Publishing Graph Data with Subgraph Differential Privacy

Binh P. Nguyen[1(✉)], Hoa Ngo[2], Jihun Kim[2], and Jong Kim[2]

[1] Division of IT Convergence Engineering, POSTECH, Pohang, Republic of Korea
phuongbinh@postech.ac.kr

[2] Department of Computer Science and Engineering, POSTECH, Pohang, Republic of Korea
{hoanx,jihun735,jkim}@postech.ac.kr

Abstract. The eruption of social networks, communication networks etc. makes them become valuable resources for the research community. However, the graph data owners hesitate to share their data due to the barrier of privacy leakage. In this work, we propose a new privacy definition, called subgraph-differential privacy (subgraph-DP), for graph data publishing based on the conventional differential privacy definition. Subgraph-DP is against the subgraph-based attacks by restricting the adversaries predict the true subgraph with a high confidence. We provide the mechanism that gives subgraph-DP in which noise will be added to a small set of edges to make sure that all k-vertices connected subgraphs are perturbed. The experimental results show that our perturbation mechanism preserves most of the important statistic features of graph while still guarantees privacy.

Keywords: Differential privacy · Graph pertubation

1 Introduction

In recent years, more and more data has been collected and stored on the Internet. These information are not independent but have relationships such as social networks, communication networks etc. Many applications are improved with information from the social networks such as network-based recommendation systems [9], sybil defenses [10]. However, the graph owners hesitate to publish their data because that may result in leaking the confidential information. Graph perturbation emerges as an inevitable solution and receives more attention from the research community. The re-identification on Netflix [6] is a typical example showing that simply anonymizating graph is not enough because adversaries can easily get desired information about a particular person if the personal privacy is not considered carefully in graph publishing.

Differential privacy [1] is an in-focus paradigm for publishing useful statistical information over sensitive data with the rigorous privacy's guarantees. Differential privacy has been successfully applied to a wide range of data analysis tasks

H. Kim and D. Choi (Eds.): WISA 2015, LNCS 9503, pp. 134–145, 2016.
DOI: 10.1007/978-3-319-31875-2_12

and found to have tight relations to other fields such as cryptography, statistics, complexity, combinatorics, mechanism design and optimization. However, it is not easy to apply differential privacy to non-tabular databases like graphs where relationships exist among separated entities.

In this paper, we propose a new privacy definition, subgraph-differential privacy, which applies the conventional differential privacy to graphs. We also introduce a mechanism satisfying subgraph-differential privacy. Finally, we evaluate the mechanism on real graphs and show that our mechanism provides strong privacy while retaining utility.

The paper is organized as follows. In Sect. 2, we summarize the related work. Section 3 reviews the basic concepts of differential privacy. In Sect. 4, we focus on our proposal, called subgraph-differential privacy, the definition and mechanism. In Sect. 5, we show our experiments and evaluate our scheme on real graphs. The conclusion and future work is addressed in Sect. 6.

2 Related Work

Several existing works try to apply the robust privacy definition, differential privacy, to graph data. There are two approaches in literature of graph data publishing: the interactive and non-interactive mechanism. In the so-called interactive setting, information is protected inside a graph handled by the data owner, and access is allowed only through an interface. In the non-interactive setting these problems are addressed by releasing once and for all the graph or its model which we think is interest to most analysts, while still preserving privacy.

In the perspective of the interactive publishing, [12] designs the node differentially private algorithms, that are, algorithms whose the output distribution does not change significantly when a node and all its adjacent edges are added/deleted to a graph. The main idea behind their techniques is to project the input graph onto a set of graphs with their maximum degree below a certain threshold. By this way, node privacy is easier to achieve in bounded-degree graphs since the insertion of one node affects only a relatively small part of the graph.

In the perspective of the non-interactive publishing, F. Ahmed et al. [4] propose a random projection approach which publishes the adjacency matrix of a given graph. This approach utilizes random matrix theory to reduce the dimensions of the adjacency matrix and achieves differential privacy by adding small amount of noise. A. Sala et al. [5] describes graph into dK-graph model. They introduce the dK-perturbation algorithm that computes the noise injected into dK-2 to obtain differential privacy. This approach becomes more efficient if dK-series is clustered.

3 Background

Differential privacy ensures that the outcome of any analysis on database is not influenced substantially by the existence of any individual. An adversary therefore hardly inference attacks on any data rows.

Definition 1. *(Differential Privacy [1]): A randomize function $\mathcal{K}$ gives ϵ-differential privacy if for all datasets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(\mathcal{K})$*

$$Pr[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) Pr[\mathcal{K}(D_2) \in S] \tag{1}$$

where $\mathrm{Range}(\mathcal{K})$ *denotes the output range of the algorithm $\mathcal{K}$.*

The most popular mechanism achieving ϵ -DP is calibrating Laplace noise to query answer. The standard deviation of noise depends on the *sensitivity* of function $\mathcal{K}$.

Theorem 1. *(Laplace mechanism [2]): For all randomize function $\mathcal{K} : D \rightarrow \mathbb{R}^d$, the following mechanism is ϵ-DP:*

$$San_{\mathcal{K}}(\boldsymbol{x}) = \mathcal{K}(\boldsymbol{x}) + (Y_1, .., Y_d) \tag{2}$$

where the Y_d are drawn i.i.d. from $Lap(\Delta\mathcal{K}/\epsilon)$

4 Subgraph-Differential Privacy

4.1 Problem

Given a graph G representing graph data, the graph owner wants to release the anonymized and possibly sanitized network graphs to commercial partners and academic researchers. Therefore, we take it for granted that attackers will access to such data.

The structural attack model on graph data is a class of attacks that adversaries somehow can collect a set of vertices and their trust relationships which are represented as edges between those vertices. [3] proposes the k-neighborhood graph attack in which adversaries possess a subgraph formed from a specific vertex and its neighborhoods in d-hops distance. Another attack assume that an adversary can collect a relatively large graph whose memberships partially overlap with the original graph [7].

In our work, we consider subgraph attack model which is similar to Neighborhood Attack Graph (NAG) in [8]. Assume that adversary can somehow collect an arbitrary connected graph made from any set of vertices. General speaking, adversary only has partial neighborhoods of a target user/vertex but he may know neighborhoods of the target's neighborhoods. Our model does not limit vertices having relationship with the target vertex.

4.2 Definition

The conventional DP limits the adversaries' ability to conclude which neighbor database the output database comes from. In context of tabular datasets, two datasets are considered as neighbor datasets if they exactly differ from one tuple. This definition is no longer appropriate in the context of graph data. We define a new neighbor graph definition in which a separated entity is a subgraph.

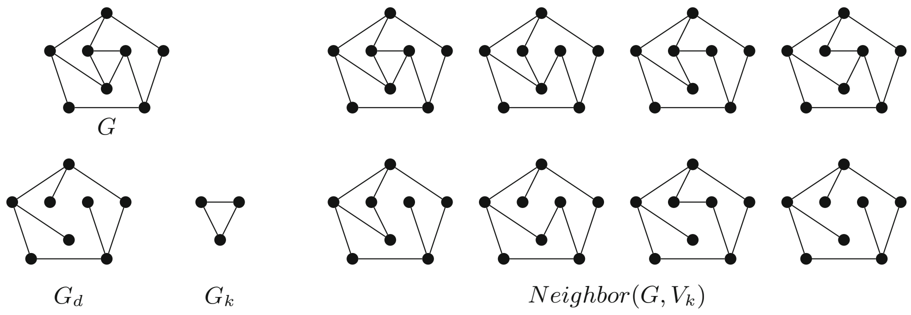


Fig. 1. Example of subgraph-based neighbor graphs

Definition 2. *(Subgraph-based neighbor graphs): Given graph* $G = (V, E)$*, the set of* k *vertices* $V_k \subseteq V$ *and* $E_k \subseteq E$ *is set of edges between vertices in* V_k*.* $Neighbor(G, V_k)$ *is defined as follows:*

$$Neighbor(G, V_k) = \{G_i | \forall G_k \in \mathbb{G}^k, G_i = G_d \parallel G_k\}$$

in which $G_d = G \setminus \{e | e \in E_k\}$

Intuitively, two graphs are neighbors if they are different from exactly one subgraph, given the set of k vertices V_k (Fig. 1).

Definition 3. *(Subgraph-Differential privacy): A randomized function* $\mathcal{K} : \mathbb{G}^n \rightarrow \mathbb{G}^n$ *is* (k, ϵ)*-subgraph-differential-privacy if given graph* $G = (V, E)$*; for all connected subgraph* $G_k = (V_k, E_k)$*, in which* $V_k \subseteq V$ *is a set of* k *vertices,* $E_k \subseteq E$ *is set of edges between vertices in* V_k*; for all pair of graphs* $G_1, G_2 \in Neighbor(G, V_k)$*; for all* $S \in Range(\mathcal{K})$

$$Pr[\mathcal{K}(G_1) \in S] \leq \exp(\epsilon) Pr[\mathcal{K}(G_2) \in S] \quad (3)$$

Subgraph-DP is against subgraph-based attacks. By observing the perturbed subgraph, adversaries cannot figure out which subgraph that observed subgraph comes from with a high confidence.

Privacy parameter ϵ and k control how much privacy leaks. Parameter k is introduced as a new parameter for graph data. Obviously, k measures how large subgraph is. In fact, the graph owners do not need configure a large value of k. We suggest $k = 3$ is enough.

4.3 Mechanism

Consider a given graph $G = (V, E)$ as a complete graph. E_r is the set of real edges in G, $E_r = E$ and E_v is the set of virtual edges which do not exist in G. The underlining idea of our mechanism is very simple. The graph is perturbed by rewiring edges. Rewiring an edge means that the edge changes its state, from real to virtual and vice versa. A set of edges, including real and virtual edges, is selected such that every k-vertices connected subgraph in G is perturbed. Each edge is assigned a weight w. The range of w is $[0, 1]$. The weight measures how "*important*" that edge is. Its weight closes to 0 means that it is not *important* and we can rewire it without too much changes in the graph features. Note that the terminology *important* here is an abstract definition that we define.

Overall, the mechanism comprises three stages. In stage 1, we select a set of edges which are injected noise in the perturbation process. The execution of our mechanism needs some parameters, therefore, these parameters are configured in the stage 2. In the final step, Laplace noise is generated and injected to weight of selected edges. Noise makes a change in the *importance* of an edge to graph features toward two tendencies, namely the edge becomes more or less *important.* A pre-defined threshold θ is used to decide whether an edge is rewired or not. We will explain in detail in the rest of this section.

Algorithm 1. Subgraph-DP Mechanism

```
 1: procedure PerturbGraph($G$,$k$,$\epsilon$)
 2:     $E_s$ = SelectEdges($G$,$k$)
 3:     $\sigma = -\frac{1}{\ln(\frac{2}{\exp(\epsilon_i)+1})}$          ▷ ($\epsilon_i = \frac{\epsilon}{N_k}$ and $N_k = \frac{k(k-1)}{2}$)
 4:     $M$ = ComputeMetric($G$)
 5:     $C$ = ComputeCost($G$)
 6:     $\alpha$ = SetParameter($E_s$,$\sigma$)
 7:     $M$ = Scale($M$,$\alpha$)          ▷ Re-compute metric $M$
 8:     $G'$ = AddNoise($G$,$E_s$,$\sigma$,$\theta = 0$)
 9:     Return $G'$
10: end procedure
```

Selection Strategy. For every vertex $v \in G$, we construct Breadth First Search (BFS) on G, starting from v. Because we consider subgraphs with exactly k vertices, we only need traversal the vertices within $(k-1)$-hops distance from v. The purpose of constructing BFS with root v is that we want to select edges which have one vertex is v, and put them in E_s. Certainly, E_s should be minimum while it still guarantees the condition. However, it is not trivial task. In our work, we propose a flexible selection strategy which does not give the optimal set but it is simple and still satisfies the condition.

For each BFS, if all real edges are selected, the condition satisfies certainly. But the graph features may change significantly due to a large amount of edges are deleted without any compensation of new edges introduced in the perturbed graph. Therefore, virtual edges should be selected even though only real edges are enough. Random selecting is the most simple way that we can consider. Selecting the edges having maximum or minimum weight among candidates is another strategy. This strategy, actually, depends on which feature we want to preserve.

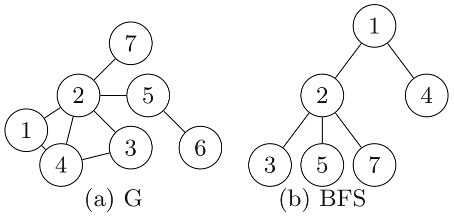


Fig. 2. An example of the edges selection. (a) The original graph G; (b) The BFS starts from 1

We introduce β, the ratio between the number of real edges and that of virtual edges when the selection performs in a particular BFS. It is noted that neither the ratio always is β in each BFS nor after the whole selection.

Figure 2 is an example of a graph G and its BFS starting from vertex 1. Every vertices, excluding 1, in BFS are candidates for selection. If the random selection strategy is used and $\beta = 1$, two edges in $\{(1,3),(1,5),(1,7)\}$ are selected randomly; for instance, $(1,3)$ and $(1,5)$ are selected. In summary, from this BFS, $\{(1,2),(1,4),(1,3),(1,5)\}$ are supplemented to E_s.

Graph Perturbation. In the final step, an appropriately pre-defined random noise will be injected into weight of each $e \in E_s$. For each edge, if the new weight is smaller than threshold θ that edge is rewired and remains unchanged if the new weight is larger than θ. The random noise follows Laplace distribution with $\mu = 0$ and σ depends on the privacy budget ϵ.

The injected noise should be large enough to guarantee subgraph-DP Eq. (3). Theoretically, we can select θ in the range of $[0,1]$, which is the range of weight. However, we fix $\theta = 0$ for following reason. Intuitively, according to the definition of subgraph-DP, when ϵ is large, injected noise becomes small, whereby the graph feature changes a little bit. Given large enough ϵ, G' should be the same as G, which means no edge or just very little edges are injected noise. If $\theta \in [0,1]$, even though given large enough ϵ, a fixed part of edges is still injected noise. This seems quite odd.

Theorem 2. *Algorithm 1 satisfies Subgraph-DP given privacy parameter* ϵ

Proof. SELECTEDGES(G,k) in line 2 guarantees that every k-vertices connected subgraph in G is perturbed as the above explanation.

Consider a set of k vertices V_k and $G_1, G_2 \in \mathit{Neighbor}(G, V_k)$ and $S \in \mathit{Range}(\mathcal{K})$.

$$Pr[\mathcal{K}(G_1) \in S] = \Pi_{i=1}^{N_k} Pr_{G_1,i}$$

in which $N_k = \frac{k(k-1)}{2}$ and $Pr_{G_1,i}$ is probability of the i^{th} edge changing its state in G_1 to that in G'

Given p_i is probability of rewiring the i^{th} edge.

$$p_i = Pr[w_i + noise \leq \theta] = Pr[Lap(\sigma) \leq \theta - w_i]$$

$\theta = 0$ and noise is Laplace noise $Lap(\sigma)$ with $\mu = 0$

Basically, p_i is the cumulative distribution at $(\theta - w_i)$

$$p_i = \frac{1}{2}\exp(-\frac{w_i}{\sigma})$$

For the i^{th} edge, $p_i \leq \frac{1}{2}$ because $w_i \in [0,1]$, therefore

$$\frac{Pr_{G_1,i}}{Pr_{G_2,i}} \leq \frac{1-p_i}{p_i} = \frac{1}{\frac{1}{2}\exp(-\frac{w_i}{\sigma})} - 1 \leq \frac{1}{\frac{1}{2}\exp(-\frac{1}{\sigma})} - 1 = \exp(\epsilon_i)$$

$$\frac{Pr[\mathcal{K}(G_1) \in S]}{Pr[\mathcal{K}(G_2) \in S]} = \frac{\Pi_{i=1}^{N_k} Pr_{G_1,i}}{\Pi_{i=1}^{N_k} Pr_{G_2,i}} \leq \Pi_{i=1}^{N_k}\exp(\epsilon_i) = \exp(\Sigma_{i=1}^{N_k}\epsilon_i) = \exp(\epsilon)$$

Parameter Setting. Given w is weight of the edge (i,j), regardless real or virtual edge, between two vertices i and j, w is computed from metric m of this edge.

$$w(i,j) = -\sigma \log(m(i,j)) \tag{4}$$

The metric $m(i,j)$ measures how strong the connection between i and j is. We introduce the normalized mutual friends as the metric.

$$m(i,j) = \frac{[\#mutualfriends]}{2}\left(\frac{1}{deg(i)} + \frac{1}{deg(j)}\right) \tag{5}$$

in which $deg(i)$ is node degree of vertex i.

Note that different metrics have different specific ranges, however, the range of the weight is fixed in $[0,1]$. Therefore, we have to translate the original range of metric to a new range $[\exp(-\frac{1}{\sigma}), 1]$.

We define a *target equation* which decides which feature we want to preserve in the perturbed graph.

$$\sum_{e \in E_s \cap E_r} p_e c_e = \sum_{e \in E_s \cap E_v} p_e c_e \tag{6}$$

in which, p_e is the probability of rewiring an edge and c_e is the cost for rewiring that edge.

The cost c_e measures how much an edge impacts a specific graph feature. The target equation means that the cost for deleting the edges should be the same as the cost for adding new edges. In fact, appraising judiciously the cost c_e is not trivial in some cases. We introduce two case studies.

Case 1: Preserving Average Node Degree. It is easily seen that the cost for deleting an edge and adding an edge is the same. Therefore, the target function is re-written as follow:

$$\sum_{e \in E_s \cap E_r} p_e = \sum_{e \in E_s \cap E_v} p_e \tag{7}$$

Case 2: Preserving the Number of Triangles. Triangles play an important role in graph analysis. If an edge (i, j) is deleted or added, the decrease/increase in the number of triangles exactly equals the number of mutual friends of i and j. Therefore, we can use the number of mutual friends as the cost of deleting/adding an edge. This cost, in fact, does not assess thoroughly triangle counting. However, our experimental results prove that the number of mutual friends is accurate enough to achieve triangle counting preservation if the number of the selected real and virtual edges approximatively are the same.

$$\sum_{e \in E_s \cap E_r} p_e mu_e = \sum_{e \in E_s \cap E_v} p_e mu_e \tag{8}$$

in which mu_e is the number of mutual friends of two vertices connected by edge e.

Unfortunately, it is difficult to achieve both Eqs. (7) and (8) in practice. To guarantee the target equation has a solution, the naive method we can consider is to scale again the range of metric to new range $[\exp(-\frac{1}{\sigma}), \alpha]$ such that the target equation satisfies. The new range is also a subset of $[\exp(-\frac{1}{\sigma}), 1]$. Solving that condition, we can get ϵ_{min} such that for $\epsilon \geqslant \epsilon_{min}$, the target equation has a solution. We do not describe in detail how to compute α and ϵ_{min} here because of the space limitation.

5 Evaluation

We collect real graphs to demonstrate that our proposed privacy definition and mechanism work well in practice, guarantee privacy while is useful in analysis. We run the mechanism in both cases: preserving average node degree and preserving triangle counting and then measure the features of both the original graphs and the perturbed graphs to evaluate the differences between them. Note that we do not consider the case of preserving number of triangles in directed graphs. We implement our mechanism using NetworkX which is a Python language software package for processing complex networks.

We use the data sets that are available on https://snap.stanford.edu/data/. The data sets and their characteristics are described in the Table 1.

Table 1. Data sets and their characteristics

Data set	Type	#nodes	#edges	Average node degree	#triangles	Clustering coefficient	Power law
Facebook	Undirected	4039	88234	43.69	1612010	0.61	1.2588
Twitter	Directed	81306	1768149	43.49	-	-	1.3189
DBLP	Undirected	425957	1049867	6.62	2224385	0.63	1.4803
Enron	Undirected	36692	183831	10.02	727044	0.50	1.5127
Stanford	Directed	281903	2312497	16.41	-	-	1.4367

"-" indicates that this value cannot be specified. We do not consider the number of triangles and thus clustering coefficient in directed graphs.

5.1 Preserving Average Node Degree

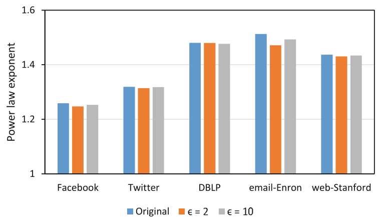
Table 2 shows the average node degree of the perturbed graphs with $k = 3$ and $\beta = 0.5$. The average node degree in all cases is preserved in general. The main trend is that the average node degree of perturbed graphs is slightly higher than that of the original graph.

Table 2. Average node degree of perturbed graphs

ϵ	Facebook	Twitter	DBLP	Enron	Stanford
1	43.54	44.45	6.99	10.93	16.57
2	43.56	44.27	6.90	10.74	16.55
3	43.81	44.09	6.85	10.59	16.53
4	43.81	43.95	6.80	10.49	16.49
5	43.67	43.87	6.75	10.35	16.46

Simultaneously, we also compute other statistical graph features to verify that how our mechanism influences other features while preserving average node degree. We compute the power law exponent and the clustering coefficient.

Even though power law exponent is also preserved in this case, the clustering coefficient changes much (Fig. 3). Note that in case of preserving average node degree, we expect the number of added and deleted edges are relatively similar. Due to the cost of real and virtual edges are generally different, namely deleting a real edge tends to lose more triangles than an new edge brings in. Therefore the number of triangles decreases dramatically, especially with a small ϵ.



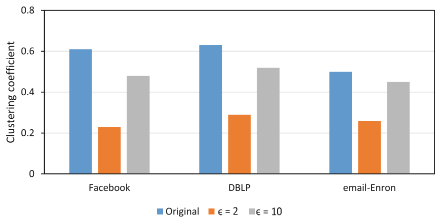


Fig. 3. Power law exponent and clustering coefficient of perturbed graphs in case of preserving average node degree

5.2 Preserving Triangle Counting

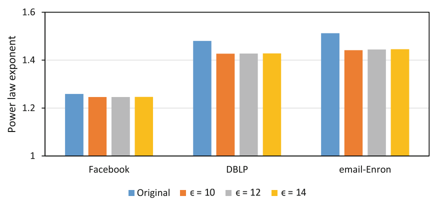
Table 3 shows the changes of the perturbed graphs in clustering coefficient. Three graphs are preserved with respect of clustering coefficient. However, all three graphs incur a relatively high ϵ_{min}.

Table 3. Clustering coefficient of perturbed graphs

ϵ	Facebook	DBLP	Enron
6	0.50	-	-
7	0.51	-	-
8	0.51	-	-
9	0.51	0.60	-
10	0.52	0.61	0.53
11		0.61	0.53
12		0.61	0.53
13		0.62	0.53
14			0.53

"-" indicates that this value is not specified because of ϵ_{min} or we do not measure

Similar to the case of preserving average node degree, power law exponent is also preserved in this case (Fig. 4). All graphs have slightly higher average node degree because the cost of a selected real edge in general is higher than that of a selected virtual edge because two vertices with relationship tend to have more mutual friends than two vertices without relationship. If we want to preserve the number of triangle, we have to add more new edges to make sure that the cost for adding and deleting are the same. However, we believe that the difference in the average node degree is acceptable.



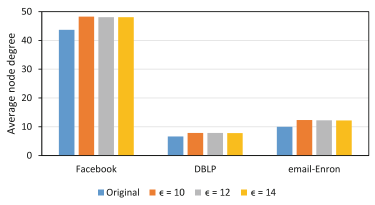


Fig. 4. Power law exponent and average node degree of perturbed graphs in case of preserving the number of triangles

6 Conclusions and Future Work

In this study, we propose a novel privacy framework, subgraph-DP, which is based on definition of differential privacy. Subgraph-DP is a robust framework for graph data where entities have relationships with others. Subgraph-DP is against subgraph-based attacks. We also propose a mechanism which gives subgraph-DP. We introduce the mechanism in two cases: preserving average node degree and preserving the number of triangles. The perturbed graph preserves most of the statistical features of graph. The database owners can appropriately adapt subgraph-DP for their purposes.

However, our work incurs some limitations. Firstly, ϵ_{min} in case of preserving the number of triangles is high, which means we cannot protect much privacy in these cases. Secondly, measuring how much an edge effects on specific graph feature is a challenge, especially with complex features such as spectral analysis and node degree distribution. Thirdly, our mechanism works well with small and medium graphs, however, with large graphs running time reaches several hours. As the future work, we plan to overcome these limitations.

Acknowledgments. We are grateful to anonymous reviewers who give us helpful advices, support us improve our work. This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. H0510-14-1004, Subgraph differential privacy for graph data publishing)

References

1. 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)
2. 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)
3. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: ICDE (2008)
4. Ahmed, F., Jin, R., Liu, A.X.: A random matrix approach to differential privacy and structure preserved social network graph publishing. ArXiv preprint (2013). arXiv:1307.0475
5. Sala, A., Zhao, X., Wilson, C., Zheng, H., Zhao, B.Y.: Sharing graphs using differentially private graph models. In: IMC, pp. 81–98 (2011)
6. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: S&P, pp. 111–125 (2008)
7. Nilizadeh, S., Kapadia, A., Ahn, Y.Y.: Community-enhanced De-anonymization of online social networks. In: CCS (2014)
8. Cheng, J., Fu, A.W.C., Fu, J.: K-Isomorphism: privacy preserving network publication against structural attacks. In: SIGMOD (2010)
9. Jamali, M., Ester, M.: TrustWalker: a random walk model for combining trust-based and item-based recommendation. In: KDD, pp. 397–406, Paris, France (2009)
10. Alvisi, L., Clement, A., Epasto, A., Lattanzi, S., Panconesi, A.: The evolution of sybil defense via social networks. In: S&P (2013)
11. Harel, D., Koren, Y.: On clustering using random walks. In: Hariharan, R., Mukund, M., Vinay, V. (eds.) FSTTCS 2001. LNCS, vol. 2245, pp. 18–41. Springer, Heidelberg (2001)
12. Kasiviswanathan, S.P., Nissim, K., Raskhodnikova, S., Smith, A.: Analyzing graphs with node differential privacy. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 457–476. Springer, Heidelberg (2013)

13. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: SIGMOD, pp. 19–30 (2009)
14. Le Ny, J., Pappas, G.J.: Differentially private filtering. In: CDC, pp. 3398–3403 (2012)
15. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-interscience, Hoboken (2006)
16. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentiallyprivate histograms through consistency. Proc. VLDB Endow. **3**(1–2), 1021–1032 (2010)
17. Chen, S., Zhou, S., Bhowmick, S.S.: Integrating historical noisy answers for improving data utility under differential privacy. In: EDBT, pp. 62–73 (2012)