Random Projections, Graph Sparsification, and Differential Privacy

Jalaj Upadhyay

David R. Cheriton School of Computer Science
University of Waterloo,
200, University Avenue West
Waterloo, ON, Canada–N2L 4X7.
{jkupadhy}@cs.uwaterloo.ca

Abstract. This paper initiates the study of preserving *differential privacy* (DP) when the data-set is sparse. We study the problem of constructing efficient sanitizer that preserves DP and guarantees high utility for answering cut-queries on graphs. The main motivation for studying sparse graphs arises from the empirical evidences that social networking sites are sparse graphs. We also motivate and advocate the necessity to include the efficiency of sanitizers, in addition to the utility guarantee, if one wishes to have a practical deployment of privacy preserving sanitizers.

We show that the technique of Blocki et al. [3] (BBDS) can be adapted to preserve DP for answering cut-queries on sparse graphs, with an asymptotically efficient sanitizer than BBDS. We use this as the base technique to construct an efficient sanitizer for arbitrary graphs. In particular, we use a preconditioning step that preserves the spectral properties (and therefore, size of any cut is preserved), and then apply our basic sanitizer. We first prove that our sanitizer preserves DP for graphs with high conductance. We then carefully compose our basic technique with the modified sanitizer to prove the result for arbitrary graphs. In certain sense, our approach is complementary to the Randomized sanitization for answering cut queries [17]: we use graph sparsification, while Randomized sanitization uses graph densification.

Our sanitizers almost achieves the best of both the worlds with the same privacy guarantee, i.e., it is almost as efficient as the most efficient sanitizer and it has utility guarantee almost as strong as the utility guarantee of the best sanitization algorithm.

We also make some progress in answering few open problems by BBDS. We make a combinatorial observation that allows us to argue that the sanitized graph can also answer (S, T)-cut queries with same asymptotic efficiency, utility, and DP guarantee as our sanitization algorithm for $S, \bar{S}$-cuts. Moreover, we achieve a better utility guarantee than Gupta, Roth, and Ullman [17]. We give further optimization by showing that fast Johnson-Lindenstrauss transform of Ailon and Chazelle [2] also preserves DP.

Keywords. Differential privacy, Graph sparsification, (S, T)-cut queries, Fast Johnson-Lindenstrauss transform.

1 Introduction

The privacy of a data is a fundamental problem in today's age of information. Many agencies collect enormous amount of data and store it in its database. These data may contain sensitive informations about an individual. However, given the benefits of analyzing these data, the problem that curators of such a database face is to provide useful information in such a manner so that no personal or sensitive information about an individual is leaked. A trivial way to guarantee this is to add a lot of noise to the database; however, nothing useful could be harnessed from such noisy database. Most of the research in this area is geared towards providing a tight utility and privacy tradeoff and only consider the query generator in mind. In this paper, we take a conceptual review and ask the practical question: what would a firm, that is going to deploy these sanitizers, demand from the group that develops these algorithms?

The question one expect to get from real firms or agencies is what extra resources they have to invest to provide this facility. This is expected in the real-world because a curator would prefer to deploy its resources to facilitate other interfaces that are primary to its business if differential private sanitizer uses a lot of resource. In general, sanitizers are polynomial time, but the exact bound on this polynomial is never made explicit in earlier works. In fact, Exponential sanitization [17, 19] may be intractable! We initiate the study of the question whether it is possible to guarantee DP that has high utility guarantee with an efficient sanitizer, emphasizing on a *concrete bound* on the efficiency parameter.

Motivation of our Problem. Our motivation of studying cut queries on sparse graphs arises from a natural problem in social networks. One of the question that is commonly asked in social network is, given a set of individuals, how many friends/acquaintance do a set of people have outside their circle? The natural approach to solve this problem is to construct a *friendship graph*, where each vertex is labeled by an individual and there is an edge between two vertices if they are friends. These graphs on social networks are usually sparse, i.e., the average degree of the graph is very small in comparison to the number of vertices.

For a concrete example, consider the *friendship graph* on Facebook. According to the recent data released by Facebook, it has around one billion active users! It is not outrageous to assume that only a small fraction of users on Facebook have more than a thousand friends. Therefore, this graph is highly sparse. The friendship graph is undirected; however, this might not always be the case. For example, consider the *following graph* based on the networking of Twitter. It is a directed graph with nodes labeled by an individual. A node is the tail of an edge if the individual follows the head of the edge. The number of active users on Twitter is few million; however, it is less likely that an individual follows more than a few hundred fellow users. Thus, the *following graph* is very sparse. In these scenarios, the difference between performing $10^{9\times 2.38}$ and 10^{18} algebraic operations is huge. Any firm, like Facebook and Twitter, which is motivated by economics is less likely to invest in the former sanitization algorithm and may consider investing in the latter one.

It could be argued that if a sanitizer works for dense graphs (and therefore, also for sparse graphs), then there is no need for a specialized sanitizer for sparse graphs. The reason why we feel it is important to study sparse graphs exclusively is that sanitizers for dense graphs do not use the structural properties present in sparse graphs. In general, sparse graphs provides faster algorithms [7, 15, 28]. For example, consider the Johnson-Lindenstrauss (JL) sanitizer [3]. The sanitization algorithm of BBDS first overlays a complete graph on top of the input graph and then applies JL transform to the columns of the Laplacian of the modified graph. The step in which we overlay the complete graph destroys all the structural properties the input graph might have.

Now consider the situation when the input graph is sparse, and a hypothetical sanitizer that does not overlay K_n on top of it. When we perform random projection on this graph, the number of operations would depend on the number of edges of the graph (more concretely, on the non-zero entries in the representative matrix of the graph which could be Laplacian or adjacency matrix). Unfortunately, this sanitizer is not differentially private if the graph is weakly connected (in graph theoretic terms, has low conductance).

To see why this hypothetical sanitization does not provide DP, consider an n-vertices graph with two connected components. If the query is to find the cut of all the set of vertices in one component, the answer is 0 with probability 1. However, for a graph that has an edge joining two vertices present in different connected components, the probability with which the response to the query is non-zero is 1. This gives an easy way to differentiate the two cases. To resolve this particular problem, BBDS overlaid an n-vertex complete graph on top of the input graph.

Unfortunately, if we overlay the complete graph on a sparse graph, then we destroy the sparsity, and lose any (possible) gain in the computational time. On the other hand, even if the graph is connected and we do not perturb the graph, chances of privacy leakage are still present. More specifically, adding a single edge in a sparse graph can potentially have more privacy leak than a corresponding change in dense graphs. For example, consider a line graph or tree. They are acyclic; however adding any edge introduces a cycle. A slight modification of the differentiating algorithm used in the case of two component graph could be used to break the DP.

Our Contributions. This work is motivated by practical scenarios in which a sanitizer might be deployed. One of the objective of this paper is to advocate that, in addition to the utility and privacy guarantee, a design methodology for sanitizers should also give a concrete analysis of the efficiency of sanitizers. We initiate this line of work by studying differentially private sanitizer for cut queries on graphs.

As mentioned above, in practice, sparse graphs are more likely to occur than dense graphs. Every sanitizer that are proposed in the literature for dense graphs also works for sparse graphs, but they are not efficient. Moreover, there are examples of sparse graphs that could leak more information in DP sense than dense graphs, mainly because an addition or deletion of an edge could change

the graph properties more dramatically in sparse graphs than in dense graphs. Thus, the problem is non-trivial, especially when we wish to construct efficient sanitizer.

On the fundamental level, we advocate the need of considering the efficiency of the sanitizer in the design methodology and give an explicit bound on the running time. The reason why we believe this is an important parameter is that, in many practical scenarios, the data-sets are held by agencies who might not have the privacy of an individual as their biggest priority. Therefore, unless a sanitizer is efficient, they might not have any incentive to perform the required sanitization. The technical contributions of this paper are as follows.

1. We show that it is possible to adapt the JL sanitizer of BBDS to answer cut queries when the graph is sparse. Additionally, our sanitizer has to perform only $O(n^{2+o(1)})$ algebraic operations. On the other hand, irrespective of whether or not the graph is sparse, the sanitizer of BBDS needs $O(rn^{2.38})$ operations[1], where r is the dimension of the subspace to which the JL transform projects the columns of the Laplacian. This improvement is *asymptotically significant*.
2. A natural question that arises next is whether our basic sanitizer is useful when the graph is fairly dense. We answer this question in affirmative. More precisely, we show that if we precondition a graph by reassigning the weights to the edges such that the transformed graph is guaranteed to be sparse and maintain the spectral properties of the graph, then applying the basic sanitizer on the conditioned graph preserves DP. This can be seen as a complementary approach to the Randomized sanitizer [17].
3. We make a simple combinatorial observation to argue that our sanitizer also preserves (S, T)-cut queries. This answers an open problem raised by BBDS.
4. Our last contribution is directed towards the optimization of the algebraic computations. We show that DP is maintained even if we replace the standard JL transform by the fast JL transform of Ailon and Chazelle [2]. This *partially answers another open problem of* BBDS.

Remark 1. An important characteristics of our preconditioning step, in item 2 above, is that it preserves the spectral properties. Any sanitizer that answers the queries based on spectral property of a graph could be transformed to first apply the preconditioning step before the sanitization step to improve the efficiency.

We note that none of our sanitizer randomly projects the vector corresponding to the column vector of the graph to a smaller dimension r. The main observation is that the mechanism is non-interactive, and in order to preserve the privacy for all set of queries, the dimension of the projected space has to be at least the dimension of the input space.

[1] Assuming that the matrix multiplication is done using Coppersmith-Winograd's algorithm.

Overview of our Techniques. We first give a brief overview of the sanitizer of BBDS. In BBDS, the sanitization algorithm first reweighs the graph by overlaying a complete graph on top of the input graph, i.e., every edge, $e = (a, b)$, with weight w_e is replaced by an edge with weight $w'_e := \frac{w}{n} w_e + \left(1 - \frac{w}{n}\right)$. In other words, the weight on the edges are redistributed such that the overlaid complete graph has an equal weight, $\frac{w}{n}$, on all its edges. This makes the graph connected, and from Lemma 4, the smallest non-zero eigenvalues of this modified graph is greater than w/n. The JL transform is then applied on the columns of the Laplacian of the modified graph.

The major challenge that we face is that sparse graphs have low conductance. When we overlay the complete graph, it increases the conductance, but simultaneously makes it inefficient to answer the cut-queries by destroying the structural properties. As an extreme, consider a line graph. The cut-queries are fairly straightforward to answer; however, if we overlay a complete graph on top of it, we destroy the structural property and need to do some extra arithmetic to answer the same set of queries. An alternative is to overlay a sparse graph that increases the conductance, but does not destroy the structure of the underlying graph by much. The most natural candidate for this is an expander graph. As we show in our analysis, this suits our purpose very well.

We modify our basic sanitization technique, as outlined above, to construct an efficient sanitization technique for dense graphs. The key idea is to use graph sparsification at an appropriate step. As a warm-up, we assume that the input graph has high conductance. Our technique in this case is simple: apply the graph sparsification algorithms followed by the JL transform. The key observation here is that conductance helps in proving that, when the sparsification technique is applied on two neighboring graphs, the corresponding sparse graphs differ on at most one edge. This allows us to use the proof of BBDS for DP. On the other hand, due to the spectral guarantee provided by the sparsification technique, we know that all the cuts of the graph is maintained within a multiplicative factor. The utility guarantee then follows using simple arithmetic.

In order to apply the above analysis to arbitrary graph, we need a high conductance graph. This directs the order of the steps we follow for arbitrary graph, i.e, we first overlay a complete graph (or an expander) on the input graph before applying the sparsification algorithm. Finally, we apply the JL transform.

Related Work. Differential privacy, introduced by Dwork et al. [9], provides a robust guarantee of privacy. Informally speaking, if a curator sanitizes a dataset, then even if an individual's data is removed from the database, none of the responses to a query is more or less likely than the others. The key idea used in Dwork et al. [9] is to add noise to an output of the query according to a Laplace distribution, where the distribution is parameterized by the *sensitivity* of the query function. The Gaussian variant of this basic sanitizer was proven to preserve DP by Dwork et al. [8] in a follow-up work. Since then, many sanitizers for preserving DP have been proposed in the literature, including the Exponential sanitizer [4, 23, 27], the Multiplicative Update sanitizer [16–19], the Median sanitizer [30], the Boosting sanitizer [10], and the Random Projection sanitizer [25].

All these sanitizers have a common theme: they perturb the output before responding to queries. Blocki et al. [3] took a complementary approach. They perturb the input by performing a random projection of the input and show that existing algorithms preserves DP if the input is perturbed in a reversible manner.

The first work to explicitly study DP when the underlying data-set is a graph or a social network was by Hay et al. [20]. They presented a differentially private sanitizer for answering the degree of a node in a graph. They were followed by works of Nissim et al. [29] and Karwa et al. [24]. Gupta et al. [17] first studied the question of answering (S, T)-cut queries. The literature of studying faster computations on a sparse variant of any mathematical objects is so extensive that we cannot hope to cover it in all details here. An extensive study of faster methods of doing linear algebra on a sparse matrix is covered in standard textbooks [7, 15]. We refer the readers to an excellent book by Nesetril and de Mendez [28] for the properties and algorithms on sparse graph.

Organization of the paper. In Section 2, we cover the basic preliminaries and definitions to the level required to understand the presentation of this paper. In Section 3, we give our basic sanitizer for sparse graph that serves as the building blocks for the sanitizers in Section 4. We conclude the paper by showing in Section 5 that the fast JL transform of Ailon and Chazelle [2] also preserves DP.

2 Preliminaries, Notations, and Basic Definitions

2.1 Privacy and Utility

In this work, we deal with privacy-preserving sanitizers for answering cut queries on graphs. The notion of differential privacy requires a definition of neighboring data-sets. Two data-sets (graphs, respectively) are *neighboring* if they differ on at most one entry (edge, respectively).

Definition 1. *A randomized algorithm $\mathcal{K}$, also called a* sanitizer*, gives ε-*DP*, if for all neighboring data-sets D_1 and D_2, and all range $S \subset Range(\mathcal{K})$, $\mathsf{Pr}[\mathcal{K}(D_1) \in S] \leq \exp(e^{\epsilon})\mathsf{Pr}[\mathcal{K}(D_2) \in S]$, where the probability is over the coin tosses of the sanitizer $\mathcal{K}$.*

In this paper, we study a natural relaxation of DP, called *approximate* DP.

Definition 2. *A randomized algorithm, $\mathcal{K}$ also called a* sanitizer*, gives (ε, δ)-*DP*, if for all neighboring data-sets D_1 and D_2, and all range $S \subset Range(\mathcal{K})$, $\mathsf{Pr}[\mathcal{K}(D_1) \in S] \leq \exp(e^{\epsilon})\mathsf{Pr}[\mathcal{K}(D_2) \in S] + \delta$, where the probability is over the coin tosses of the sanitizer $\mathcal{K}$.*

2.2 Linear Algebra

Let A be an $n \times m$ matrix. We let $\mathsf{rk}(A)$ denote the rank and $\mathsf{Tr}(A)$ denote the trace norm of the matrix A. The singular value decomposition of A is $A = V \Lambda U^T$,

where U, V are unitary matrices and Λ is a diagonal matrix. The entries of Λ, denoted by $\lambda_1(A), \cdots, \lambda_{\mathsf{rk}(A)}(A)$, are called the *singular values* of A. Since U, V are unitary matrices, one can write $A^i = V\Lambda^i U^T$ for any real value i. If A is not a full rank matrix, then its inverse is called *Moore-Penrose* inverse and is denoted by $A^\dagger$ and its determinant is called *pseudo-determinant* and is defined as $\tilde{\Delta}(A) = \Pi_{i=1}^{\mathsf{rk}(A)} \lambda_i(A)$. We let $\chi_S \in \{0,1\}^n$ denote the characteristic vector for a subset $S \subseteq \mathcal{V}$.

2.3 Gaussian Distribution

Given a random variable, X, we denote by $X \sim \mathcal{N}(\mu, \sigma^2)$ the fact that X is distributed according to a Gaussian distribution with the probability density function, $\mathsf{PDF}_X(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. The Gaussian distribution is invariant under affine transformation, i.e., if $X \sim \mathcal{N}(\mu_x, \sigma_x)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y)$, then $Z = aX + bY$ has the distribution $Z \sim \mathcal{N}(a\mu_x + b\mu_y, a\sigma_x^2 + b\sigma_y^2)$.

Multivariate Gaussian Distribution. The multivariate Gaussian distribution is a generalization of univariate Gaussian distribution. Given an m-dimensional multivariate random variable, $X \sim \mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma = \mathbb{E}[(X-\mu)(X-\mu)^T]$, the PDF of a multivariate Gaussian is given by $\mathsf{PDF}_{\mathbf{X}}(\mathbf{x}) := \frac{1}{\sqrt{2\pi\tilde{\Delta}(\Sigma)}} \exp\left(-\frac{1}{2}\mathbf{x}^T\Sigma^\dagger\mathbf{x}\right)$. It is easy to see from the description of the PDF that, in order to define the PDF corresponding to a multivariate Gaussian distribution, Σ has to have full rank. If Σ has a non-trivial kernel space, then the PDF is undefined. However, in this paper, we only need to compare the probability distribution of two random variables which are defined over the same subspace. Therefore, in those scenarios, we would restrict our attention to the subspace orthogonal to the kernel space of Σ.

Multivariate Gaussian distribution maintains many key properties of univariate Gaussian distribution. For example, any (non-empty) subset of multivariate normals is multivariate normal. Another key property that is important in our analysis is that linearly independent linear functions of multivariate normal random variables are multivariate normal random variables, i.e., if $Y = AX + \mathbf{b}$, where A is an $n \times n$ non-singular matrix and $\mathbf{b}$ is a (column) n-vector of constants, then $Y \sim \mathcal{N}(A\mu + \mathbf{b}, A\Sigma A^T)$.

2.4 Graph Theory

We reserve the symbol $\mathcal{G}$ and $\mathcal{H}$ to denote a graph. We denote by $\mathcal{G}'$ the graph formed by adding an edge to the graph $\mathcal{G}$. In the case when $\mathcal{H}$ is formed from $\mathcal{G}$ using some transformation, we denote by $\mathcal{H}'$ the graph formed by performing the same transformation on $\mathcal{G}'$. For any $S \subseteq \mathcal{V}(\mathcal{G})$, the *cut* of the set of vertices S, denoted it by $\Phi_{\mathcal{G}}(S)$, is the weight of the edges that are present between S and $V \backslash S$.

We follow the same terminology of BBDS to define the utility guarantee.

Definition 3. *We say a sanitizer $\mathcal{K}$ gives a (η, τ, ν)-*approximation for cut queries*, if for every non-empty set $S \subseteq \mathcal{V}$, it holds that*

$$\mathsf{Pr}[(1-\eta)\Phi_{\mathcal{G}}(S) - \tau \leq \mathcal{K}(S, \mathcal{G}) \leq (1+\eta)\Phi_{\mathcal{G}}(S) + \tau] \geq 1 - \nu.$$

For the entire paper, we fix $w = \Theta\left(\frac{\log(1/\delta)+\log(1/\nu)}{\varepsilon}\right)$.

Laplacian of a Graph. For a weighted graph $\mathcal{G} := (\mathcal{V}, \mathcal{E}, w)$, its adjacency matrix $A_{\mathcal{G}}$ is given by $A_{\mathcal{G}}(i,j) = w_{ij}$ if $(i,j) \in \mathcal{E}$. The degree matrix of a weighted graph $\mathcal{G}$ is given by a diagonal matrix $D_{\mathcal{G}}$ such that the diagonal entries (i,i) is $\sum_j A_{\mathcal{G}}(i,j)$. The signed-edge matrix, $B_{\mathcal{G}}$, is constructed in the similar fashion as in BBDS: let $\mathcal{O}$ be an arbitrary orientation of edges. For an edge $e = (u,v)$, place $\sqrt{w_e}$ at position (e,v) if the edge e has v as its head and $-\sqrt{w_e}$ if it has v as its tail. For the other (e,i) when $i \neq u, v$, place 0.

The matrix for the Laplacian of a weighted graph, denoted by $L_{\mathcal{G}}$, is defined as $D_{\mathcal{G}} - A_{\mathcal{G}}$. One of the most useful form of Laplacian of a graph is the following form: $L_{\mathcal{G}} = \sum_{(a,b)\in E} w_{ab} L_{ab} = B_{\mathcal{G}}^T B_{\mathcal{G}}$, where L_{ab} is the Laplacian of a graph with a single edge (a,b). Many interesting properties of the Laplacian of a graph follows from this representation. For example, Laplacian of a graph is positive semi-definite, i.e., all the eigenvalues are non-negative. For a set S of vertices, its cut-set is $\Phi_{\mathcal{G}}(S) = \chi_S^T L_{\mathcal{G}} \chi_S$. Moreover, for $S, T \subseteq \mathcal{V}$, the sum of the weights of the edges with one end in S and other in T is denoted by $\Phi_{\mathcal{G}}(S,T)$. We explore this in detail later in Section 4.3.

We let $\lambda_i(\mathcal{G})$ denote the eigenvalues of $L_{\mathcal{G}}$ for $1 \leq i \leq n$. Next we present few lemmata that are useful in our analysis. In our analysis, we analyze multivariate Gaussian distributions that are linear combination of the Laplacian of two graphs. In order to analyze the two distributions, the corresponding covariance matrices must span the same subspace. The first lemma allows us to work on the same subspace, that is, the subspace orthogonal to $\mathsf{Span}\{\mathbf{1}\}$.

Lemma 1. [11, 12] *Let $0 = \lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \cdots \leq \lambda_n(\mathcal{G})$ be the n eigenvalues of $L_{\mathcal{G}}$. Then $\mathcal{G}$ is connected iff $\lambda_2 > 0$ and the kernel space of a connected graph is $\mathsf{Span}\{\mathbf{1}\}$. More generally, if a graph has k components, then the multiplicity of eigenvalue 0 is k.*

The following two lemmata are useful in giving the upper bound while proving the DP of our sanitizer.

Lemma 2. *Let $\mathcal{G}$ and $\mathcal{G}'$ be two graphs, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by adding one edge joining two distinct vertices of $\mathcal{G}$. Then*

$$\lambda_2(\mathcal{G}) \leq \lambda_2(\mathcal{G}') \leq \lambda_2(\mathcal{G}) + 2.$$

Lemma 3. *Let $\mathcal{G}'$ be formed by adding an edge (u,v) to $\mathcal{G}$. For any vector $\boldsymbol{x} \in \mathbb{R}^n$, we have $\mathsf{Tr}(L_{\mathcal{G}'}) \leq \mathsf{Tr}(L_{\mathcal{G}}) + 2$.*

The following lemma is particularly useful in arguing that the lowest non-zero eigenvalues of all the graphs is bounded from below by a constant (which is the second smallest eigenvalue of an expander).

Lemma 4. (Eigenvalue Interlacing). *Let $\mathcal{G}$ and $\mathcal{G}'$ be two graphs, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by adding one edge joining two distinct vertices of $\mathcal{G}$. Then*

$$\lambda_i(\mathcal{G}) \leq \lambda_i(\mathcal{G}') \leq \lambda_{i+1}(\mathcal{G}).$$

In particular, if $\mathcal{H}$ be a subgraph of $\mathcal{G}$, then $\lambda_i(\mathcal{H}) \leq \lambda_i(\mathcal{G}) \forall 1 \leq i \leq n$.

We refer the readers to the excellent book by Godsil and Royle [14] for a comprehensive treatment of the algebraic properties of graphs.

Graph Approximation. A graph $\mathcal{H}$ is said to ϵ-approximate a graph $\mathcal{G}$ if $\mathcal{H}$ approximates the spectral properties of $\mathcal{G}$, i.e.,

$$(1-\epsilon)\mathbf{x}^T L_{\mathcal{G}}\mathbf{x} \leq \mathbf{x}^T L_{\mathcal{H}}\mathbf{x} \leq (1+\epsilon)\mathbf{x}^T L_{\mathcal{G}}\mathbf{x} \qquad \forall \mathbf{x} \in \mathbb{R}^n.$$

We denote it by $(1-\epsilon)L_{\mathcal{G}} \preceq L_{\mathcal{H}} \preceq (1+\epsilon)L_{\mathcal{G}}$.

Electrical Flows and Resistance. We need the concept of electrical flow in graphs at various points for the analysis of Theorem 7. Intuitively, electrical flow of a graph measures how easy or difficult it is to move from one vertex to the other. If the "resistance" (as described later) between two vertices is high, then it is more difficult to reach from one vertex to the other, and vice versa. We give a brief exposition of the electrical flow that is required to understand this paper. Let $\mathbf{i}$ be the vector of current injected at the vertices of the graph $\mathcal{G}$. Then the effective resistance between two vertices u and v is defined as the potential difference induced between them when a unit current is injected at one vertex and extracted from the other. For any pair of vertices u and v, the effective resistance,

$$R_{uv} = (\chi_u - \chi_v)^T L_{\mathcal{G}}^{\dagger}(\chi_u - \chi_v) = \|B_{\mathcal{G}} L_{\mathcal{G}}^{\dagger}(\chi_u - \chi_v)^2\|_2^2.$$

Conductance. At the intuitive level, the conductance of a graph is the inverse of the resistance. For a graph, $\mathcal{G} = (V, E)$, let d_v denote the degree of vertex $v \in V$. Let $Vol(S) = \sum_{i \in S} d_i$, then the conductance of a set of vertex S, denoted by $\mathsf{cond}_S(G)$ is defined by

$$\mathsf{cond}_S(G) := \frac{|\Phi_{\mathcal{G}}(S)|}{\min\{Vol(S), Vol(V-S)\}}.$$

The conductance of a graph $\mathcal{G}$ is then given by $\mathsf{cond}(G) := \min_{S \subset V, |S| \geq 1} \mathsf{cond}_S(G)$. The conductance of a graph has a strong relation to the smallest non-zero eigenvalue of its Laplacian and we use it implicitly or explicitly in all of our analyses.

Theorem 1. (Cheeger's Inequality). *For a graph $\mathcal{G}$, $\mathsf{cond}(G)^2/2 \leq \lambda_2(L_{\mathcal{G}}) \leq 2\mathsf{cond}(G)$.*

2.5 JL Transform

The famous JL transform [1, 2, 5, 6, 21, 22] can be seen as a random projection of d points from a n-dimensional space to a lower dimensional space such that the Euclidean distance between any two pairs of points is maintained.

Theorem 2. *Fix any $\eta \in (0, 1/2)$ and M be a $k \times n$ matrix, whose entries are chosen from $\mathcal{N}(0, 1)$. Then $\forall \boldsymbol{x} \in \mathbb{R}^n$, we have*

$$\mathsf{Pr}_M \left[(1-\eta)\|\boldsymbol{x}\|^2 \leq \frac{1}{k}\|M\boldsymbol{x}\|^2 \leq (1+\eta)\|\boldsymbol{x}\|^2 \right] \geq 1 - 2\exp(-\eta^2 k/8). \quad (1)$$

Fast JL *Transform.* Ailon and Chazelle [2] gave an elegant transform that is asymptotically faster than the traditional JL transform. It involves preconditioning the input. In this section, m denotes the number of n-dimensional points on which the transform is applied and k denotes the target dimension. More specifically, the fast JL transform is $M = PWD$, where (i) W is a $n \times n$ normalized Walsh-Hadamard matrix, (ii) D is a $n \times n$ diagonal matrix, where $\mathsf{Pr}[D_{ij} = 1] = \mathsf{Pr}[D_{ij} = -1] = 1/2$, and (iii) P is a $k \times n$ matrix whose elements are independently distributed as follows. With probability $1 - q$, set $P_{ij} = 0$; otherwise draw P_{ij} from a normal distribution of expectation 0 and variance $1/q$. The constant q is called the *sparsity* constant and is set to $q = \Theta\left(\frac{\eta^{p-2}\log^p n}{n}\right)$, where p is the norm we wish to preserve. Since W encodes the discrete Fourier transform, using fast Fourier transform, Ailon and Chazelle [2] proved that the transform satisfies equation (1), and takes time $\tilde{O}(n + qm/\eta^2))$.

3 Sanitizer for Cut Queries

In this section, we give our basic sanitizer for sparse graphs. Our key observation is that, in the sanitizer of BBDS, the overlay of the complete graph is required to maintain high conductance, and the result regarding the second smallest eigenvalue of the Laplacian follows immediately from the fact that a complete graph is a subgraph of the resulting graph. Unfortunately, this perturbation, when applied to sparse graphs, destroys the structural benefits of sparsity.

We get the same two objectives by overlaying an expander graph. An expander graph makes the graph connected while the smallest non-zero eigenvalue has the desired lower bound if we chose our expander graph with care. Recently, Friedman [13] proved that a random graph is an expander graph with high probability. In fact, he showed that such graphs are Ramanujan graphs. Marcus, Spielman, and Srivastava [26] recently proved the existential result for bipartite expander that matches the Ramanujan bound for every degree d.

We first derive a connection between the spectral properties of an expander graph and a complete graph. The most useful relation for this derivation is an alternate definition of an expander graph, i.e., *a d-regular graph $\mathcal{G}$ is an expander if $\lambda_2(\mathcal{G}) \geq (1 - \epsilon')d$ for some arbitrary constant ϵ'*. We give our basic sanitizer for sparse graphs in Figure 1.

Input. A n-vertex sparse graph $\mathcal{G}$, and parameters $\varepsilon, \delta, \eta$.
Output A Laplacian of a graph $\tilde{L}$.

1. Pick a d-regular expander graph E.
2. Set $L_{\mathcal{H}} \leftarrow \frac{w}{d} L_E + \left(1 - \frac{w}{d}\right) L_{\mathcal{G}}$
3. Pick an $n \times n$ matrix M with each entries picked from $\mathcal{N}(0, 1)$.
4. Output $\tilde{L} = (M^T L_{\mathcal{H}} M)/n$

The sanitizer publishes $\tilde{L}$ and anyone with a set of vertices S as input can compute $\Phi_{\mathcal{G}}(S)$ as below

$$\Phi_{\mathcal{G}}(S) = \frac{1}{1 - \frac{w}{d}} \left(\chi_S^T \tilde{L} \chi_S - \frac{ws(n-s)}{n} \right),$$

where $|S| = s$.

Fig. 1. The Basic Sanitizer

Theorem 3. *The basic algorithm in Figure 1 preserves* (ε, δ)-DP*, provides an utility of* (η, τ, ν)*-approximation, where* $\tau \leq O((\eta + \epsilon')ws)$*, and runs in time* $O(n^{2+o(1)})$.

Proof. We first perform the complexity analysis of the above sanitizer. For a sparse graph, $m = O(n)$; therefore, using [34], it takes $\tilde{O}(n)$ time to compute the JL transform (since every column in the Laplacian of a sparse graph has $\tilde{O}(1)$ entries). Since, matrix multiplication takes $\Omega(n^2)$, this is almost tight for any sanitizer design that uses noise multiplication for answering cut queries.

The proof of DP proceeds in the similar manner as in BBDS. This is because our change still fulfills the requirement for which BBDS introduced the complete graph, i.e., it makes the graph connected. Using Lemma 1, the kernel space is Span$\{\mathbf{1}\}$. Also, by a suitable choice of the expander graph, i.e., one with $1 - \epsilon' \geq w/d$, Lemma 4 guarantees that all the eigenvalues of $\mathcal{H}$ is greater that w, which is required in the privacy analysis of BBDS.

Our proof of utility guarantee develops on a useful relation between an expander graph and a complete graph. Let E be a d-regular expander graph such that the eigenvalues of A_E are $\leq \epsilon' d$. From the expression, $L_E = D_E - A_E$, where D_E is the degree matrix of E, we have that all the non-zero eigenvalues of L_E are between $(1-\epsilon')d$ and $(1+\epsilon')d$. Therefore, from Courant-Fischer formula,

$$(1 - \epsilon')d\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T L_E \mathbf{x} \leq (1 + \epsilon')d\mathbf{x}^T\mathbf{x} \qquad \forall \mathbf{x} \in \mathbb{R}^n. \tag{2}$$

We wish to relate this to the complete graph. For the complete graph, K_n, the eigenvalues are 0 with multiplicity 1 and n with multiplicity $n-1$. Therefore,

$$\mathbf{x}^T L_{K_n} \mathbf{x} = n\mathbf{x}^T\mathbf{x} \qquad \Rightarrow \frac{d}{n}\mathbf{x}^T L_{K_n} \mathbf{x} = d\mathbf{x}^T\mathbf{x}. \tag{3}$$

Plugging equation (3) in equation (2), we have

$$(1-\epsilon')\frac{d}{n}\mathbf{x}^T L_{K_n}\mathbf{x} \leq \mathbf{x}^T L_E \mathbf{x} \leq (1+\epsilon')\frac{d}{n}\mathbf{x}^T L_{K_n}\mathbf{x} \qquad \forall \mathbf{x} \in \mathbb{R}^n. \tag{4}$$

One way to look at equation 4 is that an expander graph is a sparsified complete graph. Using equation 4, we have the following approximation:

$$\begin{aligned}\left((1-\epsilon')\frac{w}{n}L_{K_n} + \left(1-\frac{w}{d}\right)L_{\mathcal{G}}\right) &\preceq \left(\frac{w}{d}L_E + \left(1-\frac{w}{d}\right)L_{\mathcal{G}}\right) \\ &\preceq \left((1+\epsilon')\frac{w}{n}L_{K_n} + \left(1-\frac{w}{d}\right)L_{\mathcal{G}}\right). \end{aligned}\tag{5}$$

We can now calculate the utility guarantee using equation (5) and similar arithmetic as in BBDS. More specifically, the upper bound on the utility guarantee can be calculated as below.

$$\begin{aligned}(1+\eta)\chi_S^T L_{\mathcal{H}}\chi_S &= (1+\eta)\chi_S^T\left(\frac{w}{d}L_E + \left(1-\frac{w}{d}\right)L_{\mathcal{G}}\right)\chi_S \\ &\leq (1+\eta)\chi_S^T\left((1+\epsilon')\frac{w}{n}L_{K_n} + \left(1-\frac{w}{d}\right)L_{\mathcal{G}}\right)\chi_S \\ &\leq (1+\eta)(1+\epsilon')\frac{w}{n}s(n-s) + \left(1-\frac{w}{d}\right)(1+\eta)\Phi_{\mathcal{G}}(S).\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{1}{1-\frac{w}{d}}\left(\chi_S^T\tilde{L}\chi_S - \frac{ws(n-s)}{n}\right) &= \left(\frac{w(\eta+\epsilon'+\eta\epsilon')s}{(1-\frac{w}{d})}\right)\left(1-\frac{s}{n}\right) + (1+\eta)\Phi_{\mathcal{G}}(S) \\ &\leq 2(\eta+\epsilon'+\eta\epsilon')ws + (1+\eta)\Phi_{\mathcal{G}}(S).\end{aligned}$$

This gives an additive approximation of $O((\eta+\epsilon')ws)$. The proof of the lower bound is similar.

4 Differential Privacy by Sparsification

In Section 3, we showed that a simple change to the sanitizer of BBDS gives an efficient sanitizer for sparse graphs. In this section, we consider the case of an arbitrary graph. In particular, we show that various graph sparsification techniques also preserves DP. This serves as the second main contribution of this paper. Intuitively, the result in this section follows from the observation that, for large enough n, the sparsification techniques can be seen as a random projection. Thus, sparsification composed with our basic scheme should preserve DP by the composition theorem [10] and Theorem 3.

In certain sense, our approach is complementary to the approach used in Randomized sanitization. The Randomized sanitization [17] constructs a weighted graph, $\mathcal{H} = (\mathcal{V}, \mathcal{E}', w')$, such that $\forall u, y \in \mathcal{V}$, the weight of edge (u, v) in $\mathcal{H}$ is distributed as per the following distribution: $\mathsf{Pr}[w'_{uv} = 1] = (1 + \varepsilon w_{uv})/2$ and $\mathsf{Pr}[w'_{uv} = -1] = (1 - \varepsilon w_{uv})/2$.

Input. A n-vertex graph $\mathcal{G}$ with high conductance, and parameters $\varepsilon, \delta, \eta$.
Output A Laplacian of a graph $\tilde{L}$.

1. Convert $\mathcal{G}$ to a $\mathcal{H}$ using [31] or [32], such that $\mathcal{H}$ is an ϵ-sparsification of $\mathcal{G}$.
2. Pick an $n \times n$ matrix M with each entries picked from $\mathcal{N}(0, 1)$.
3. Output $\tilde{L} = (M^T L_{\mathcal{H}} M)/n$.

The sanitizer publishes $\tilde{L}$ and anyone who has as input the set of vertices S can compute $\Phi_{\mathcal{G}}(S) = \left(1 - \frac{w}{n}\right) \chi_S^T \tilde{L} \chi_S$.

Fig. 2. Sanitizer for Graph With High Conductance

4.1 Sanitization of Graphs with High Conductance

Every randomized sparsification technique picks an edge to be included in a sparsified graph with some specified probability distribution. At a high level, the distribution can be defined either dependent on the local structure or on the global structure of the graph. We give our sanitizer for both types of distribution, picking the most efficient one for instantiation. In this section, we analyze our sanitizer, stated in Figure 2, for graphs with high conductance.

The utility guarantee follows from the sparsification guarantee provided by the respective sparsifiers and the JL transform. The efficiency guarantee is straightforward from the observation that there are $\tilde{O}(1)$ entries in every row or column of the Laplacian of sparse graphs, and the run time of the step 3 in the Figure 2 is governed by the number of non-zero entries in the Laplacian. More concretely, assume that the sparsification algorithm takes $\tilde{O}(m)$ time to output a graph with $\tilde{O}(n)$ edges (as we will see, both the techniques, Spielman and Teng [32] based on local properties of the graph (Theorem 5), and Spielman and Srivastava [31] based on global properties of the graph (Theorem 6), satisfies these two conditions). Therefore, even if the graph is dense, i.e., $m = O(n^2)$, the run time for sparsification is $\tilde{O}(n^2)$. Therefore, the time taken by the sanitizer is bounded by $\tilde{O}(n^2)$ (since, in expectation, every column in the Laplacian of the sparse matrix has $\tilde{O}(1)$ entries).

The tricky part is to prove the privacy guarantee. For the privacy guarantee, we prove that for two neighboring graphs $\mathcal{G}$ and $\mathcal{G}'$, the respective sparse graphs differ on at most one edge. We can then apply Lemma 3, and the rest of the proof follows along the same line as BBDS. For both the sparsifier, we prove that if the graph has high enough conductance, then the probability distribution on edges with which the sparsification algorithm picks an edge does not differ by a lot. We then analyze two types of edges: (i) the edge (a, b) that is present in $\mathcal{G}'$ but not in $\mathcal{G}$ and (ii) the edges that are in both $\mathcal{G}$ and $\mathcal{G}'$. In the first case, the probability that the edge (a, b) is present in $\mathcal{H}'$ is non-zero and is identically zero in $\mathcal{H}$. We then prove that if the probability distribution on the edges does not differ by "lot", then with all but negligible probability, the respective sparse graphs will differ on at most one edge, i.e., only due to the (possible) presence of the edge (a, b) in $\mathcal{H}'$. The privacy guarantee follows using the proof of BBDS.

Using Local Sparsification Techniques: Construction of Spielman and Teng [32]. Spielman and Teng [32] proved the following result for any graph.

Theorem 4. [32] *There exists an $\tilde{O}(m)$ time algorithm which on input $0 < \epsilon, p < 1/2$ and a graph $\mathcal{G}$ with n-vertices and m-edges, outputs a sparse graph $\mathcal{H}$ with $\tilde{O}(n/\epsilon^2)$ edges such that $\mathcal{H}$ is an ϵ-approximation of $\mathcal{G}$.*

The main construction of Spielman and Teng [32] for arbitrary graphs is little complicated and uses techniques of graph decomposition and contraction. In this section, we will use the construction that works for a graph with high conductance. In this construction, every edge, $e = (i, j)$, is picked with probability, $p_{ij} := \frac{144k^2}{\epsilon^2 \lambda_2(G)^2 \min\{d_i, d_j\}}$, where $k = \max\{\log_2(3/p), \log_2 n\}$, d_i denotes the degree of the vertex i, and p is an arbitrary constant between $(0, 1/2)$. It is an easy exercise to check that the probability distribution on the edges that are already present does not change by a lot when a new edge is added due to the dependence only on the local structure of the graph (the eigenvalue changes by at most two by Lemma 2, and degree of only the two end vertices changes).

Using the proof outline mentioned above, we have the following theorem for the sanitizer in Figure 2 when we use the sparsifier of Spielman and Teng [32].

Theorem 5. *The algorithm in Figure 2 preserves (ε, δ) DP, provides an answer that is $((1+\eta)(1+\epsilon), 0, \nu)$-approximation, and runs in time $O(n^{2+o(1)})$ when using [32] sparsifier.*

Using Global Sparsification Techniques: Construction of Spielman and Srivastava [31]. We first recall the spectral sparsifier of Spielman and Srivastava [31]. One alternative way to see their sparsification is that $\mathcal{H}$ is a random projection of the edge matrix of $\mathcal{G}$, where edges are picked according to their importance in the original graph. The sparsifier construct a graph $\mathcal{H}$ by picking every edge, $e \in \mathcal{E}(\mathcal{G})$, with probability $p_e = w_e R_e/(n-1)$ to be included in $\mathcal{H}$, where R_e is the effective resistance across the edge e and w_e is the weight of the edge e. The effective resistance on an edge can be computed as

$$R_e = b_e L_{\mathcal{G}}^{\dagger} b_e^T \qquad \Rightarrow \qquad R = B_{\mathcal{G}} L_{\mathcal{G}}^{\dagger} B_{\mathcal{G}}^T. \tag{6}$$

Using the above probability distribution, Spielman and Srivastava [31] proved the following for any arbitrary graph.

Theorem 6. [31] *There exists an $\tilde{O}(m(\log r)/\epsilon^2)$ algorithm which on input $\epsilon > 1/\sqrt{n}$ and an n vertex, m edges graph $\mathcal{G}$, with the ratio of maximum weight to minimum weight r, outputs a sparse graph $\mathcal{H}$ such that $\mathcal{H}$ is an ϵ-approximation of $\mathcal{G}$.*

Consider the matrix, $\Pi = BL_{\mathcal{G}}^{+}B^T$, where B is the signed edge-vertex matrix. It is easy to see that Π is a projection matrix and has a well defined spectrum: eigenvalue 1 with multiplicity $(n-1)$ and 0 otherwise. Also, it has a nice relation to the probability with which an edge is picked to be placed in $\mathcal{H}$: $\Pi_{e,e} = W_{e,e}^{1/2} R_e W_{e,e}^{1/2} = w_e R_e$ for every edge $e = (a, b)$. Moreover, since the trace of Π

is $n-1$; therefore, $p_e = \Pi_{e,e}/(n-1) = \|BL^{\dagger}(\chi_a - \chi_b)\|/(n-1)$, where χ_a is the characteristic vector of the vertex a.

We have the following theorem for DP when using the sparsification technique of Spielman and Srivastava [31].

Theorem 7. *If the input graph has high conductance, then the algorithm in Figure 2 runs in time $O(n^{2+o(1)})$ and preserves (ε, δ) differential privacy with an utility of $((1+\eta)(1+\epsilon), 0, \nu)$ approximation when using [31] sparsifier.*

Few remarks are in order regarding Theorems 5 and 7. The theorems state that the sanitizer allows zero additive error. Therefore, if we have a graph with high conductance, we can get an answer with only multiplicative approximation for as low tolerance as possible. This stands in stark contrast with Theorem 3 and Theorem 9 where there is an additive approximation that governs the tolerance achieved by the sanitizer. The reason is that, as the underlying graph has high conductance, the smallest non-zero eigenvalue is large. This allows us to remove the step where we overlay an expander graph!

4.2 Sanitizer for Arbitrary Graph

Before we move to arbitrary graphs, we give an alternative for the local sparsification technique of Spielman and Teng [31]. They (and Trevisan [33], independently) proved an important combinatorial property of an arbitrary graph, which can be used, in composition to their basic technique for high conductance graph, to prove the sparsification result for any arbitrary graph.

Theorem 8. [32, 33] *Let $\mathcal{G} = (V, E)$ be an arbitrary graph. Then there exists a set $\mathcal{E}' \subset \mathcal{E}$ of $\kappa|\mathcal{E}|$ edges, such that removal of these edges decomposes the graph in some components, each of which have an smallest non-zero eigenvalue at least $\kappa^2/72 \cdot (\log|\mathcal{E}|)^2$. Furthermore, these edges can be found in polynomial time.*

The above theorem could be used to get sparsification algorithm for an arbitrary graph. Let $\mathcal{E}'$ be the set of edges found by the algorithm guaranteed in Theorem 8. First apply the sparsification algorithm of [32] on all the components with high conductance, neglecting the edges in the set $\mathcal{E}'$. We recursively apply the sparsification algorithm on $\mathcal{E}'$ until we get a sparse graph, i.e, one with $|\mathcal{E}'| \leq \tilde{O}(n)$. The recursion depth is at most $O(\log n)$ rounds, so the overall run time of the algorithm is still under the bound guaranteed by Theorem 4. Therefore, one could use the complete sparsification technique that decomposes the graph in to graphs of high conductance with few bridge edges between the components before the third step of Figure 1. The DP of the sanitizer would follow the same idea as in the proof of Theorem 5 because the probability of picking edges depends on local graph structure, and the utility guarantee would follow from the partition-then-sample lemma of Spielman and Teng [32] and the guarantee of JL transform.

The idea of constructing sparse graphs using recursion also applies to the sparsifier of Spielman and Srivastava [31], but does not help us in proving the

DP because of a subtle reason: the probability distribution in [32] depend locally on the graph structure, i.e., only on the degree of the end-points of the edges (it also depends on the smallest non-zero eigenvalue of the Laplacian, but from Lemma 3 and Lemma 4, it is easy to prove that the eigenvalue change by at most 2). Thus, the probability distribution changes by any significant amount only for the edge that is added.

On the other hand, the probability distribution on an edge can change drastically for Spielman and Srivastava's sparsifier [31]. This is because the probability distribution depends globally on the whole graph and if an edge is added, it is possible that the effective resistance along a different edge changes by a lot. For example, if a unit weight edge (a, b) is added, then any edge (u, v) that is parallel to (a, b) sees an effective drop to less than 1. If the conductance of the graph is not large, then this drop is significant.

The key idea to work around this problem is to maintain the conductance of the graph. We do this by using appropriate order of composition: we first overlay a expander (or complete) graph on top of the input graph and then apply the Spielman-Srivastava's sparsifier [31].

The utility guarantee follows by incorporating the approximation guarantee provided by Spielman and Teng [32] or Spielman and Srivastava [31] in the analysis of Theorem 3.

Theorem 9. *The algorithm in Figure 3 preserves* (ε, δ)-DP*, provides an utility of* (η, τ, ν) *approximation, where* $\tau \leq O((\eta+\epsilon)ws)$*, and runs in time* $O(n^{2+o(1)})$*.*

Input. A n-vertex graph $\mathcal{G}$, and parameters $\varepsilon, \delta, \eta$.
Output A Laplacian of a graph $\tilde{L}$.

1. Pick a d-regular expander (or n-vertices complete) graph E.
2. Set $L_{\mathcal{G}} \leftarrow \frac{w}{d} L_E + \left(1 - \frac{w}{d}\right) L_{\mathcal{G}}$.
3. Convert $\mathcal{G}$ to a $\mathcal{H}$ using [31] or [32], such that $\mathcal{H}$ is an ϵ-sparsification of $\mathcal{G}$.
4. Pick an $n \times n$ matrix M with each entries picked from $\mathcal{N}(0, 1)$.
5. Output $\tilde{L} = (M^T L_{\mathcal{H}} M)/n$

The sanitizer publishes the matrix $\tilde{L}$. For an input $S \subset V$, one can compute the number of vertices that crosses the cut as below

$$\Phi_S(\mathcal{G}) = \frac{1}{1 - \frac{w}{n}} \left(\chi_S^T \tilde{L} \chi_S - \frac{ws(n-s)}{n} \right).$$

Fig. 3. Sanitizer for Arbitrary Graph

Remark. Note that we can perform the complexity analysis of all the sanitizer mentioned in Sections 3 and 4 using the optimization mentioned in Section 5.

4.3 Answering (S,T)-cut Queries

One of the open problems listed by BBDS was to construct a sanitizer that answers (S,T)-cut queries on arbitrary graphs. Their concern for using the JL transform based mechanism is related to the inner product problem in JL transform. We get around this problem by making a simple combinatorial observation.

Let $S,T \subseteq \mathcal{V}$ be the set of vertices and we wish to find $\Phi_{\mathcal{G}}(S,T) = \sum_{s\in S,t\in T} w_{st}$. Note that

$$\Phi_{\mathcal{G}}(S) = \sum_{s\in S,u\in V\setminus S} w_{su} \qquad \textit{and} \qquad \Phi_{\mathcal{G}}(T) = \sum_{t\in T,v\in V\setminus T} w_{st}.$$

Therefore, $\Phi_{\mathcal{G}}(S) + \Phi_G(T)$ counts the weight of the edges that are crossing the boundaries of either S or T. These edges includes two types of edges: one that are crossing the boundaries of either S or T but do not have end vertices in both the sets and the one that have one end in S and the other in T. Note that we are interested in counting the weight of the edges of the latter form. Therefore, $\Phi_{\mathcal{G}}(S) + \Phi_{\mathcal{G}}(T) - \Phi_{\mathcal{G}}(S\cup T)$ is the sum of the weights of the edges between S and T, counted twice. This observation gives us that $\Phi_{\mathcal{G}}(S,T) = (\Phi_{\mathcal{G}}(S) + \Phi_{\mathcal{G}}(T) - \Phi_{\mathcal{G}}(S\cup T))/2$. Therefore, anyone with a set of vertices S and T as input and the sanitized graph from any of the mechanisms in this paper can compute $\Phi_{\mathcal{G}}(S,T)$ as below

$$\begin{aligned}\Phi_{\mathcal{G}}(S,T) = {} & \frac{1}{2(1-\frac{w}{n})}\left(\left(\chi_S^T\tilde{L}\chi_S + \chi_T^T\tilde{L}\chi_T - \chi_{S\cup T}^T\tilde{L}\chi_{S\cup T}\right)\right) \\ & - \frac{1}{2(1-\frac{w}{n})}\left(\frac{ws(n-s)}{n} + \frac{wt(n-t)}{n} - \frac{w(s+t)(n-(s+t))}{n}\right).\end{aligned}$$

Since computing the (S,T)-cut is three sequential applications of our basic sanitizer, DP follows from Theorem 3 (Theorem 5 and 7, respectively) for sparse graphs (high conductance graphs and arbitrary graphs, respectively) and the composition theorem of Dwork, Rothblum, and Vadhan [10, Theorem III.1]. The utility guarantee and efficiency guarantee are straightforward, giving the following theorem.

Theorem 10. *We can preserves (ε,δ) DP and provides an utility of (η,τ,ν) approximation, where $\tau \le O((\eta+\epsilon)w\max\{s,t\})$ in time $O(n^{2+o(1)})$ for answering (S,T)-cut queries.*

4.4 Comparison With Other Algorithms

In Table 1, we compare our sanitizer algorithms with other sanitizers that are proposed in the literature. It is not clear how to compare interactive and non-interactive sanitizers; therefore, for the additive errors, we have a column when total number of cut queries are at most k.

It is easy to see from Table 1 that our sanitizer almost matches the best of both the worlds; it is almost as efficient as Randomized Response and the

utility guarantee is as high as JL transform for constant ϵ', ϵ. For JL mechanism, we assume that for publishing the matrix, the sanitizer uses the Coppersmith-Winograd's matrix multiplication algorithm.

Method	τ for any k	Curator's Run Time
Randomized Response [17]	$O(\sqrt{sn \log k/\varepsilon})$	$O(n^2)$
Exponential Sanitizer [4, 27]	$O(n \log n/\varepsilon)$	Intractable
Multiplicative Weight [17, 19]*	$\tilde{O}(\sqrt{\lvert\mathcal{E}\rvert} \log k/\varepsilon)$	$O(n^2)$
JL [3]	$O(s\sqrt{\log k}/\varepsilon)$	$O(rn^{2.38})$
Basic Scheme	$O(s(\eta+\epsilon')\sqrt{\log k}/\varepsilon)$	$O(n^{2+o(1)})$
Using Sparsifier	$O(s(\eta+\epsilon')\sqrt{\log k}/\varepsilon)$	$O(n^{2+o(1)})$

Table 1. Comparison Between our Sanitizers and Other Sanitizers.

5 Optimization Using Fast-JL Transform

Our last major contribution is to explore whether some other variants of JL transform also preserve DP. We show a positive result for fast JL-transform of Ailon and Chazelle [2]; thereby, partially answering an open problem of BBDS.

Due to lack of space, we just give an overview of our proof. Recall that the fast-JL transform is the product PWD. The intuitive reason why fast JL transform preserves privacy is that fast-JL transform preconditions the input by performing a random projection by matrix D. This is a random projection by the result of Achlioptas [1]. It then applies an unitary matrix that is a FFT and then another random projection matrix P. Thus, it can be seen as the application of two random projections. Using Theorem III.1 of Dwork, Rothblum, and Vadhan [10] and the main observation of BBDS, it preserves DP. This is our intuition behind the proof.

The exact proof uses case analysis. Consider the edge (a, b) that is present in $\mathcal{G}'$ and absent in $\mathcal{G}$. Let $d_1, \cdots, d_n$ be the diagonal entries of the matrix D. The proof proceeds by consider two cases: when $d_a = d_b$ and when $d_a \neq d_b$. The first case is almost the same as in BBDS because WD is an unitary matrix. The upper bound when $d_a \neq d_b$ is also immediate. However, for lower bound, we need to analyze the terms in the decomposition of matrix $WDL_{ab}D^TW^T$, and the eigenvalues of the projection of $WDL_{\mathcal{G}}$ on the co-ordinates a, b.

6 Open Problems

Our technique of using spectral sparsification is very general. We believe it could be used as a subroutine in many sanitization algorithms, which are designed to answer queries based on the spectral properties, to improve their run time. It

would be interesting to investigate other such spectral properties. For example, one possible candidate for this improvement could be the differentially private low rank approximation algorithm of Kapralov and Talwar [23]. This is because Kapralov and Talwar [23] assume that their private matrices are covariance matrices and publish the low rank approximation by computing the singular vectors. Since covariance matrices are symmetric, one could compute the spectral sparsification.

Another aspect that is still open is to investigate whether other off-the-shelf JL transforms also preserve privacy or not. We have partially answered this question by studying fast JL transform, but there are many other variants that have applicability in different domains of computer science. In particular, we believe that any positive result for sparse JL transforms will be a significant step in improving the efficiency of our sanitizers and help in better understanding the relation between JL transforms and DP.

Acknowledgements

The author would like to thank the anonymous reviewers of ASIACRYPT, 2013 for many useful comments. The author would also like to thank the attendees of the C&O reading group and the members of CrySP and CACR for the useful discussions during the author's informal presentations.

References

1. Achlioptas, D. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.
2. Ailon, N. and Chazelle, B. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.
3. Blocki, J., Blum, A., Datta, A., and Sheffet, O. The johnson-lindenstrauss transform itself preserves differential privacy. In *FOCS*, pages 410–419, 2012.
4. Blum, A., Ligett, K., and Roth, A. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.
5. Dasgupta, A., Kumar, R., and Sarlós, T. A sparse johnson: Lindenstrauss transform. In *STOC*, pages 341–350, 2010.
6. Dasgupta, S. and Gupta, A. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.
7. Davis, T. *Direct Methods for Sparse Linear Systems.* SIAM. Part of the SIAM Book Series on the Fundamentals of Algorithms., Philadelphia, U.S.A., September 2008.
8. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.
9. Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
10. Dwork, C., Rothblum, G., and Vadhan, S. Boosting and DP. In *FOCS*, pages 51–60, 2010.

11. Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.
12. Fiedler, M. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, 25(100):618–633, 1975.
13. Friedman, J. A proof of alon's second eigenvalue conjecture. In *STOC*, pages 720–724, 2003.
14. Godsil, C. and Royle, G. *Algebraic Graph Theory*. Springer, 2001.
15. Golub, G. and Loan, C. *Matrix Computations*. John Hopkins University Press, Maryland, U.S.A., 1996.
16. Gupta, A., Hardt, M., Roth, A., and Ullman, J. Privately releasing conjunctions and the statistical query barrier. In *STOC*, pages 803–812, 2011.
17. Gupta, A., Roth, A., and Ullman, J.. Iterative constructions and private data release. In *TCC*, pages 339–356, 2012.
18. Hardt, M., Ligett, K., and McSherry, F.. A simple and practical algorithm for differentially private data release. In *NIPS*, pages 2348–2356, 2012.
19. Hardt, M. and Roth, A.. Beating randomized response on incoherent matrices. In *STOC*, pages 1255–1268, 2012.
20. Hay, M., Li, C., Miklau, G., and Jensen, C. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.
21. Johnson, W. and Lindenstrauss, J. Extensions of Lipschitz mapping into Hilbert space. In *Conf. in modern analysis and probability*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.
22. Kane, D. and Nelson, J. Sparser johnson-lindenstrauss transforms. In *SODA*, pages 1195–1206, 2012.
23. Kapralov, M. and Talwar, K. On differentially private low rank approximation. In *SODA*, pages 1395-1414, 2013.
24. Karwa, K., Raskhodnikova, S., Smith, A., and Yaroslavtsev, G. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.
25. Kenthapadi, K., Korolova, A., Mironov, I., and Mishra, N.. Privacy via the johnson-lindenstrauss transform. *CoRR*, abs/1204.2606, 2012.
26. Marcus, A, Spielman, D., and Srivastava, N.. Interlacing families i: Bipartite ramanujan graphs of all degrees. To *Appear in FOCS*, 2013.
27. McSherry, F. and Talwar, K. Mechanism design via DP. In *FOCS*, pages 94–103, 2007.
28. Nesetril, J. and Mendez, P. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.
29. Nissim, K., Raskhodnikova, S., and Smith, A. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
30. Roth, A. and Roughgarden, T. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.
31. Spielman, D. and Srivastava, N. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.
32. Spielman, D. and Teng, S. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.
33. Trevisan, L. Approximation algorithms for unique games. *Theory of Computing*, 4(1):111–128, 2008.
34. Yuster, R. and Zwick, U. Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, 2005.