# Anonymization and de-Anonymization

Department of Computer Science
Georgia State University

## Introduction

Definition:

Data anonymization is a type of information sanitization whose intent is **privacy protection**. It is the process of either encrypting or removing personally **identifiable information** from data sets, so that the people whom the data describe remain **anonymous**.

## Introductions

- Organizations typically need to publish microdata e.g., census data or election data
- Data publishing gives useful information to researchers and analyzers
- At the same time, it extends a privacy risk to individuals whose data is being published
- We need strong privacy notions that enable us to confine the disclosure risk while simultaneously maximize the benefits

## Three types of Anonymization

| Mask identifier in unstructured data | Privacy preserving data analysis | Transforming structured data |
|---|---|---|
| Subject: clinical notes… | Subject: query results… | Subject: tabular data… |
| Method: machine learning, regular expression | Method: interactive differential privacy, query-set-size control… | Method: generalization, suppression, randomization |
| Implementation: MIST, MITdeid, NLM Scrubber | Implementation: Aircloak, Airavat, FUZZ, PINQ, HIDE | Implementation: AnonTool, ARX, PARAT |

## Multiple aspects have to be balanced

**Main Goal: Achieve a balance between data utility and privacy**

**Complex Tasks:**

- **Many different types of methods need to be applied in an integrated manner**
- **Methods may need be parameterized**
- **Different aspects are interrelated**

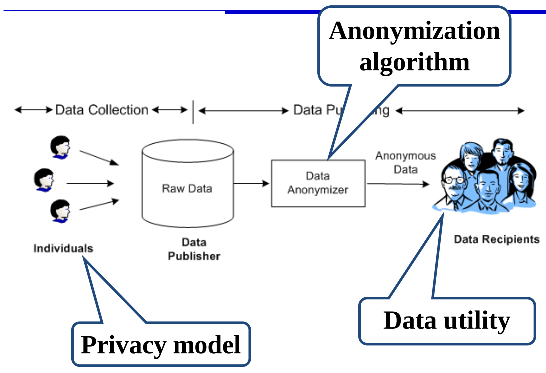
## The Players

- ✓ The *user* (whose privacy we want to protect)
- ✓ The *publisher* or *data owner* (role: anonymize data before publication) G -> anonymized G*
- ✓ The *attacker* or *adversary*
- ✓ The *analyst*

## Basic element

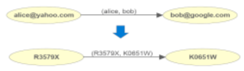
1. Models of Privacy
   - What pieces of information, we want to protect – usually, information is classified as sensitive (i.e., private) and not sensitive
2. Background Knowledge
   - What an adversary may know
3. Models of Utility
   - Use of the published data

## Overview

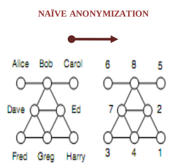


## Pseudo- or Naive anonymization

replace identifying attributes with synthetic (encrypting) identifiers



*Is this enough?*



## Privacy Model

- Privacy classified into

1. Identity disclosure: the *identity* of an individual who is associated with a node is revealed (or, just whether an individual participates or not)
2. Link disclosure: the sensitive *relationship* between individuals is disclosed, and
3. Content disclosure: the *sensitive data* associated with each node or edge is compromised, for example, the email message sent and/or received, the label (e.g., age, political beliefs, etc) associated with a node, properties such as the degree of a node, and general graph properties, etc

## The attacker

1. *Background Knowledge* (what the attacker knows besides the released network)
   e.g., the age of the victim, the number of her friends, etc
   also, combine from participation in many networks

2. Acquired through *malicious actions* of the adversary called *attacks*

   ❖active: an adversary tries to compromise privacy by strategically creating new user accounts and links ***before*** the anonymized network is released

   ❖passive: try to learn the identities of nodes only ***after*** the anonymized network has been released

❖ a closed world adversary, in which case, external information is complete and absent facts are considered false,
❖ an open world adversary, in which case, external information may be incomplete and absent facts are simply unknown

## Utility model

*information loss* or *anonymization quality*

▪ *Preserve general graph properties* (diameter, distribution of node degrees, number of nodes/edges, average path length, clustering coefficient, etc)

▪ *Quality of the results of queries*. E.g., Aggregate network *queries*: compute the aggregate on some path or subgraph that satisfies some given condition

Example: the average distance from a medical doctor vertex to a teacher vertex

Useful in many applications, such as customer relationship management

## Anonymization methods

Initial Graph G -> Anonymized Graph G*

- Clustering-based approaches: clusters nodes and edges into groups and replaces a sub-graph with a super-node (Generalization)
- Graph modification approaches: modifies (inserts or deletes) edges and nodes in the graph (Add noise or "perturb" the graph)

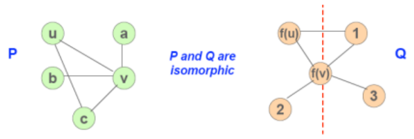
## Mappings that preserve the graph structure

Mappings that preserve the graph structure

A graph homomorphism $f$ from a graph $G = (V, E)$ to a graph $G' = (V', E')$, is a mapping f: G → G', from the vertex set of G to the vertex set of G' such that

$(u, u') \in G \Rightarrow (f(u), f(u')) \in G'$

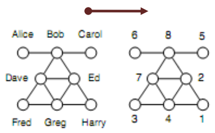
If the homomorphism is a bijection whose inverse function is also a graph homomorphism, then $f$ is a graph isomorphism $[(u, u') \in G \Leftrightarrow (f(u), f(u')) \in G']$



## Naiv anonymization produce an isomorphic

Data owner (publisher) Data Analyst

NAÏVE ANONYMIZATION



Good utility: *Published graph isomorphic to input graph*

The general graph isomorphic problem which determines whether two graphs are isomorphic is NP-hard

## Other issues

Beside privacy-aware publishing (non-interactive) mechanisms

Interactive mechanism

a question is posed, the exact answer is computed by the curator, and then a noisy (anomymized) version of the true answer is returned to the user

Beside publishing:

- private attributes
- access control

Location privacy

## Outline

1. An example of an **active attack**
2. General “structural” **k-anonymity**
3. An example of combining information from **more than one network**
4. (besides data publishing) How to infer **private attributes**

## Active and passive attacks

Definition:

Identify type of attacks that even from a **single anonymized copy** of a social network, it is possible for an adversary to **learn whether edges exist or not** between specific targeted pair of nodes

✓Note that the adversary may be a user of the system being anomymized

## Passive attacks

Learn the identities of the nodes ***after*** the anonymized network has been released

How?

Users ***find themselves*** in the released network and from this discover the existence of ***edges among users to whom they are linked***

Based on the observation that:

most nodes in real social networks already belong to *a small subgraph*, thus if a user can collude **with a coalition of *k*-1 friends** after the release, he/she is able to identify *additional nodes that are connected to this coalition*

## Active attacks

compromise privacy by strategically ***creating new user accounts*** and links ***before*** the anonymized network is released

1. Chooses an arbitrary set of users whose privacy it wishes to violate,
2. Creates a small number of new user accounts with edges to those targeted users, and
3. Creates ***patterns of links*** among the new accounts to make it ***identifiable*** in the anomymized graph structure

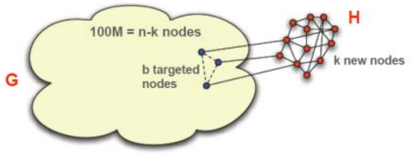
## Passive attack VS Active attack

✓ Active work in with high probability in any network – passive rely on the chance that a use can uniquely find themselves after the network is released

✓ Passive attacks can compromise the privacy only of users linked to the attacker

✓ Passive attacks no observable wrong doing

## Active Attacks: Overview

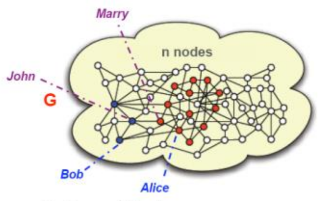


Anonymized network G of n-k nodes

Attacker
- Choose *b* targeted users
- Create a sub-graph *H* containing *k* nodes
- Attach *H* to the targeted nodes

**How to create the sub-graph *H***

## Active attacks: overview

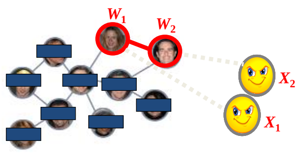


After the anonymized network is released:
- Find the sub-graph *H* in the graph G
- Follow edges from *H* to locate the *b* target nodes and their true location in G
- Determine all edges among these *b* nodes (privacy breach)

**How to locate the sub-graph *H***

## Walk-based attacks---simplified version

- Construction:
  - Pick target users $W = \{w_1,\ldots,w_k\}$
  - Create new users $X = \{x_1,\ldots,x_k\}$ and random subgraph <u>***G*[*X*] = *H***</u>
  - Add edges $(x_i, w_i)$



- Recovery
  - Find *H* in *G* ↔ No subgraph of G isomorphic to H
  - Label *H* as $x_1,\ldots,x_k$ ↔ No automorphisms
  - Find $w_1,\ldots,w_k$

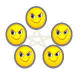
## Active attack

*What is a good H?*

With high probability, *H* must be:

- Uniquely identifiable in *G*
  - For any *G*, *there is no* $S \neq X$ *such that G[S] and G[X] are isomorphic (if we found a copy of H, this is the correct one)*
- Efficiently locatable in G
  - Tractable instance of sub-graph isomorphism
- Subgraph H has non trivial automorphism
  - *H has no non trivial automorphism (isomorphism to itself (relabeling of the nodes)]*
- But undetectable
  - From the point of view of the *data curator*

## Active attacks --- approaches



- Basic idea: *H* is *randomly generated*
  - Start with *k* nodes, add edges independently at random
- Two variants:
  - $k = \Theta(\log n)$ de-anonymizes $\Theta(\log^2 n)$ users
  - $k = \Theta(\sqrt{\log n})$ de-anonymizes $\Theta(\sqrt{\log n})$ users
    - H needs to be "more unique"
    - Achieved by "thin" attachment of H to G

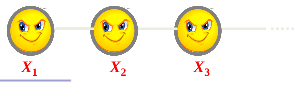
The "Walk-based" attack – better in practice

The "Cut-based" attack – matches theoretical bound

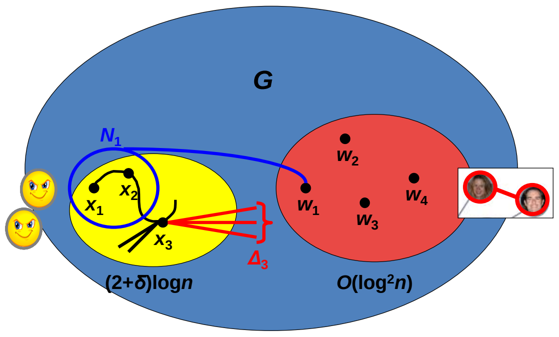
## Walk-based attack—construction of H

- Construction:
  - Pick target users $W = \{w_1,\ldots,w_b\}$
  - Create new users $X = \{x_1,\ldots,x_k\}$ and *H*
  - Connect $w_i$ to a *unique subset* $N_i$ of *X*
  - Between *H* and *G* – *H*
    - Add $\Delta_i$ edges from $x_i$ where $d_0 \leq \Delta_i \leq d_1 = O(\log n)$
  - Inside *H*, add edges $(x_i, x_{i+1})$

To help find *H*



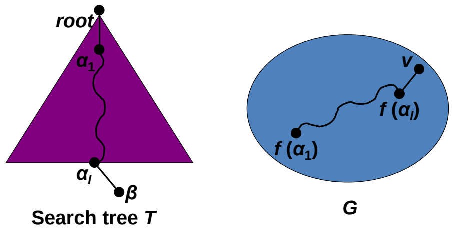
## Walk-based attack—construction of H



- Total degree of $x_i$ is $\Delta'_i$

## Walk-based attack—Recovering H

- Search G based on:
  - Degrees $\Delta'_i$
  - Internal structure of *H*



## Walk-based attack—Analysis

- Theorem 1 [*Correctness*]:
  With high probability, *H* is unique in *G*. Formally:
  - *H* is a **random** subgraph
  - *G* is **arbitrary**
  - Edges between H and G – H are **arbitrary**
  - There are edges $(x_i, x_{i+1})$
  - ➢ Then WHP no subgraph of *G* is isomorphic to *H*.
- Theorem 2 [*Efficiency*]:
  Search tree *T* does not grow too large. Formally:
  - For every $\varepsilon$, WHP the size of *T* is $O(n^{1+\varepsilon})$

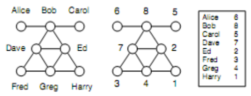
## K-structure anonymity

A number of approaches based on *k*-anonymity: at least *k* elements satisfy a constraint
Hide the private information with these *k* elements

Methods based on ***"structural"*** k-anonymity (the attacker knowledge concerns the structure of the graph)
- k-candidate
- k-degree
- k-neighborhood

## K-candidate anonymity--- introduction

$G_a$ the naive anonymization of G through an anonymization mapping f



| | |
|---|---|
| Alice | 6 |
| Bob | 8 |
| Carol | 5 |
| Dave | 7 |
| Ed | 2 |
| Fred | 3 |
| Greg | 4 |
| Harry | 1 |

- An individual $x \in V$ called the target has a candidate set, denoted cand(x) which consists of the nodes of $G_a$ that could possibly correspond to x

Given an ***uninformed adversary***, each individual has the same risk of re-identification, cand(x) = $V_a$

In practice, some knowledge, examples:
- ✓ Bob has three or more neighbors, cand(Bob) =?
- ✓ Greg is connected to at least two nodes, each with degree 2, cand(Greg) =?

## K-candidate anonymity--- introduction

- Target x ∈ V
- cand(x) nodes of $G_a$ that could possibly correspond to x

**Topic of this work:**
✓Definition of the problem: size of cand(x) (at least size k)
✓Model background knowledge
✓ Study both real (experimentally) and synthetic (analytically) graphs in terms of the size of cand(x)

## K-candidate anonymity--- introduction

- (background knowledge) structural re-identification where the information of the adversary is about graph structure
- (utility) analysis about structural properties: finding communities, fitting power-law graph models, enumerating motifs, measuring diffusion, accessing resiliency

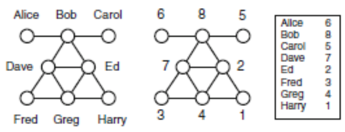
Two important factors

- descriptive power of the external information
- structural similarity of nodes

## Automorphic Structural Similarity

A strong form of structural similarity.

[**automorphic equivalence**]. Two nodes x, y ∈ V are automorphically equivalent (denoted x ≡ y) if there exists an isomorphism from the graph onto itself that maps x to y.



| | |
|---|---|
| Alice | 6 |
| Bob | 8 |
| Carol | 5 |
| Dave | 7 |
| Ed | 2 |
| Fred | 3 |
| Greg | 4 |
| Harry | 1 |

Example: Fred and Harry, but not Bob and Ed

## Automorphic Structural Similarity

Automorphic equivalence induces **a partitioning on V** into sets whose members have identical structural properties.
Size of each partition at least ***k***

An adversary —even with exhaustive knowledge of the structural position of a target node — cannot identify an individual beyond the set of entities to which it is automorphically equivalent.

- Some special graphs have large automorphic equivalence classes.
  - E.g., complete graph, a ring
- In general, an ***extremely strong notion*** of structural similarity.

## Model of Adversary knowledge

An adversary access a source that provides answers to a **restricted knowledge query Q** evaluated for a **single target node** of the **original graph G**.

*knowledge gathered by the adversary is accurate.*

For target x, use Q(x) to refine the candidate set.

[CANDIDATE SET UNDER Q]. For a query Q over a graph, the candidate set of x w.r.t Q is candQ(x) = $\{y \in V_a \mid Q(x) = Q(y)\}$.

## Model of Adversary knowledge

Adversary knowledge about a targeted individual tends to be *local* to the targeted node

Three Types of Queries

1. Vertex Refinement Queries
2. Subgraph Queries
3. Hub Fingerprint Queries

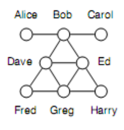
## Vertex Refinement

A class of queries of increasing power which report on the *local structure* of the graph around a node.

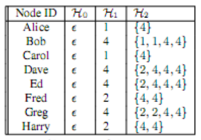
- **$H_0$**, simply returns the label of the node.
- **$H_1$(x)** returns the degree of x,
- **$H_2$(x)** returns the multiset of each neighbors' degree,
- **$H_i$(x)** returns the multiset of values which are the result of evaluating $H_{i-1}$ on the nodes adjacent to x

**H*** Iterative computation of H until no new vertices are distinguished.

## Vertex Refinement



(a) graph



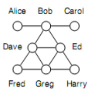
| Node ID | $\mathcal{H}_0$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
|---|---|---|---|
| Alice | $\epsilon$ | 1 | $\{4\}$ |
| Bob | $\epsilon$ | 4 | $\{1,1,4,4\}$ |
| Carol | $\epsilon$ | 1 | $\{4\}$ |
| Dave | $\epsilon$ | 4 | $\{2,4,4,4\}$ |
| Ed | $\epsilon$ | 4 | $\{2,4,4,4\}$ |
| Fred | $\epsilon$ | 2 | $\{4,4\}$ |
| Greg | $\epsilon$ | 4 | $\{2,2,4,4\}$ |
| Harry | $\epsilon$ | 2 | $\{4,4\}$ |

(b) vertex refinements

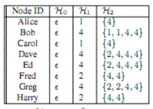
Unlabeled graph, $H_0 = \{\varepsilon\}$
$H^* = H_2$

## Vertex Refinement

DEFINITION 2 (RELATIVE EQUIVALENCE). Two nodes x, y in a graph are equivalent relative to $H_i$, denoted $x \equiv_{Hi} y$, if and only if $H_i(x) = H_i(y)$.

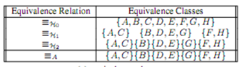


(a) graph



| Node ID | $\mathcal{H}_0$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
|---|---|---|---|
| Alice | $\epsilon$ | 1 | $\{4\}$ |
| Bob | $\epsilon$ | 4 | $\{1, 1, 4, 4\}$ |
| Carol | $\epsilon$ | 1 | $\{4\}$ |
| Dave | $\epsilon$ | 4 | $\{2, 4, 4, 4\}$ |
| Ed | $\epsilon$ | 4 | $\{2, 4, 4, 4\}$ |
| Fred | $\epsilon$ | 2 | $\{4, 4\}$ |
| Greg | $\epsilon$ | 4 | $\{2, 2, 4, 4\}$ |
| Harry | $\epsilon$ | 2 | $\{4, 4\}$ |

(b) vertex refinements



| Equivalence Relation | Equivalence Classes |
|---|---|
| $\equiv_{\mathcal{H}_0}$ | $\{A, B, C, D, E, F, G, H\}$ |
| $\equiv_{\mathcal{H}_1}$ | $\{A, C\}$ $\{B, D, E, G\}$ $\{F, H\}$ |
| $\equiv_{\mathcal{H}_2}$ | $\{A, C\}\{B\}\{D, E\}\{G\}\{F, H\}$ |
| $\equiv_{\mathcal{A}}$ | $\{A, C\}\{B\}\{D, E\}\{G\}\{F, H\}$ |

(c) equivalence classes

Proposition: Let $x, x' \in V$, if $x \equiv_{Hi} x'$, then $cand_{Hi}(x) = cand_{Hi}(x')$

*To an adversary limited to knowledge query $H_i$, nodes equivalent with respect to $H_i$ are indistinguishable.*

Equivalence under H* is very likely to coincide with automorphic equivalence.

## Subgraph Queries

Limitation of vertex refinement:

- always provide complete information about the nodes adjacent to the target (closed-world).

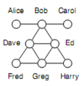
- arbitrarily large subgraphs around a node if that node is highly connected
  E.g., if $H_1(x) = 100$ vs $H_1(y) = 2$

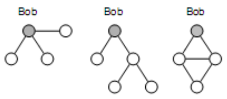
Class of queries about the **existence of a subgraph** around the target node.

Measure their descriptive power by counting ***edge facts*** (# edges in the subgraph).

## Subgraph Queries

Example: Three subgraph queries centered around Bob with 3, 4 and 5 edge facts





❖ may correspond to *different strategies* of knowledge acquisition by the adversary.
including breadth-first exploration, induced subgraphs of radius 1 and 2. etc -- For a given number of edge facts, some queries are more effective at distinguishing individuals.

❖ may be *incomplete* (open-world)

## Hub Fingerprint

**Hub:** a node with high degree and high betweenness centrality (the proportion of shortest paths in the network that include the node)

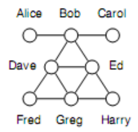
✓ are *often outliers*, making it difficult to protect their identity through anonymization.

**Hub fingerprint** for a target node x: a description of the connections of x to a set of designated hubs in the network.

$\mathbf{F_i(x)}$ hub fingerprint of x to a set of designated hubs, where *i* limit on the maximum distance

## Hub Fingerprint

Hubs: Dave and Ed
$F_1$(Fred) = (1; 0)
$F_2$(Fred) = (1; 2)



- both an open and a closed world.

Example:
*open world*, if the adversary knows $F_1$(Fred) = (1; 0) then nodes in the anonymized graph with $F_1$ fingerprints of (1; 0) is both candidates for Fred.

## Disclosure in Real Networks

- Study ***three networked data sets***, from diverse domains.

For each data set,

- Consider each node in turn as a target.
- *Assume the adversary computes* a vertex refinement, a subgraph, or a hub fingerprint query on that node, and compute the corresponding candidate set for that node.
- *Report the distribution of candidate set sizes* across the population of nodes to characterize how many nodes are protected and how many are identifiable.

## Disclosure in Real Networks

**Hep-Th database:** papers and authors in theoretical high-energy physics, from the arXiv archive, linked if at least two papers together.

**Enron dataset:** from a corpus of email sent to and from managers at Enron Corporation -- Two individuals connected if they corresponded at least 5 times.

**Net-trace dataset:** from an IP-level network trace collected at a major university. monitors traffic at the gateway; a bipartite graph between IP addresses internal to the institution, and external IP addresses.

187 internal addresses from a single campus department and the 4026 external addresses to which at least 20 packets were sent on port 80 (http traffic).

undirected edges, self-loops removed, eliminated a small percentage of disconnected nodes.

## Disclosure in Real Networks: Vertex refinement

- *very low percentage of high-risk nodes* under a reasonable assumption about adversary knowledge.
  Two datasets meet that requirement for H1 (Hep-Th and Net-trace), but no datasets meet that requirement for H2.
- *significant variance* across different datasets in their vulnerability to different adversary knowledge.
- the most significant change in re-identification is *from H1 to H2*
- Re-identification tends to *stabilize after H3* — more information in the form of H4 does not lead to an observable increase in re-identification
- a substantial number of nodes are *not uniquely identified even with H4*

## Disclosure in Real Networks: Subgraph queries

▪ *disclosure is substantially lower* than for vertex refinement queries.

To select candidate sets of size less than 10 requires a subgraph query of size 24 for Hep-Th, size 12 for Enron, and size 32 for Net-trace.

▪ The *smallest subgraph query* resulting in a unique disclosure was size 36 for Hep-Th and 20 for Enron. The smallest candidate set witnessed for Net-trace was size 2, which resulted from a query consisting of 88 edge facts.

▪ *Breadth-first exploration* led to selective queries across all three datasets.
- ▪ asserts lower bounds on the degree of nodes.
- ▪ In Enron, the most selective subgraph queries witnessed;
- ▪ for Hep-Th and Net-trace, the more selective subgraph queries asserted the existence of two nodes with a large set of common neighbors.

## Disclosure in Real Networks: Hub fingerprints

▪ disclosure is low using hub fingerprints.
At distance 1, 54% of the nodes in Enron were not connected to any hub and therefore hub fingerprints provide no information.
This was 90% for Hepth and 28% for Net-trace.

▪ connectivity to hubs was fairly uniform across individuals.
For example, the space of possible fingerprints at distance 1 for Hepth and Net-trace is $2^{10}$= 1024.
Of these, only 23 distinct fingerprints were observed for Hepth and only 46 for Net-trace.

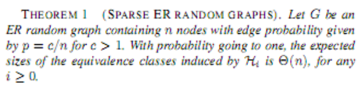
▪ hubs themselves stand out, but have high-degrees, thus connections to a hub are shared by many.

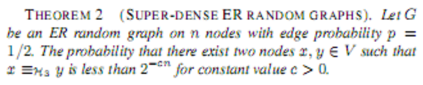
## Re-identification in Random Graphs

Erdos-Renyi (ER) random graphs
*n* nodes by sampling each edge independently with probability *p*
How *p* scales with n
*sparce* p = c/n, *dense* = clogn/n, *super-dense* p = c (c is a constant)

c>1,
include a giant connected component of size Θ(n), and a collection of smaller components (sparse)
completed connected (dense)

## Re-identification in Random Graphs

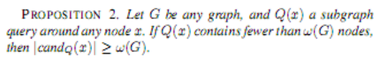

THEOREM 1 (SPARSE ER RANDOM GRAPHS). *Let $G$ be an ER random graph containing $n$ nodes with edge probability given by $p = c/n$ for $c > 1$. With probability going to one, the expected sizes of the equivalence classes induced by $\mathcal{H}_i$ is $\Theta(n)$, for any $i \geq 0$.*


THEOREM 2 (SUPER-DENSE ER RANDOM GRAPHS). *Let $G$ be an ER random graph on $n$ nodes with edge probability $p = 1/2$. The probability that there exist two nodes $x, y \in V$ such that $x \equiv_{\mathcal{H}_3} y$ is less than $2^{-cn}$ for constant value $c > 0$.*

For dense, nodes cannot be identified for $H_1$ for any c>0, but all nodes are re-identifiable for $H_2$ for any c>1

## Re-identification in Random Graphs

ω(G) the number of nodes in the largest clique


PROPOSITION 2. *Let $G$ be any graph, and $Q(x)$ a subgraph query around any node $x$. If $Q(x)$ contains fewer than $\omega(G)$ nodes, then $|cand_Q(x)| \geq \omega(G)$.*

Any subgraph query matching fewer than ω(G) nodes, will match any node in the clique

## *k*-degree Anonymity

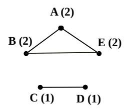
K. Liu and E. Terzi, *Towards Identity Anonymization on Graphs,* SIGMOD 2008

## *k*-degree Anonymity

- Assume that adversary **A** knows that **B** has *327* connections in a social network! (background knowledge)
- If the graph is released by removing the identity of the nodes
  - **A** can find all nodes that have degree *327*
  - If there is only one node with degree *327*, **A** can identify this node as being **B**.

## *k*-degree Anonymity

*k-degree anonymity* A graph ***G(V, E)*** is ***k****-degree anonymous* if every node in ***V*** has the same degree as ***k-1*** other nodes in ***V***.



2-degree anonymous

Prop 1: If G is k1-degree anonymous, then it is also k2-degree anonymous, for every k2 ≤ k1

**[Properties] It prevents the re-identification of individuals by adversaries with *a priori* knowledge of the degree of certain nodes.**

## *k*-degree Anonymity: Anonymization problem

Given a graph *G(V, E)* and an integer *k*,

modify G via a set of edge addition or deletion operations

to construct a new graph *k-degree anonymous* graph G' in which every node u has the same degree with at least *k*-1 other nodes

Why not simply transform G to the complete graph?

## *k*-degree Anonymity: Anonymization problem

Given a graph *G(V, E)* and an integer *k*, modify *G* via a *minimal set* of edge addition or deletion operations to construct a new graph *G'(V', E')* such that

1) *G'* is *k*-degree anonymous;

2) *V' = V*;

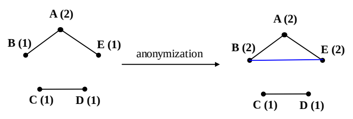
3) (utility) The *symmetric difference* of *G* and *G'* is as small as possible

$$\text{SymDiff}(G', G) = (E' \setminus E) \cup (E \setminus E')$$

Assumption: G: undirected, unlabeled, no self-loops or multiple-edges

Only edge **additions** -- SymDiff(G', G) = |E'| - |E|

## *k*-degree Anonymity

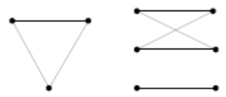


## *k*-degree Anonymity: degree sequence

**[k-anonymous sequence] A sequence of integers *d* is *k-anonymous* if every distinct element value in *d* appears at least *k* times.**

**[100,100, 100, 98, 98,15,15,15]**

A graph G(V, E) is *k*-degree anonymous if its degree sequence is *k*-anonymous

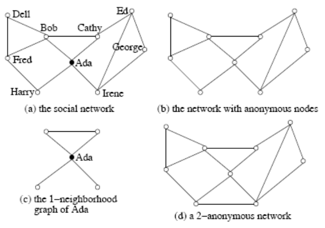


## k-neighborhood Anonymity

B. Zhou and J. Pei, *Preserving Privacy in Social Networks Against Neighborhood Attacks*, ICDE 2008

## Introduction

The neighborhood of u ∈ V(G) is the induced subgraph of the neighbors of u, denoted by $\text{Neighbor}_G(U) = G(N_u)$ where $N_u = \{v \mid (u,v) \in E(G)\}$.



(a) the social network (b) the network with anonymous nodes

(c) the 1–neighborhood graph of Ada (d) a 2–anonymous network

1-neighborhood attacks

For example, the adversary knows that:

- Ada has 2 friends who know each other and another two friends who do not know each other (1-neighborhood graph – induced subgraph)
- Bob can also be identified from its 1-neighborhood graph

## Graph Model

Graph G= (V, E, L, F),

V a set of nodes,
$E \subseteq V \times V$ set of edges,
**L** is a set of labels, and
F a labeling function **F: V →L** assigns each node a label.

*edges do not carry labels*

❖ **Items in L form a hierarchy.**

E.g., if occupations as labels, L contains not only the specific occupations [such as dentist, general physician, optometrist, high school teacher, primary school teacher, etc] but also general categories [such as, medical doctor, teacher, and professional}.

* ∈ L -> most general category generalizing all labels.

## Graph Model

Given a graph = ($V_H$, $E_H$, L, F ) and a social network G = (V, E, L, L), an instance of H in G is a tuple (H', f) where H' = ($V_{H'}$ ,$E_{H'}$ ,L, F) is a subgraph in G and f: $V_H \rightarrow V_{H'}$, is a bijection function such that

(1) for any $u \in V_H$, $F(f(u)) \leq F(u)$, /* the corresponding labels in H' are more general */ and

(2) $(u, v) \in E_H$ if and only if $(f(u), f(v)) \in E_{H'}$.

## *k*-neighborhood anonymity

G -> G' through a bijection (isomorphism) A

[***k*-neighborhood anonymity**] A vertex u ∈ V (G), u is k anonymous in G' if there are at least (k − 1) other vertices $u_1, \dots, u_{k-1} \in V(G)$ such that $\text{Neighbor}_{G'}(A(u))$, $\text{Neighbor}_{G'}(A(u_1))$, . . ., $\text{Neighbor}_{G'}(A(u_{k-1}))$ are isomorphic.

**G′ is k-anonymous if every vertex in G′ is k-anonymous**.

Property 1 (k-anonymity) Let G be a social network and G′ an anonymization of G. If G′ is k-anonymous, then with the neighborhood background knowledge, any vertex in G cannot be re-identified in G′ with confidence larger than 1/k .

## *k*-neighborhood anonymity

Given a social network G, the k-anonymity problem is to compute an anonymization G′ such that

(1) G′ is *k*-anonymous;
(2) each node in G is anonymized to a node in G′ and G′ does not contain any fake nodes; *(no node addition)*
(3) every edge in G is retained in G′; *(no edge deletion)*; and
(4) the number of edges to be added is minimized.

## *k*-neighborhood anonymity

Utility

Aggregate queries:
compute the aggregate on some paths or subsgraphs satisfying some given conditions
E.g., Average distance from a medical doctor to a teacher

Heuristically, when the number of edges added is as small as possible, G′ can be used to answer aggregate network queries accurately

## *k*-neighborhood anonymization method

Two steps:

**STEP 1**
Extract the neighborhoods of all nodes in the network
Encode the neighborhood of each node (to facilitate the comparison between neigborhoods)

**STEP 2**
Greedily, organize nodes into groups and anonymize the neighborhoods of nodes in the same group

## *k*-neighborhood anonymization method

### Step 1: Neighborhood Extraction and Coding

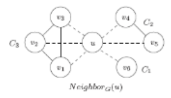
General problem of determining whether two graphs are isomorphic is NP-complete

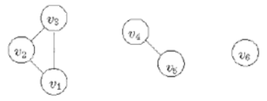
Goal: **Find a coding technique** for neighborhood subgraphs so that whether two neighborhoods are isomorphic can be determined by the corresponding encodings

## *k*-neighborhood anonymization method

### Step 1: Neighborhood Extraction and Coding

A subgraph C of G is a **neighborhood component** of u ∈ V (G), if C is a maximal connected subgraph in $\text{Neighbor}_G(u)$.





Neighborhood components of u

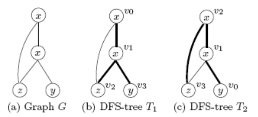
1. Divide the neighborhood of v into neighborhood components
2. To code the whole neighborhood, first code each component.

## *k*-neighborhood anonymization method

### Step 1: Neighborhood Extraction and Coding

**Encode** the edges and vertices in a graph based on its **depth-first search tree** (DFS-tree).

All the vertices in G can be encoded in the pre-order of T .



(a) Graph $G$ (b) DFS-tree $T_1$ (c) DFS-tree $T_2$

*Thick edges are those in the DFS-trees (forward edges),*
*Thin edges are those not in the DFS-trees (backward edges)*

*vertices encoded u0 to u3 according to the pre-order of the corresponding DFS-trees.*

The DFS-tree is generally not unique for a graph -> minimum DFS code (based on an ordering of edges) – select the lexically minimum DFS code – DFS(G)

- Two graphs G and G’ are isomorphic, if and only if, DFS(G) = DFS(G’)

## *k*-neighborhood anonymization method

### Step 2: Social Network Anonymization

Each **node must be grouped with a least (k-1) other nodes** such their anonymized neighborhoods are isomorphic

Very few nodes have **high degrees**, **process them first** to keep information loss for them low

Many vertices of low degree, easier to anonymize

## *k*-neighborhood anonymization method

Step 2: Anonymizing 2 neighborhoods

First, find **all perfect matches** of neighborhood components (perfectly match=same minimum DFS code)

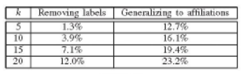
**For unmatched**, try to pair "similar" components and anonymize them

How**: greedily**, starting with two nodes with the same degree and label in the two components to be matched (if ties, start from the one with the highest degree - If there are no such nodes: choose the one with minimum cost)

Then a BFS to match nodes one by one, if we need to add a node, consider nodes in V(G)

## *k*-neighborhood anonymization method

Co-authorship data from KDD Cup 2003 (from arXiv, high-energy physics)
Edge – co-authored at least one paper in the data set.
57,448 vertices
120,640 edges
average number of vertex degrees about 4.



| $k$ | Removing labels | Generalizing to affiliations |
|---|---|---|
| 5 | 1.3% | 12.7% |
| 10 | 3.9% | 16.1% |
| 15 | 7.1% | 19.4% |
| 20 | 12.0% | 23.2% |

TABLE I
THE PERCENTAGES OF VERTICES VIOLATING $k$-ANONYMITY IN THE CO-AUTHORSHIP DATA.

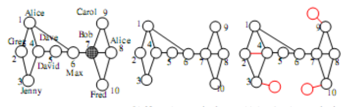
## k-Automorphism

L. Zhu, L. Chen and M. Tamer Ozsu, *k-automorphism: a general framework for privacy preserving network publication,* PVLDB 2009

## K-Automorphism

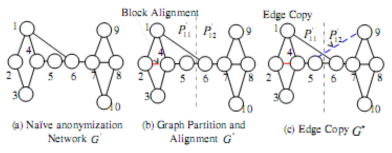
Considers any subgraph query - any structural attack

At least k symmetric vertices no structural differences



(a) Original Network $G$ (b) Naive Anonymized Network $G'$ (c) Another Anonymized Network $\hat{G}$

## K-Match (KM) Algorithm


Block Alignment
Edge Copy
(a) Naïve anonymization Network G'
(b) Graph Partition and Alignment G'
(c) Edge Copy G*

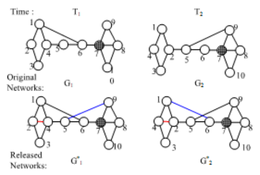
Step 1: Partition the original network into blocks

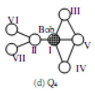
Step 2: Align the blocks to attain isomorphic blocks (add edge (2,4))

Step 3: Apply the "edge-copy" technique to handle matches that cross the two blocks

## Dynamic Releases

Example:
Individually satisfy 2-automorphism
**Assume that an adversary knows that sub-graph Q4 exists around target Bob at both time T1 and T2.**
At time T1, an adversary knows that there are two candidates vertices (2, 7)
Similarly, at time T2, there are still two candidates (4, 7)
**Since Bob exists at both T1 and T2, vertex 7 corresponds to Bob**





## Evaluation

Prefuse (129 nodes, 161 edges)
Co-author graph (7995 authors in database and theory, 10055 edges)

Synthetic
Erdos Renyi, 1000 nodes
Scale free, $2 < \gamma < 3$

All $k$ = 10 degree anonymous, but no sub-graph anonymous

## *k*-degree Anonymity: degree sequence anonymization

✓ Increase/decrease of degrees correspond to additions/deletions of edges

[degree-sequence anonymization] Given degree sequence ***d***, and integer ***k***, construct a ***k***-anonymous sequence ***d'*** such that **||*d'*-*d*||** (i.e., $L_1(d' - d)$) **is minimized**

$|E'| - |E| = ½ L_1(d' – d)$

Relax graph anonymization: E' not a supergraph of E

## *k*-degree Anonymity: algorithm

Two steps

**Input:** Graph ***G*** with degree sequence ***d***, integer ***k***
**Output:** ***k***-degree anonymous graph ***G'***

[STEP 1: **Degree Sequence Anonymization**]:
Construct an (optimal) k-anonymous degree sequence ***d'*** from the original degree sequence ***d***

[STEP 2: **Graph Construction**]:
[**Construct**]: Given degree sequence ***d'***, construct a new graph ***$G^0(V, E^0)$*** such that the degree sequence of ***$G^0$*** is ***d'***
[***Transform***]: *Transform* ***$G^0(V, E^0)$*** *to* ***G'(V, E')*** *so that* ***SymDiff***(*G',G*) *is minimized.*

## *k*-degree Anonymity Algorithm: degree sequence

Degree sequence anonymization

$\boldsymbol{d}(1) \geq \boldsymbol{d}(2) \geq \ldots \geq \boldsymbol{d}(i) \geq \ldots \geq \boldsymbol{d}(n)$ : original degree sequence.
$\boldsymbol{d'}(1) \geq \boldsymbol{d'}(2) \geq \ldots \geq \boldsymbol{d'}(i) \geq \ldots \geq \boldsymbol{d'}(n)$ : *k*-anonymized degree sequence.

If we only add edges, d'(i) ≥ d(i)

Observation 1, if d'(i) = d'(j) with i < j, then d'(i) = d'(i+1) = .. . d'(j-1) = d(j)

***I(i, j)***: anonymization cost when all nodes i, i+1, …, j are put in the same anonymized group

$$I(i,j) = \sum_{\ell=i}^{j} \left(d(i) - d^*\right)$$

## *k*-degree Anonymity Algorithm: degree sequence

Example
6 6 5 5 4 4 4 4 4 (k=3)
6 6 6 ↑

Merge with previous group: 6 6 6 6, or
Start new group: 6 6 6 5

**Greedy**
Form a group with the first k, for the k+1, consider
$C_{merge}$ = (d(1) – d(k+1)) + I(d(k+2, 2k+1))
$C_{new}$= I(d(k+1, 2k)

## *k*-degree Anonymity Algorithm: degree sequence

***DA(1, j)***: the optimal degree anonymization of subsequence d(1, j)
***DA(1, n)***: the optimal degree-sequence anonymization cost

***I(i, j)***: anonymization cost when all nodes i, i+1, …, j are put in the same anonymized group

For i < 2k (impossible to construct 2 different groups of size k)

$$DA(1,i) = I(1,i)$$

For i ≥ 2k

$$DA(1,i) = \min\ \{ \min_{k \le t \le i-k} \{DA(1,t) + I(t+1,i)\}, I(1,i)\}$$

## *k*-degree Anonymity Algorithm: degree sequence

$$DA(1,i) = I(1,i)$$
$$DA(1,i) = \min\ \{\min_{k \le t \le i-k} \{DA(1,t) + I(t+1,i)\}, I(1,i)\}$$

Can be improved, no anonymous groups should be of size larger than 2k-1

We do not have to consider all the combinations of I(i, j) pairs, but for every i, only j's such that k ≤ j – i + 1 ≤ 2k-1

O($n^2$) -> (Onk)

$$DA(1,i) = \min_{\max\{k,i-2k+1\} \le t \le i-k} \{DA(1,t) + I(t+1,i)\}$$

Additional bookkeeping -> Dynamic Programming with **O(nk)**

## *k*-degree Anonymity Algorithm: graph construction

- A degree sequence ***d*** is **realizable** if there exists a simple undirected graph with nodes having degree sequence ***d.***

*Are all degree sequences realizable?*

- Not all vectors of integers are realizable degree sequences
  - d = {4, 2, 2, 2, 1} ?
- How can we decide?

## *k*-degree Anonymity Algorithm: graph construction

[**Erdös and Gallai**] A degree sequence ***d*** with $\boldsymbol{d}(1) \ge \boldsymbol{d}(2) \ge \ldots \ge \boldsymbol{d}(i) \ge \ldots \ge \boldsymbol{d}(n)$ and $\Sigma \boldsymbol{d}(i)$ even, is realizable if and only if

$$\sum_{i=1}^{l} \boldsymbol{d}(i) \le l(l-1) + \sum_{i=l+1}^{n} \min\{l, \boldsymbol{d}(i)\}, \text{ for every } 1 \le l \le n-1.$$

For each subset of the *l* highest degree nodes, the degrees of these nodes can be "absorbed" within the nodes and the outside degrees

## *k*-degree Anonymity Algorithm: graph construction

**Input:** Degree sequence ***d'***
**Output:** Graph ***G⁰(V, E⁰)*** with degree sequence ***d'*** or ***NO!***

✓General algorithm, create a graph with degree sequence d'

In each iteration,

- pick an arbitrary node u
- add edges from u to d(u) nodes of highest residual degree, where d(u) is the residual degree of u

Is an oracle

We also need G' such that E' ⊇ E

Thus, we **start with the edges** of E already in

Is not an oracle

## *k*-degree Anonymity Algorithm: graph construction

**Input:** Degree sequence ***d'***
**Output:** Graph ***$G^0$(V, $E^0$)*** with degree sequence ***d'*** or ***NO!***

→If the degree sequence ***d'*** is NOT realizable?

•Convert it into a realizable and ***k***-anonymous degree sequence

Slightly increase some of the entries in d via the **addition of uniform noise**

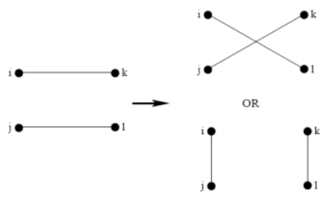
In the implementation, examine the nodes in increasing order of their degrees, and slightly increase the degrees of a single node at each iteration (in real graph, few high degree nodes – rarely any two of these exactly the same degree)

## *k*-degree Anonymity Algorithm: graph transformation

- GreedySwap transforms ***$G^0$ = (V, $E^0$)*** into ***G'(V, E')*** with the same degree sequence ***d'***, and min symmetric difference ***SymDiff(G',G)*** .
- GreedySwap is a greedy heuristic with several iterations.
- At each step, GreedySwap swaps a pair of edges to make the graph more similar to the original graph ***G***, while leaving the nodes' degrees intact.

## *k*-degree Anonymity Algorithm: graph construction

Valid swappable pairs of edges

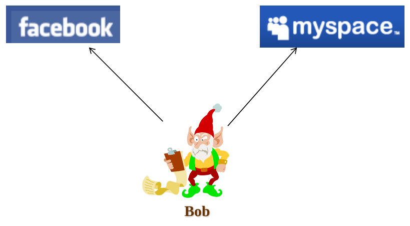


A swap is ***valid*** if the resulting graph is simple

## De-anonymization

A. Narayanan, V. Shmatikov, *De-anonymizing Social Networks*. International Symposium on Security and Privacy, 2009

## Introduction: new type of passive attack



Bob has accounts in both facebook and myspace

Background knowledge: another social network with partially overlapping membership

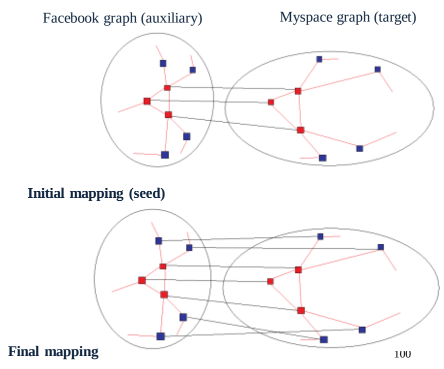
98

## Introduction: goal

Given the **auxiliary information**, (nodes, edges) about one social network (auxiliary network) and **a small number of members of target network**, the attacker wants to learn sensitive information about other members of target network.

99

## Introduction: goal



100

## Algorithm

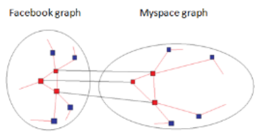
2 stages

1. Seed identification
2. Propagation

101

## Algorithm: Seed identification

- Identify a small number of seed nodes in both target graph and auxiliary graph, and map them to each other
- assumes a clique of *k* nodes in both graphs
- suffices to know the degree of each node and the number of common neighbors for each pair



## Algorithm: Seed identification

Input:

1. the target graph
2. *k* seed nodes in the auxiliary graph
3. *k* nodes' information, such as, degree values and pairs of common-neighbor counts
4. Error parameter ε

Method

Search the target graph for a unique *k*-clique with matching (within a factor of 1 ± ε) nodes degrees and common-neighbor counts

Output

If found, it maps the nodes in the clique to the corresponding nodes in the auxiliary graph

## Algorithm: Seed identification

- does not guarantee a unique k-clique in target graph.
- the running time is exponential in k.
  - Once we find a matched clique in target graph, stop searching

## Algorithm: Propagation

Input

1. G1(V1, E1)
2. G2(V2, E2)
3. A partial "seed" mapping between the two.

No distinction which is the auxiliary graph or the target graph

Output

mapping μ, focus on deterministic

## Algorithm: Propagation

Finds new mappings using the topological structure of the network and the feedback from previously constructed mappings.

At each iteration,

start with the list of mapped pairs between V1 and V2.

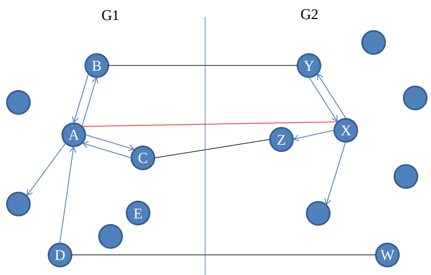
**Pick an arbitrary unmapped node** u in V1 and computes a **score** for **each unmapped node v in V2**, equal to **the number of neighbors** of u that have been mapped to neighbors of v.

If the strength of the match is above a **threshold**,

the mapping between u and v is added to the list, and the next iteration starts.

106

## Algorithm: Propagation



Mapping list={(B,Y), (C, Z),(D, W)}
Score(A, X) = 2, Score(E, Z) = 0

107

## Algorithm: Propagation

Lots of scores. Which mapping should we keep?

Additional details

1. **Edge directionality**. Score = incoming edge score + outgoing edge score.

2. **Node degrees**. (to compensate for the bias towards high degree nodes)
Score(u, $v_i$)=score(u, $v_i$)/√degree of $v_i$

3. **Eccentricity.** It measures how much an item in a set X "stands out" from the rest.

$$\begin{matrix}\max(X) \quad \max_2(X)\\ \sigma(X)\end{matrix}$$

where max and max2 denote the highest and second highest values, respectively, and σ denotes the standard deviation.

If > θ, keep the mapping; otherwise, it is rejected, where θ is a parameter.

108

## Algorithm: Propagation

4. Does not matter whether G1 is the target and G2 is the auxiliary, each time u is mapped to v, **switch the input graph**, if v gets mapped back to u, the mapping is retained; otherwise, it is rejected.

5. **Revisiting nodes**. As the number of mapped nodes increases, we need to revisit already mapped nodes.

6. Do the iteration until convergence.

109

## Algorithm: Propagation

- Complexity
  O((|E1|+|E2|)*d1*d2) where d1 is a bound on the degree of nodes in G1. D2 is a bound on the degree of nodes in G2.
- Without revisiting nodes and reverse matches
  O(|E1|*d2)
- Without reverse matches
  O(|E1|*d1*d2)

## Propagation

- Auxiliary graph: Flickr. Target graph: Twitter

ground truth (matches based on username, name, location)

27,000 mappings

- Seed mapping consists of 150 pairs of nodes with the constraints that the degree of each node in auxiliary graph is at least 80.

## Result of accuracy

- 30.8% of mappings(27,000) were re-identified correctly, 12.1% were identified incorrectly, and 57% were not identified.
- 41% of the incorrectly identified mappings were mapped to nodes which are at a distance 1 from the true mapping.
- 55% of the incorrectly identified mappings were mapped to the nodes where the same location was reported.
- The above two categories overlap; only 27% of incorrect mappings are completely erroneous.