Publishing Attributed Social Graphs with Formal Privacy Guarantees

Zach Jorgensen
North Carolina State University, USA
zjorgen@ncsu.edu

Ting Yu
Qatar Computing Research Institute, Qatar
tyu@qf.org.qa

Graham Cormode
University of Warwick, UK
g.cormode@warwick.ac.uk

ABSTRACT

Many data analysis tasks rely on the abstraction of a graph to represent relations between entities, with attributes on the nodes and edges. Since the relationships encoded are often sensitive, we seek effective ways to release representative graphs which nevertheless protect the privacy of the data subjects. Prior work on this topic has focused primarily on the graph structure in isolation, and has not provided ways to handle richer graphs with correlated attributes.

We introduce an approach to release such graphs under the strong guarantee of differential privacy. We adapt existing graph models, and introduce a new one, and show how to augment them with meaningful privacy. This provides a complete workflow, where the input is a sensitive graph, and the output is a realistic synthetic graph. Our experimental study demonstrates that our process produces useful, accurate attributed graphs.

1. INTRODUCTION

Social network analysis (SNA) is an important tool with diverse applications, from marketing to counter-terrorism to the prediction of disease outbreaks. Recent years have seen massive growth in online social networking, which has created unique opportunities for SNA at unprecedented scales. Although the vast stores of social data hold great potential for research, they also present significant concerns for privacy. Social networks encode complex relationships among individuals (e.g., friendships, aquaintances, sexual relationships, disease transmission), which may be sensitive. Moreover, the nodes (e.g., users) in real-world social networks may be associated with various sensitive attributes, such as age, location or sexual preference. In order for the full benefit of SNA to be realized, effective privacy-preserving analysis techniques for this type of data are critical.

Much prior research on privacy for SNA has focused on graph anonymization techniques (e.g., [17, 38, 39]). Unfortunately, such approaches are now known to be vulnerable to deanonymization attacks (e.g., [1, 24]). Consequently, recent efforts have sought to provide the more rigorous guarantees of differential privacy for graph analysis. Two directions have emerged: (1) methods for privately computing specific statistics over sensitive graphs, such as degree distribution [11], subgraph counting [13, 2], clustering coefficient [35], and frequent subgraph mining [33]; and (2) private graph release, which typically involves (privately) fitting a generative graph model to input graphs in order to sample a synthetic graph, which can be used in analyses as a proxy for a real input graph (e.g., [23, 30, 34, 4, 36, 18, 29]). We follow this latter approach. Specifically, we address the important challenge of privately modeling social graphs in which the vertices have *sensitive attributes* that are correlated with the structure of the graph. Our goal is to generate synthetic graphs that mimic an input graph in terms of these important characteristics and that can be published without compromising the privacy of the individuals and relationships described in the input graph.

Three important motivations for producing a synthetic graph are highlighted by Hay [9]: (1) first, some analyses require running complex algorithms or simulations over actual network data, rather than via statistics. Using synthetic graphs in place of the true input allows an analyst to protect privacy without needing to analyze the privacy properties of each algorithmic approach; (2) synthetic graphs facilitate "exploratory, open-ended, and iterative" analyses, which can be very difficult to accommodate directly on an input graph with limited privacy budget under differential privacy; and (3) it eliminates the need to share the details of proprietary analyses with the data owner.

Existing work on differentially private graph learning and synthesis (cited above) has focused on modeling network structure alone, without taking into account *vertex attributes* and their correlations with graph structure. However, real-world graphs have vertex attributes and do exhibit such correlations. For example, in social graphs, user nodes often have many attributes such as age, gender, sexual preference, etc. and are well known to exhibit *homophily*, or the tendency for nodes with similar attributes to form connections [20]. Existing differentially private graph models do not capture such properties. However, there are many relevant analyses that rely on the presence of these correlations. In the field of relational machine learning, such correlations are exploited to predict missing or future attribute values; information diffusion in social networks has been shown to be influenced by both the graph structure as well as the attributes of the nodes.

In this work, we develop a differentially private framework for synthesizing attributed social graphs to mimic the structure and attribute correlations observed in an input graph. Our solution prevents disclosure of individual relationships (edges) and node attribute values associated with the individuals in the input graph. This is the first differentially private approach that targets both structural properties and attribute correlations.

To this end, we adapt the recent Attributed Graph Model (AGM) of Pfeiffer et al. [27] to add strong differential privacy guarantees

SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-3531-7/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2882903.2915215

while preserving the utility of the synthesized graphs. AGM models a social network using three sets of parameters that describe (1) the distribution of attributes over the nodes, (2) the correlations between node attributes and edges, and (3) the modeling parameters for an underlying generative structural model. We show how to effectively compute each of these model parameters under differential privacy. While satisfying differential privacy for the attribute distribution is relatively straightforward, the other two sets of parameters present a significant challenge. Modeling the attribute correlations involves computing the fraction of edges that exist between nodes with every possible attribute configuration. Changing the attribute values associated with a single node can change many of these values, depending on the degree of the node. This translates to a high global sensitivity and standard techniques, such as the Laplace mechanism, will destroy utility. We propose a novel application of the the notion of edge truncation to avoid this problem, and show that it is superior to the more standard approaches of smooth sensitivity and sample-and-aggregate.

To capture the structural properties of the input graph, AGM incorporates an underlying generative structural model. Although several prior works have focused on extending existing structural models to satisfy differential privacy, none of these models sufficiently capture properties of typical networks, which are characterized by heavy tailed degree sequences and large amounts of clustering. Moreover, methods that do attempt to capture this structure appear less amenable to differential privacy. In response, we propose a *new* structural model, called TriCycLe, that captures the degree distribution and clustering coefficients of a social network, and is parameterized by statistics for which good differentially private estimators can be found. Specifically, TriCycLe extends the simple Chung-Lu random graph model with a triangle construction phase, in which existing edges in the Chung-Lu graph are iteratively rewired to create approximately the same number of triangles observed in the input graph. The model parameters are the triangle count and degree sequence of the input graph, both of which can be accurately estimated under differential privacy. We integrate this new model into AGM and outline the end-to-end workflow for generating differentially private synthetic graphs.

We evaluate the efficacy of the proposed framework on four real-world social network datasets and demonstrate that it is able to produce synthetic graphs that accurately mimic the structure and attribute correlations of an input graph, while also ensuring a strong level of privacy.

Outline. In the next section, we describe our problem and setting in more detail and introduce notation; we also provide an overview of AGM, the attributed graph model that we adapt to satisfy differential privacy. In Section 3 we present our approach for computing differentially private estimates of the three sets of modeling parameters used by AGM. In Section 4 we put the pieces together and outline the end-to-end workflow for synthesizing differentially private attributed graphs. Section 5 presents our experimental study on four real-world datasets. Related work and conclusions are in Sections 6 and 7 respectively.

2. PRELIMINARIES

2.1 Setting and Notation

We represent a social network (social graph) as an attributed simple graph (i.e., undirected, unweighted edges with no multiple or self-edges) denoted by the triple $G = (N, E, X)$. We use the terms network and graph interchangeably in the rest of the paper. $N = \{v_1, \ldots, v_n\}$ is a set of vertices (nodes), where each node typically represents a person or user. We assume that the number of nodes, n, is publicly known. $E = \{e_1, \ldots, e_m\}$ is a set of edges, where we write e_{ij} to refer to an undirected edge between nodes v_i and v_j (i.e., $e_{ij} \equiv e_{ji}$ and $e_{ij} \in E$ iff $e_{ji} \in E$). We write $\Gamma(v_i)$ to mean the set of v_i's neighbors, i.e., $\Gamma(v_i) = \{v_j \in N | e_{ij} \in E\}$, and we use $d_i = |\Gamma(v_i)|$ to denote the degree of v_i. $X = \{x_1, \ldots, x_n\}$ is a set of w-dimensional attribute vectors, where the vector $x_i = \langle x_{i1}, \ldots, x_{iw} \rangle$ contains the attributes associated with node v_i. In what follows, we assume that attributes are binary, i.e, $x_i \in \{0,1\}^w$; however, the ideas easily extend to attributes with larger domains.

Table 1: Summary of notation

N	Set of nodes, $\{v_i\}$
n	Number of nodes, i.e., $\lvert N \rvert$
$\Gamma(v_i)$	Set of nodes incident to node v_i
d_i	Degree of node v_i
d_{max}	Maximum degree among nodes in a graph
E	Set of edges, $\{e_{ij}\}$
e_{ij}	An (undirected) edge between v_i and v_j
E_{ij}	Binary random variable, $E_{ij} = 1$ if $e_{ij} \in E$
m	Number of edges, i.e., $\lvert E \rvert$
X	Set of attribute vectors, $\{x_i\}$
x_i	Attribute vector associated with node v_i
x_{ij}	The jth attribute in $x_i \in X$
w	Number of attributes
Y_w	Set of elements representing possible node attribute configurations
f_w	Function that maps an x_i to an element of Y_w
Y_w^F	Set of elements representing possible edge attribute configurations
F_w	Function that maps an e_{ij} to an element of Y_w^F
Θ_X	Set of attribute parameters used by AGM
Θ_F	Set of attribute correlation parameters used by AGM
M	Underlying structural model used by AGM
Θ_M	Set of structural model parameters
k	Truncation parameter
ε	Privacy parameter
n_Δ	Number of triangles in a graph
$\mathcal{S}$	Unordered degree sequence of a graph

Table 1 summarizes the notation introduced above, as well as other notation frequently used throughout this paper. We sometimes augment the notation with a superscript to explicitly connect the notation with a specific graph, e.g., $N^{(G_1)}$ refers to the set of nodes of graph G_1 and $d_i^{(G_1)}$ to the degree of node $v_i \in N^{(G_1)}$.

2.2 Attributed Graph Model (AGM)

In this work, we extend the recent Attributed Graph Model (AGM) of Pfeiffer et al. [27] to enable generation of synthetic graphs with formal differential privacy guarantees. AGM is capable of capturing both the structure and attribute correlations of a given input graph and efficiently sampling graphs from the learned model.

To simplify notation in the rest of the paper, we first define two functions to map a node attribute vector or an undirected edge to a unique integer. We use $f_w(x_i)$ to denote the function that maps maps a w-dimensional attribute vector x_i bijectively to an element of the set Y_w. We write $F_w(x_i, x_j)$ to mean the function that maps a pair of attribute vectors x_i, x_j associated with an edge to an element of Y_w^F, ignoring the direction of the edge.

AGM models graphs using parameters Θ_X, Θ_M, and Θ_F:

1. A set of attribute parameters Θ_X is used for modeling the distribution of attributes on the vertices, i.e., $\Pr[X|\Theta_X]$. Θ_X is computed from graph G as: $\Theta_X(y) = \frac{\sum_{v_i \in N}[f_w(x_i)=y]}{n}, \forall_{y \in Y_w}$, where $[\cdot] = 1$ if the statement within the brackets evaluates to

true, and 0 otherwise. In other words, $\Theta_X(y)$ is the fraction of total nodes whose attribute vectors are encoded as y.

2. A set of edge parameters Θ_M models the distribution of edges, $\Pr[E|\Theta_M]$. This uses an appropriate underlying generative structural model, M, parameterized by Θ_M. In general, M can be any generative model in which a graph can be generated by repeatedly sampling edges from the conditional distribution $\Pr[E_{ij}|\Theta_M, M]$, where $E_{ij} \in \{0,1\}$ denotes a binary random variable such that $E_{ij} = 1$ means $e_{ij} \in E$. Note that graphs generated by M are independent of X (i.e., the node attribute vectors).

3. Θ_F is the set of parameters for modeling the correlations between attributes and edges, $\Pr[F_w(x_i,x_j)|E_{ij} = 1, \Theta_M, \Theta_F]$. We can construct Θ_F from G as: $\Theta_F(y) = \frac{\sum_{e_{ij}\in E}[F_w(x_i,x_j)=y]}{m}$, $\forall_{y\in Y_w^F}$. In other words, $\Theta_F(y)$ is the fraction of all edges that connect two nodes whose associated pair of node attribute vectors is encoded as y.

Given the three sets of parameters, the procedure to generate a synthetic graph is relatively straightforward. We create a fresh set of nodes N' of the desired size, and sample a new set of attribute vectors X' for the nodes in N' independently from $\Pr[X|\Theta_X]$. Next, we sample a graph G' based on $\Pr[E_{ij} = 1|F_w(x'_i,x'_j), \Theta_M, \Theta_F]$. It is not convenient to sample from this distribution directly, so instead the approach of accept/reject sampling can be used. Essentially, the randomly produced attribute correlations observed in the generated graph are used to derive acceptance probabilities, which are in turn used to generate a new output graph [26].

2.3 Differential Privacy

The tools of differential privacy [7, 8] are by now well-known, so we give only a short summary of the main concepts. It provides a statistical notion of privacy, based on the concept of *neighboring data sets*, which differ in at most one element. We write $D \sim D'$ to indicate that datasets D and D' are neighbors.

A randomized algorithm (mechanism) $\mathcal{M}$ is said to be *differentially private* (DP) if the probability distribution $\mathcal{M}(D)$ on any data set D is approximately the same as $\mathcal{M}(D')$, up to a factor of e^{ε}, for every $D \sim D'$. In other words, the algorithm's behavior should be (mostly) insensitive to the presence or absence of any one tuple in the input. Here, $\varepsilon > 0$ is a publicly known *privacy parameter* that controls the strength of the differential privacy guarantee: a larger ε yields weaker privacy, while a smaller ε leads to stronger privacy.

For real valued functions, i.e., $f : \mathcal{D} \to \mathbb{R}^d$, the most common way to satisfy differential privacy is to inject carefully chosen random noise into the output. The magnitude of the noise is adjusted according to the *global sensitivity* of the function, defined as $\Delta_f = \max_{D\sim D'} \|f(D) - f(D')\|_1$ ($\|\cdot\|_1$ is the L_1 norm). This measures the maximum extent to which any one tuple in the input can affect the output. A function f can be made ε-differentially private by adding random noise drawn from the Laplace distribution with mean zero and scale $\lambda = \frac{\Delta_f}{\varepsilon}$, denoted $\mathrm{Lap}(\lambda)$, to its output.

Global sensitivity can be too blunt a tool when a few rare inputs can push up its value. *Smooth sensitivity* adapts the notion of sensitivity to the given data [25]. For $\beta > 0$, the β-smooth sensitivity of a function f at D is $S_{f,\beta} = \max_{D'\in\mathcal{D}} \left(LS_f(D') \cdot e^{\beta d(D,D')}\right)$ where $d(D,D')$ is the distance between D and D', i.e., the number of additions or deletions needed to turn D into D'. Adding noise from $\mathrm{Lap}\left(\frac{2S_{f,\beta}(D)}{\varepsilon}\right)$ satisfies the slightly weaker model of (ε,δ)-differential privacy, where $\beta = \varepsilon/2\ln(1/\delta)$ [25]. Smooth sensitivity has proven useful for certain tasks, such as the median and minimum spanning tree [25] and various graph analysis tasks [35]. But it is non-trivial to compute for arbitrary functions, in general; moreover, it can also be computationally hard to compute for some functions (e.g., [18]).

Properties of Differential Privacy. When multiple differentially private outputs are combined, there are simple rules for reasoning about the privacy level that results. The most basic is sequential composition [21]. If we run k mechanisms $\mathcal{M}_1,\ldots,\mathcal{M}_k$ on an input D in sequence and each independently satisfies ε_i-differential privacy, then the full output is ε'-differentially private, where $\varepsilon' = \sum_{i=1}^k \varepsilon_i$. That is, the privacy loss accumulates linearly. More strongly, if the mechanisms are applied to a set of *disjoint* inputs, the resulting sequence of computations is $(\max_i \varepsilon_i)$-differentially private. Lastly, any post-processing of the output of a DP algorithm does not impact the privacy guarantee [15]. Using these composition theorems, one can easily construct complex differentially private algorithms from individual differentially private components.

2.4 Differential Privacy for Attributed Graphs

Differential privacy was originally defined in the context of tabular data, where each row corresponds to an individual, so neighboring datasets differ in the presence of one individual [7, 8]. Translating differential privacy to the graph analysis domain requires defining what it means for two graphs to be neighboring. Most prior work in this area considers two graphs to be neighboring if they differ in the presence of a single edge; that is, given a graph G, the neighboring graph G' is formed by adding (or deleting) any one edge to (from) G. This *edge-differential privacy* [11] definition protects individual relationships from disclosure—an adversary will not be able to determine with high probability whether an edge exists between any pair of nodes (in the input graph).

Unlike most prior work (apart from [2]), we concentrate on graphs that have one or more attributes associated with each node. Edge-differential privacy is only concerned with edges; however, node attributes are typically highly sensitive as well, and should be protected. Thus, we adopt a notion for neighboring graphs from [2].

DEFINITION 1 (EDGE-ADJACENT ATTRIBUTED GRAPHS). *Two attributed graphs G, G' are said to be* edge-adjacent *(or neighboring) if they differ in the presence of a single edge or in the attribute vector associated with a single node.*

A core challenge is that the sensitivities of many graph-structural analyses are proportional to the number of nodes, n, since the impact of a node often depends on its degree, which is bounded by $n-1$. For example, in the subgraph counting task the goal is to count the number of subgraphs of a distinct shape (e.g., triangles) contained in a graph; in the worst case, it is possible to have a graph where the addition of a single edge could complete $n-2$ triangles.

3. COMPUTING MODEL PARAMETERS

In this section, we extend AGM to enforce differential privacy for the synthesized graphs. To do so, we develop procedures for learning differentially private approximations for each of the three model parameters, based on the sensitive input graph. Having approximated the model parameters in a differentially private way, we can then use them with the sampling algorithm (Section 2.2) to obtain a synthetic graph that closely approximates the input graph without loss of privacy. For clarity of exposition, formal descriptions of algorithms and their proofs are postponed to Appendix C.

3.1 Attribute-Edge Correlations

Recall that the attribute correlation parameters Θ_F describe the correlations observed between edges and node attributes in the input graph. Such correlations are a key characteristic of social graphs, which are well-known to exhibit phenomena such as homophily (the tendency for similar individuals to associate with one another). To compute Θ_F, we first find the connection probabilities $\Theta_F(y_j) = \frac{\sum_{e_{ij} \in E}[F_w(x_i,x_j)=y_j]}{m}$, for each $y_j \in Y_w^F$. Since we are assuming undirected graphs with w binary node attributes, we need to compute $y = \binom{2^w+1}{2}$ connection probabilities.

We can view the problem as computing a set of counting queries $Q_F = \{q_1, \ldots, q_y\}$ that count the edges between nodes with a particular attribute configuration y_i, i.e., $q_i = \sum_{e_{jk} \in E}[F_w(x_j,x_k) = y_i]$. Probabilities can be derived from private counts by normalizing. However, while adding or removing a single edge only increases or decreases one count by at most one, changing the attributes of a single node can have a significant impact on many counts (depending on the degree of the node). In the language of differential privacy, this entails a large global sensitivity. In fact, a naïve attempt to satisfy differential privacy through a direct application of the Laplace mechanism (Section 2.3), e.g. by computing Q_F and adding independent Laplace noise to each q_i will completely destroy the utility of the value. To understand why, consider an input graph containing a degree $n-1$ hub node; changing the attributes of this hub node will cause some subset of the counts to decrease by a total of $n-1$ and another subset to increase by the same amount, which means the global sensitivity is $2n-2$. Therefore, achieving differential privacy with good utility in this setting will require either finding a way to exploit the fact that such worst case nodes are extremely rare in real-world networks, or somehow limiting the worst-case impact of a single node by way of a transformation applied to the input graph.

We introduce a new approach to this problem, based on the idea of edge truncation. This generates a private approximation of the set of connection counts Q_F (i.e., noisy counts). These are then divided by their sum to get an approximation of the probability distribution Θ_F. In Appendix B, we describe two alternate approaches based on the DP techniques of smooth sensitivity and sample-and-aggregate, and show that edge truncation is preferable in practice.

Edge Truncation. Our approach enforces low global sensitivity by truncating high-degree nodes. The general idea of projecting an input graph of arbitrary degree onto the set of k-bounded graphs—that is, the set of graphs with degree at most k—was recently proposed in recent work [14, 2]. Specifically, [2] proposed a general edge truncation operation for transforming graphs in the edge adjacency model, which we extended to our task of computing Q_F.

DEFINITION 2 (EDGE TRUNCATION [2]). *Given a graph G and a truncation parameter $k > 0$, the truncation algorithm $\mu(G,k)$ starts by fixing a canonical ordering over all of the edges in E. Then, iterating through each edge $e_{ij} \in E$ in order, an edge is deleted if and only if $d_i > k$ or $d_j > k$.*

The notion of *restricted sensitivity* in [2] restricts the global sensitivity to consider just the set H_k of graphs with degree at most k. An important subtlety is that transformations applied to the input graph, such as μ from Definition 2, may amplify the restricted sensitivity of a function that is executed on the resulting truncated graph. For a function f with restricted sensitivity $RS_f(H_k)$, it is shown in [2] that the algorithm that applies this truncation operation to G and then runs f on the truncated graph, i.e., $A_f(G,k) = f(\mu(G,k))$, has a global sensitivity of $3 \cdot RS_f(H_k)$. The factor of 3 arises as adding or removing a single edge, prior to truncation, results in a difference of up to three edges in the truncated graph [2].

Our result improves over the general results in [2]. We show that for the special case of Q_F the truncation operation is "for free", resulting in the global sensitivity being equal to the restricted sensitivity, which is $2k$. The intuition is that, although the truncation routine can amplify the effects of an edge addition/deletion in the input graph, the global sensitivity is still dominated by the impact of changing an attribute vector, which is not affected by truncation.

PROPOSITION 1. *Let μ be the edge truncation algorithm of Definition 2 and let $k > 1$. The* global sensitivity $GS(A_{\mu,Q_F}(G,k))$ *of algorithm $A_{\mu,Q_F}(G,k) = f_{Q_F}(\mu(G,k))$, which truncates graph G then computes Q_F over the resulting k-bounded graph, is $2k$.*

PROOF. By Definition 1, neighboring graphs differ in the presence of a single edge *or* in the attributes associated with one node. Let v_i, v_j be two nodes in input graph G such that $d_i = d_j = k$ and no edge exists between v_i and v_j. Let G' be the neighboring graph formed by adding edge e_{ij} to G. Let $O = e_0, e_1, \ldots, e_m$ be the canonical ordering of the edges in G and let $O' = O + e_{ij}$ be the ordering for G'. Finally, assume that $e_0 = e_{rs}$ and $e_1 = e_{uv}$ such that $s \neq u$, $v \neq r$, $d_s \leq k$, $d_v \leq k$ and that $f_w(x_s) \neq f_w(x_v)$ (i.e., their attribute values are different). Observe that the graph resulting from truncating G, $\mu(G,k)$, contains edges e_0, e_1 since the nodes at both end points have degree $\leq k$, by our assumption. On the other hand, truncating G' results in both e_0 and e_1 being deleted, since at the time they are processed, both v_r and v_u have degree $k+1$ in G'. Moreover, by the time edge e_{ij} is considered, both u and v will have degree $\leq k$, so e_{ij} will not be deleted. Also observe that all other edges that are deleted from G are also deleted from G' and vice versa. Consequently, two of the connection counts will decrease by a total of two, while one count will increase by one. Therefore the outputs of $\mu(G,k)$ and $\mu(G',k)$ differ in exactly three edges.

Now consider the case in which G and G' are size n graphs and are identical except that the attribute vector associated with one node, say v_i, is changed from x_i to x_i', such that $f_w(x_i) \neq f_w(x_i')$, in G'. Assume that $d_i \geq k$. First observe that the difference in the attribute vectors has no impact on the edges that are deleted by μ, since μ only looks at the degrees, and after truncation, v_i will have degree k in both graphs. Since the attribute vector associated with v_i is changed in G', the k edges incident to v_i in $\mu(G',k)$ will now contribute to a different subset of the counts than the edges incident to u in $\mu(G,k)$; that is, one subset of the counts will increase by k and another disjoint subset will decrease by k, for a total difference of $2k$. Finally, since $2k > 3$ (for $k > 1$), the impact of changing the attributes of a single node dominates that of adding/removing an edge and we have $GS(A_{\mu,Q_F}) = 2k$. $\square$

We provide pseudocode for this procedure (Alg. 4), and prove its privacy properties (Thm. 7) in Appendix C. Compared to using smooth sensitivity (Appendix B), we expect this approach to perform better when k is close to the true maximum degree, since smooth sensitivity with the Laplace mechanism requires using $\varepsilon/2$, whereas we can use the full ε with this approach. However, directly using the true maximum degree to set k does not satisfy differential privacy. Moreover, social networks tend to have a power law degree distribution, and the average degree is significantly smaller than the maximum degree. Therefore, setting $k \ll d_{\max}$, so that a few very high degree nodes are truncated while most nodes are untouched may give better performance due to the reduced sensitivity.

The best choice for k depends on ε and on the characteristics of the input graph; however, tuning k for the given input would consume privacy budget. Instead, we find that using the data-independent

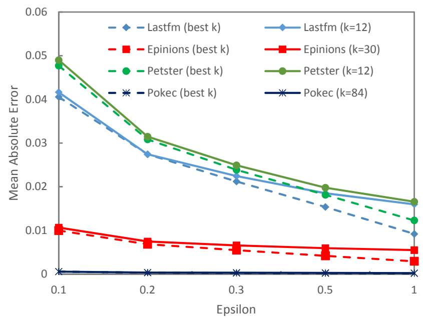


Figure 1: MAE of truncation approach when using the best truncation parameter k (dashed lines) vs. $k = n^{\frac{1}{3}}$ (solid lines).

heuristic $k = n^{\frac{1}{3}}$ works well in practice (a similar heuristic is given in [2]). A preliminary experiment to measure the Mean Absolute Error (MAE) between the input Θ_F and the noisy output over four datasets (described in Appendix A) is shown in Fig. 1. Since real-world social networks tend to have power-law degree distributions—meaning that only a small fraction of the nodes have very high degrees—this choice of k significantly lowers the sensitivity without deleting too many edges. For sufficiently large graphs (e.g., Pokec), the difference in error rates between the best k and the heuristically chosen k is negligible. Note that since n is assumed to be public in this work, setting k as a function of n does not affect the privacy guarantee of Thm. 7.

3.2 Node Attribute Distribution

The distribution of attributes on the nodes is modeled by Θ_X. Recall that Y_w is the set representing the 2^w possible attribute configurations that nodes can take, on a dataset with w binary attributes. We compute Θ_X as: $\Theta_X(y_i) = \frac{\sum_{v_i \in N}[f_w(x_i)=y_i]}{n}$, for $y_i \in Y_w$.

We can view the task of privately computing Θ_X as answering a set of $y = 2^w$ counting queries $Q_X = \{q_1, \ldots, q_y\}$ (where $q_i = \sum_{v_j \in N}[f_w(x_j) = y_i]$) under differential privacy. Here, we are counting disjoint sets of nodes, which has low global sensitivity. Observe that changing the attributes of a single node may, in the worst case, reduce at most one count $q_i \in Q_X$ by one and increase another count q_j by one, which means the global sensitivity is just *2*. We are only counting *nodes*, so adding/removing edges has no impact on the counts. Therefore, we can obtain a private approximation of Θ_X, denoted by $\widetilde{\Theta}_X$, by computing counts on G, adding noise from $\mathrm{Lap}(0, 2/\varepsilon)$, and finally dividing the noisy counts by their sum. As with Θ_F, we clamp the noisy counts to the range $(0, n)$ before dividing by the sum. For completeness, pseudocode of this procedure (Alg. 5) and a proof of its privacy properties (Thm. 8) are provided in Appendix C.2.

3.3 Structural Model: TriCycLe

AGM requires an underlying generative graph model, M, to capture the structural properties of the input graph. M is responsible for proposing new edges, which AGM then accepts or rejects to achieve the desired attribute correlations as it constructs a synthetic graph. An attractive feature of AGM is that it supports the use of any generative structural model in which a graph can be generated by repeatedly sampling edges. Therefore, our objective in this section is to (a) identify an appropriate structural model, and (b) to devise a way to compute its required model parameters, Θ_M, efficiently and accurately under differential privacy. For our purposes, the selected model should (1) capture structural features of social networks, including heavy-tailed degree distribution and high clustering; (2) be reasonably scalable (i.e., able to handle hundreds of thousands of nodes and millions of edges); and (3) be amenable to computation under differential privacy.

Recent work has extended popular structural models to support differential privacy, including exponential random graph models (ERGMs) [18] and the Kronecker graph model (KGM) [23]; however neither of these models meets our needs. ERGMs can reproduce the degree distribution and clustering coefficients of an input graph, but they do not scale well beyond a few thousand nodes. Conversely, KGMs are more scalable but fail to capture the high clustering characteristic of real-world social networks [32, 26].

From the broader graph analysis literature, two candidate models stand out: the Block Two-Phase Erdős-Rényi (BTER) model [31, 16], and the Transitive Chung-Lu (TCL) model [26]. Both allow efficient generation of synthetic graphs that mimic the degree distribution as well as the distribution of clustering coefficients in an input graph. The models both take as input the target degree sequence, and a measure of clustering. At a high level, BTER models a graph as a collection of dense Erdős-Rényi (ER) subgraphs that are sparsely interconnected to form the global graph structure and aims to reproduce the *degree-wise clustering coefficients*, i.e., $C_D = \{c_i\}$, where c_i is the ratio of the number of triangles involving nodes of degree i over the number of length-two paths centered at nodes of degree i. To do so, it requires C_D, along with the degree sequence $\mathcal{S}$, to be measured from the input graph and supplied as input to the algorithm. TCL, on the other hand, aims to reproduce the distribution of *local clustering coefficients* observed in the input graph. It extends the well-known Chung-Lu (CL) random graph model [6] by adding a transitive closure paramater, ρ, which controls the probability with which new edges create triangles (by connecting a node to a random two-hop neighbor), versus connecting two random nodes across the graph. The ρ parameter is a single value learned from the input graph via expectation-maximization.

From the privacy perspective, the degree sequence can be estimated from an input graph with reasonably high accuracy under differential privacy [11]. However, the clustering parameters used by these models are problematic. Under BTER, a single edge may be part of many triangles, and hence many coefficients can be affected. Moreover, adding or removing an edge e_{ij} will change the degrees of nodes v_i and v_j, which could cause a significant change in four different c_i's (i.e., those of the old degrees d_i, d_j and the new degrees $d_i \pm 1, d_j \pm 1$). This translates into a high global sensitivity, and it is unclear whether it is possible to circumvent this issue. Under TCL, the main difficulty comes from the fact that the ρ parameter is tuned through multiple iterations of an expectation maximization algorithm. The tools do not currently exist to bound the impact of such a change on the final value of ρ.

Proposed Approach. Motivated by the difficulty of incorporating differential privacy guarantees into existing generative social network models, we design a new model around a statistic for which accurate differentially private estimators are already known, namely the *triangle count*. Our model, which we call TriCycLe, is inspired by TCL. Rather than learning a transitive closure parameter, TriCycLe first estimates the number of triangles in the input graph and then rewires edges in a Chung-Lu seed graph until the desired number of triangles are present. In addition to being more amenable to differential privacy, we have found that TriCycLe often attains a better fit to the local clustering coefficient distribution than TCL.

In what follows, we give a brief overview of TCL and the CL model that it extends. We then describe how TriCycLe modifies TCL to replace the transitive closure parameter with the triangle count, and how that statistic is used to introduce the expected amounts of clustering into the output graph. The details of how the required model parameters—the degree distribution and number of triangles—can be estimated privately and efficiently are presented in Appendix C.3. Section 4 then explains how TriCycLe can be integrated into AGM as the underlying structural model.

A Review of CL and TCL. In the CL model, a graph is generated by (1) assigning a desired degree to every node based on the degree sequence of the input graph, and (2) sampling edges with a probability proportional to the desired degrees of the endpoint nodes; specifically, an edge e_{ij} is sampled with probability $\frac{d_i d_j}{2m}$. A graph matching a given degree sequence is generated in $O(m)$ steps by repeatedly sampling node pairs v_i, v_j independently from the distribution π, where $\pi(i) = \frac{d_i}{2m}$, and then adding edge e_{ij} to the sample graph. [28] gives an efficient implementation of the above idea, called Fast Chung-Lu (FCL). In FCL, nodes are sampled from π in constant time by constructing a vector of length $2m$ in which the ID of a node v_i is repeated d_i times, and then drawing nodes uniformly from this vector. Although CL therefore matches the desired degree distribution, it does not attempt to reproduce the amount of clustering in the input graph.

TCL extends CL to incorporate transitive edge formation, i.e., when a node connects to a friend of a friend. A transitive closure probability, ρ, controls the probability by which an edge is added between a randomly selected node and a two-hop neighbor (i.e., a friend of a friend), rather than to another randomly selected node, as in the CL model. Given a degree sequence $\mathcal{S}$ and ρ (learned from the input graph), TCL generates a synthetic graph in two phases. First, a *seed graph* of m edges is created using the CL procedure. Then TCL refines the seed graph by introducing triangles while preserving the expected degree sequence. Node v_i is first sampled from the π distribution; then with probability ρ, v_k is selected uniformly from v_i's neighbors and an edge is added between v_i and a uniformly selected neighbor of v_k, v_j (if it doesn't already exist), which results in at least one new triangle. With probability $1-\rho$, the endpoint v_j is sampled from π instead. After each new edge addition, the oldest edge in the graph is deleted to ensure that the expected degree sequence is maintained. The process of replacing edges continues until *all* old (seed) edges have been replaced.

TriCycLe Random Graph Model. Similarly to TCL, TriCycLe is based on the intuition that clustering in social graphs is attributable to triangles that form when a user is linked to a two-hop neighbor (i.e., a friend-of-a-friend). Therefore, TriCycLe attempts to reproduce the total number of triangles seen in the input graph by connecting nodes in this way. Like TCL, TriCycLe starts with a CL seed graph and iteratively replaces edges to create triangles. However, rather than using a transitive closure probability, we count the number of triangles in the input graph and then add transitive edges until the number of triangles in the resulting graph matches that of the input graph. The number of triangles in a graph can be accurately estimated under differential privacy using a recent technique [37] (Appendix C.3). Like TCL, for each transitive edge we add, we remove a seed edge to maintain the expected degree distribution; however, since the deleted edge may itself be part of one or more existing triangles, we reject proposed replacements that decrease the net triangle count. This check ensures that the algorithm terminates with the desired number of triangles.

Algorithm 1 gives the procedure for generating a graph with TriCycLe. The inputs to the algorithm are the degree sequence $\mathcal{S} = \{d_i | v_i \in G\}$ and the desired number of triangles n_Δ. For now, we assume that these two quantities have been directly measured from the input graph G (i.e., non-privately); in Appendix C.3, we explain how they can be accurately and efficiently estimated with

Algorithm 1 TriCycLe Graph Generation

```
Input: $N, \Theta_M = \{\mathcal{S}, n_\Delta\}$
Output: $E_T$

 1: Compute $\pi$ distribution from $N$ and degree sequence $\mathcal{S}$
 2: $E_T \leftarrow \mathrm{CL}(\mathcal{S}, \pi)$
 3: $\tau \leftarrow \textsc{CountTriangles}(E_T)$
 4: while $\tau < n_\Delta$ do
 5:     $v_i \leftarrow \textsc{Sample}(\pi)$
 6:     $\gamma_i \leftarrow \Gamma(E_T, v_i) \backslash \{v_i\}$
 7:     $v_k \leftarrow \textsc{SampleUniform}(\gamma_i)$
 8:     $\gamma_k \leftarrow \Gamma(v_k)$
 9:     $v_j \leftarrow \textsc{SampleUniform}(\gamma_k)$
10:     if $e_{ij} \notin E_T$ then
11:         $e_{qr} \leftarrow \textsc{OldestEdge}(E_T)$        ▷ Get oldest edge to replace
12:         $CN_{qr} \leftarrow |\Gamma(v_q) \cap \Gamma(v_r)|$ ▷ Count common neighbors of $v_q$ & $v_r$
13:         $E_T \leftarrow E_T \backslash e_{qr}$
14:         $CN_{ij} \leftarrow |\gamma_i \cap \Gamma(v_j)|$     ▷ Would proposed edge decrease $\tau$?
15:         if $CN_{ij} \geq CN_{qr}$ then
16:             $E_T \leftarrow E_T \cup e_{ij}$                ▷ Update triangle count
17:             $\tau \leftarrow \tau + CN_{ij} - CN_{qr}$
18:         else
19:             $E_T \leftarrow E_T \cup e_{qr}$     ▷ Undo edge removal, making $e_{qr}$ the
            youngest edge in $E_T$
20: return $E_T$
```

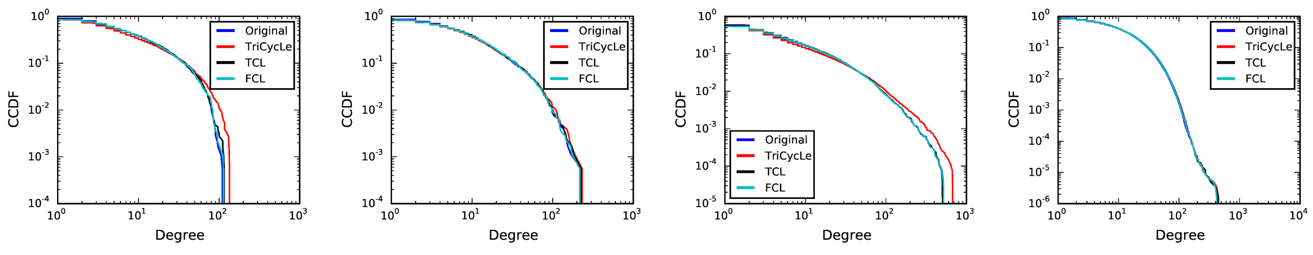
differential privacy. In line 2, a seed graph is sampled from the CL model[1], yielding edge set E_T. Recall that in CL, $m = \frac{1}{2}\sum_{d_i \in \mathcal{S}} d_i$ edges are sampled from the π distribution that is created from $\mathcal{S}$. In the main loop (line 4), we add transitive edges one at a time until E_T contains the desired number of triangles. A new edge is proposed by sampling a node v_i from the π distribution (i.e., with probability proportional to node degree) and then selecting a random "friend of a friend", v_j. Assuming the proposed edge e_{ij} does not already exist in E_T, we select the oldest edge e_{qr} (line 11) to be deleted in order to maintain the expected degree distribution. In lines 12–14 we calculate the number of triangles that would be created (destroyed) by adding (deleting) edge e_{ij} (e_{qr}). Note that we delete e_{qr} in line 13 before computing CN_{ij}, since its presence may affect the number of triangles created by connecting v_i and v_j. As long as the proposed edge addition/deletion does not decrease the number of triangles (line 15), we proceed with adding e_{ij} in line 16 and updating the triangle count in line 17. Otherwise, we undo the deletion of e_{qr} in line 19 before proceeding to the next iteration. Note that whenever the deletion of e_{qr} is reverted, its timestamp is reset so that it becomes the *youngest* edge in E_T; this is an important detail, since without it, the algorithm could get stuck, as there may not (at that moment) be any other edges in the graph that participate in more triangles than e_{qr}.

Extension to Handle Orphaned Nodes. A practical issue with the CL model, as well as TCL and TriCycLe, is that graphs generated by these models often contain many disconnected or *orphaned* nodes[2], compared to the input graph. Some fraction of nodes will not be assigned any edges, or will be connected only to nodes that are themselves disconnected from the rest of the graph. For large social graphs with many low-degree nodes, these models can generate many orphaned nodes. For example, on the large Pokec social network dataset (Appendix A), over 100K of its 720K nodes (many degree-one and degree-two nodes) are orphaned in the output.

We propose an extension to TriCycle to address this issue (it

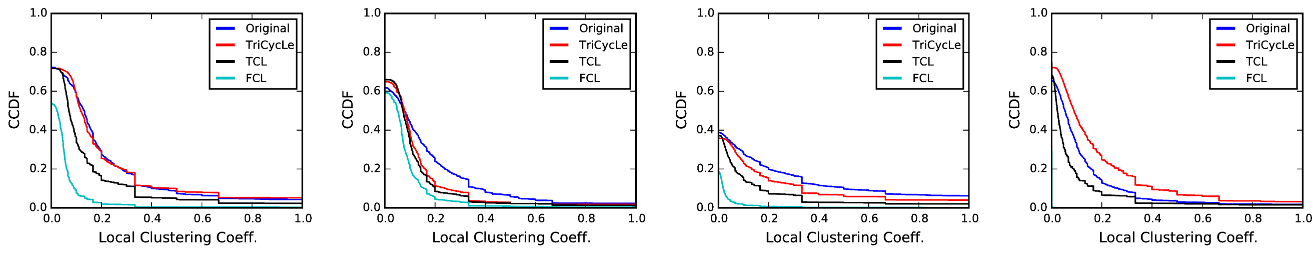
[1]We use the FCL variant with bias correction (cFCL) in [26].

[2]A node is orphaned if it is not part of the main connected component of the graph. In this work, we assume the input graph is connected, and so the output graph should be too.



(a) Last.fm (b) Petster (c) Epinions (d) Pokec

Figure 2: Comparison of structural models in terms of their ability to reproduce the degree distribution



(a) Last.fm (b) Petster (c) Epinions (d) Pokec

Figure 3: Comparison of structural models in terms of their ability to reproduce the distribution of local clustering coefficients.

Algorithm 2 PostProcessGraph

```
Input: $N, E, \pi, \mathcal{S}$
Output: $E'$
 1: $m \leftarrow \frac{1}{2}\sum_{v_i \in N} \mathcal{S}_{v_i}$
 2: $E' \leftarrow E$
 3: $m' \leftarrow |E'|$
 4: while graph $G \leftarrow (N, E')$ is disconnected do
 5:     Select an orphaned node $v_i$
 6:     if $d_i^{(G)} > 0$ then
 7:         $m' \leftarrow m' - d_i^{(G)}$
 8:         Delete edges incident to $v_i$
 9:     for $j \leftarrow 1 \ldots \mathcal{S}_{v_i}$ do
10:         repeat
11:             $v_k \leftarrow$ SAMPLE($\pi$)
12:         until $d_k^{(G)} < \mathcal{S}_{v_k}$
13:         $E' \leftarrow E' \cup e_{ik}$
14:         if $m' = m$ then
15:             Delete a random edge from $E'$
16:         else
17:             $m' \leftarrow m' + 1$
18: return $E'$
```

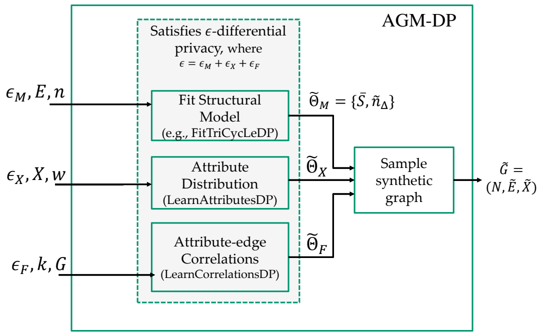
could also be applied to CL and TCL). We observe that (1) most nodes that end up orphaned are those with a degree of one in the input graph, and (2) since degree-one nodes cannot be part of any triangle, there is no reason to pick such nodes in line 5 of Algorithm 1. Let N_1 denote the set of degree-one nodes in the input graph. We make the following changes to Algorithm 1: (1) exclude degree-one nodes from the π distribution computed on line 1; (2) when generating the CL graph in line 2, generate $m - |N_1|$ edges, rather than m edges; (3) wire up the degree-one nodes, and any other orphaned nodes, in a separate post-processing step, given by Algorithm 2. Note that we apply post-processing to both the CL graph generated in line 2 (of Algorithm 1), and to the final output graph, before it is returned in line 20.

Post-processing takes as input nodes N and edges E, along with π and the degree sequence of the original input graph, which corresponds to the *desired* degrees for the nodes[3]. The main loop (lines 4–17) processes each orphaned node v_i by deleting any existing incident edges (which could only be connected to other orphaned nodes), then connecting v_i to the main connected component by adding edges to nodes in the main component (for which their desired degree has not yet been met) until v_i's desired degree is met. Upon adding a new edge, if we have more than the desired number of edges overall, we delete a randomly selected edge (line 15). Note that deleting existing edges (line 15) may result in additional orphaned nodes, therefore, we repeat this process as long as the graph is still disconnected. This process may destroy some triangles, but we found the impact to be minor in practice.

Empirical Evaluation. We validated the (non-private) graph models by generating synthetic versions of four real-world social graphs (see Appendix A) and comparing structural characteristics of the synthesized graphs to those of the original graphs. Figs. 2 (a)–(d) plot the degree distributions for the four datasets (on log-log scale), while Figs. 3 (a)–(d) plot the local clustering coefficient distributions. The y-axes give the complementary cumulative degree distribution (CCDF), i.e., the fraction of nodes with a greater degree or clustering coefficient than the x-value.

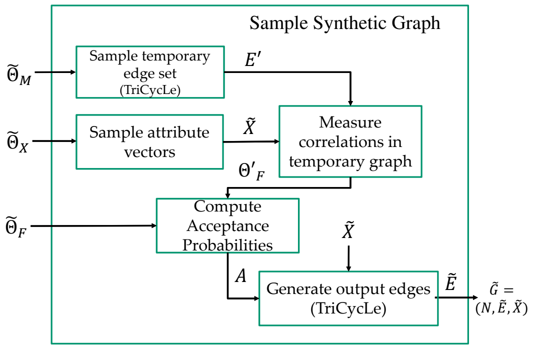
Figures 3(a)–(d) show that, as expected, the clustering coefficient distributions from TCL and TriCycLe were much closer to that of the input than for FCL (which does not model clustering). On the two largest graphs, the clustering coefficients for FCL were all very close to zero. Compared to TCL, TriCycLe more closely appoximated clustering coefficients of Last.fm and Epinions graphs, while its performance was comparable to that of TCL on the Pet-

[3]Degree-one nodes are included in $\mathcal{S}$, but excluded from π.



AGM-DP
Satisfies ϵ-differential privacy, where $\epsilon = \epsilon_M + \epsilon_X + \epsilon_F$
ϵ_M, E, n
ϵ_X, X, w
ϵ_F, k, G
Fit Structural Model (e.g., FitTriCycLeDP)
Attribute Distribution (LearnAttributesDP)
Attribute-edge Correlations (LearnCorrelationsDP)
$\widetilde{\Theta}_M = \{\tilde{S}, \tilde{n}_\Delta\}$
$\widetilde{\Theta}_X$
$\widetilde{\Theta}_F$
Sample synthetic graph
$\tilde{G} = (N, \tilde{E}, \tilde{X})$

(a) Overview of AGM-DP



Sample Synthetic Graph
$\widetilde{\Theta}_M$
$\widetilde{\Theta}_X$
$\widetilde{\Theta}_F$
Sample temporary edge set (TriCycLe)
E'
Sample attribute vectors
$\tilde{X}$
Measure correlations in temporary graph
Θ'_F
Compute Acceptance Probabilities
$\tilde{X}$
A
Generate output edges (TriCycLe)
$\tilde{E}$
$\tilde{G} = (N, \tilde{E}, \tilde{X})$

(b) Overview of Sampling Step

Figure 4: Fig. 4a gives an overview of our DP of AGM. Fig. 4b depicts the sampling step (corresponding to the right-most block in 4a).

ster graph. For Pokec, TriCycLe seems to over-estimate the clustering coefficients. All three models approximate the degree distributions reasonably well (Figs. 2(a)–(d)), though on two datasets, TriCycLe generated slightly more high degree nodes. The bias is a consequence of requiring edge replacements to increase the triangle count; removing edges from high degree nodes tends to remove more triangles, so fewer such edges are removed. There seems to be an inherent tension between the fidelity of the degree distribution and that of the clustering characteristics. Nevertheless, TriCycLe captures both properties sufficiently well for the purposes of this work, which focuses on the privacy aspect; therefore, we defer further analysis and improvement of TriCycLe to future work.

4. AGM-DP: PUTTING IT ALL TOGETHER

In the previous section we described how we model attributes on nodes and the graph structure, and learn the model parameters under DP. In this section, we put the pieces together and outline the end-to-end workflow for synthesizing differentially private graphs that mimic the attribute correlations and structural properties of a given social network. For concreteness, the presented algorithm uses TriCycLe as the structural model; however, other generative structural models (with DP) can be used in its stead.

The original AGM algorithm generates a synthetic graph based on G by repeatedly sampling edges from an underlying structural model M. Each sampled edge is either acceped or rejected to the output graph based on the acceptance probabilities A. This process ends when the desired number of edges (i.e., $m^{(G)}$) are generated. However, TriCycLe (Section 3.3) does not fit this pattern directly, as it continues to refine an initial seed graph by replacing edges to create new triangles until the desired number of triangles exists in the output graph. To accommodate models such as TriCycLe, we move the accept/reject step into the sampling algorithm for the structural model—that is, we still compute acceptance probabilities as in the original AGM algorithm (with the alterations required to satisfy differential privacy), but we then pass the probabilities A into M's specific sampling algorithm when generating the final output graph.[4] Since TriCycLe only replaces edges until the desired number of triangles exist, there may be many edges that are not replaced and thus are not subjected to the acceptance probabilities. Therefore, we also pass the acceptance probabilities into the CL seed graph generator on line 2 of Algorithm 1.

[4]Specifically, we change the condition in the *if* statement on line 10 of Algorithm 1 to "$e_{ij} \notin E_T \wedge \text{SampleUniform}(0,1) \leq A(F_w(\tilde{x}_i, \tilde{x}_j))$", where $\tilde{x}_i$ and $\tilde{x}_j$ are the attribute vectors generated for nodes v_i and v_j in Algorithm 5.

Algorithm 3 AGM-DP-TriCycLe

```
Input: $G = (N, E, X), w, \varepsilon, k$
Output: An $\varepsilon$-DP graph $\widetilde{G} = (\widetilde{N}, \widetilde{E}, \widetilde{X})$
 1: $\widetilde{N} \leftarrow N$
 2: $\varepsilon_X, \varepsilon_F, \varepsilon_M \leftarrow$ SPLITBUDGET($\varepsilon$)    ▷ Split privacy budget $\varepsilon$ among the
    parameters to be learned
 3: $\widetilde{\Theta}_X \leftarrow$ LEARNATTRIBUTESDP($\varepsilon_X, X, w$)  ▷ Learn DP approximations
    for each parameter from input graph $G$
 4: $\widetilde{\Theta}_M \leftarrow$ FITTRICYCLEDP($\varepsilon_M, E$)
 5: $\widetilde{\Theta}_F \leftarrow$ LEARNCORRELATIONSDP($\varepsilon_F, G, k$)
 6: Sample new attribute vectors $\widetilde{X}$ using $\widetilde{\Theta}_X$
 7: $\widetilde{E} \leftarrow$ TRICYCLE($\widetilde{N}, \widetilde{\Theta}_M$)
 8: $A \leftarrow \emptyset$
 9: for Iterations do
10:     Compute $\Theta'_F$ from the graph $\widetilde{G} = \{\widetilde{N}, \widetilde{E}, \widetilde{X}\}$
11:     for $y_i \in Y_F$ do
12:         $R(y_i) \leftarrow \frac{\widetilde{\Theta}_F(y_i)}{\Theta'_F(y_i)}$
13:         if $A_{old} \neq \emptyset$ then
14:             $R(y_i) \leftarrow R(y_i) \times A_{old}(y_i)$
15:     for $y_i \in Y_F$ do
16:         $A(y_i) \leftarrow \frac{R(y_i)}{\text{SUP}(R)}$          ▷ Compute the acceptance probabilities
17:     $\widetilde{E} \leftarrow$ TRICYCLE($\widetilde{N}, \widetilde{\Theta}_M, \widetilde{X}, A$)     ▷ Sample a new edge set using
    acceptance probabilities
18:     $A_{old} \leftarrow A$
19: return $\widetilde{G} = (\widetilde{N}, \widetilde{E}, \widetilde{X})$
```

Our end-to-end differentially private workflow, AGM-DP, is given in Algorithm 3 and illustrated in Fig. 4. It takes input graph G with w attributes, a global privacy budget ε, and a truncation parameter k. We split the privacy budget ε evenly in our empirical analysis (Section 5), i.e, $\varepsilon_X = \varepsilon_F = \frac{1}{4}\varepsilon$, $\varepsilon_M = \varepsilon_S + \varepsilon_\Delta = \frac{1}{4}\varepsilon + \frac{1}{4}\varepsilon$, which seems to work well in practice; though other strategies could also be used. In lines 3–5, we learn the three sets of modeling parameters from the input graph using their respective differentially private learning procedures. Note that after line 5 we never look back at the raw input graph. Next we assign attribute vectors to nodes by sampling randomly from $\widetilde{\Theta}_X$. Temporary edge set E' is sampled independently of the node attributes. The loop beginning at line 9 computes edge acceptance probabilities by measuring the attribute correlations between Θ'_F in the current graph and $\widetilde{\Theta}_F$ derived from G Finally, new edges are sampled using the modified structural model discussed above (line 17). The process is iterated until the acceptance probabilities A converge (for simple models like FCL, only one iteration is required). In our experimental study, we found that A tended to converge after just a few iterations. The output graph $\widetilde{G}$ approximates a joint sampling of the structural model and the attribute correlations, and satisfies differential privacy.

THEOREM 2. *Algorithm 3 satisfies ε-differential privacy.*

PROOF. The privacy budget ε is split into three parts, $\varepsilon_X, \varepsilon_F, \varepsilon_M$, which are used in the learning procedures for the three sets of model parameters (lines 3–5). The fact that the learning procedures satisfy differential privacy with privacy parameters ε_F, ε_X, and ε_M, respectively, follows from Thm. 8, Thm. 7, and Thm. 9, respectively. After line 5, the algorithm never looks back at the raw input data again. Thus, by sequential composition and post-processing invariance, we have that the algorithm as a whole satisfies ε-DP. □

The time costs of our approach are discussed in Appendix C.4.

5. EMPIRICAL ANALYSIS

We present an experimental study of our DP adaptation of AGM on the four real-world social network datasets described in Appendix A. Our goal is to demonstrate that the synthetic graphs produced by AGM-DP reasonably preserve attribute correlations and structural properties of an input graph under privacy. We instantiate AGM-DP with the differentially private version of TriCycLe (referred to as AGMDP-TriCL henceforth), as well as a DP version of the simple FCL model (AGMDP-FCL henceforth) as the underlying structural models. For the latter, we modified the bias-corrected FCL algorithm (cFCL) described in [26] to take a noisy degree sequence as input, generated using the constrained inference approach of [11], just as was done for TriCycLe (Appendix C.3).

We experimented with four different privacy budgets, $\varepsilon \in \{0.2, 0.3, \ln(2), \ln(3)\}$; for the larger Pokec dataset, we used $\varepsilon \in \{0.01, 0.05, 0.1, 0.2\}$[5]. For AGM-DPTriCL, we divided the overall privacy budget ε evenly among the four model parameters: attribute distribution Θ_X, attribute correlations Θ_F, degree sequence $\mathcal{S}$ and number of triangles n_Δ. For AGMDP-FCL does not use n_Δ, so we allocated half of the budget to $\mathcal{S}$ and split the other half evenly between Θ_F and Θ_X. The truncation parameter, k, used in computing the private attribute correlations, was set to $n^{\frac{1}{3}}$ for all experiments. For both AGMDP-TriCL and AGMDP-FCL, on each setting of ε, we report average results over 1,000 synthetic graphs (100 for Epinions and Pokec). For comparison, we also report the results of AGM instantiated with the non-private versions of FCL and TriCycLe, denoted AGM-FCL and AGM-TriCL, respectively.

5.1 Statistics Evaluated

We compare synthetic and original graphs via several statistics.

Clustering Coefficients. We consider two measures of clustering. The *global clustering coefficient* (also called *transitivity*) of a graph G is defined as $C(G) = \frac{3 \times n_\Delta}{n_W}$ where n_Δ is the number of triangles in G and n_W is the number of wedges (i.e., length two paths) in G. Another measure is the average of the *local clustering coefficients*. The local clustering coefficient of a node $v_i \in N$ is given by $C_i = \frac{2|\{e_{jk} \in E | v_j, v_k \in \Gamma(v_i)\}|}{|\Gamma(v_i)| \times (|\Gamma(v_i)| - 1)}$, and the average of the local clustering coefficients is denoted $\overline{C} = \frac{1}{n}\sum_{v_i \in N} C_i$. The reason for using both these measures is that the former tends to emphasize the low-degree nodes while the latter emphasizes high-degree nodes. We report the *mean relative error* (MRE) of C and $\overline{C}$ between the synthetic graphs and the original graphs (denoted C and $\overline{C}$ in the tables).

[5]The large size of this dataset makes it much more robust to injected noise, so we use smaller (stronger) values for ε.

Degree Distribution. To evaluate how well a synthetic graph captures the degree distribution of the input graph, we use the *Kolmogorov-Smirnov* (KS) statistic, which quantifies the maximum distance between the two degree distributions. Let $F_{\mathcal{S}}$ and $F_{\widetilde{\mathcal{S}}}$ denote the cumulative distribution functions estimated from the sorted degree sequences of the original and synthetic graphs, respectively. Then $KS(\mathcal{S}, \widetilde{\mathcal{S}}) = \max_d |F_{\mathcal{S}}(d) - F_{\widetilde{\mathcal{S}}}(d)|$. Since the KS statistic is known to be less sensitive to differences in the tails of the distributions, we also report the *Hellinger distance* between the two degree distributions (denoted $D_{\mathcal{S}}, D_{\widetilde{\mathcal{S}}}$), defined as

$$H_{\mathcal{S}} = \frac{1}{\sqrt{2}}\sqrt{\sum_d \left(\sqrt{D_{\mathcal{S}}(d)} - \sqrt{D_{\widetilde{\mathcal{S}}}(d)}\right)^2}.$$

The smaller the values of both statistics, the closer (more similar) the degree distributions of the synthetic and original graphs.

Edge Count and Triangle Count. We report the mean relative error (MRE) for the number of edges (denoted simply as m in the tables) and the number of triangles (denoted as n_Δ in the tables) in the synthetic graphs relative to the original input graphs.

Attribute Correlations. To quantify the error in the attribute correlations in the synthetic graph, relative to the original graph, we report the MRE (denoted as simply Θ_F in the tables), as well as the Hellinger distance, defined as

$$H_{\Theta_F} = \frac{1}{\sqrt{2}}\sqrt{\sum_{y_i \in Y_w^F} \left(\sqrt{\Theta_F(y_i)} - \sqrt{\widetilde{\Theta}_F(y_i)}\right)^2},$$

where Θ_F and $\widetilde{\Theta}_F$ are the discrete probability distributions for the original and synthetic graphs, respectively. H_{Θ_F} lies in the range $[0, 1]$, with a value near 0 meaning that the correlations in the synthetic graph closely approximate those in the input graph.

5.2 Results

Tables 2–5 summarize the experimental results on the four datasets. For each setting of the overall privacy budget (ε) we report the average error rates for the graphs synthesized by both AGMDP-TriCL and AGMDP-FCL. Recall that a smaller ε translates to a *stronger* privacy guarantee; so the level of privacy (and, in general, the amount of error) increases as we move down the tables. To make the impact of the privacy mechanism clear, we include, at the top of each table, the error rates for the non-private version of AGM instantiated with FCL and TriCycLe (denoted AGM-FCL and AGM-TriCL, respectively).

Clustering. Recall that the FCL structural model includes no mechanism for capturing and reproducing clustering in the input graph, so the error rates for the triangle count (n_Δ), average clustering coefficient ($\overline{C}$) and the global clustering coefficient (C) are larger for FCL-based models than for the TriCycLe-based models, as expected. For that reason, the error rates for AGMDP-FCL on those three statistics serve as a good reference point for interpreting the corresponding results of AGMDP-TriCL. For instance, looking at the results on the Last.fm and Epinions datasets (Tables 2 and 4, respectively) we see that even with a reasonably strong privacy budget of $\varepsilon = 0.2$, AGMDP-TriCL could reproduce the clustering observed in the input graph significantly better than AGMDP-FCL. For the large Pokec dataset, the error rates remained significantly below those of AGMDP-FCL, even with the extreme setting of $\varepsilon = 0.01$, typically the smallest ε seen in the DP literature. In

Table 2: Results for AGMDP-FCL and AGMDP-TriCL structural models on the **Last.fm** dataset for different privacy settings, ε.

Last.fm	Model	Θ_F	H_{Θ_F}	$KS_{\mathcal{S}}$	$H_{\mathcal{S}}$	n_Δ	$\overline{C}$	C	m
non-private	**AGM-FCL**	0.00	0.01	0.05	0.15	0.59	0.75	0.61	0.0000
	AGM-TriCL	0.00	0.02	0.08	0.16	0.05	0.04	0.24	0.0001
$\varepsilon = \mathbf{ln\ 3}$	**AGMDP-FCL**	0.02	0.14	0.06	0.16	0.59	0.75	0.60	0.0076
	AGMDP-TriCL	0.02	0.14	0.09	0.17	0.05	0.07	0.23	0.0147
$\varepsilon = \mathbf{ln\ 2}$	**AGMDP-FCL**	0.03	0.18	0.07	0.17	0.56	0.74	0.58	0.0120
	AGMDP-TriCL	0.03	0.18	0.10	0.18	0.06	0.10	0.23	0.0222
$\varepsilon = 0.3$	**AGMDP-FCL**	0.05	0.27	0.09	0.19	0.42	0.68	0.48	0.0248
	AGMDP-TriCL	0.05	0.28	0.12	0.21	0.18	0.30	0.24	0.0499
$\varepsilon = 0.2$	**AGMDP-FCL**	0.06	0.32	0.11	0.20	0.39	0.65	0.43	0.0374
	AGMDP-TriCL	0.06	0.33	0.16	0.24	0.35	0.38	0.27	0.0769

Table 3: Results for AGMDP-FCL and AGMDP-TriCL on the **Petster** dataset for different privacy settings, ε.

Petster	Model	Θ_F	H_{Θ_F}	$KS_{\mathcal{S}}$	$H_{\mathcal{S}}$	n_Δ	$\overline{C}$	C	m
non-private	**AGM-FCL**	0.00	0.02	0.04	0.17	0.13	0.51	0.15	0.0001
	AGM-TriCL	0.00	0.01	0.05	0.17	0.00	0.35	0.07	0.0000
$\varepsilon = \mathbf{ln\ 3}$	**AGMDP-FCL**	0.03	0.16	0.05	0.17	0.18	0.51	0.19	0.0075
	AGMDP-TriCL	0.03	0.16	0.06	0.18	0.09	0.26	0.08	0.0136
$\varepsilon = \mathbf{ln\ 2}$	**AGMDP-FCL**	0.03	0.20	0.06	0.18	0.16	0.49	0.17	0.0117
	AGMDP-TriCL	0.04	0.20	0.07	0.19	0.13	0.29	0.10	0.0221
$\varepsilon = 0.3$	**AGMDP-FCL**	0.05	0.29	0.08	0.20	0.21	0.44	0.18	0.0268
	AGMDP-TriCL	0.06	0.30	0.12	0.22	0.39	0.32	0.24	0.0486
$\varepsilon = 0.2$	**AGMDP-FCL**	0.07	0.35	0.10	0.21	0.32	0.41	0.24	0.0388
	AGMDP-TriCL	0.07	0.36	0.14	0.24	0.72	0.32	0.40	0.0766

fact, the error rates for those statistics are nearly unchanged (compared to the non-private counterpart) until ε was set ≤ 0.05. This is because the amount of noise added to preserve privacy is independent of the size of the input graph, so larger graphs achieve a more favorable signal-to-noise ratio for the model parameters. On the other hand, we see that on the smaller Petster dataset, the clustering produced by AGMDP-TriCL was no better than that produced by AGMDP-FCL, when $\varepsilon < \ln(2)$ (≈ 0.69).

Attribute-Edge Correlations. A baseline for attribute-edge correlations is to set all correlation probabilities to be equal[6], independent of the data. This yields an average Hellinger distance of 0.37, 0.45, 0.55 and 0.5 and an MAE of 0.09, 0.11, 0.13 and 0.12 on Last.fm, Petster, Epinions and Pokec, respectively. Comparing against H_{Θ_F} and Θ_F in the tables, we see that the error rates for the synthetic graphs generated by AGMDP-FCL and AGMDP-TriCL were significantly lower than the baseline error rates on all datasets, even under the strongest privacy settings tested. The difference was most dramatic on the larger datasets (Epinions and Pokec).

Degree Statistics. The error in degree statistics ($H_{\mathcal{S}}$ and $KS_{\mathcal{S}}$) grows more slowly for AGMDP-FCL than for AGMDP-TriCL (as ε decreases). This is because we allocated *half* of the privacy budget for computing the noisy degree sequence with AGMDP-FCL, while for AGMDP-TriCL we allocated only *a quarter* (the other quarter is used to estimate triangle counts, not needed by FCL). Thus, the degree sequence has more noise in AGMDP-TriCL for the same ε. To calibrate the error-rates, we note that the baseline model that assigns edges to nodes uniformly at random achieves KS error 0.51, 0.51, 0.61 and 0.43 and a Hellinger distance of 0.64, 0.63, 0.64 and 0.56 on Last.fm, Petster, Epinions and Pokec, respectively. The KS statistic was substantially below that of the baseline for both AGMDP-FCL and AGMDP-TriCL at $\varepsilon = 0.2$: the KS was ≤ 0.16 on Last.fm and Petster (Tables 2 and 3), and ≤ 0.09 on Epinions and Pokec (Tables 4 and 5). Similarly, the Hellinger distance for both models was less than half that of the baseline on all datasets: on Last.fm and Petster, we had $H_{\mathcal{S}} \leq 0.24$, and on Epinions and Pokec we had $H_{\mathcal{S}} \leq 0.12$. The number of edges in the synthetic graphs (m) is close to the input for small privacy budgets ($\varepsilon = 0.2$ for Last.fm, Petster and Epinions, and $\varepsilon = 0.01$ for Pokec) with the MRE below 0.08 on the small Last.fm and Petster datasets, below 0.04 on Epinions and below 0.085 on Pokec.

[6]So for $w = 2$ attributes, we set each of the ten probabilities to 0.1.

5.3 Discussion

Our experiments demonstrate that it is possible to generate synthetic attributed social graphs with reasonably high fidelity, without sacrificing privacy. An important takeaway from the results is that, while the proposed framework performs well even on very small graphs, its power really becomes evident when applied on moderate to large input graphs. In particular, we saw that on the reasonably large Pokec social network, we generated very accurate synthetic graphs even under the strongest privacy regime for DP used in the literature (i.e., $\varepsilon = 0.01$).

We focused our empirical analysis on graphs with $w = 2$ node attributes, however our framework is not limited to two-dimensional attribute vectors. Although the sensitivity of the approach is not affected by the number of attributes, we can expect the error rates to increase as w increases, since the number of counts that must be

Table 4: Results for AGMDP-FCL and AGMDP-TriCL on the **Epinions** dataset for different privacy settings, ε.

Epinions	Model	Θ_F	H_{Θ_F}	KS_S	H_S	n_Δ	$\overline{C}$	C	m
non-private	**AGM-FCL**	0.01	0.03	0.06	0.08	0.82	0.92	0.83	0.0008
	AGM-TriCL	0.00	0.01	0.04	0.08	0.27	0.06	0.53	0.0008
$\varepsilon = \mathbf{ln\ 3}$	**AGMDP-FCL**	0.01	0.06	0.06	0.08	0.83	0.92	0.84	0.0019
	AGMDP-TriCL	0.01	0.06	0.04	0.08	0.28	0.10	0.53	0.0057
$\varepsilon = \mathbf{ln\ 2}$	**AGMDP-FCL**	0.01	0.08	0.05	0.08	0.82	0.92	0.84	0.0042
	AGMDP-TriCL	0.01	0.09	0.04	0.09	0.28	0.11	0.52	0.0084
$\varepsilon = 0.3$	**AGMDP-FCL**	0.01	0.12	0.06	0.10	0.81	0.92	0.83	0.0128
	AGMDP-TriCL	0.02	0.14	0.07	0.11	0.23	0.14	0.48	0.0218
$\varepsilon = 0.2$	**AGMDP-FCL**	0.02	0.13	0.08	0.11	0.78	0.91	0.80	0.0157
	AGMDP-TriCL	0.02	0.17	0.09	0.12	0.20	0.19	0.46	0.0345

Table 5: Results for AGMDP-FCL AGMDP-TriCL on the **Pokec** dataset for different privacy settings, ε.

Pokec	Model	Θ_F	H_{Θ_F}	KS_S	H_S	n_Δ	$\overline{C}$	C	m
non-private	**AGM-FCL**	0.00	0.01	0.04	0.05	1.00	1.00	1.00	0.0000
	AGM-TriCL	0.00	0.02	0.03	0.04	0.24	0.37	0.30	0.0000
$\varepsilon = 0.2$	**AGMDP-FCL**	0.00	0.02	0.05	0.05	1.00	1.00	1.00	0.0020
	AGMDP-TriCL	0.00	0.03	0.03	0.05	0.24	0.39	0.30	0.0052
$\varepsilon = 0.1$	**AGMDP-FCL**	0.00	0.03	0.05	0.06	1.00	1.00	1.00	0.0057
	AGMDP-TriCL	0.00	0.04	0.05	0.07	0.24	0.37	0.30	0.0101
$\varepsilon = 0.05$	**AGMDP-FCL**	0.00	0.06	0.05	0.07	1.00	1.00	1.00	0.0108
	AGMDP-TriCL	0.01	0.09	0.07	0.10	0.25	0.35	0.33	0.0189
$\varepsilon = 0.01$	**AGMDP-FCL**	0.02	0.14	0.11	0.15	1.00	1.00	1.00	0.0510
	AGMDP-TriCL	0.03	0.20	0.15	0.19	0.46	0.37	0.50	0.0833

computed for Θ_X and Θ_F increase exponentially with w, leading to smaller counts and hence a higher noise-to-signal ratio. Likewise, we expect error rates to decrease when using $w = 1$.

6. RELATED WORK

Prior studies have shown that simple graph anonymization techniques are vulnerable to a number of attacks that can allow an attacker to reidentify individuals and determine the presence of specific edges [1, 10, 24]. Initial attempts to address these types of attacks (e.g., [17, 38, 39]) made strong assumptions about the attacker's background knowledge and lacked formal privacy guarantees. More recently, focus has shifted to extending the strong formal privacy guarantees of differential privacy to graph data. This work has generally pursued one of two directions: methods for privately computing specific statistics, and private graph release.

Private Graph Statistics. Hay et al. [11] considered releasing the degree distribution of a graph under edge-differential privacy using a constrained inference technique. The idea is to impose an ordering constraint on the pre-noise degree sequence, add Laplace noise to the sorted sequence then post-process the noisy sequence to restore the order constraint.

Subgraph counting queries ask how many "edge-induced isomorphic copies" of a query graph Q (e.g., triangle), are present in G. Although subgraph counting has high global sensitivity in general, Karwa et al. extend smooth sensitivity for specific queries such as triangles and k-stars. Wang et al. outlined a general, divide and conquer approach using smooth sensitivity for graph analysis tasks that have low *local* sensitivity, such as clustering coefficient [35]. Zhang et al. [37] defined the *Ladder* framework for producing accurate DP estimates of subgraph counting queries, including triangles and k-stars. The Ladder framework combines the concept of "local sensitivity at distance t" [25] with the exponential mechanism [22]. Shen and Yu [33] consider finding frequently occurring subgraphs under differential privacy.

Private Graph Release. Mir and Wright [23] focused on generating representative synthetic graphs via the *Kronecker graph model*. Their approach estimates the (single) model parameter from the input graph under edge-differential privacy. Sala et al.'s *Pygmalion* model uses the dK-series of the input graph, which records the distribution of the pairs of degrees observed on edges [30]. The proposed approach is based on local sensitivity, and so does not provide differential privacy. Subsequent work combined the dk-series model that constructs a synthetic graph via smooth sensitivity [34].

Xiao et al. encode the structure of a graph via private edge-counting queries, under the hierarchical random graph (HRG) model, and claim better results than the dK-series approach [36]. Chen et al. [4] use the exponential mechanism to sample an adjacency matrix after clustering the input graph. Proserpio et al. [29] suggest non-uniformly down-weighting the edges of a graph in order to reduce high global sensitivity due to the possibility of very high degree nodes. They demonstrate the approach, combined with MCMC-based sampling, to generate private synthetic graphs. Our work differs from all these prior works, in particular as we aim to (privately) model both structure and attribute correlations.

Node Differential Privacy. The prior work discussed above adopts edge-differential privacy. In the more challenging model of node-differential privacy, neighboring graphs differ in a single node and *all* of its incident edges. While smooth sensitivity has been useful for taming high global sensitivity under edge-differential privacy, it appears less effective for node differential privacy. This is because in node-differential privacy, even the local sensitivity of basic statistics can be very high. For example, the number of edges in a graph can increase by n (the number of nodes) if we add a new node connected to all others, so even the local sensitivity of edge counting is n. To disallow such cases, Chen and Zhou propose a *recursive mechanism* based on the notion of *local empirical sensitivity*, which considers only the effects of *deleting* of an existing node from the input graph [5]. Kasiviswanathan et al. [14] and Blocki et al. [2] independently proposed the idea of using low-sensitivity transforms to project an input graph onto the set of degree-bounded graphs. By bounding the maximum degree of the input graph, the sensitivity becomes bounded and can be significantly reduced. When the degree bound, D, is set so that most of the nodes in the graph have degree $\leq D$, very little error is introduced by the projection operation; however, choosing an appropriate bound can be difficult in practice.

7. CONCLUSIONS AND FUTURE WORK

This work proposed a differentially private adaptation of the Attributed Graph Model (AGM) of [27]. Our framework is capable of modeling and synthesizing attributed social networks that mimic the structural properties and attribute correlations of an input network, without disclosing individual relationships and attributes. We explored three different approaches for accurately modeling the attribute correlations under differential privacy, identifying edge truncation to be the most effective choice. We introduced a new structural model capable of reproducing the important structural properties of social networks, including the degree distribution and the distribution of local clustering coefficients, and showed that the parameters required by the model can be accurately estimated under differential privacy. Finally, we presented an end-to-end workflow for generating private, synthetic social graphs with our framework and demonstrated its efficacy through experiments on four real-world social network datasets.

We believe this work lays a good foundation for providing differential privacy guarantees for attributed graph analysis; however, there remain many avenues for future work. We limited our focus to achieving edge differential privacy for undirected social graphs with a relatively small number of binary node attributes. We conclude by briefly discussing what would be needed to remove each of these restrictions.

Other Graph Types. We chose to focus on attributed social networks in this work for concreteness; however the proposed framework could be extended to support other types of attributed graphs by plugging in an appropriate underlying structural model for M. As long as the structural model satisfies differential privacy, the framework as a whole will still satisfy differential privacy.

Non-Binary Attributes. Our framework can support non-binary categorical or continuous attributes by simply converting each attribute to a series of binary attributes, one per category or range[7]. Although increasing the number of attributes, w, has no impact on the sensitivity of our approach, it does increase the number of counts that need to be computed for Θ_X and Θ_F; thus we can expect the accuracy of the generated graphs to degrade as w increases, due to a higher noise-to-signal ratio for the counts. In general, the larger the graph, the more attributes that could be supported accurately, since the amount of noise is also independent of the size of the graph. It may also be of interest to support edge attributes (e.g., trust or friendship strength), which would need an additional correlation model.

[7]For example, marital status could be represented by the following three binary attributes: *isMarried*, *isDivorced*, *isSingleOrWidowed*.

Directed Edges. Modeling attribute-edge correlations under directed edges increases the number of counts computed for Θ_F from $\binom{2^w+1}{2}$ to 4^w. However, the sensitivity does not change. The structural model also needs to capture directionality. This is more challenging, since the current approaches (TriCycle and FCL) make use of the degree sequence. Switching to using the in-degree and out-degree sequence is not compatible with the constrained inference of [11], so this helpful post-processing step would be lost.

Node Differential Privacy. In this work we focused on satisfying edge-differential privacy, in which neighboring graphs differ in a single edge or the attributes of a single node. A stronger notion of adjacency which has been studied in a few recent works is node-differential privacy, in which neighboring graphs differ in a single node and all of its adjacent edges (and/or the attributes associated with one node). This is a more difficult definition to satisfy, as it translates to a much higher sensitivity for many tasks. A promising approach for computing Θ_F under node-privacy is to use the same edge truncation approach but then to add noise according to its smooth sensitivity in the node-adjacency model. In a preliminary experiment we found that using this approach, the Hellinger distance between the original and noisy correlation probabilities was still better than the baseline when $\varepsilon \geq \ln(2)$ on Last.fm, when $\varepsilon \geq 0.3$ on Petster, when $\varepsilon \geq 0.2$ on Epinions, and when $\varepsilon \geq 0.05$ on Pokec; δ was fixed at 0.01 for the experiment. The problem of developing a node-private approach for fitting a suitable structural model remains a challenging open problem.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful feedback and suggestions. The work of Cormode is supported in part by European Commission Marie Curie Integration Grant PCIG13-GA-2013-618202 and a Royal Society Wolfson Research Merit Award.

9. REFERENCES

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou R3579X?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[2] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, 2013.

[3] I. Cantador, P. Brusilovsky, and T. Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec). In *RecSys*, 2011.

[4] R. Chen, B. C. Fung, S. Y. Philip, and B. C. Desai. Correlated network data publication via differential privacy. *VLDB Journal*, pages 1–24, 2013.

[5] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *SIGMOD*, 2013.

[6] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. volume 99, pages 15879–15882. National Acad Sciences, 2002.

[7] C. Dwork. Differential privacy. *ICALP*, 2006.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, 2006.

[9] M. Hay. *Enabling accurate analysis of private network data*. PhD thesis, University of Massachusetts Amherst, 2010.

[10] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *VLDB*, 1(1):102–114, 2008.

[11] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, 2009.

[12] Z. Jorgensen and T. Yu. A privacy-preserving framework for personalized, social recommendations. In *EDBT*, 2014.

[13] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 2011.

[14] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, 2013.

[15] D. Kifer and B.-R. Lin. Towards an axiomatization of statistical privacy and utility. In *PODS*, 2010.

[16] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. SIAM J. Sci. Comput., 36(5), C424–C452, 2014.

[17] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.

[18] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *KDD*, 2014.

[19] P. Massa and P. Avesani. Trust-aware bootstrapping of recommender systems. In *Workshop on recommender systems*, 2006.

[20] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[21] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, 2009.

[22] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[23] D. J. Mir and R. N. Wright. A differentially private graph estimator. In *ICDMW*, 2009.

[24] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy*, 2009.

[25] Nissim, Raskhodnikova, and Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[26] J. J. Pfeiffer, T. La Fond, S. Moreno, and J. Neville. Fast generation of large scale social networks while incorporating transitive closures. In *(PASSAT)*, 2012.

[27] J. J. Pfeiffer III, S. Moreno, T. La Fond, J. Neville, and B. Gallagher. Attributed graph models: modeling network structure with correlated attributes. In *WWW*, 2014.

[28] A. Pinar, C. Seshadhri, and T. G. Kolda. The similarity between stochastic kronecker and chung-lu graph models. *CoRR, abs/1110.4925*, 2011.

[29] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis. In *VLDB*, 2014.

[30] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *IMC*, 2011.

[31] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of erdős-rényi graphs. *Physical Review E*, 85(5):056109, 2012.

[32] C. Seshadhri, A. Pinar, and T. G. Kolda. An in-depth study of stochastic kronecker graphs. In *ICDM*, 2011.

[33] E. Shen and T. Yu. Mining frequent graph patterns with differential privacy. In *KDD*, 2013.

[34] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy*, 6(2):127, 2013.

[35] Y. Wang, X. Wu, J. Zhu, and Y. Xiang. On learning cluster coefficient of private networks. Social Network Analysis and Mining (3), pages 925–938, 2013.

[36] Q. Xiao, R. Chen, and K.-L. Tan. Differentially private network data release via structural inference. In *KDD*, 2014.

[37] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Private release of graph statistics using ladder functions. In *SIGMOD*, 2015.

[38] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.

[39] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. PVLDB (2), 946–957, 2009.

APPENDIX

A. DATASETS

We make use of four real-world social network datasets:

Last.fm[8] is a social music platform where users listen to and rate music, as well as share their favorite music with friends. This publicly available dataset [3] contains undirected friendship edges between user nodes, as well as a set of "listened to" relations indicating which artists a user has listened to. Although the dataset does not contain node attributes, we created two attributes by selecting the two most popular artists and creating *listenedToArtistX* attributes, with a value of 1 if a user had listened to artist *X* at least once.

The Epinions dataset [19] was crawled from Epinions.com, a website where users review and rate products. The dataset contains directed edges between users that represent positive trust statements, e.g., that user *A* trusts the reviews/ratings produced by user *B*. We converted the directed edges to undirected edges by keeping only the *mutual* trust relationships, i.e., we keep edge e_{ij} only if both e_{ij} and e_{ji} were present. Similarly to the Last.fm dataset, we created binary node attributes by selecting the two most frequently-rated products in the dataset and creating attributes with a value of 1 if a user had rated the product.

Petster is a website where users create web pages for their pets and form friendships links with other pet owners[9]. The dataset[10] is a crawl of the undirected friendship network of pet hamster owners and includes node attributes such as the hamster's name, sex, age, etc. We used two binary attributes: *sex* and *is-living*, where is-

[8]http://www.last.fm

[9]www.petster.com.

[10]http://konect.uni-koblenz.de/networks/petster-friendships-hamster

Table 6: Dataset properties. The columns are number of nodes n, number of edges m, maximum degree $d_{\max}$, average degree d_{avg}, number of triangles n_Δ, and average local clustering coefficient $\overline{C}$.

Dataset	n	m	d_{max}	d_{avg}	n_Δ	$\overline{C}$
Last.fm	1,843	12,668	119	6.9	19,651	0.183
Petster	1,788	12,476	272	7.0	16,741	0.143
Epinions	26,427	104,075	625	3.9	231,645	0.138
Pokec	592,627	3,725,424	1,274	6.3	2,492,216	0.104

living was set to 0 if the value of the age attribute was "gone to hamster heaven", or 1 otherwise.

Pokec is a popular Slovakian online social network. This dataset is publicly available as part of the Stanford Large Network Dataset Collection[11] and contains directed friendship edges and node attributes such as gender, age, hobbies, etc. As with the Epinions dataset, we converted the edges to undirected by keeping mutual friendships. We used two attributes: *sex* and *age*. We converted age into a binary attribute by assigning a value of 1 for ages ≤ 30 and 0 for ages greater than 30. We ignored nodes with missing values for gender or age.

For all datasets, we considered only the main connected component (after converting to undirected edges, where applicable). Table 6 summarizes the four (pre-processed) datasets, with respect to various statistics.

B. ATTRIBUTE-EDGE CORRELATIONS

In this Appendix, we consider other approaches to generate the needed attribute edge correlation distribution Θ_F, and compare them to our proposed Edge Truncation method. The first generates connection counts which can be normalized to generate Θ_F, while the second directly produces a probability distribution.

B.1 Direct Approach with Smooth Sensitivity.

In real-world social graphs, most nodes share edges with only a small fraction of the other nodes in the graph. *Smooth sensitivity*, proposed by Nissim et al. [25], offers a general approach for getting around a high global sensitivity due to unlikely inputs. We add noise according to a smooth upper bound on the *local sensitivity*, which is based on the actual input rather than an improbable worst-case input. However, there is no automatic procedure for determining the smooth sensitivity of a computation, so we must derive it manually for our specific task. We begin by observing that the local sensitivity f_{Q_F} (that is, the function that computes the set Q_F) is twice the maximum node degree.

LEMMA 3. *The* local sensitivity *of f_{Q_F} is $LS_{Q_F}(G) = 2d_{\max}$, where $d_{\max} = \max_{v_i \in N^{(G)}} d_{v_i}$.*

PROOF. The greatest impact to Q_F comes from changing the attribute vector associated with the highest degree node, which would have the effect of reducing some subset of the counts by an amount equal to $d_{\max}$, and increase another subset of the counts by the same amount. The impact of adding or removing an edge is just an increasing or decrease of one to a single count, which is negligible by comparison. $\square$

Adding noise directly calibrated to the local sensitivity *does not* satisfy differential privacy, because the magnitude of the noise may itself leak information about the underlying data. Rather, smooth sensitivity can be expressed in terms of the local sensitivity at distance t from the input graph (Section 2.3).

PROPOSITION 4. *The β-*smooth sensitivity *of f_{Q_F} is*

$$\begin{aligned} S^*_{Q_F,\beta}(G) &= \max_{t\geq 0} e^{-t\beta} LS^t_{Q_F}(G) \\ &= \max_{t\geq 0} e^{-t\beta} \max_{G':d(G,G')\leq t} LS_{Q_F}(G') \\ &= \max_{t\geq 0} e^{-t\beta} \min(2d_{\max}+2t, 2n-2) \end{aligned}$$

where, $d(G,G')$ is fewest number of edges that would need to be added/removed or the fewest number of nodes whose attributes would need to be changed to transform G into G'.

For typical graphs and typical settings of ε, $\delta \approx 0$, this function is maximized when $t = 0$, yielding twice the maximum degree. Setting the derivative of the right hand side to zero and solving, we arrive at the following corollaries.

COROLLARY 5. *For a graph G with maximum degree $d_{\max}$,*

$$S^*_{Q_F,\beta}(G) = \begin{cases} 2d_{\max} & \text{if } \frac{1}{\beta} \leq 2d_{\max} \\ \frac{2e^{(\beta d_{\max}-1)}}{\beta} & \text{otherwise} \end{cases}$$

COROLLARY 6. *For a graph G with maximum degree $d_{\max}$, $\beta = \frac{\varepsilon}{2\ln(2/\delta)}$, and fixed ε, we have $S^*_{Q_F,\beta}(G) = 2d_{\max}$, for any $\delta \geq 2e^{-(\varepsilon d_{\max})/2}$.*

For many real-world social networks, we can expect the maximum degree to be a small fraction of the total number of nodes, and thus using smooth sensitivity should work reasonably well.

B.2 SA-Based Approach.

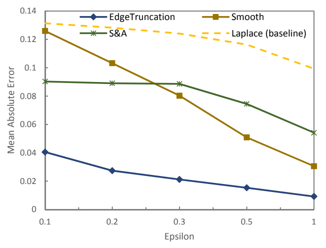
In the context of traditional databases, the *sample and aggregate framework (SA)* was originally proposed by Nissim et al. [25] as a general procedure for satisfying differential privacy in situations where determining the global (or smooth) sensitivity of a function is difficult or inefficient. The idea is as follows: to compute a differentially private estimate of a function f on dataset D, where $n = |D|$, first partition D into $t = n/k$ smaller datasets of k records each (for some k), denoted $D_1, \ldots, D_t$. Then, compute f on each of the t smaller datasets, yielding the set $Z = \{z_1, \ldots, z_t\}$, where $z_i = f(D_i)$. Finally, $f(D)$ is estimated by applying an aggregation function Agg (e.g., mean) to the set Z and adding noise according to the global (or smooth) sensitivity of Agg. For example, if the range of f is $[0,1]$ and A is the mean function, then we could add noise with sensitivity $1/t$ to the output of Agg(Z).

We extend this idea to the graph analysis domain to get around the high global sensitivity of Θ_F, as follows. For a given input graph G with n nodes, we first randomly partition the nodes into $t = n/k$ disjoint groups of k nodes each. Let $g_1, \ldots, g_m$ be the set of induced subgraphs, such that g_i is the subgraph induced by the nodes in group i. Note that (a) partitioning the nodes randomly does not impact the sensitivity, and (b) by working with the induced subgraphs, we ensure that a change to one node in the input graph G will only impact a single subgraph. Thus, we can apply a function f to each subgraph, aggregate the individual answers and add noise.

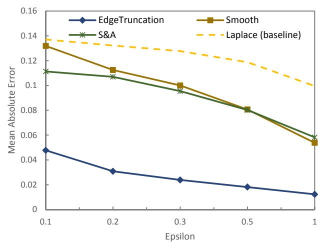
To compute Θ_F, we find the $\binom{2^w+1}{2}$ connection *probabilities* (not the *counts*) for each subgraph, average the corresponding probabilities across all subgraphs and add Laplace noise to each average. This approach has a global sensitivity of $\frac{2}{t}$, as illustrated by the following: let G, G' be a pair of neighboring graphs, where v_i is the node whose attributes were changed to get G' from G. Observe that, in the worst case, v_i is a member of a subgraph in which all of the internal edges are incident to v_i; if v_i's attribute values are different in G and G', then the L_1 difference of the corresponding sets of probabilities from G and G' will be 2. Since we are averaging over t subgraphs, the total impact on the set of average probabilities is $\frac{2}{t}$. Finally, after adding noise to the probabilities, we must divide them by their sum to ensure that they add up to one.

This approach introduces two types of error into the resulting probabilities: *estimation error*, due to averaging over outputs from the subgraphs, and *perturbation error* due to the Laplace noise.

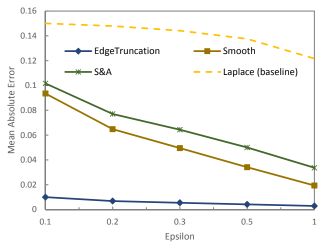
[11] http://snap.stanford.edu/data/



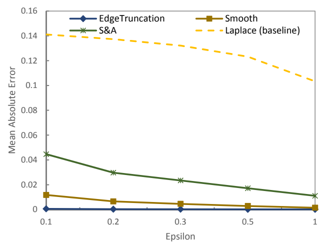
(a) Last.fm dataset



(b) Petster dataset



(c) Epinions dataset



(d) Pokec dataset

Figure 5: Comparison of mean absolute error for different approaches for computing $\widetilde{\Theta}_F$ on four different datasets, under different privacy regimes. Note that privacy becomes weaker as we move rightward along the x-axis. The parameters for each approach were tuned empirically.

The accuracy will thus depend on balancing the two sources of error by selecting an appropriate size k for the subgraphs. A larger k gives fewer but bigger subgraphs, which reduces estimation error at the cost of increasing the perturbation error; conversely, a smaller k gives a greater number of small subgraphs, which reduces perturbation error at the cost of more estimation error.

B.3 Comparison of Approaches.

Figures 5(a)–5(d) compare the mean absolute errors[12] (MAE) of the three proposed approaches for different privacy settings, ε, on four datasets (Last.fm, Petster, Epinions and Pokec, described in Appendix A) with $w = 2$ attributes. The error-rates are taken over 10,000 trials (1,000 for the larger Epinions and Pokec datasets). In the plots, *Smooth* denotes the smooth sensitivity-based approach, *EdgeTrunc* is the edge truncation approach, and *S&A* is the sample and aggregate-based approach. The truncation parameter for EdgeTrunc and the group-size parameter for S&A were tuned empirically to minimize the MAE. Note that such tuning should be counted against the privacy budget for a real release; here, we discount this to see the relative performance. For the smooth sensitivity approach, we used negligible $\delta = 10^{-6}$.

As a baseline, we included results for the naïve approach (dashed line) in which noise from Lap $\left(\frac{2n-2}{\varepsilon}\right)$ is simply added to each connection count and then normalized by the sum to get Θ_F. A useful approach should have an MAE well below this line. The main takeaways from these results are that (1) all of the approaches work reasonably well at the weakest privacy setting tested ($\varepsilon = 1$), but edge truncation appears to be the best choice across all datasets and privacy settings tested; (2) in general, the larger the input graph, the lower the error. On the large Pokec dataset ($m = 3,725,424$) all of the approaches perform well, even with the strongest privacy setting tested. For EdgeTrunc, the MAE comes very close to zero, even for $\varepsilon = 0.1$. Given the superior performance of EdgeTrunc, we chose to limit our focus to that approach for privately computing Θ_F in the remainder of this work.

C. DP ALGORITHMS

For completeness, we summarize the procedure for computing a differentially private version of Θ_F, denoted $\widetilde{\Theta}_F$, with edge truncation, as described in Section 3.1. The pseudocode is in Algorithm 4 and we prove its privacy guarantee in Theorem 7.

C.1 Computing Correlations Privately

THEOREM 7. *Algorithm 4 satisfies ε-differential privacy.*

[12]That is, the mean error between privately computed probabilities and true probabilities computed directly from the input graph.

Algorithm 4 LearnCorrelationsDP

```
Input: $\varepsilon, G, k, w$
Output: $\widetilde{\Theta}_F$
1: $G' \leftarrow \mu(G,k)$          ▷ Truncate G using the truncation operator $\mu$ from Def. 2
2: $Q_F \leftarrow f_{Q_F}(G')$      ▷ Compute the set of counts on truncated graph $G'$
3: for $y_i \in Y_w^F$ do
4:     $\widetilde{Q}_F(y_i) \leftarrow \min\left(n, \max\left(0, Q_F(y_i) + \text{LAP}(0, \frac{2k}{\varepsilon})\right)\right)$   ▷ Add Laplace noise to each count and clamp range
5: $q_{sum} \leftarrow \max_{y_i} \widetilde{Q}(y_i)$
6: for $y_i \in Y_w^F$ do
7:     $\widetilde{\Theta}_F(y_i) \leftarrow \frac{\widetilde{Q}_F(y_i)}{q_{sum}}$
8: return $\widetilde{\Theta}_F$
```

PROOF. The global sensitivity of truncating G to get k-bounded graph G' followed by computing Q_F on G' (lines 1–2) was shown to be $2k$ by Proposition 1. In line 4, we add independent noise from $Lap\left(\frac{2k}{\varepsilon}\right)$, which gives ε-differential privacy (Laplace mechanism). The remainder of the algorithm, including clamping the noisy counts to the range $(0,n)$, operates only on the noisy counts, and therefore has no impact on the privacy guarantee. Thus, Algorithm 4 satisfies ε-differential privacy. ☐

C.2 Private Release of Attribute Distribution

Algorithm 5 outlines the procedure for computing the attribute distribution, $\widetilde{\Theta}_X$ that was described in Section 3.2. The formal statement of its privacy guarantee is then proved with Theorem 8.

THEOREM 8. *Algorithm 5 satisfies ε-differential privacy.*

PROOF. Let $G = (N, E, X)$ be an arbitrary input graph and let $G' = (N, E, X')$ be the neighboring graph created by changing the attributes of one node, say $v_i \in N$, from x_i to x_i', where $f_w(x_i) = y_i$ and $f_w(x_i') = y_j$ where $y_i, y_j \in Y_w, y_i \neq y_j$. Then we have that $q_i' = q_i - 1$ and $q_j' = q_j + 1$, and $q_k' = q_k, \forall_k k \neq i, k \neq j$. Thus, the global

Algorithm 5 LearnAttributesDP

```
Input: $\varepsilon, X, w$
Output: $\widetilde{\Theta}_X$
1: $Q_X \leftarrow f_{Q_X}(X)$          ▷ Compute the set of counts from $X$
2: for $y_i \in Y_w$ do          ▷ Add Laplace noise to each count and clamp
3:     $\widetilde{Q}_X(y_i) \leftarrow \min\left(n, \max\left(0, Q_X(y_i) + \text{LAP}(0, \frac{2}{\varepsilon})\right)\right)$
4: $q_{sum} \leftarrow \sum_{y_i} \widetilde{Q}_X(y_i)$
5: for $y_i \in Y_w$ do
6:     $\widetilde{\Theta}_X(y_i) \leftarrow \frac{\widetilde{Q}_X(y_i)}{q_{sum}}$          ▷ Normalize
7: return $\widetilde{\Theta}_X$
```

sensitivity of $f_{Q_X}(X)$ (line 1) is 2. In line 3, we add independent Laplace noise with mean zero and scale equal to $\frac{2}{\varepsilon}$, which gives ε-differential privacy. (Laplace mechanism). The remainder of the algorithm, including clamping the noisy counts to the range $(0,n)$, operates on the noisy counts, and so does not impact privacy. □

C.3 Privately Fitting TriCycLe

The two inputs required by the algorithm are the degree sequence $\mathcal{S}$ and the number of triangles n_Δ, measured from the input graph G. Here we discuss how to compute differentially private estimates of these statistics using existing techniques, resulting in a DP graph generation process.

C.3.1 Degree Sequence

The degree sequence of input graph G is the multiset $\mathcal{S} = \{d_i | v_i \in N\}$. A straightforward way to satisfy DP for $\mathcal{S}$ is to apply the Laplace mechanism, adding independent noise from $\text{Lap}(2/\varepsilon)$ to each $d_i \in \mathcal{S}$; the global sensitivity is *two* because adding or removing an edge of G changes the degrees of exactly two nodes by one. However, this approach introduces a lot of noise, especially in low degree nodes, which are abundant in real social graphs. Instead, we can improve the estimate by observing that the mapping between a specific node and its degree is unimportant for how TriCycLe uses the degree sequence—we only need the sequence of degrees, the order is unimportant. [11] proposes an approach for accurately estimating the degree sequence of a graph based on constrained inference—recall that ε-differential privacy is invariant under post-processing, so this step does not affect the guarantee. The high-level idea is to sort the degree sequence prior to adding noise and then to post-process the noisy sequence to enforce the ordering condition, thereby canceling out much of the noise.

The algorithm in [11] starts by computing the actual degrees for each node, forming degree sequence S. It then sorts $\mathcal{S}$ in non-decreasing order and adds independent noise drawn from $\text{Lap}(2/\varepsilon)$ to each degree, yielding noisy degree sequence $\tilde{\mathcal{S}}$; the addition of noise may cause the ordering constraint to be violated. A constrained inference procedure is applied to $\tilde{\mathcal{S}}$ to find the $\overline{\mathcal{S}}$ that is "closest" to $\tilde{\mathcal{S}}$ (i.e., the minimum L_2 distance) that also satisfies the ordering constraint. A dynamic programming solution given in [11] performs the constrained inference operation in linear time.

C.3.2 Triangle Count

TriCycLe also needs the number of triangles in the input graph. The challenge here, from the privacy perspective, is that the global sensitivity of triangle counting is prohibitively high, since in the worst case there could be a single edge that is shared by every triangle in the graph—for a graph with n nodes, adding or removing a single edge could change the triangle count by up to $n-2$. Consequently, direct application of the Laplace mechanism yields a very inaccurate count. Fortunately, real-world graphs typically do not contain such pathological structures, and tend to have local sensitivities that are much lower. Triangle counting has been extensively studied in the context of differential privacy (e.g., [25, 37, 13, 5]). The most effective approaches take advantage of the above observation in some way to reduce the amount of noise required to satisfy differential privacy.

The state-of-the-art approach for differentially private triangle counting is based on the *Ladder* framework [37], which effectively combines the concept of "local sensitivity at distance t", from the smooth sensitivity framework [25], with the exponential mechanism [22]. In the context of triangle counting, it has been shown to provide better accuracy and has the added benefit that it provides pure differential privacy, rather than the weaker notion of (ε,δ)-

Algorithm 6 FitTriCycLeDP

```
Input: $\varepsilon, E, n$
Output: $\widetilde{\Theta}_M = \{\bar{\mathcal{S}}, \widetilde{n}_\Delta\}$
 1: $\varepsilon_{\mathcal{S}} \leftarrow \frac{\varepsilon}{2}$
 2: $\varepsilon_\Delta \leftarrow \frac{\varepsilon}{2}$        ▷ Split the privacy budget (evenly) between the degree
    sequence and triangle count
 3: $\mathcal{S} \leftarrow \langle d_1, \dots, d_n \rangle$
 4: $\mathcal{S} \leftarrow \text{SORTASCENDING}(\mathcal{S})$
 5: $\widetilde{\mathcal{S}} \leftarrow \mathcal{S} + \text{LAP}(\frac{2}{\varepsilon_{\mathcal{S}}})^n$        ▷ Add independent Laplace noise to each
    coordinate of degree sequence $\mathcal{S}$
 6: Apply constrained inference (Algorithm 1 from [11]) to $\widetilde{\mathcal{S}}$ to get $\bar{\mathcal{S}}$
 7: for $\bar{d}_i \in \bar{\mathcal{S}}$ do
 8:     Round $\bar{d}_i$ to the nearest integer in $\{0, \dots, n-1\}$
 9: $\widetilde{n}_\Delta \leftarrow \text{NOISESAMPLE}(f_{n_\Delta}, E, \varepsilon_\Delta)$        ▷ Get a differentially private
    estimate of the triangle count using Algorithm 1 from [37]
10: return $\widetilde{\Theta}_M = \{\bar{\mathcal{S}}, \widetilde{n}_\Delta\}$
```

differential privacy provided by the smooth sensitivity framework. The local sensitivity of a function f at distance t (from the input graph) is denoted $LS_f^t(G)$, and quantifies the maximum local sensitivity among all graphs that can be formed from G by adding or deleting up to t edges. The resulting mechanism allows us to sample an approximate value for the triangle count, such that better approximations are exponentially more likely to be sampled.

C.3.3 Algorithm for Fitting TriCycLe-DP.

Algorithm 6 summarizes the procedure for fitting the parameters used by TriCycLe from an input graph G, while Thm. 9 shows that the algorithm satisfies ε-differential privacy.

THEOREM 9. *Algorithm 6 satisfies ε-differential privacy.*

PROOF. Let $G = (N,E,X)$ be an arbitrary input graph and, without loss of generality, suppose we form a neighboring graph G' by adding an arbitrary edge e_{ij}, such that the degrees of the two incident nodes in G are d, d' respectively. Let $\mathcal{S}$ and $\mathcal{S}'$ denote the degree sequence vectors for G and G' sorted in non-descending order, respectively. Thus, relative to $\mathcal{S}$, two values in $\mathcal{S}'$ are different: the right-most d has changed to $d+1$ and the right-most d' has become $d'+1$. Thus, the L_1 difference $|\mathcal{S}' - \mathcal{S}| = 2$, which gives a global sensitivity of 2. In line 5 we add Laplace noise to each coordinate of the sorted degree sequence vector. Since the global sensitivity is 2 and we add Laplace noise with a scale of $\frac{2}{\varepsilon_{\mathcal{S}}}$, giving $\varepsilon_{\mathcal{S}}$-differential privacy for the degree sequence. Performing the constrained inference procedure on the noisy degree sequence (line 6) has no effect on the privacy guarantee, since it does not look at the original degree sequence; likewise for the subsequent rounding of the post-processed degrees. Line 9 computes an ε_Δ-differentially private estimate of the number of triangles in G using the Ladder framework of [37]. By sequential composition, these two computations together satisfy ε-differential privacy, where $\varepsilon = \varepsilon_{\mathcal{S}} + \varepsilon_\Delta$. □

C.4 Running Time Analysis

Relative to the non-private version of AGM, the privacy mechanisms in AGM-DP add relatively little computational overhead. The truncation operation in LearnCorrelationsDP runs in $O(m)$ and sampling noise from the Laplace distribution takes constant time. The underlying structural model dominates the running time of our framework. Quantitatively, we note that on the largest graph in our experiments (Pokec), generating a synthetic graph with AGM-DP-TriCycLe took about 85 minutes on a Core i7 desktop with 12GB of RAM. Apart from sampling the edges, computing the private triangle count was the most expensive operation ($\approx$ 12 minutes using the Ladder framework of [37]). Our prototype was implemented in Python and was not optimized for speed; implementation in a faster language, such as C++, would likely result in significant speedups.