

Delineating Social Network Data Anonymization via Random Edge Perturbation

Mingqiang Xue
I²R, Singapore
xuem@i2r.a-star.edu.sg

Panagiotis Karras
Rutgers University, USA
karras@business.rutgers.edu

Chedy Raïssi
INRIA Nancy, France
chedy.raïssi@inria.fr

Panos Kalnis
KAUST, Saudi Arabia
panos.kalnis@kaust.edu.sa

Hung Keng Pung
NUS, Singapore
dcsphk@nus.edu.sg

ABSTRACT

Social network data analysis raises concerns about the privacy of related entities or individuals. To address this issue, organizations can publish data after simply replacing the identities of individuals with pseudonyms, leaving the overall structure of the social network unchanged. However, it has been shown that attacks based on structural identification (e.g., a *walk-based attack*) enable an adversary to re-identify selected individuals in an anonymized network. In this paper we explore the capacity of techniques based on *random edge perturbation* to thwart such attacks. We theoretically establish that *any* kind of structural identification attack can effectively be prevented using random edge perturbation and show that, surprisingly, important properties of the whole network, as well as of subgraphs thereof, can be accurately calculated and hence data analysis tasks performed on the perturbed data, given that the legitimate data recipient knows the perturbation probability as well. Yet we also examine ways to enhance the *walk-based attack*, proposing a variant we call *probabilistic attack*. Nevertheless, we demonstrate that such probabilistic attacks can *also* be prevented under sufficient perturbation. Eventually, we conduct a thorough theoretical study of the probability of success of *any* structural attack as a function of the perturbation probability. Our analysis provides a powerful tool for delineating the identification risk of perturbed social network data; our extensive experiments with synthetic and real datasets confirm our expectations.

Categories and Subject Descriptors

H.2.7 [Database Management]: Database Administration—*Security, integrity, and protection*; H.2.8 [Database Management]: Database applications—*Data mining*; K.4.1 [Computers and Society]: Public Policy Issues—*Privacy*

Keywords

privacy, social network, random perturbation, graph utility

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...\$15.00.

1. INTRODUCTION

Today *data owners* (e.g., private corporations, public organizations) can store large volumes of digital human interactions. Examples include interactions among groups of players in an online game, for file-sharing in P2P networks, and among individuals in online social networks. In most cases, the network at hand can be represented by a graph $G = (V, E)$, where vertices stand for individuals and edges for interactions or relations. Data owners wish to publish such graphs, e.g. for social studies and marketing.

Nevertheless, publication should not reveal sensitive personal information contained in the network. A *naïve graph anonymization* approach is to remove identifying attributes such as names, e-mails or IP addresses, and still publish the network in a way that allows all kind of analysis of its graph properties.

Unfortunately, this naïve anonymization approach does not guarantee privacy protection. As Hay et al. [8] discerned, the structural characteristics of the published graph, combined with background knowledge, can expose individuals. For instance, assume an adversary has access to graph G_n in Figure 1(b), the naïvely anonymized version of the graph in Figure 1(a) and knows that Suzanne has 3 friends, none of whom has more than 2 friends. With this knowledge, the adversary can identify Suzanne as node 3.

Backstrom et al. [1] have shown that a practical way for an adversary to gain structural knowledge is to embed a known subgraph (e.g., register dummy users in an online social network and link them to victims). Figure 1(c) shows our example graph including an embedded subgraph (nodes {11, 12, 13, 14, 15}). Malicious nodes were connected to Suzanne and Robert (e.g., by tricking them to respond to a bogus friendship invitation). An adversary capable to locate the inserted subgraph efficiently can also identify Suzanne and Robert and hence infer other attributes of theirs.

Let $G_A = (V_A, E_A)$ be the inserted subgraph and $k = |V_A|$ the number of vertices in G_A . Typically, it suffices for k to be in the order of $\sqrt{\log |G_o|}$, as the number of possible k -node subgraphs, hence the probability that the inserted subgraph is unique, grows exponentially with k^2 . Locating G_A is an intractable problem [7], hence impractical for large graphs. Therefore, [1] only considers subgraphs having a *connected backbone* (i.e., a Hamiltonian path of their k nodes), and describes an efficient *walk-based attack*: an adversary can locate G_A by searching for k -node paths having the same *degree sequence* as G_A (e.g., the subgraph of nodes {11...15} in Figure 1(c) has degree sequence [2, 5, 3, 4, 2]).

In this paper, we study the effectiveness of random edge perturbation as a tool for preventing structural identification in an any-

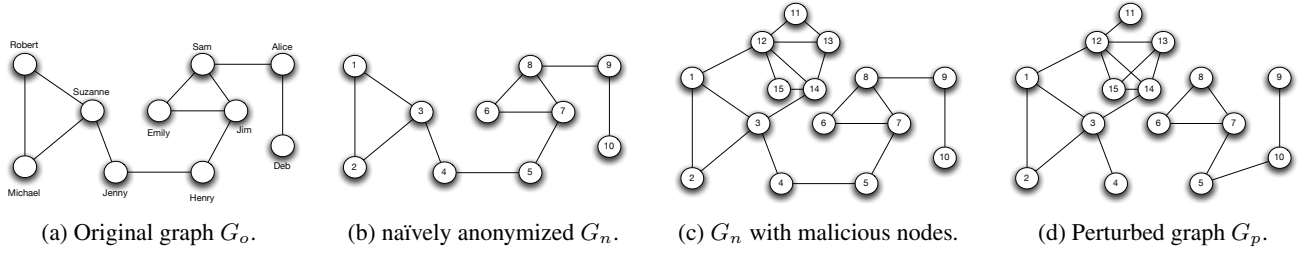


Figure 1: Example of a social network graph

mized graph. We assume that the original graph G_o is first transformed to G_n by naïve anonymization, and then G_n is altered to G_p by random edge perturbation with perturbation probability μ . Figure 1(d) shows a perturbed graph G_p , in which edges (4, 5), (8, 9) and (11, 13) in G_n have been removed, whereas edges (13, 15) and (5, 10) have been added. Fittingly, the degree sequence of G_A cannot be found in G_p ; therefore the *walk-based attack* fails.

Remarkably, Ying and Wu [12] reject random edge perturbation as a method for privacy preservation on the grounds that it obfuscates important graph characteristics; Bonchi and Tassa arrive at a similar, albeit more positive result [2]. We inspect such claims by showing that estimation algorithms can accurately *recover* important graph properties (e.g., density, degree distribution, transitivity, modularity, etc) from the perturbed graph, not only for the complete original graph, but more crucially, even for subgraphs thereof - for which publishing such measures directly to the end-user would not be a solution. We apply this methodology on several graph metrics used in graph analysis [5]. Our estimation algorithms are not applicable only on the presented metrics; in addition, we introduce a generic framework for estimating a class of utility metrics.

Nevertheless, we are wary that an adversary can employ a methodology similar to ours so as to launch sophisticated attacks. We demonstrate this potential by generalizing the *walk-based attack* of [1] to a *probabilistic walk-based attack*, which examines multiple possible degree sequences, each with its own probability of appearance. We show that, while being more computationally intensive than the deterministic variant, our probabilistic attack is practicable in perturbed graphs with higher success probability. Furthermore, we generalize our study to take into account *any* possible structural attack, even by extremely powerful adversaries having the computational power to enumerate all possible subgraphs of size k inside G_n . Our analysis delineates the probability of success in that case, for subgraph size k and perturbation probability μ .

To summarize, our contributions are outlined as follows: We show how important graph properties can be accurately **recovered** from a perturbed graph, *even though this perturbed graph does not look like the original*. We introduce a novel **probabilistic attack**, more powerful than the one [1]. We compute the probability of success of *any* structural attack, offering a yardstick for **assessing the privacy risk** entailed by the publication of perturbed graph data.

2. RELATED WORK

The idea that privacy can be preserved after randomization while data characteristics can be partially recovered, which we espouse, was debuted in the context of association rule mining [6, 11].

Backstrom et al. [1] were the first to present an attack demonstrating that naïve anonymization is insufficient to ensure privacy in graph data. They emphasize the differences between *active* attacks, where the adversary adds nodes and edges before publication, and *passive* attacks, where they do not, and propose the *walk-based* and *cut-based* attacks; the extent to which random edge perturbation can protect from this family of active attacks was not exam-

ined in [1]. Hay et al. [9] proposed a privacy protection technique based on random edge insertions and deletions, similar to the random edge perturbation that we study. However, the efficiency and utility preservation of their scheme were not studied. In [8], Hay et al. formalize two adversary knowledge models based on subgraph queries and hub fingerprints, and propose an approach using structure aggregation and sampling to protect from such attacks; however this method does not preserve the complete graph structure. Ying and Wu [12] assert that random edge perturbation significantly degrades a graph's utility; this assertion is true if utility metrics are measured on the perturbed graph; however [12] does not attempt to recover graph properties by estimation algorithms.

Zhou and Pei [13] use edge addition (along with generalization of node labels) in order to ensure that, for every node in the graph, there exist at least $k-1$ other nodes that share isomorphic neighborhoods. Similarly, Liu and Terzi [10] bring a graph in a form such that, for each node, there exist at least $k-1$ other nodes having the same degree, while Cormode et al. [4] study the anonymization problem for bipartite graphs and propose the notion of *safe groupings*. Zou et al. [14] propose the model of k -automorphism, which guarantees that the anonymized graph consists of k structurally indistinguishable subgraphs; Cheng et al. [3] expand on this concept with k -isomorphism, in which a graph is published in the form of k identical disjoint subgraphs. Unfortunately, such *syntactic* anonymization methods can severely alter the nature of the published network so as to conform to the privacy guarantee they provide. For example, by k -isomorphism, a connected graph is converted to an assortment of k identical disjoint graphs; thus, the published graph is *not* an anonymized version of the *whole* original, but only of $\frac{1}{k}$ thereof. Besides, the computational complexity of these syntactic schemes renders them inapplicable on very large graphs.

Recently, Bonchi et al [2] used an entropy-based privacy metric to evaluate several graph anonymization techniques; they conclude that techniques based on random perturbation are comparable to syntactic methods. Our work builds upon these findings by inspecting the potential to estimate a graph's original properties and the viability of structural attacks after random perturbation.

3. UTILITY MEASURE ESTIMATION

Let $G_o = (V, E)$ be an undirected graph representing the original social network and let $N = |V|$. Without loss of generality, we assume that V is ordered and $V[i]$ refers to the i^{th} node in the set. Naïve anonymization replaces all labels in G_o with random pseudonyms. The resulting graph is denoted by G_n , where new labels are given by $L(V[i], G_n)$, $i \in [1, N]$. As G_n and G_o contain the same set of nodes and edges, $G_n = (V, E)$.

Random edge perturbation yields a perturbed graph $G_p = (V, E_p)$ from a naïvely anonymized graph G_n by adding or removing edges. Let $\mu \in [0, \frac{1}{2}]$ be a user-defined *perturbation probability*, known to the legitimate data recipient, and $(V[i], V[j])$ denote

the edge between nodes $V[i]$ and $V[j]$. For every pair of nodes $V[i], V[j] \in V$ it holds that:

$$\begin{aligned} \Pr((V[i], V[j]) \notin E_p \mid (V[i], V[j]) \in E) &= \mu \\ \Pr((V[i], V[j]) \in E_p \mid (V[i], V[j]) \notin E) &= \mu \end{aligned}$$

We assume that the adversary knows a subgraph $G_A = (V_A, E_A)$ in the original graph G_o . G_A contains k ordered nodes, with labels $L(V_A[i], G_A)$ known to the adversary; the latter aims to identify G_A in G_p , i.e., an ordered set Y of k nodes in G_p , such that $L(Y[i], G_o) = L(V_A[i], G_A)$, for $1 \leq i \leq k$. We are interested to quantify the probability that the adversary will succeed.

Before we study the adversary's success probability, we study the effect of perturbation on different graph utility metrics. No single metric for measuring a graph's utility exists, as organizations may use a graph for different purposes. Still, there are a few widely accepted metrics [5]. We study four popular metrics: density, degree distribution, transitivity, and modularity. While past research has asserted that such metrics rapidly degrade with random edge perturbation [12], we outline how, even when the metrics measured on the perturbed graph vary greatly from those of the original, we can still accurately *estimate* their original values.

3.1 Density Estimation

The density metric for a general graph is the ratio of the number of edges in the graph over the number of possible edges. It describes the average level of connectivity between nodes. Formally, the density value is defined as follows:

$$density = \frac{2|E|}{|V|(|V| - 1)} \quad (1)$$

The density metric is widely used in social network data analysis. Admittedly, due to perturbation, the number of edges in the perturbed graph will differ from that in the original graph. However, we can use the known perturbation probability to estimate the original density. Since the density value depends on the graph size and the number of edges, when the graph size is known, we only need to estimate the original number of edges. Let $h = |E|$ be the (unknown) number of edges in G_o , and $h_p = |E_p|$ the observed number of edges in the perturbed graph G_p . We can estimate the value of h via the maximum-likelihood estimator \hat{h} , defined as:

$$\hat{h} = \arg \max_h (\Pr(h_p|h)) \quad (2)$$

where $\Pr(h_p|h)$ is the probability of observing h_p edges in the perturbed graph when the original graph has h edges. Intuitively, the estimator \hat{h} is the number of edges in the original graph that is most likely to result in h_p edges in the perturbed graph.

To estimate \hat{h} , we need to find a value h that maximizes $\Pr(h_p|h)$. Alternatively, we can find the value of \hat{h} by deriving an equation with the following rationale: since each edge removal and addition during perturbation can be viewed as an independent Bernoulli trial, the value of h that maximizes $\Pr(h_p|h)$ is the one that renders the *expected* number of edges in the perturbed graph equal to h_p . Given that each existing edge survives with probability $(1 - \mu)$ and each non-existing one is created with probability μ , the equality in consideration is expressed as follows:

$$h_p = \hat{h} \cdot (1 - \mu) + (M - \hat{h}) \cdot \mu \quad (3)$$

where $M = \frac{N(N-1)}{2}$ is the number of possible edges in the graph. The quantity on the right-hand side of the equation is the

expected number of edges in G_p when the number of edges in G_o is \hat{h} . Solving Equation 3 for \hat{h} , we get:

$$\hat{h} = \frac{h_p - M \cdot \mu}{1 - 2\mu} \quad (4)$$

3.2 Degree Distribution Estimation

The *degree* is a cardinal feature of a node. In social networks, degrees describe the number of friends a person has. In many real-world networks, degrees exhibit a power-law distribution.

Our estimation for degree distribution is similar to that for density estimation, as a node's original degree can be estimated via its degree in the perturbed graph. While there may be a high error in the estimation for an individual node, the total degree distribution can still be accurately estimated. We first focus on estimating the original degree of a single node. Let d_p be a node's observed degree in G_p , and \hat{d}_o our estimator of its degree in G_o , respectively. As in our density estimation solution, we can estimate \hat{d}_o by setting d_p to be equal to the *expected* degree of the node at hand in G_p when its original degree is \hat{d}_o . Therefore it is:

$$d_p = \hat{d}_o \cdot (1 - \mu) + (N - 1 - \hat{d}_o) \cdot \mu \quad (5)$$

Solving the latter for \hat{d}_o , we get:

$$\hat{d}_o = \frac{d_p - (N - 1) \cdot \mu}{(1 - 2\mu)} \quad (6)$$

Let $\hat{d}_o[i]$ be the estimator of the i^{th} node's degree. Then $[\hat{d}_o[1], \hat{d}_o[2], \dots, \hat{d}_o[N]]$ forms an estimation of the degree sequence of all nodes in G_o , whence we derive an estimate of the distribution.

3.3 Transitivity Estimation

Transitivity measures the incidence of order-3 loops in a graph. In a social network, if v_1 is connected to both v_2 and v_3 , then there is a relatively high probability that v_2 and v_3 are also connected (i.e., *a friend's friend is a friend*). Most social networks present high transitivity. Formally, a network's transitivity \mathcal{T} is defined as:

$$\mathcal{T} = \frac{3N_{\Delta}}{N_3} \quad (7)$$

In the above formula, N_{Δ} is the number of *triangles* in the network, and N_3 is the number of *connected triplets*, i.e., subgraphs of exactly 3 connected nodes (albeit not necessary triangular).

To compute a graph's transitivity, we have to count the number of triangles and the number of connected triplets therein. Such structures may be destroyed post-perturbation, yet we can still count the number of different triplet structures in G_p , and thereby estimate the number of triangles and connected triplets in G_o .

Given any triplet of nodes in G_o , the possible edge connections therein fall into the following four patterns:

- *Pattern 1:* They form a *triangle* (i.e., a triplet of *three edges*). We denote the number of triangles in G_o as T_o and their *observed* number in G_p as T_p .
- *Pattern 2:* They form a *connected triplet* with *two edges*. Let X_o (X_p) be the original (observed) number of connected triplets in G_o (G_p).
- *Pattern 3:* They form a disconnected triplet with only *one edge*. Again, let I_o (I_p) be the original (observed) numbers of this pattern in G_o (G_p).














		T_p	X_p				I_p			D_p
										
T_o		$(1-\mu)^3$	$(1-\mu)^2\mu$	$(1-\mu)^2\mu$	$(1-\mu)^2\mu$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	μ^3	μ^3
X_o		$(1-\mu)^2\mu$	$(1-\mu)^3$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	μ^3	$(1-\mu)^2\mu$	$(1-\mu)^2\mu$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$
I_o		$(1-\mu)\mu^2$	$(1-\mu)^2\mu$	μ^3	$(1-\mu)^2\mu$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	$(1-\mu)^3$	$(1-\mu)^3$	$(1-\mu)^3$
D_o		μ^3	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	$(1-\mu)\mu^2$	$(1-\mu)^2\mu$	$(1-\mu)^2\mu$	$(1-\mu)^2\mu$	$(1-\mu)^3$	$(1-\mu)^3$

Figure 2: Converting a pattern in G_o to another

- *Pattern 4*: They are completely disconnected with *no edges*. Similarly, we denote the original (observed) numbers of this pattern in G_o (G_p) as D_o (D_p).

Given a triplet pattern in G_o , we can calculate the probability that it will be transformed into one of the three other patterns in G_p . For instance, a triangle in G_o can become a triplet of either *Pattern 2*, *Pattern 3* or *Pattern 4*, or remain unchanged. Figure 2 depicts the calculations of the probabilities that a pattern in G_o is converted to another pattern in G_p . We emphasize that these calculations are obtained under the simplistic assumption that each triplet conversion is independent of others. This assumption serves our objective of deriving an estimation, which is ultimately tested in our experiments. Thus, for example, the probability that a triangle in G_o remains unchanged in G_p is $(1-\mu)^3$; the probability that a triangle becomes a triplet of *Pattern 2* (for which three possible cases as shown in Figure 2) is $3(1-\mu)^2\mu$. Given the counts of triplets in different patterns (T_p , X_p , I_p and D_p) in G_p , and the pattern conversion probabilities in Figure 2, we can devise a system of linear equations with four unknown estimator variables \hat{T}_o , \hat{X}_o , \hat{I}_o and \hat{D}_o , as we did for density and degree distribution estimation. Solving this system of equations, we get:

$$\hat{T}_o = \frac{1}{(2\mu-1)^3} (D_p\mu^3 - I_p\mu^2 + I_p\mu^3 - T_p + 3\mu T_p - 3\mu^2 T_p + \mu^3 T_p + \mu X_p - 2\mu^2 X_p + \mu^3 X_p) \quad (8)$$

$$\hat{X}_o = \frac{1}{(2\mu-1)^3} (2I_p\mu - 3D_p\mu^2 - 4I_p\mu^2 + 3D_p\mu^3 + 3I_p\mu^3 + 3\mu T_p - 6\mu^2 T_p + 3\mu^3 T_p - X_p + 3\mu X_p - 5\mu^2 X_p + 3\mu^3 X_p) \quad (9)$$

Then, given the maximum-likelihood estimators of triangles \hat{T}_o and connected triplets \hat{X}_o , we estimate the transitivity of G_o as:

$$\hat{\tau} = \frac{3\hat{T}_o}{\hat{X}_o} \quad (10)$$

3.4 Modularity Estimation

Social networks exhibit community structures. A characteristic of such a community is that intra-community links are more than inter-community links. The modularity is a metric measuring whether a partition of graph exhibits community properties. To compute modularity, the network is partitioned into a fixed number of communities. A symmetric matrix A is formed such that the elements $A[i, i]$ (i.e., the diagonal of matrix A) are the fractions of links between the nodes in the same community i . The other elements $A[i, j]$ are the fractions of links between communities i and j . The network's modularity is defined as:

$$\mathcal{M} = \sum_i [A[i, i] - (\sum_j A[i, j])^2] \quad (11)$$

The modularity value depends on the entries in matrix A . To estimate modularity, we first create an estimator \hat{A} for A . To do so, we use an auxiliary symmetric matrix, B , in which $B[i, i]$ is the number of edges within community i and $B[i, j]$ the number of edges between community i and community j . B_p and \hat{B} refer to the matrix in the perturbed graph and the estimator for B , respectively. Once \hat{B} is computed, \hat{A} can be derived from \hat{B} by dividing each entry by the estimated total number of edges. Since the entries in B are counts of edges between nodes, we can apply similar technique as in density estimation to estimate \hat{B} . B_p is directly computed by counting the edges within and between partitions in the perturbed graph. The relation between $\hat{B}[i, i]$ and $B_p[i, i]$ is:

$$\hat{B}[i, i] = \frac{2B_p[i, i] - (z_i^2 + z_i) \cdot \mu}{2 - 4\mu} \quad (12)$$

In the above equation z_i is the number of nodes in partition i . Assuming the number of nodes in community i (j) is z_i (z_j), the relation between $\hat{B}[i, j]$ and $B_p[i, j]$ is:

$$\hat{B}[i, j] = \frac{B_p[i, j] - z_i \cdot z_j \cdot \mu}{1 - \mu} \quad (13)$$

Each entry in \hat{A} is then computed by dividing the corresponding entry in \hat{B} by the estimated number of edges in the network.

3.5 A Generic Estimation Framework

While it is impossible to provide an exhaustive enumeration of estimation methods for every possible graph utility metric, in this section we provide a generic framework for the estimation a particular class of such metrics.

A common characteristic of the above four metrics is that the utility value relies on the counts of certain substructures (a sub-graph) in the graph. For example, the density value relies on the count of disconnected pairs of nodes and the count of connected pairs of nodes. Similarly, the transitivity value relies on the count of connected triplets and the count of triangles in the graph. Generally, the class of the utility metrics whose values rely on the counts of substructures can be estimated using a generic framework. The generic framework for estimation is described as follows.

Let \mathcal{S} be a set of s_{max} substructures relevant to a particular utility metric, where \mathcal{S}_i refers to the i^{th} substructure. For example, in transitivity estimation, $s_{max} = 2$, and the two substructures are triangles and connected triplets. $\mathcal{S}_{i.c}$ refers to the count of the i^{th} substructure in the original graph. Therefore, the utility value is a function of the counts of different structures in \mathcal{S} , i.e. $f(\mathcal{S}_{1.c}, \mathcal{S}_{2.c}, \dots, \mathcal{S}_{s_{max}.c})$. To estimate the utility value, we need to estimate the value of $\mathcal{S}_{i.c}$ for all substructures in \mathcal{S} . Let $\mathcal{S}_{i.s}$ be the number of nodes in the i^{th} substructure. For transitivity estimation, the number of nodes in the two substructures (triangles and triplets) are both 3. In the perturbed graph, we count the number of all substructures involving $\mathcal{S}_{1.s}, \mathcal{S}_{2.s}, \dots, \mathcal{S}_{s_{max}.s}$ nodes. The count of all the substructures in the perturbed graph is denoted as $p_1, p_2, \dots, p_{t_{max}}$. For transitivity estimation, they are the counts of the four patterns involving three nodes. The $\hat{\mathcal{S}}_{i.c} \forall i$, are maximum likelihood estimations for the parameters $\mathcal{S}_{i.c}$, and can be derived by solving the following maximization problem:

$$\hat{\mathcal{S}}_{1.c}, \hat{\mathcal{S}}_{2.c}, \dots, \hat{\mathcal{S}}_{s_{max}.c} = \arg \max_{\mathcal{S}_{i.c} \forall i} (\Pr(p_1, p_2, \dots, p_{t_{max}} | \mathcal{S}_{i.c} \forall i))$$

Given the estimation $\hat{\mathcal{S}}_{i.c} \forall i$, we can compute the estimated utility value using function f . Note that the counting of all substructures is only feasible for substructures involving a small number of

nodes. For example, in density, we count substructures involving two nodes, and in transitivity substructures involving three nodes only. For utility metrics involving substructures of a larger number of nodes, we can use sampling (defining an upper limit for counting or execution time) to estimate their values.

While not all utility metrics can be effectively recovered, there exists an ineffective algorithm (impractical due to the computation cost) that follows standard procedures to estimate all the metrics. The algorithm is described as follows: Let \mathcal{G} denote the set of all possible graphs on N number of nodes. For each possible graph $\mathcal{G}[i]$, there is a probability that $\mathcal{G}[i] = G_o$, which can be computed based on $\mathcal{G}[i]$, G_p and μ , and denoted by $\Pr(\mathcal{G}[i] = G_o|G_p)$. Consider a particular metric Z , we can measure its value $val_i(Z)$ on each $\mathcal{G}[i]$. Lastly, the sum of $val_i(Z) \cdot \Pr(\mathcal{G}[i] = G_o|G_p)$, $\forall i$ forms an estimation to the metric Z in the original graph. Although theoretically sound, this algorithm is impracticable as it requires the enumeration of all possible graphs on N nodes. A more efficient recovery of specific other metrics is left as an open problem.

The quality of our estimations is also a concern. To assess that quality, we can compute the standard error of the mean, $StErr$. For example, from Equation 4, we estimate the variance of \hat{h} as:

$$\sigma^2(\hat{h}) = \frac{\sigma^2(h_p)}{(1 - 2\mu)^2} \quad (14)$$

Let $R = 1$ ($R = 0$) denote the event there exists (does not exist) an edge between a particular pair of nodes. Therefore, $\Pr(R = 1) = \frac{h}{M} \cdot (1 - \mu) + (1 - \frac{h}{M}) \cdot \mu$, and $\Pr(R = 0) = \frac{h}{M} \cdot \mu + (1 - \frac{h}{M}) \cdot (1 - \mu)$. Therefore, $\sigma^2(h_p) = M \cdot \Pr(R = 1) \cdot \Pr(R = 0)$. Substituting in Equation 14, we get:

$$\sigma(\hat{h}) = \sqrt{M \cdot \left[\frac{1}{16 \cdot (\frac{1}{2} - \mu)^2} - \left(\frac{h}{M} - \frac{1}{2} \right)^2 \right]} \quad (15)$$

4. ATTACKING THE PERTURBED GRAPH

As we argued, by carefully designing estimation algorithms, many of the original graph properties can be accurately estimated. Nevertheless, the same potential can also be exploited by an adversary to launch more sophisticated attacks.

4.1 Principles of the Probabilistic Attack

The *probabilistic attack* starts out from the observation that an adversary can confidently estimate a degree interval (i.e., range) for each embedded malicious node. Using an approach similar to that of the *walk-based attack* [1], the adversary can efficiently enumerate a list of candidate *degree sequences* that will include, with high probability, the one that represents the embedded subgraph G_A . In most cases, the adversary can arrive at a single *degree sequence* that represents with high probability the subgraph embedded in G_o . This disposition is enabled by the fact that candidate degree sequences are *filtered* out in our attack methodology using two tests: the *interval degree check* and the *error-tolerant edge check*. These tests work differently from those of the *walk-based attack* [1].

	$k = 10$	$k = 20$	$k = 30$
$\mu = 0.0001$	0.9991	0.9981	0.9971
$\mu = 0.001$	0.9910	0.9812	0.9714
$\mu = 0.01$	0.9135	0.8262	0.7472

Table 1: Probability that k -path in G_A is preserved

We face a few challenges in demonstrating the feasibility of the attack: first, the prediction of degree ranges should be correct with

high probability. Second, the length of the predicted interval for an adversary's node should be as small as possible, as a large interval may result in a large number of nodes passing the *degree check*, causing a severe penalty to the attack's time complexity. Last, in order for our attack to succeed, the perturbation may alter the attacker's subgraph but must not have destroyed the k -path (i.e., path of k nodes) therein. In Table 1 we show the probability that this k -path is preserved (i.e., all its edges are preserved after perturbation) for several combinations of different μ and k , computed as $(1 - \mu)^{k-1}$ under perturbation probability μ . As this probability is quite high, the adversary can reasonably assume that the embedded subgraph's k -path is preserved in G_p after perturbation.

4.2 Predicting the Degree Interval

Let d_o be the degree of a malicious node in G_o and d_p the degree of the same node after perturbation. In the following, we first compute $\Pr(d_p|d_o)$, i.e., the probability that, given that the node's original node degree is d_o , its degree after perturbation is d_p .

Let r be the number of neighbors *eventually* removed from the set of d_o 's neighbors in G_o , and a the number of new neighbors added, due to perturbation. Without loss of generality, suppose that the d_p neighbors of the malicious node at hand are generated in two steps: first, r neighbors are disconnected and the number of remaining neighbors is $d_o - r$; then, $a = d_p - (d_o - r)$ nodes are connected and become neighbors, so the total number of neighbors is $d_o - r + a = d_p$. Then the following three inequalities hold:

$$r \geq 0, \quad d_o - r \geq 0, \quad d_p - (d_o - r) \geq 0 \quad (16)$$

It follows that r is in the range of $[\max\{0, d_o - d_p\}, d_o]$. The r neighbors can be removed in $\binom{d_o}{r}$ ways, and new neighbors added in $\binom{N - d_o - 1}{d_p - d_o + r}$ ways. Hence, $\Pr(d_p|d_o)$ is equal to:

$$\sum_r \binom{d_o}{r} \binom{N - d_o - 1}{d_p - d_o + r} \cdot (1 - \mu)^{N - 1 - (d_p - d_o + 2r)} \mu^{d_p - d_o + 2r}$$

Thus, an adversary can efficiently compute $\Pr(d_p|d_o)$. The possible values of d_p after perturbation ranges from 1 to $N - 1$. Yet the distribution of these values is not uniform. For each embedded node, the adversary can select a small subset of d_p values and build an interval \mathcal{I} representing the range of possible degrees for that node. We check inclusion in this interval as our *degree check*.

The removal and addition of neighbors of an embedded node can be viewed as two *independent Binomial processes*. The expected values for r and a are $E[r] = d_o \cdot \mu$ and $E[a] = (N - d_o - 1) \cdot \mu$, respectively. $\Pr(d_p|d_o)$ is maximized for $r = E[r]$ and $a = E[a]$ (hence $d_p = E[d_p]$). Then the chosen interval \mathcal{I} for the embedded node at hand is centered at $d_p = E[d_p]$, with w other values to its left and right, where w is a small non-negative integer. Eventually, the predicted degree interval for a selected malicious node in G_p is $\mathcal{I} = [E[d_p] - w, E[d_p] + w]$. Let $\Pr(d_p \in \mathcal{I})$ be the probability that the embedded node's degree is in \mathcal{I} after perturbation. An effective attack is possible if the adversary can find a fine-tuned value of w such that $\Pr(d_p \in \mathcal{I})$ is sufficiently large and yet the width of \mathcal{I} is small. Let $\mathcal{I}_{V_A[i]}$ be the degree interval for embedded node $V_A[i]$, and \mathcal{D}_p be the degree sequence of the embedded graph G_A in G_p . Then the probability that *all* embedded nodes' degrees fall into their respective intervals is $\prod_{i=1}^k \Pr(\mathcal{D}_p[i] \in \mathcal{I}_{V_A[i]})$. This calculation is made under the assumption that the events of different embedded nodes' degrees falling into their respective intervals are independent of each other, which helps us derive our estimate.

Table 2 shows the values of $\Pr(d_p \in \mathcal{I})$ for selected values of μ and w with $k = 12$. We observe that, when the number of nodes in the network is 10,000, the perturbation probability is 0.001 and $w = 4$, then the probability that a *single* embedded node falls into the interval \mathcal{I} is close to 1. Moreover, the probability that *all* the embedded nodes' degrees fall into their respective intervals after perturbation is also close to 1 in the same configuration. We conclude that, with this configuration, the attacker is almost sure that all embedded nodes will pass the *interval degree check*.

μ	w	$\Pr(d_p \in \mathcal{I})$	$\prod_{i=1}^k \Pr(\mathcal{D}_p[i] \in \mathcal{I}_{V_A[i]})$
$\mu = 0.001$	0	0.3670	5.9643×10^{-6}
$\mu = 0.001$	2	0.9814	0.7983
$\mu = 0.001$	4	0.9994	0.9931
$\mu = 0.01$	0	0.1246	1.3935×10^{-11}
$\mu = 0.01$	4	0.8488	0.1399
$\mu = 0.01$	8	0.9927	0.9158

Table 2: $\Pr(d_p \in \mathcal{I})$ with $N = 10,000$ and $d_o = 50$

For example, in the graph in Figure 1(c), the adversary's *degree sequence* is $[2, 5, 3, 4, 2]$. Yet in the perturbed graph G_p the degree of the node labeled 11 has become 1. Then, if a *walk-based attack* is launched, this node will not be detected. Still, with a *probabilistic attack*, the adversary is able to estimate the degree interval for each embedded node. For example (after integer rounding) the estimated degree intervals can be $[1, 2], [4, 5], [3, 4], [3, 4], [2, 3]$. In this scenario node 11 can still be accepted by the adversary as a candidate embedded node, allowing for a successful attack.

Algorithm 1: The *probabilistic attack*

Data: G_p, G_A, μ, w as chosen, $m = 0, k$ -path;
Result: A k -path containing identifiers of nodes in V_A ;
1 while k -path not found and w_{max}, m_{max} unreached **do**
2 $\mathcal{T} = \text{new Tree}()$; $level = 0$;
3 **foreach** $V[i]$ in G_p **do**
4 $\text{localSearch}(V[i], level, \mathcal{T}.root())$;
5 **if** $w < w_{max}$ **then**
6 $w++$;
7 **else if** $m < m_{max}$ **then**
8 $m++$;

Function $\text{localSearch}(currnode, level, parent)$

1 if $level = k$ **then**
2 **return**;
3 if $currnode$ passes *Interval Degree* and *ET-Edge checks* **then**
4 $\mathcal{T}.add(currnode, parent)$;
5 **foreach neighbor nb of** $currnode$ **do**
6 $\text{localSearch}(nb, level++, currnode)$;

4.3 Attack Algorithm

Algorithm 1 describes our *probabilistic attack*. \mathcal{T} is the tree of all candidate subgraphs, w is the width parameter for the degree intervals used in the *interval degree check* (chosen by the adversary as discussed) and m is the maximum number of errors that we allow in the employed *error-tolerant edge check*. The main block of the algorithm is a loop that continues until a k -path is found in \mathcal{T} by

the *localSearch* function (shown below), or both w and m reach their predefined maximum thresholds w_{max} and m_{max} .

Our probabilistic attack algorithm follows the pattern of a walk-based attack [1], with the key difference being on the two tests it performs (Line 3 in *localSearch*). To pass the *interval degree check*, a node's degree should fall in the predicted degree interval $V_A[level]$. To pass the *error-tolerant edge check*, the number of errors in *edge checks* accumulated in the path from this node to the root should not be larger than m . At each iteration of the main loop, if a k -path is not found, we relax the searching condition by either increasing w or m . However, using large w and m enlarges search space. The maximum w and m values that can be used depend on the attacker's computational resources.

4.4 Building Edges to Target the Victims

In order to compromise the victims' privacy, the adversary has to correctly identify the victims. However, due to perturbation, the links between the victim and the adversary's nodes may have changed which raises new challenges for identifying the victims. We propose a method that minimizes the impact of perturbation and establishes robust links against perturbation. Let the set of nodes that represent the victims in the network be $V_T = \{\tau_1, \tau_2, \dots, \tau_q\}$. $S_{\tau_i} \subset V_A$ is the set of maliciously embedded nodes that are linked to victim τ_i , $1 \leq i \leq q$. Our approach is as follows: we define two parameters ρ_1 and ρ_2 , where ρ_1 defines the minimum size of S_{τ_i} , and ρ_2 defines the minimum number of different members between the two sets S_{τ_i} and S_{τ_j} , for $i \neq j$. Formally:

$$\begin{cases} |S_{\tau_i}| \geq \rho_1 & \forall i \in [1, q] \\ |S_{\tau_i} \setminus S_{\tau_j}| \geq \rho_2 & \forall i, j \in [1, q] \text{ and } i \neq j \end{cases} \quad (17)$$

Moreover, we require that none of the adversary's nodes share common neighbors other than the nodes in V_T and V_A . To prove the robustness of the links between the victims and the adversary's nodes, we show analytically that the probabilities of the three events that affect the identification of the victims are negligible.

Claim 1: The probability that S_{τ_i} for any τ_i changes due to perturbation tends to be 0 when $\mu \rightarrow 0$.

PROOF. S_{τ_i} is preserved for any τ_i if the edge relations between this τ_i and all the adversary's nodes are preserved (i.e., $(1 - \mu)^k$). Therefore, the probability that S_{τ_i} changes is $1 - (1 - \mu)^k$. When $\mu \rightarrow 0$, $1 - (1 - \mu)^k \rightarrow 0$. \square

Claim 2: The probability that there is a node v outside sets V_A and V_T such that E_v , the set of edges between v and the nodes from V_A , is equal to S_{τ_i} for the victim τ_i , decreases fast with the increase of ρ_1 .

PROOF. Let us consider a particular node v which already has an edge with a node in S_{τ_i} . The probability that it forms new edges with all other nodes in S_{τ_i} but not with the nodes in $V_A - S_{\tau_i}$ is at most $\mu^{\rho_1-1} \cdot (1 - \mu)^{k-\rho_1+1}$. Moreover, the total number of such possible nodes v in the network is $N - k - q$. Therefore, $(N - k - q) \cdot \mu^{\rho_1-1} \cdot (1 - \mu)^{k-\rho_1+1}$ is the probability for the event in this claim. When $\mu = \frac{c}{N}$, this probability at most $\frac{c^{\rho_1-1}}{N^{\rho_1-1}}$ (by taking $N - k - q$ as N and $(1 - \mu)^{k-\rho_1+1}$ as 1), which decreases fast with the increase of ρ_1 . \square

Claim 3: The probability that the set S_{τ_i} of malicious nodes connected to victim τ_i becomes the same as the set S_{τ_j} of malicious nodes connected to victim τ_j after perturbation decreases fast with the increase of ρ_2 .

PROOF. Let the number of non-common elements in S_{τ_i} and S_{τ_j} be x_{ij} . Similarly to the derivation of equation 27, S_{τ_i} is con-

verted to S_{τ_j} by perturbation if and only if an x_{ij} number of edge additions and deletions occurs. Therefore, after perturbation, $\Pr(S_{\tau_i} = S_{\tau_j}) = (1 - \mu)^{k-x_{ij}} \cdot \mu^{x_{ij}}$. Since $\rho_2 \leq x_{ij}$, $\Pr(S_{\tau_i} = S_{\tau_j}) \leq \mu^{\rho_2}$, which decreases fast with the increase of ρ_2 . \square

A trivial algorithm that builds our robust links operates as follows: for each victim τ_i , repetitively select a subset of nodes in V_A at random, until a subset that satisfies both ρ_1 and ρ_2 requirements is found; then link τ_i to all the nodes in this subset of V_A .

4.5 Preventing the Probabilistic Attack

The adversary can successfully identify his subgraph based on the assumption that the k -path is not broken. However, the publisher can increase the perturbation probability so that, with high probability, the k -path is broken and therefore the *probabilistic attack* is infeasible. Let ε , the secure parameter, be the maximum probability that the k -path is preserved. Therefore,

$$(1 - \mu)^{k-1} \leq \varepsilon \quad (18)$$

The following inequality gives the minimum μ to be used so as to prevent the *probabilistic attack* with probability no less than $1 - \varepsilon$:

$$\mu \geq 1 - \sqrt[k-1]{\varepsilon} \quad (19)$$

5. GENERIC STRUCTURAL ATTACK

We now provide a generic analysis of a structural attack's probability of success. We assume that any structural attack can be translated into an instance of the graph isomorphism problem and show that an adversary who can enumerate all permutations of k nodes in G_p , will also be able, with high probability, to detect the embedded nodes under a random perturbation scheme for low μ .

Let Y_i be the i^{th} permutation of k nodes in the network. We assume an extreme case where the adversary has infinite computational power and is able to enumerate all permutations of k nodes in the network: $\mathcal{Y} = \{Y_i : 1 \leq i \leq P_k^N\}$ where $P_k^N = \frac{N!}{(N-k)!}$ is the total number of permutations. The adversary will then choose a particular permutation $Y \in \mathcal{Y}$ as a candidate for V_A and will assume that $Y[i]$ is $V_A[i]$. However, the adversary faces the following two challenges when choosing Y : first, the perturbation may have changed the embedded graph G_A in such a way that G_A cannot be detected in G_p . Second, even if the adversary succeeds in locating a permutation Y that matches subgraph G_A in appearance, there is still a probability that Y is not the actual V_A , but only a look-alike.

We define λ_Y as the probability that the chosen Y is V_A , given G_p . For the sake of conciseness, we use E_A^Y to denote the set of edges among the nodes in Y , after perturbation, in the event that this set has been created by perturbing the set E_A , i.e., in the event that $Y = V_A$. Then $\lambda_Y = \Pr(E_A^Y | E_p)$.

Let E_p^Y be the set of edges among the nodes in Y in the perturbed graph E_p . Then the following theorem holds:

THEOREM 5.1. *For a perturbed graph G_p of size N with a perturbation value μ , the probability that an adversary detects the embedded subgraph G_A , for a permutation of k nodes Y , is:*

$$\lambda_Y = \frac{\Pr(E_p^Y | E_A^Y)}{\sum_{i=1}^{P_k^N} \Pr(E_p^{Y_i} | E_A^{Y_i})} \quad (20)$$

PROOF. First, we rewrite the expression using Bayes' theorem,

$$\Pr(E_A^Y | E_p) = \frac{\Pr(E_p | E_A^Y) \cdot \Pr(E_A^Y)}{\sum_{i=1}^{P_k^N} \Pr(E_p | E_A^{Y_i}) \cdot \Pr(E_A^{Y_i})} \quad (21)$$

In the above equation, $\Pr(E_A^{Y_i})$ is the prior probability of Y being the attacker's nodes, which is the same for all i . Therefore:

$$\Pr(E_A^Y | E_p) = \frac{\Pr(E_p | E_A^Y)}{\sum_{i=1}^{P_k^N} \Pr(E_p | E_A^{Y_i})} \quad (22)$$

We first focus on the numerator $\Pr(E_p | E_A^Y)$ in Equation 22. We split the set of edges in the perturbed graph into two sets: E_p^Y , the edges between the nodes in Y , only and $\overline{E_p^Y}$, the other edges in E_p . By definition, $\overline{E_p^Y} = E_p - E_p^Y$. E_p^Y and $\overline{E_p^Y}$ are independent, hence:

$$\Pr(E_p) = \Pr(E_p^Y) \cdot \Pr(\overline{E_p^Y}) \quad (23)$$

Inserting the conditional variable E_A^Y to Equation 23, we get:

$$\Pr(E_p | E_A^Y) = \Pr(E_p^Y | E_A^Y) \cdot \Pr(\overline{E_p^Y} | E_A^Y) \quad (24)$$

Since $\overline{E_p^Y}$ and E_A^Y are independent, $\Pr(\overline{E_p^Y} | E_A^Y) = \Pr(\overline{E_p^Y}) = \frac{\Pr(E_p)}{\Pr(E_p^Y)}$ (by Equation 23). Then Equation 24 becomes:

$$\Pr(E_p | E_A^Y) = \Pr(E_p^Y | E_A^Y) \cdot \frac{\Pr(E_p)}{\Pr(E_p^Y)} \quad (25)$$

Combining Equations 22 and 25, and using $E_p^{Y_i}$ for E_p^Y , we get:

$$\Pr(E_A^Y | E_p) = \frac{\Pr(E_p^Y | E_A^Y)}{\sum_{i=1}^{P_k^N} \Pr(E_p^{Y_i} | E_A^{Y_i})} \quad (26)$$

\square

The intuition behind the derived equation for λ_Y is the following: the numerator $\Pr(E_p^Y | E_A^Y)$ is the likelihood of the particular chosen permutation Y to be V_A . The denominator is the sum of the likelihoods of each permutation Y_i in the network being V_A . The ratio describes the probability of success of the particular selection Y being V_A . Thus, the value of λ_Y depends on the value of $\Pr(E_p^Y | E_A^Y)$ and the sum of $\Pr(E_p^{Y_i} | E_A^{Y_i})$ for all i . In effect, the computation of the exact λ_Y would require the enumeration of all permutation of k nodes in the graph. In the following, we study the conditional probability $\Pr(E_p^Y | E_A^Y)$ for a particular Y .

Given that Y is the set of embedded nodes and E_A^Y is the set of edges among the nodes in Y before perturbation, $\Pr(E_p^Y | E_A^Y)$ is the probability that the set of edges among the nodes in Y becomes E_p^Y after perturbation. With this fact in mind, the derivation of $\Pr(E_p^Y | E_A^Y)$ becomes easy: let m be the number of *non-common* edges in E_p^Y and E_A^Y . The minimum value of m is 0, achieved when E_A^Y and E_p^Y are identical, and its maximum value is $M = \frac{k^2-k}{2}$, obtained when E_A^Y and E_p^Y are totally foreign to each other. Since each edge removal or addition occurs with probability μ , the probability that E_A^Y is converted to E_p^Y is:

$$\Pr(E_p^Y | E_A^Y) = \mu^m \cdot (1 - \mu)^{M-m} \quad (27)$$

5.1 Estimation of λ_Y

From a (computationally powerful) adversary's point of view, Equation (20) can be used to compute λ_Y for each $Y = Y_i$. The attacker will then assume that the set Y that gives the maximal value of λ_Y is the embedded subgraph V_A in G_p . In particular, given that the denominator is constant for a given E_A and G_p , the

μ	ℓ	$\hat{\lambda}_Y$ when $m = \ell$	$\Pr(m \leq \ell)$
0.0001	0	1	0.9955
0.0001	5	1	1
0.0001	10	1	1
0.001	0	1	0.9559
0.001	5	1	1
0.001	10	0.0031	1

Table 3: λ_Y with $k = 10$, $M = 45$, $N = 10,000$

adversary will simply opt for the set Y that maximizes the numerator in Equation (20). The best-case scenario for an adversary is that $m = 0$ (i.e., E_p^Y and E_A^Y are identical). Otherwise, an adversary will choose the Y that gives the most similar subgraph to G_A .

In the following we provide a simple method for estimating the value of λ_Y . For any permutation of k nodes Y_i , let $m_{Y_i} = |E_{A^Y}^{Y_i} \cup E_p^{Y_i} - E_A^{Y_i} \cap E_p^{Y_i}|$. We consider $E_p^{Y_i}$ as a random subset chosen from all possible edges among the nodes of Y_i . Then the expected value of m_{Y_i} is $\frac{M}{2}$ and the expected value of $\Pr(E_p^{Y_i} | E_A^{Y_i})$ is $\mu^{\frac{M}{2}} \cdot (1 - \mu)^{\frac{M}{2}}$. Then our simple estimate of λ_Y is:

$$\hat{\lambda}_Y = \min \left\{ 1, \frac{1}{P_k^N} \left(\frac{1 - \mu}{\mu} \right)^{\frac{M}{2} - m} \right\} \quad (28)$$

Thus, our estimate of λ_Y depends on μ and m . Table 3 lists different λ_Y estimates with respect to different μ and m combinations in a network of 10,000 nodes with 10 malicious nodes. Intuitively, m can be seen as the number of altered edges from E_A^Y to E_p^Y . We note that, unless both μ and m are high, $\hat{\lambda}_Y$ approaches 1.

We observe that λ_Y tends to be large only when m is small. If the lowest value of m is kept large (e.g., greater than 10, as in the last row of Table 3), then $\hat{\lambda}_Y$ is kept low. In effect, to assess the potential for a successful attack, we study the distribution of m under random edge perturbation. This perturbation can be seen as a binomial process that adds and deletes edges with probability μ . Then the probability distribution of m is:

$$\Pr(m \leq \ell) = \sum_{m=0}^{\ell} \binom{M}{m} (1 - \mu)^{M-m} \mu^m \quad (29)$$

The last column of Table 3 shows instances of the probability distribution for m with the values of k and μ fixed. We observe that $\Pr(m = 0)$ equals 0.9955 and 0.9559 for $\mu = 0.0001$ and $\mu = 0.001$. This result shows that m tends to be rather small with high probability. Thus, we conclude that λ_Y tends to be large for such perturbation probabilities. This is good news for an adversary who has the computational powers to enumerate all subgraphs and choose the one most similar to the embedded one.

6. EXPERIMENTAL EVALUATION

We now present our experimental evaluation. First, we investigate the probability of success of the *probabilistic attack* under different values of μ , and measure its execution time. Next, we investigate the effect of perturbation on the graph properties.

All experiments ran on a 2.33GHz CPU, Windows-XP machine with 3.25GB RAM. We employ two real datasets: The Enron dataset¹ is the graph of email exchange among employees of Enron, having 4,644 accounts. Each account corresponds to a node and two accounts are linked if they have exchanged emails in both directions. The DBLP dataset² is a random subset of 20,000 authors from the

DBLP bibliography. Each author corresponds to a node and two authors are linked if they are coauthors in at least one paper. The Wiki dataset³ is a network Wikipedia encyclopedia writers around the world. It consists of 7,115 nodes and 103,689 edges.

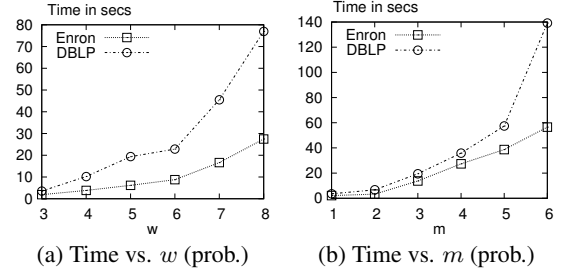


Figure 3: Efficiency of the probabilistic attack

6.1 Assessing the Probabilistic Attack

In our first experiment, we assess the probability of success of our *probabilistic attack* as opposed to that of the classical *walk-based attack* [1]. We first test the *walk-based attack* on the Enron and DBLP data, measuring its success rate in trials of 200 separate attack runs, as a function of the perturbation probability μ . An attack run is considered to be successful, if the adversaries can detect the sequence of embedded nodes and re-identify at least one victim node. In [1], the suggested number of malicious nodes k is $\Theta(\log(N))$ and the number of victim nodes is $q = O(\log^2(N))$. Following this suggestion, we vary k at values 20, 25, 30 for both graphs, with number of victims 100, 157 and 225, respectively.

Figures 4a,e show our results, which provide a glimpse of the probability that an adversary successfully identifies the embedded nodes in perturbed DBLP and Enron data using a walk-based attack. When μ is 0, all attacks are 100% successful. Still, already for rather small values of μ (10^{-7} to 10^{-6}), the success rate drops drastically to very low values. In addition, the success rate is lower for larger k under the same perturbation value μ ; that is because, with larger k , the node degree sequence of the malicious nodes is more likely to be changed or the backbone to be broken, making the attack more likely to fail. In effect, the walk-based attack can be effectively prevented through random edge perturbation, with minimal impact on the graph's structure (as μ is negligible).

On the other hand, Figures 4b-d,f-h show the success rate of the *probabilistic attack* on the same DBLP and Enron data, again in trials of 200 runs each. We show results for several values of the interval-width parameter w . As the search space of the attack algorithm grows with w , the success rate also rises with it. For $\mu \simeq 10^{-4}$, the probabilistic attack succeeds in almost 100% of the cases, in stark contrast to the walk-based one. Still, as μ grows further, the observed success rate swiftly drops for all values of w . As with the walk-based attack, the success rate falls as k grows.

Figures 3a,b show the execution time of our attack as a function of interval-width w and error-tolerance m , with $\mu = 10^{-3}$. The number of malicious nodes is $k = 4 \log(N)$ and the number of victims $q = \log^2(N)$. The algorithm's search space grows with both w and m (Section 4.2), hence the execution time also ascends with them, yet remains lower than 3 minutes, rendering the attack rather feasible on reasonably-sized real-world data sets.

An adversary who successfully identifies the embedded subgraph inside the perturbed graph may yet not locate the target victims, as the edges between the embedded nodes and the victims may have been removed. Table 4 shows the measured percentage of victims that can be identified in a successful attack. The number of malicious nodes and the victims remain at $k = 4 \log(N)$ and at

¹ <http://www.cs.cmu.edu/~enron>

² <http://dblp.uni-trier.de/xml/>

³ <http://snap.stanford.edu/data/wiki-Vote.html>

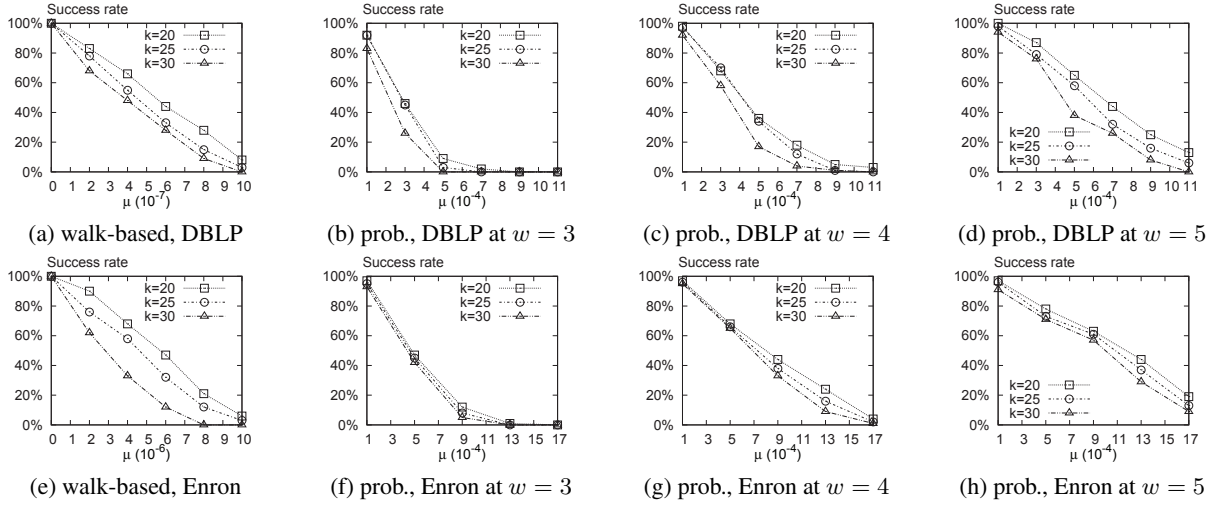


Figure 4: Evaluation of Probabilistic Attack

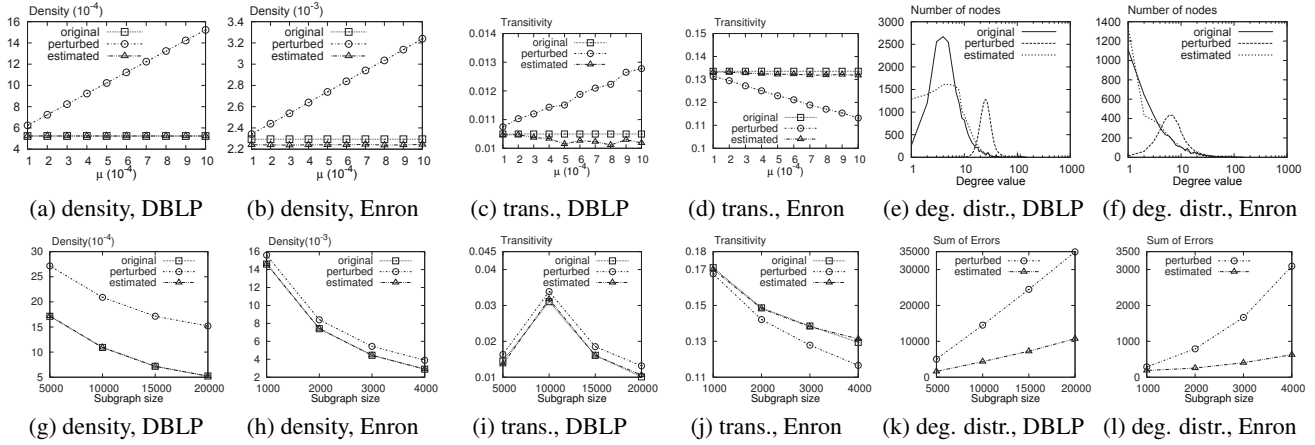


Figure 5: Evaluation of utility preservation

$q = \log^2(N)$. As the table shows, more than 91% of victims are identified when the attack succeeds.

μ	Enron	DBLP
1×10^{-4}	95.2%	98.3%
2×10^{-4}	93.6%	98.3%
3×10^{-4}	92.7%	96.7%
4×10^{-4}	91.9%	94.2%

Table 4: Percentage of affected victims

The error-tolerance m also affects an attack’s probability of success. Table 5 shows an instance of this effect: in a trial of 100 attacks with $m = 0$ on the Enron graph perturbed with $\mu = 0.04$, there are 53 successes, 35 failures due to false predicted interval, and 6 due to broken path or edge check failures. Still, when we relax the requirement for passing the edge check test, we can increase the number of successes to 58 and 59.

Events	$m=0$	$m=1$	$m=2$
Success	53	58	59
False prediction	35	35	35
Broken path	6	6	6
Edge check fail	6	1	0

Table 5: Effect of m

To sum up, the probabilistic attack is more effective than the

walk-based one *and* feasible in terms of runtime. Still, *both* can be prevented under random perturbation with sufficiently large μ .

6.2 Assessing Utility Preservation

We use the perturbed data derived in previous experiments to evaluate the extent to which graph properties are preserved. Figures 5a-f show the density, transitivity and degree distribution for perturbed DBLP and Enron data. Each figure shows the original, perturbed, and estimated values. Density and transitivity values vary with the perturbation probability μ , while the degree distribution is given as a single snapshot for $\mu = 10^{-3}$. We observe that, while graph properties deviate significantly from the originals as μ grows, our derived *estimates* are resilient to perturbation and approximate the original values well. Significantly, our estimation algorithms provide accurate approximations even under perturbation probabilities that render probabilistic attacks unviable.

Furthermore, we illustrate the benefit of our method in terms of measuring the properties of *subgraphs* of the original graph. This application of our technique offers an undeniable benefit, as publishing the graph cannot be substituted by directly publishing the measures of interest. We measure the same six structural properties over subgraphs of different sizes extracted from the original DBLP and Enron graphs, and average our results over 10 different randomly extracted subgraphs. Figures 5g-l show our results as a function of subgraph size, with $\mu = 0.001$. For density and transi-

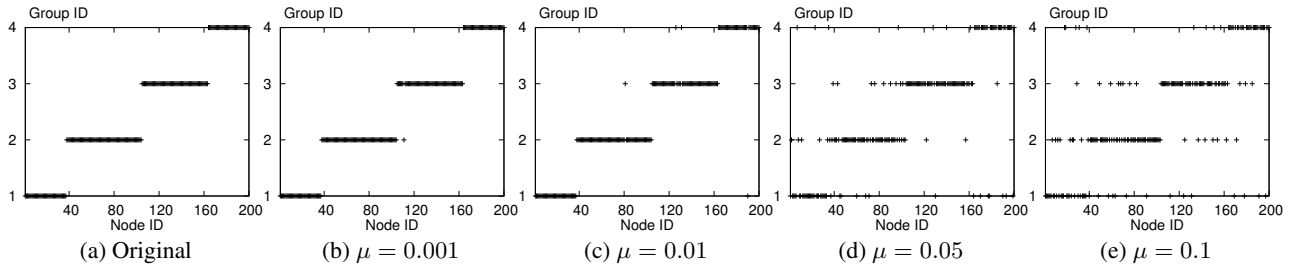


Figure 6: Classification of nodes under perturbation

tivity, we show again the original, perturbed, and estimated values, for each subgraph size. For degree distribution, we now present the sum of absolute errors between the perturbed/estimate and original distribution values for each degree. Our results reconfirm our findings even with subgraphs extracted from the parent graphs.

Overall, our results corroborate our claim that a published graph allows accurate estimations of graph properties not only for the whole graph, but also for subgraphs thereof, where publishing the measures directly to end-users would not be a satisfactory solution.

6.3 Distance-based Classification

We now attempt to perform a specific data mining task, distance-based node classification, over perturbed data.

Social networks often possess hubs, i.e., nodes with very high degree. A particular person’s connectivity pattern to the hubs indicates that person’s interests. For example, in a social election, a person’s voting pattern may indicate its political views. Thus, node classification based on such patterns is useful. We consider a classification of nodes based on the distance between their hub connectivity pattern (HCP) and some target patterns (TPs). Given a set of hubs, a node’s HCP is the subset of hubs that this node has connectivity to. Each TP_i is a subset of the hubs defined by the analyst. Given a set of k hubs, HCP and TP_i are k -dimensional binary vectors. The distance between HCP and a particular TP_i is the edit distance between the two vectors. For each TP_i , a group of nodes G_i is formed by assigning group membership to the nodes that have closest distances to TP_i than to all other $TP_j (i \neq j)$. We aim to classify each node to the right group.

We use the Wiki graph. Hubs are chosen as the nodes that have top-10 degrees in the graph (ranging from 482 to 1,053). We extract a subset of 200 nodes for classification, each having at least 4 connections to the hubs. For instance, the 10-dimensional binary data (0 0 1 0 1 0 0 1 1 0) represent the HCP for a node that has connectivity to the 3rd, 5th, 8th and 9th hubs. We define four target patterns, $TP_1 = (0 0 0 0 0 0 0 0 0 0)$, $TP_2 = (0 1 0 1 0 1 0 1 0 1)$, $TP_3 = (1 1 1 1 1 1 1 1 1 1)$ and $TP_4 = (1 0 1 0 1 0 1 0 1 0)$, hence 4 classes of nodes. We assign IDs to the nodes so that nodes that are classified into the same group have consecutive IDs. Figure 6(a) visualizes the original classification. Figures 6(b,c,d,e) show the classification obtained from the perturbed graph with increasing μ . Remarkably, the classification is faithful even with $\mu = 0.01$, with which both *walk-based attack* and *probabilistic attack* fail (see Figure 4). Classification error becomes apparent with larger μ as Figures 6(d,e) show, while most nodes are still correctly classified.

7. CONCLUSIONS

In this paper we have delineated the effectiveness of random edge perturbation as a tool for privacy-preserving publication of graph data in the face of structural attacks. We have shown that sophisticated attacks based on probabilistic heuristics are feasible when the perturbation probability is sufficiently low, but can be

thwarted as that probability grows. Moreover, we conducted an in-depth theoretical study of the probability of success for *any* structural attack. Our analysis offers a yardstick for assessing the privacy risk entailed by the publication of perturbed graph data. On the other hand, we developed methods that accurately estimate the properties of the original graph from the perturbed data. A thorough experimental study validates our analysis and explores the tradeoff between privacy and utility. Last, we have effectively demonstrated that data mining tasks such as distance-based node classification can be successfully performed even under random edge perturbation probabilities with which structural attacks typically fail.

8. ACKNOWLEDGEMENTS

We thank Dr. Yufei Tao for sharing insights on this topic, and Dr. Ee-Chien Chang and Dr. Hai Feng Yu for their constructive feedback. Mingqiang Xue is supported by I²R grant.

9. REFERENCES

- [1] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.
- [2] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In *ICDE*, 2011.
- [3] J. Cheng, A. W.-C. Fu, and J. Liu. k -isomorphism: privacy preserving network publication against structural attacks. In *SIGMOD*, 2010.
- [4] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *VLDB*, 2008.
- [5] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Adv. Phys.*, 2007.
- [6] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, 2002.
- [7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- [8] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *PVLDB*, 1(1):102–114, 2008.
- [9] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *Technical Report 07-19*, 2007.
- [10] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.
- [11] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, 2002.
- [12] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SIAM SDM*, 2008.
- [13] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.
- [14] L. Zou, L. Chen, and M. T. Özsu. k -automorphism: A general framework for privacy preserving network publication. In *VLDB*, 2009.