give a formal definition of

DPTD. The definition is parametrized by three parameters α, β and γ. The parameters α and β play the same role as in the differential privacy definition, while the parameter γ bounds the probability of the "bad" set B of databases on which the differential privacy condition fails to hold.

Definition 3.2 ((α, β, γ)-Differential Privacy on Typical Datasets (DPTD)). *Let Π be a fixed distribution on the domain $\mathcal{X}^d$ of datasets. A randomized algorithm $\mathcal{A}$ is (α, β, γ)-differentially private on typical datasets, if there exists a subset $B \subseteq \mathcal{X}^d$ satisfying $\Pr_{x \sim \Pi}[x \in B] \le \gamma$ such that condition (1) of Definition 3.1 holds for any two datasets $x, x' \in \mathcal{X}^d \setminus B$ and all measurable sets $\mathcal{Z} \subseteq Range(\mathcal{A})$. The probability in (1) is over the randomness of the algorithm $\mathcal{A}$.*

Our main observation is that for algorithms which output values in a finite range, testing DPTD can be reduced to testing the Lipschitz property (of a family of functions). Assume again that the output space of $\mathcal{A}$ is a finite set $\mathcal{Z}$ and define functions f_z as above.

Observation 3.2 (DPTD for Algorithms with Finite Range). Algorithm $\mathcal{A}$ is $(\alpha, 0, \gamma)$-differentially private if and only if the following two conditions hold: (i) there exists a subset $B \subseteq \mathcal{X}^d$ such that $\Pr_{x \sim \Pi}[x \in B] \le \gamma$; and (ii) for every $z \in \mathcal{Z}$, function f_z is α-Lipschitz on the set $\mathcal{X}^d \setminus B$.

Recall that a function f is ϵ-close to property $\mathcal{P}$ with respect to distribution Π if there is a function g which satisfies $\mathcal{P}$ and $\Pr_{x \sim \Pi}[f(x) \neq g(x)] \le \epsilon$. Observation 3.2, in particular, implies the following. If algorithm $\mathcal{A}$ satisfies $(\alpha, 0, \gamma)$-DPTD, then for every $z \in \mathcal{Z}$, function f_z is γ-close to the α-Lipschitz property with respect to the distribution Π. However, to apply a Lipschitz tester, we need a converse of this statement. The following lemma gives the converse.

Lemma 3.1 (Connection between DPTD and Testing the Lipschitz Property). *If for every $z \in R$, function f_z is ϵ_z-close to the α-Lipschitz property with respect to the distribution Π, then $\mathcal{A}$ is $(\alpha, 0, \sum_z \epsilon_z)$-DPTD. In particular, if $\mathcal{A}$ is* not *$(\alpha, 0, \gamma)$-DPTD, then there exists $z \in \mathcal{Z}$ such that f_z is $\gamma/|\mathcal{Z}|$-far from the α-Lipschitz property.*

Proof. Since every f_z is ϵ_z-close to the α-Lipschitz property with respect to the distribution Π, there exists B_z corresponding to each f_z such that (i) f_z is α-Lipschitz on $\mathcal{X}^d \setminus B_z$; and (ii) $Pr_{x \sim \Pi}[x \in B_z] \le \epsilon_z$. Let B be the union over all z of the sets B_z. Applying the union bound, we get $\Pr_{x \sim \Pi}[x \in B] \le \sum_z \epsilon_z$. Then the first part of the lemma follows from Observation 3.2 with B as the required set. The second part of the lemma follows from an averaging argument. □

3.1 Discussion of Differential Privacy on Typical Datasets

Differential privacy on typical datasets (DPTD) in Definition 3.2 is very similar to the definition of differential privacy, except in DPTD there exists a set of datasets B where the differential privacy condition (i.e., Equation 1 of Definition 3.1) does not hold. Moreover, the probability mass of B under the data generating distribution Π is at most γ. If we assume $\beta = 0$ for simplicity, then at a high-level DPTD implies that for any two datasets x and x' from the set $\mathcal{X}^d \setminus B$, which have sufficient probability mass under Π and differ in k-entries, the distribution of $\mathcal{A}(x)$ and $\mathcal{A}(x')$ have a statistical distance of $2k\alpha$ when $k\alpha$ is less than one.

In differential privacy, the scale of the parameters α and β are typically chosen as follows: α is chosen to be some small constant and β is chosen to be $O(1/d^2)$. With this choice of parameters, differential privacy ensures that *even in the presence of any auxiliary information, from the output of the algorithm $\mathcal{A}$, an adversary draws the same conclusions about any entry in the data set irrespective of its presence or absence.* (See [22] for more discussion.) Since α and β play the same role in DPTD, we think of α and β of the same order as discussed for differential privacy. Additionally, throughout this paper we think of γ and β to be of the same order.

In [3], a generalization of DPTD has been stated under the name of *generalized differential privacy* (GDP). Under suitable choice of auxiliary information (i.e., the random variable $\mathcal{A}ux$), the definition of [3] reduces to Definition 3.2. More precisely, for every data entry x_i, the auxiliary information in the GDP condition (of the definition of GDP) corresponds to all entries in the data set x except x_i.

4 Testing and Reconstruction of Differential Privacy on Typical Datasets

In this section we present an algorithm $\mathcal{T}_{priv}$ (Algorithm 1) which "tests" whether a given algorithm $\mathcal{A}$ is private. The guarantee of the testing algorithm $\mathcal{T}_{priv}$ is "asymmetric", i.e., if algorithm $\mathcal{A}$ is α-differentially private, then the tester accepts, and if the algorithm $\mathcal{A}$ is not $(\alpha, 0, \gamma)$-DPTD, then the tester rejects with high probability. It is worth highlighting that this style of utility guarantee deviates from the conventional utility guarantees in property testing literature (where the utility guarantee is symmetric over a particular property $\mathcal{P}$).

Next we use $\mathcal{T}_{priv}$ as a subroutine to design a "reconstruction" algorithm $\mathcal{A}_{\boldsymbol{privGen}}$ (Algorithm 2). The algorithm $\mathcal{A}_{\boldsymbol{privGen}}$ is guaranteed to be differential private on typical datasets under the data generating distribution Π. Moreover, if $\mathcal{A}$ is differentially private, then the output of $\mathcal{A}_{\boldsymbol{privGen}}$ equals the output of $\mathcal{A}$. (See Theorem 4.2 for the exact parameters.)

4.1 Tester for Differential Privacy on Typical Datasets

In this section, we prove the following theorem.

Theorem 4.1 **($(\alpha, \beta, \gamma, \delta)$-Privacy testing).** *Let $\mathcal{A}$ be a randomized algorithm which outputs values in the finite set $\mathcal{Z}$. Let Π be a data generating distribution. Let $\mathcal{T}_{Lip}$ be a δ-approximate Lipschitz tester. Suppose there is an oracle $\mathcal{O}_{\mathcal{A}}$ which for every value $z \in \mathcal{Z}$ and for every $x \in \mathcal{X}^d$ allows constant time access to the value $\Pr(\mathcal{A}(x) = z)$ (where the probability is only over the randomness of the algorithm $\mathcal{A}$). Then algorithm $\mathcal{T}_{priv}$ (Algorithm 1), given access to $\mathcal{O}_{\mathcal{A}}$, satisfies the following.*

– *If algorithm $\mathcal{A}$ is α-differentially private, then $\mathcal{T}_{priv}$ accepts.*
– *If algorithm $\mathcal{A}$ is not $(\alpha(1+\delta), 0, \gamma)$-DPTD, then the tester rejects with probability at least $1-\beta$.*

The algorithm $\mathcal{T}_{priv}$ uses $\mathcal{T}_{Lip}$ as a subroutine and runs in time

$$O(|\mathcal{Z}| \cdot \textit{Run-time}\left(\mathcal{T}_{Lip}(\frac{\gamma}{|\mathcal{Z}|}, \beta, \delta, d)\right).$$

At high level, algorithm $\mathcal{T}_{priv}$ (Algorithm 1) does the following. For each possible output $z \in \mathcal{Z}$, it defines a function f_z (with the domain $\mathcal{X}^d$). It then invokes algorithm $\mathcal{T}_{Lip}$ to test f_z for the Lipschitz property. It accepts iff $\mathcal{T}_{Lip}$ accepts all f_z.

Algorithm 1 $\mathcal{T}_{priv}$: Tester of Differential Privacy on Typical Data Sets

```
Require: Algorithm $\mathcal{A}$, data generating distribution $\Pi$, data domain $\mathcal{X}^d$, output range $\mathcal{Z}$, privacy
    parameters $\alpha, \gamma \in (0,1]$, failure probability $\beta \in (0,1]$ and approximation parameter $\delta$.
1: Let $\mathcal{T}_{Lip}$ be a $\delta$-approximate Lipschitz tester defined in Definition 2.3.
2: for all values $z \in \mathcal{Z}$ do
3:     Define function $f_z : \mathcal{X}^d \to \mathbb{R}$ by setting $f_z(x) = \frac{1}{\alpha}\log \Pr(\mathcal{A}(x) = z)$.
4:     Run $\mathcal{T}_{Lip}$ on $f_z$ with proximity parameter $\frac{\gamma}{|\mathcal{Z}|}$ and failure probability $\beta$.
5:     If $\mathcal{T}_{Lip}$ rejects, then reject.
6: end for
7: Accept.
```

Proof. We use Lemma 3.1 and Observation 3.1 to prove the theorem. To prove the first item, assume $\mathcal{A}$ is α-differentially private. Then Observation 3.1 implies that for every $z \in \mathcal{Z}$, function f_z (in Line 4 of Algorithm 1) is Lipschitz. Since $\mathcal{T}_{priv}$ always accepts a Lipschitz function, we get that $\mathcal{T}_{priv}$ accepts, as required. For the second item, assume $\mathcal{A}$ is *not* $(\alpha, 0, \gamma)$-DPTD. Then Lemma 3.1 implies that there exists z^* such that f_{z^*} is $\gamma/|\mathcal{Z}|$-far from being $(1+\delta)$-Lipschitz. From definition of $\mathcal{T}_{Lip}$, it follows that $\mathcal{T}_{Lip}$ rejects f_{z^*} with probability at least $1-\beta$, and therefore, so does $\mathcal{T}_{priv}$. The running time of $\mathcal{T}_{priv}$ follows from the fact that the tester $\mathcal{T}_{Lip}$ is invoked at most $|\mathcal{Z}|$ times. □

The proof above used the second part of Lemma 3.1 and used an *arbitrary* Lipschitz tester $\mathcal{T}_{Lip}$. It is possible to obtain faster privacy testers using the (stronger) statement given in the first part of Lemma 3.1. This requires making mild assumptions about the guarantees of the Lipschitz tester. We defer this analysis to the full version.

4.2 Application of DPTD Tester to Ensure Privacy of a Given Candidate Algorithm

In this section we will demonstrate how one can use algorithm $\mathcal{T}_{priv}$ (Algorithm 1) designed in the previous section to guarantee (α, β, γ)-differential privacy on typical datasets to the output produced by a candidate algorithm $\mathcal{A}$. The details are given in Algorithm 2.

```
Algorithm 2 $\mathcal{A}_{privGen}$: DPTD mechanism
Require: Dataset $x$, candidate algorithm $\mathcal{A}$, testing algorithm $\mathcal{T}_{priv}$, data generating distribution
    $\Pi$, data domain $\mathcal{X}^d$, output set $\mathcal{Z}$, privacy parameters $\alpha, \beta, \gamma$
1: Run $\mathcal{T}_{priv}$ with parameters $\mathcal{A}, \Pi, \mathcal{X}^d, \mathcal{Z}$, privacy parameters $\alpha, \gamma$, and failure parameter $\beta$
2: If $\mathcal{T}_{priv}$ accepts, then output $\mathcal{A}(x)$. Otherwise, output FAIL.
```

The guarantees for Algorithm 2 are given below.

Theorem 4.2 ($((1+\delta), \alpha, \beta, \gamma)$**-DPTD mechanism).** *Let $\mathcal{T}_{Lip}$ be a $(1+\delta)$-approximate Lipschitz tester (see Definition 2.3) used in the testing algorithm $\mathcal{T}_{priv}$ (Algorithm 1). Under the assumptions of Theorem 4.1, algorithm $\mathcal{A}_{privGen}$ (Algorithm 2) satisfies:*

– ***(privacy)*** *Algorithm $\mathcal{A}_{privGen}$ (Algorithm 2) is $(\alpha(1+\delta), \beta, \gamma)$-DPTD.*
– ***(utility)*** *If the candidate algorithm $\mathcal{A}$ is α-differentially private, then the output distributions of algorithm $\mathcal{A}_{privGen}$ (Algorithm 2) and $\mathcal{A}$ are identical.*

We defer the proof of this theorem to the full version.

5 Lipschitz Property Testing on the Hypercube under Product Distribution

In this section, we present a δ-approximate Lipschitz tester (see Definition 2.3) for functions defined on $\mathcal{X}^d = \{0,1\}^d$ when the underlying distribution on $\{0,1\}^d$ is an *unknown product* distribution. Specifically, the points in the dataset are distributed according to the product distribution $\Pi = Ber(p_1) \times Ber(p_2) \times ..., \times Ber(p_d)$ where $Ber(p)$ denotes the Bernoulli distribution with probability p. Namely, $Ber(p)$ is 1 with probability p and 0 with probability $1-p$. Therefore, each vertex in $x \in \{0,1\}^d$ has an associated probability mass $p_x = \prod_{i \in [d]} p_i^{x_i}(1-p_i)^{1-x_i}$.

In this section, we view the domain $\{0,1\}^d$ as vertices of the hypercube graph $\mathcal{H}_d = (\{0,1\}^d, E)$. The edge set E consists of pairs $\{x, y\}$ of vertices $x, y \in \{0,1\}^d$ which differ in exactly one coordinate (i.e., there exists $i \in [d]$ such that $x_i = y_i$ and for all $j \neq i$, $x_j = y_j$). Observe that f is Lipschitz on $\{0,1\}^d$ if and only if for every edge $\{x, y\} \in E$, the following holds: $|f(x) - f(y)| \leq 1$. An edge which does not satisfy this condition is called a *violated edge*.

5.1 Algorithm for Testing the Lipschitz Property on the Hypercube

In this section, we prove the following theorem which gives a 1-approximate Lipschitz tester for $\delta\mathbb{Z}$-valued functions. A function is $\delta\mathbb{Z}$ valued if it produces outputs in integral multiples of δ. The running time of our tester is stated in terms of the *image diameter* of the input function f.

Definition 5.1 (Image Diameter). *The image diameter of a function $f : \mathcal{X}^d \to \mathbb{R}$, denoted by $ImD(f)$, is the difference between the maximum and the minimum values attained by f, i.e.,* $\max_{x \in \mathcal{X}^d} f(x) - \min_{x \in \mathcal{X}^d} f(x)$.

Theorem 5.1. *Let $\{0,1\}^d$ be the domain from which the dataset are drawn according to a product probability distribution $\Pi = Ber(p_1) \times Ber(p_2) \times ..., \times Ber(p_d)$. The Lipschitz property of functions $f : \{0,1\}^d \to \delta\mathbb{Z}$ on these datasets can be tested non-adaptively and with one sided error probability ρ in $O(\frac{d \cdot \min\{d, ImD(f)\}}{\delta(\epsilon - d^2\delta)} \ln(\frac{2}{\rho}))$ time for $\delta \in (0, \frac{\epsilon}{d^2})$. Here ImD is the image diameter defined in Definition 5.1.*

By discretizing, as in proof of Corollary 1.2 in [21], we obtain a $(1+\delta)$-approximate Lipschitz tester for real-valued functions.

Corollary 5.1. *Let $\{0,1\}^d$ be the domain from which the dataset are drawn according to a product probability distribution $\Pi = Ber(p_1) \times Ber(p_2) \times ..., \times Ber(p_d)$. There is an algorithm that on input parameters $\delta \in (0, \frac{\epsilon}{d^2}), \epsilon \in (0,1), d$ and oracle access to a function $f : \{0,1\}^d \to \mathbb{R}$ has the following behavior: It accepts if f is Lipschitz and rejects with probability at least $1 - \rho$ if f is ϵ-far (with respect to the distribution Π) from $(1+\delta)$-Lipschitz and runs in $O(\frac{d \cdot \min\{d, ImD(f)\}}{\delta(\epsilon - d^2\delta)} \ln(\frac{2}{\rho}))$ time. Here ImD is the image diameter defined in Definition 5.1.*

Theorem 5.1 is proved in Section 5.2. To state the proof we need the following technical result stated in lemma 5.1.

We define a distribution D_E on edges of the hypercube where the probability mass of an edge $\{x, y\}$ is given by $\frac{p_x + p_y}{d}$. Note that $\sum_{(x,y) \in E(H_d)} \frac{(p_x + p_y)}{d} = 1$. Thus, D_E is well-defined. Our tester is based on detecting violated edges (that is, edges which violate the Lipschitz property) sampled from distribution D_E. Our main technical lemma (Lemma 5.1) gives a lower bound on the probability of sampling a violated edge according to distribution D_E for a function that is ϵ-far from Lipschitz. (Recall that ϵ-far is measured with respect to the distribution Π.)

Lemma 5.1. *Let function $f : \{0,1\}^d \to \delta\mathbb{Z}$ be ϵ-far from Lipschitz. Let $V(f)$ denote the set of edges in $\mathcal{H}_d$ violated by f. Then*

$$\sum_{(x,y) \in V(f)} \frac{(p_x + p_y)}{d} \geq \frac{\delta(\epsilon - d^2\delta)}{d \cdot ImD(f)}$$

Here ImD is the image diameter defined in Definition 5.1.

We prove the above lemma in section 5.3.

5.2 Lipschitz Tester

In this section we prove Theorem 5.1 and Corollary 5.1. We first present the algorithm stated in Theorem 5.1.

Algorithm 3 Lipschitz Tester

```
Require: Data domain $\{0,1\}^d$, product distribution on dataset $\Pi = Ber(p_1) \times Ber(p_2) \times ... \times Ber(p_d)$, failure probability $\rho$, proximity parameter $\epsilon'$, discretization parameter $\delta$.
1: Set $\epsilon = \epsilon' - d^2\delta$.
2: Sample $t = \left\lceil \frac{2}{\epsilon} \ln(\frac{2}{\rho}) \right\rceil$ vertices $z_1, z_2, ..., z_t$ independently from $\mathcal{H}_d$ according to the distribution $\Pi$.
3: Let $r = \max_{i=1}^t f(z_i) - \min_{i=1}^t f(z_i)$.
4: If $r > d$, reject.
5: Sample $\left\lceil \frac{dr}{\delta\epsilon} \ln(\frac{2}{\rho}) \right\rceil$ edges independently with each edge $(x, y)$ picked with probability $\frac{(p_x+p_y)}{d}$ from the hypercube $\mathcal{H}_d$.
6: If any of the sampled edges is violated, then reject, else accept.
```

Proof (of Theorem 5.1). First observe that if input function f is Lipschitz then Algorithm 3 always accepts. This is because a Lipschitz function f has image diameter (see Definition 5.1) at most d and hence cannot be rejected in Step 4. Moreover, it does not have any violated edges and hence cannot be rejected in Step 6. Next consider the case when f is ϵ-far from Lipschitz. Towards this we first extend Claim 3.1 of [21] about sample diameter r to our setting where the distance (in particular, the notion of ϵ-far) is measured with respect to a product distribution.

Claim. Value r computed on Line 3 is at most $ImD(f)$ and with probability at least $1 - \frac{\rho}{2}$, f is ϵ-close to having diameter at most r.

Proof. Sort the points in $\{0,1\}^d$ according to their function values in non-decreasing order. Let L be the first ℓ-points such that their *probability mass* sums up to $\frac{\epsilon}{2}$ and R be the set of last ℓ' points such that their *probability mass* sums up to $\frac{\epsilon}{2}$. The rest of the proof is very similar to the proof of Claim 3.1 in [21], so we omit the details here. □

Having established Claim 5.2, the rest of the proof of Theorem 5.1 is identical to [21]. We omit the details. □

5.3 Repair Operator and Proof of Lemma 5.1

We show a transformation of an arbitrary function $f : \{0,1\}^d \to \delta\mathbb{Z}$ into a Lipschitz function by changing f on certain points, whose probability mass is related to the probability mass (with respect to D_E) of the violated edges of $\mathcal{H}_d$. This is achieved by repairing one dimension of $\mathcal{H}_d$ at a time as explained henceforth. To achieve this, we define an **asymmetric** version of the basic operator of [21]. The operator redefines function values so that it reduces the gap asymmetrically according to the Hamming weights (and probability masses in-turn) of the endpoints of the violated edge. This is the main difference from previous approaches ([21], [2]) which do not work if applied directly, because of the varying probability masses of the vertices with respect to the Hamming weight, defined as $|x|$ for a vertex x. We first define the building block of the repair operator which is called the asymmetric basic operator.

Definition 5.2 (Asymmetric basic operator). *Given $f : \{0,1\}^d \to \delta\mathbb{Z}$, for each violated edge $\{x,y\}$ along dimension i, where $f(x) < f(y) - 1$, define B_i as follows.*

1. *If $|x| > |y|$, then $B_i[f](x) = f(x) + (1 - p_i)\delta$ and $B_i[f](y) = f(y) - p_i\delta$*
2. *If $|x| < |y|$, then $B_i[f](x) = f(x) + p_i\delta$ and $B_i[f](y) = f(y) - (1 - p_i)\delta$*

Now we define the repair operator.

Definition 5.3 (Repair Operator). *Given $f : \{0,1\}^d \to \delta\mathbb{Z}$, $A_i[f](x)$ is obtained from f by several applications of the asymmetric basic operator (see Definition 5.2) B_i along dimension i followed by a single application of the rounding operator. Specifically, let f' be the function obtained from f by applying B_i repeatedly until there are no violated edges along the i-th dimension. Then, $A_i[f]$ is defined to be $\mathbf{R}[f']$ where the rounding operator $\mathbf{R}$ rounds the function values to the closest $\delta\mathbb{Z}$-valued function.*

In effect, we have the following picture for the repair operation.

$$f = f_0 \xrightarrow{A_1} f_1 \xrightarrow{A_2} f_2 \to \cdots \to f_{d-1} \xrightarrow{A_d} f_d.$$

Now we define a measure called violation score which will be used to show the progress of repair operation. As shown later, the violation score is approximately preserved along any dimension $j \neq i$ when we apply the repair operator to repair the edges along dimension i. Note that the violation score closely resembles the violation score in [21] except that it depends on the function value as well as the probability masses of the end-points of the edge.

Definition 5.4. *The violation score of an edge with respect to function f, denoted by $vs(\{x,y\})$, is $\max(0, (p_x + p_y)(|f(x) - f(y)| - 1))$. The violation score along dimension i, denoted by $VS^i(f)$, is the sum of violation scores of all edges along dimension i*

The violation score of an edge $\{x,y\}$ is positive iff it is violated and violation score of a $\delta\mathbb{Z}$ valued function is contained in the interval $[\delta(p_x + p_y), ImD(f)(p_x + p_y)]$. Let $V^i(f)$ denote be the set of edges along dimension i violated by f. Then

$$\delta \cdot \sum_{\{x,y\} \in V^i(f)} (p_x + p_y) \leq VS^i(f) \leq \sum_{\{x,y\} \in V^i(f)} (p_x + p_y) \cdot ImD(f) \qquad (2)$$

Lemma 5.2 shows that A_i does not increase the violation score in dimensions other than i more than the additive value of δ.

Lemma 5.2. *For all $i, j \in [d]$, where $i \neq j$, and every function $f : \{0,1\}^d \to \delta\mathbb{Z}$, the following holds.*

– ***(progress)*** *Applying the repair operator A_i does not introduce new violated edges in dimension j if the dimension j is violation free, i.e. $VS_j(f) = 0 \Rightarrow VS_j(A_i[f]) = 0$.*
– ***(accounting)*** *Applying the repair operator A_i does not increase the violation score in dimension j by more than δ, i.e. $VS_j(A_i[f]) \leq VS_j(f) + \delta$.*

Application of the repair operator A_i entails several applications of the basic operator B_i followed by a single application of the rounding operator $\mathbf{R}$. In Lemma 5.3, we show that the applications of B_i does not increase the violation score along the remaining dimensions. In Claim 5.3, we show that the second step (rounding) is not too harmful for the remaining dimensions either. Finally, we use Lemma 5.3 and Claim 5.3 to prove Lemma 5.2.

Lemma 5.3. *Suppose $f : \mathcal{H}_d \to \mathbb{R}$ is such that for every edge $\{x, y\}$ in $\mathcal{H}_d$, the difference $f(x) - f(y) \in \delta\mathbb{Z}$. Let f' be the function obtained from f by applying B_i repeatedly until there are no violated edges along the i-th dimension. Then for every dimension $j \neq i$, $VS_j(f') \leq VS_j(f)$.*

Proof. First observe that it is sufficient to prove the lemma for a single application of the basic operator B_i. This is because for every edge $\{x, y\}$, the following holds: $f(x) - f(y) \in \delta\mathbb{Z} \Rightarrow B_i[f](x) - B_i[f](y) \in \delta\mathbb{Z}$. To see this observe that, by definition of the basic operator, $B_i[f](x) - B_i[f](y)$ is either $f(x) - f(y) + \delta$ or $f(x) - f(y) - \delta$ and we already started with the assumption that $f(x) - f(y) \in \delta\mathbb{Z}$. Note that before the application of repair operations, the function f has range in $\delta\mathbb{Z}$ and the assumption holds true. Also, it holds true for the further applications of B_i as shown above. Next we prove the lemma for one step of the basic operator.

Following the proof outline of a similar proof in [21], we show that application of the asymmetric basic operator in dimension i does not increase the violation score in dimension $j \neq i$. Standard arguments [16,9,21,2] show that it is enough to analyze the effect of applying B_i on one fixed disjoint square formed by adjacent edges that cross dimensions i and j. (This is because edges along dimensions i and j form disjoint squares in the hypercube. So proving the statement for one fixed square of the hypercube, the full claim follows by summing up the inequalities over all such squares.)

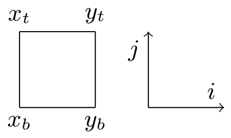

Fig. 1. Image courtesy [21]

Consider the two dimensional function $f : \{x_b, x_t, y_b, y_t\} \to \mathbb{R}$ where $\{x_b, x_t, y_b, y_t\}$ are as positioned in the figure. Assume that the basic operator is applied along the dimension i. We show that the violation score along dimension j does not increase. Assume that the violation score along edge $\{x_b, x_t\}$ increases. First, assume that the $B_i[f](x_t) > B_i[f](x_b)$. (The other case is very similar and we will prove it later.) Then B_i increases $f(x_t)$ and/or decreases $f(x_b)$. Assume that B_i increases $f(x_t)$. (The other case is symmetrical.) This implies that $\{x_t, y_t\}$ is violated and $f(x_t) < f(y_t)$. Recall that the repair operator is applied only if the edge is violated. Therefore $f(y_t) - f(x_t) > 1$. Since $f(y_t) - f(x_t) \in \delta\mathbb{Z}$ and $\frac{1}{\delta}$ is an integer, we have

$$f(y_t) \geq f(x_t) + 1 + \delta$$

The above inequality is crucial for the remaining proof of the lemma 5.1. Now consider the cases when either the bottom edge is also violated or is not violated.

If the bottom edge is not violated then we have $f(x_b) \geq f(y_b) - 1$ and $f(x_b)$ and $f(y_b)$ are not modified by the basic operator. Since $vs(\{x_t, x_b\})$ increases, $f(x_t) > f(x_b) + 1 - p_i\delta$. Combining the above inequalities, we get $f(y_t) \geq f(x_t) + 1 + \delta > f(x_b) + 2 + (1 - p_i)\delta \geq f(y_b) + 1 + (1 - p_i)\delta > f(y_b) + 1$. Thus the violation score increases along $\{x_t, x_b\}$ by $(p_{x_b} + p_{x_t})p_i\delta$ and decreases along $\{y_b, y_t\}$ by $(p_{y_b} + p_{y_t})(1 - p_i)\delta = (p_{x_b} + p_{x_t})\left(\frac{p_i}{1-p_i}\right)(1 - p_i)\delta$ which is same as $(p_{x_b} + p_{x_t})p_i\delta$, keeping the violation score along the dimension j unchanged.

If the bottom edge is violated, then the increase in $vs(\{x_b, x_t\})$ implies that $f(x_b)$ must decrease (after application of B_i) by $p_i\delta$ (since $|x_b| < |y_b|$) implying $f(y_b) + 1 < f(x_b))$. Therefore $f(x_t) + p_i\delta > f(x_b) + 1 - p_i\delta$ or $f(x_t) > f(y_t) + 1 - 2p_i\delta$. Therefore $f(y_t) > f(x_t) + 1 > f(x_b) + 2 - 2p_i\delta \geq f(y_b) + 3 - 2p_i\delta + \delta \geq f(y_b) + 1 + \delta$. The last inequality is true since $\delta \leq 1$ and $p_i \leq 1$. Thus, $vs(\{x_t, x_b\})$ increases by at most $(p_{x_b} + p_{x_t})2p_i\delta$ while $vs(\{y_t, y_b\})$ decreases by $(p_{y_t} + p_{y_b})2(1 - p_i)\delta = (p_{x_b} + p_{x_t})2p_i\delta$, ensuring that violation score along the vertical dimension does not increase.

Now we turn to the case when $B_i[f](x_t) < B_i[f](x_b)$. By the arguments very similar to the first case, it can be proved that $f(x_t) \geq f(y_t) + 1 + \delta$ and the application of basic operator decreases $f(x_t)$ by $p_i\delta$ and increases $f(y_t)$ by $(1 - p_i)\delta$.

If the bottom edge is not violated then $f(y_b) \geq f(x_b) - 1$ and $f(x_b)$ and $f(y_b)$ are not modified by the basic operator. Since $vs(\{x_t, x_b\})$ increases, $f(x_b) > f(x_t) + 1 - p_i\delta$. Combining the above inequalities, we get $f(y_b) \geq f(x_b) - 1 > f(x_t) - p_i\delta \geq f(y_t) + 1 + \delta(1 - p_i)$. Thus the violation score increases along $\{x_t, x_b\}$ by $(p_{x_b} + p_{x_t})p_i\delta$ and decreases along $\{y_b, y_t\}$ by $(p_{y_b} + p_{y_t})(1 - p_i)\delta = (p_{x_b} + p_{x_t})\left(\frac{p_i}{1-p_i}\right)(1 - p_i)\delta$ which is same as $(p_{x_b} + p_{x_t})p_i\delta$, keeping the violation score along the dimension j unchanged.

If the bottom edge is violated, then the increase in $vs(\{x_b, x_t\})$ implies that $f(x_b)$ must increase implying $f(y_b) > f(x_b) + 1$. Therefore, the increase in $vs\{x_b, x_t\}$ implies that $f(x_b) + p_i\delta > f(x_t) - p_i\delta + 1$ or $f(x_b) > f(x_t) - 2p_i\delta + 1$. Combining the above inequalities, we get $f(y_b) > f(x_b) + 1 > f(x_t) - 2p_i\delta + 2 \geq f(y_t) + 3 + \delta - 2p_i\delta \geq f(y_t) + 1 + \delta$. The last inequality is true since $\delta \leq 1$ and $p_i \leq 1$. Thus, $vs(\{x_t, x_b\})$ increases by at most $(p_{x_b} + p_{x_t})2p_i\delta$ while $vs(\{y_t, y_b\})$ decreases by $(p_{y_t} + p_{y_b})2(1 - p_i)\delta = (p_{x_b} + p_{x_t})2p_i\delta$, ensuring that violation score along the vertical dimension does not increase. □

Claim (Rounding is safe). Given $a, b \in \mathbb{R}$ satisfying $|a - b| \leq 1$, let a' (respectively, b') be the value obtained by rounding a (respectively, b) to the closest $\delta\mathbb{Z}$ integer. Then $|a' - b'| \leq 1$.

Proof. Assume without loss of generality $a \leq b$. For $x \in \mathbb{R}$, let $\lfloor x \rfloor_\delta$ be the largest value in $\delta\mathbb{Z}$ not greater than x. Observe that $a' \in \{\lfloor a \rfloor_\delta, \lfloor a \rfloor_\delta + \delta\}$. Using the fact that $\lfloor a \rfloor_\delta \leq b' \leq \lfloor a \rfloor_\delta + 1 + \delta$, we see that if $a' = \lfloor a \rfloor_\delta + \delta$ then $|b' - a'| \leq 1$ always holds. Therefore, assume $a' = \lfloor a \rfloor_\delta$. This can happen only if $a \leq \lfloor a \rfloor_\delta + \delta/2$. The latter implies $b \leq \lfloor a \rfloor_\delta + 1 + \delta/2$ (using the fact that $b - a \leq 1$). That is $b' \neq \lfloor a \rfloor_\delta + 1 + \delta$. In other words, $b' \leq \lfloor a \rfloor_\delta + 1$ again implying $b' - a' \leq 1$, as required. □

Proof (of Lemma 5.2). Let f' be the function from the statement of Lemma 5.3. Then function $A_i[f]$ is obtained by rounding the values of f' to the closest $\delta\mathbb{Z}$ values. Since rounding can never create new edge violations by Claim 5.3, we immediately get the first part of the lemma. The second part follows from the observation that the rounding step modifies each function value by at most $\delta/2$. Correspondingly, the violation score of an edge along the j-th dimension changes by at most $2\cdot(\delta/2)\cdot(p_u+p_v)$ where the factor 2 comes because both endpoints of an edge may be rounded. Summing over all edges in the j-th dimension, we get, increase in violation score $\leq \sum_{\{u,v\}} \delta \cdot (p_u + p_v) = \delta$ where the last equality holds because edges along the j-th dimension form a perfect matching and therefore the probabilities $p_u + p_v$ sum to 1. □

Proof of Lemma 5.1. Using the arguments very similar to [21] as given below, we can get the following sequence of inequalities

$$Dist(f_{i-1}, f_i) = Dist(f_{i-1}, A_i(f_{i-1})) \leq \sum_{(x,y)\in V_i(f_{i-1})} (p_x + p_y)$$

$$\leq \frac{1}{\delta} VS^i(f_{i-1}) \leq \frac{1}{\delta} VS^i(f) + (d-i)\delta \leq \frac{1}{\delta} \sum_{(x,y)\in V^i(f)} (p_x + p_y)\cdot ImD(f) + (d-i)\delta$$

Here functions $\{f_i\}_{i=0}^{i=d}$ are defined in the same way as [21]. The first inequality holds because A_i modifies f only at the endpoints points x and y of violated edge (x, y) along dimension i, thus paying $p_x + p_y$. The second and fourth inequalities follow from Equation (2) and the third inequality holds because of Lemma 5.2. Therefore, by triangle inequality, we have

$$\begin{aligned} Dist(f, f_d) &\leq \sum_{i\in[d]} Dist(f_{i-1}, f_i) \\ &\leq \sum_{i\in[d]} \left(\sum_{(x,y)\in V^i f(H)} (p_x + p_y)\cdot \frac{ImD(f)}{\delta} \right) + (d-i)\delta \\ &\leq \left(\sum_{(x,y)\in V(f))} (p_x + p_y)\cdot \frac{ImD(f)}{\delta} \right) + d^2\delta \end{aligned}$$

For a function which is ϵ-far from Lipschitz, we have $Dist(f, f_d) \geq \epsilon$. Therefore, from the above inequality, we have

$$\sum_{(x,y)\in V(f)} \frac{(p_x + p_y)}{d} \geq \frac{\delta(\epsilon - d^2\delta)}{d \cdot ImD(f)}$$

6 Instantiation of Privacy Tester Using Lipschitz Testers

In this section, we instantiate the privacy tester of Section 4 with both known Lipschitz testers as well as the Lipschitz tester developed in this work. The table below compares the current state of art Lipschitz testers on the hypercube domain. The third column gives the “approximation factor” as defined in Definition 2.3 for the various testers. The last row gives the result of Lipschitz tester (Section 5) developed in this work. The discussion about the instantiations follows.

Reference	Functions	Approx. Factor	Distribution	Tester running time
[7]	$\{0,1\}^d \to \mathbb{R}$	1	Uniform	$O(\frac{d}{\epsilon})$
This work	$\{0,1\}^d \to \mathbb{R}$	$(1+\delta)$	**Product**	$O\left(\frac{d \cdot ImD(f)}{(\epsilon - d^2\delta)\delta}\right)$

First we analyze the result for the case when points are sampled from $\{0,1\}^d$ according to the uniform distribution. In this case, the running time for for $(\alpha, \beta, \gamma, \delta)$-privacy testing of $\mathcal{T}_{priv}$ (defined in theorem 4.1) is $O(\frac{|Z|^2 d}{\gamma})$. Let us now analyze the running time for the same when applied to the datasets coming from the hypercube domain according to some (possibly unknown) product distribution. We use the tester given in the algorithm 3. The running time in this case is given by $O(|\mathcal{Z}| \cdot$ Run-time $\left(\mathcal{T}_{Lip}(\frac{\gamma}{|\mathcal{Z}|}, \beta, \delta, d)\right)$. Choosing $\delta = \frac{\gamma}{4d^2|Z|}$, one gets the running time of the $\mathcal{T}_{priv}$ to be $O(\frac{d^4|Z|^3}{\gamma^2})$. In general, δ can be made smaller at the cost of higher running time of tester. This clearly shows the trade off between the privacy guarantee and running time of the tester.

7 Discussions and Open Problems

In this section we discuss some of the interesting implications of our current work and some of the new avenues it opens up. Also we state some of the open problems that remains unresolved in our work.

Privacy: In this work, we took the first step towards designing efficient testing algorithm for statistical data privacy. Our work indicates that it is indeed possible to design efficient testing algorithms for some existing notions of statistical data privacy (e.g., differential privacy on typical datasets). It is important that the current paper should be treated as an initial study of the problem and in no way should be interpreted as conclusive. It is interesting (and also important) to explore other rigorous notions of data privacy, their implications, and design testers for them.

In this paper, we test for differential privacy on typical datasets, which is a relaxation of differential privacy. It remains an open problem to design a privacy tester for *pure* (α, β)-differential privacy.

Lipschitz Testing: This work presents the first Lipschitz property tester for the setting where the domain points are sampled from a distribution that is not uniform. Because of possible applications to statistical data privacy, this work has motivated the design of such Lipschitz testers for other domains, e.g. hypergrid. Also, this paper mainly shows the tester for the product distribution over the hypercube domain, but it still remains open to design testers for other distributions that may be correlated in some way (e.g., pairwise correlation). In the current work, we have designed testers where the domain of the dataset is finite. A natural question that arises is that if we can extend the current results to design privacy testers when the datasets are drawn from continuous domain.

Other Limitations of our Approach: The current work on testing of privacy properties have several limitations which are worth highlighting. These limitations need to be addressed in order to allow our current testing algorithms to be used in practice. Firstly, our current testing algorithm works on discrete range space S. This significantly limits the applicability of our testing algorithm in various applications which are meaningful over continuous range spaces (e.g., various machine learning problems). Also the running time of our tester depends polynomially on the size of the range space S. Secondly, our testing algorithm needs oracle access to the probability measure induced on the range space of the untrusted algorithm $\mathcal{A}$. In general it is not clear how to come up with a computationally efficient oracle given just the algorithm $\mathcal{A}$. Thirdly, the current results are only for discrete domain of datasets $\mathcal{X}^d$. This restricts the applicability of our approach to many interesting applications (e.g., Gaussian process regression).

Acknowledgements. We would like to thank Adam Smith for suggesting the name Differential Privacy on Typical Datasets and for various other suggestions and comments.

References

1. Awasthi, P., Jha, M., Molinaro, M., Raskhodnikova, S.: Limitations of local filters of lipschitz and monotone functions. In: Gupta, et al.: [18], pp. 374–386
2. Awasthi, P., Jha, M., Molinaro, M., Raskhodnikova, S.: Testing lipschitz functions on hypergrid domains. In: Gupta, et al.: [18], pp. 387–398
3. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S., Thakurta, A.: Noiseless database privacy. Cryptology ePrint Archive, Report 2011/487 (version: 20120524:110619) (2011), `http://eprint.iacr.org/`
4. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S., Thakurta, A.: Noiseless Database Privacy. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 215–232. Springer, Heidelberg (2011)
5. Bhowmick, A., Dwork, C.: Natural differential privacy. Personal Communication (2012)
6. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: "you might also like: " privacy risks of collaborative filtering. In: IEEE Symposium on Security and Privacy, pp. 231–246 (2011)
7. Chakrabarty, D., Seshadhri, C.: Optimal bounds for monotonicity and lipschitz testing over the hypercube. CoRR abs/1204.0849 (2012)
8. Dixit, K., Jha, M., Raskhodnikova, S., Thakurta, A.: Testing the lipschitz property over product distributions with applications to data privacy (2013), `http://arxiv.org/abs/1209.4056`
9. Dodis, Y., Goldreich, O., Lehman, E., Raskhodnikova, S., Ron, D., Samorodnitsky, A.: Improved Testing Algorithms for Monotonicity. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM-APPROX 1999. LNCS, vol. 1671, pp. 97–108. Springer, Heidelberg (1999)

10. Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
11. Dwork, C.: Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
12. Dwork, C.: The Differential Privacy Frontier (Extended Abstract). In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 496–502. Springer, Heidelberg (2009)
13. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
14. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
15. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: KDD, pp. 265–273 (2008)
16. Goldreich, O., Goldwasser, S., Lehman, E., Ron, D., Samorodnitsky, A.: Testing monotonicity. Combinatorica 20(3), 301–337 (2000)
17. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM 45(4), 653–750 (1998)
18. Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.): APPROX 2012 and RANDOM 2012. LNCS, vol. 7408. Springer, Cambridge (2012)
19. Halevy, S., Kushilevitz, E.: Distribution-free property-testing. SIAM J. Comput. 37(4), 1107–1138 (2007)
20. Halevy, S., Kushilevitz, E.: Distribution-free connectivity testing for sparse graphs. Algorithmica 51(1), 24–48 (2008)
21. Jha, M., Raskhodnikova, S.: Testing and reconstruction of lipschitz functions with applications to data privacy. In: Ostrovsky, R. (ed.) FOCS, pp. 433–442. IEEE (2011)
22. Kasiviswanathan, S.P., Smith, A.: A note on differential privacy: Defining resistance to arbitrary side information. CoRR abs/0803.3946 (2008)
23. Korolova, A.: Privacy violations using microtargeted ads: A case study. In: ICDMW, pp. 474–482 (2010)
24. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE, p. 24 (2006)
25. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: SIGMOD, pp. 19–30 (2009)
26. Mohan, P., Thakurta, A., Shi, E., Song, D., Culler, D.: Gupt: privacy preserving data analysis made easy. In: SIGMOD, pp. 349–360 (2012)
27. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC, pp. 75–84 (2007)
28. Parnas, M., Ron, D.: Testing the diameter of graphs. Random Struct. Algorithms 20(2), 165–183 (2002)
29. Reed, J., Pierce, B.C.: Distance makes the types grow stronger: a calculus for differential privacy. In: ICFP, pp. 157–168 (2010)
30. Roy, I., Setty, S.T.V., Kilzer, A., Shmatikov, V., Witchel, E.: Airavat: Security and privacy for mapreduce. In: NSDI, pp. 297–312 (2010)
31. Rubinfeld, R., Sudan, M.: Robust characterization of polynomials with applications to program testing. SIAM J. Comput. 25(2), 252–271 (1996)
32. Smith, A.: Privacy-preserving statistical estimation with optimal convergence rates. In: STOC, pp. 813–822 (2011)
33. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557–570 (2002)