De-anonymizing Social Networks

Arvind Narayanan and Vitaly Shmatikov

The University of Texas at Austin

--presented by: Li Shuhe

content

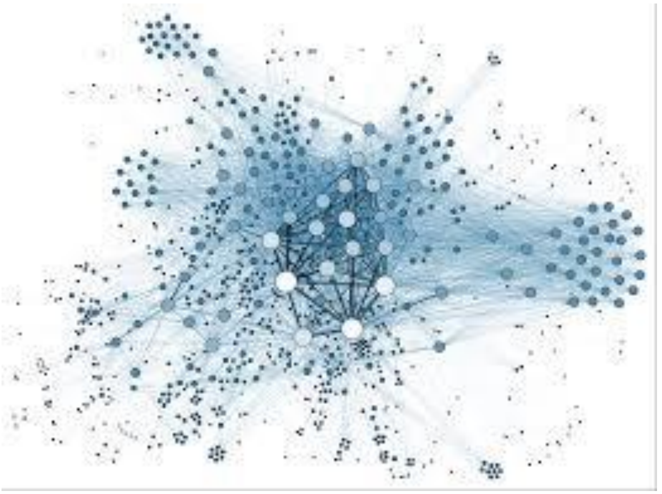
- Background
- Related work
- De-anonymization Algorithm
- Experimental Results
- Conclusion

Background

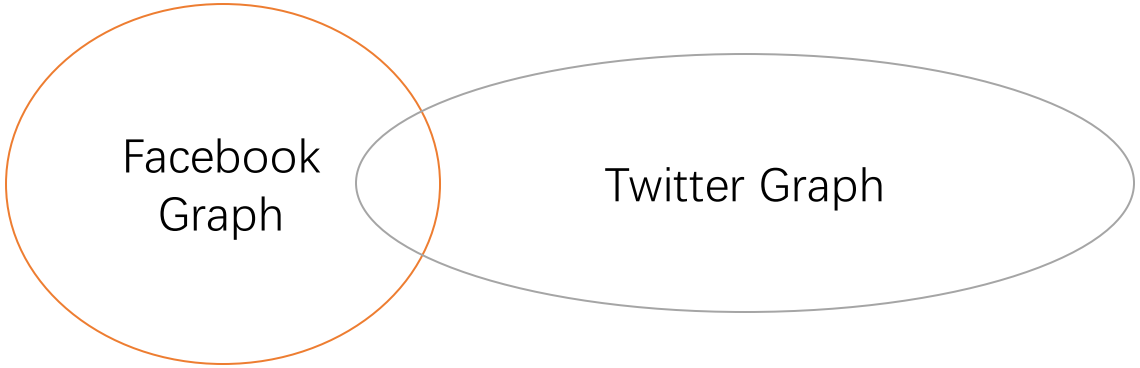
- Social networks

  Graph consists of nodes(individuals), edges(relations) and information associated with each node and edge.

  Facebook, Twitter, Myspace



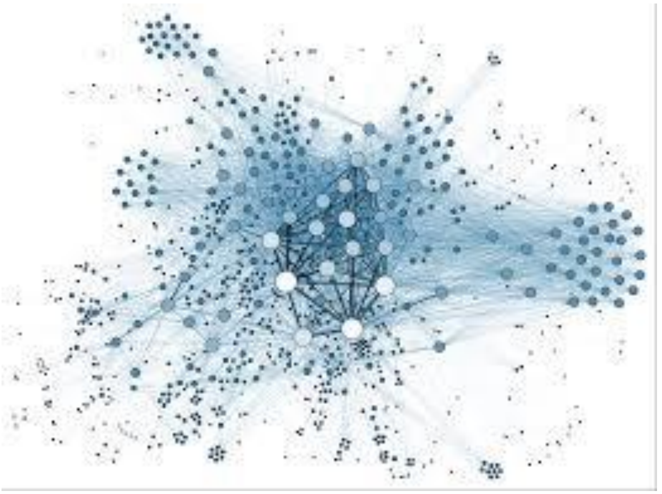
- Overlap



Background

- Privacy and anonymity

  A privacy breach occurs when someone accesses information without permission.



  Any privacy breach much include learning some identifying information about the endpoints.

  De-anonymity is guaranteed to violate any reasonable definition of privacy.

Related work

- Active attacks

  Create a small number of new user accounts with edges to targeted users and create a pattern of links among the new accounts.

  For example, create 7-node sub graphs containing a Hamiltonian path.

  The attacks are restricted to online social networks.

  Attacks are easy to detect by operators.

  Many online social networks require a link to be mutual before the information is made available in any form.

Related work

- Passive attacks

  A small coalition of users discover their location in the anonymized graph by utilizing the knowledge of the network structure around them.

  Only works on a small scale.

De-anonymization algorithm

- Model

  Input:

  Auxiliary graph $G_{aux}=\{V_{aux},E_{aux}\}$

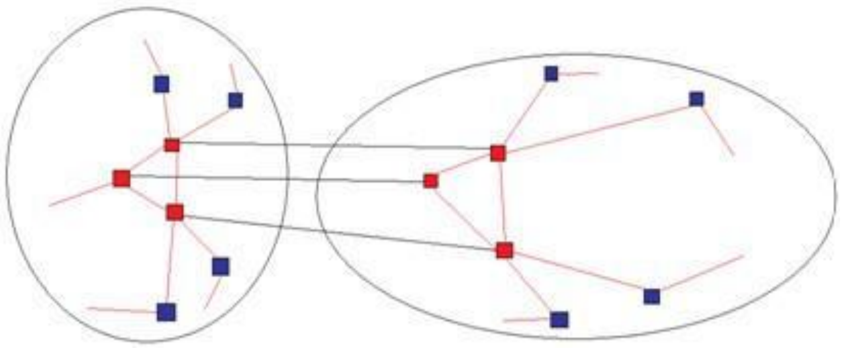
  Node attribute Aux_X

  Edge attribute Aux_Y

  Anonymized graph $G_{sen}=\{V_{sen},E_{sen}\}$

  Goal:

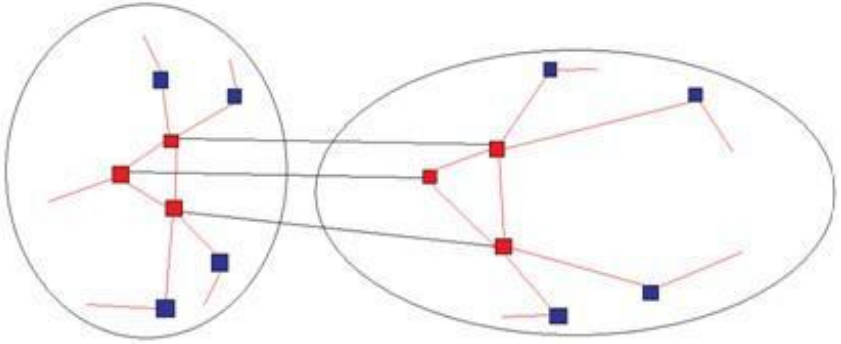
  Generate $\mu: V_{san} \times V_{aux} \rightarrow [0, 1]$ where $\mu(v_{aux}, v_{san})$ is the probability that v_{aux} is mapped to v_{san}.



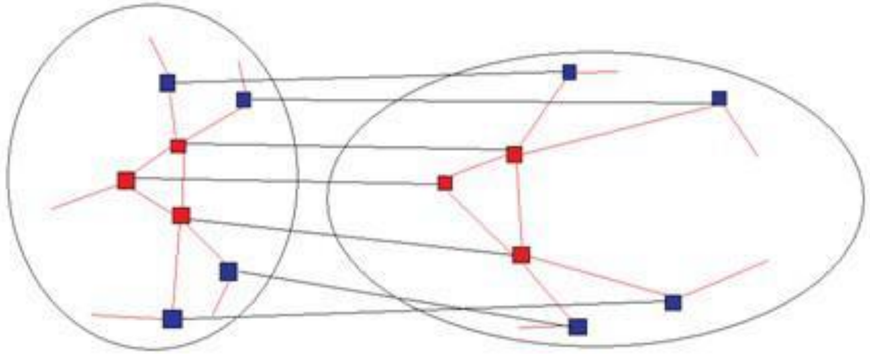
De-anonymization algorithm

- Algorithm

1. Seed identification



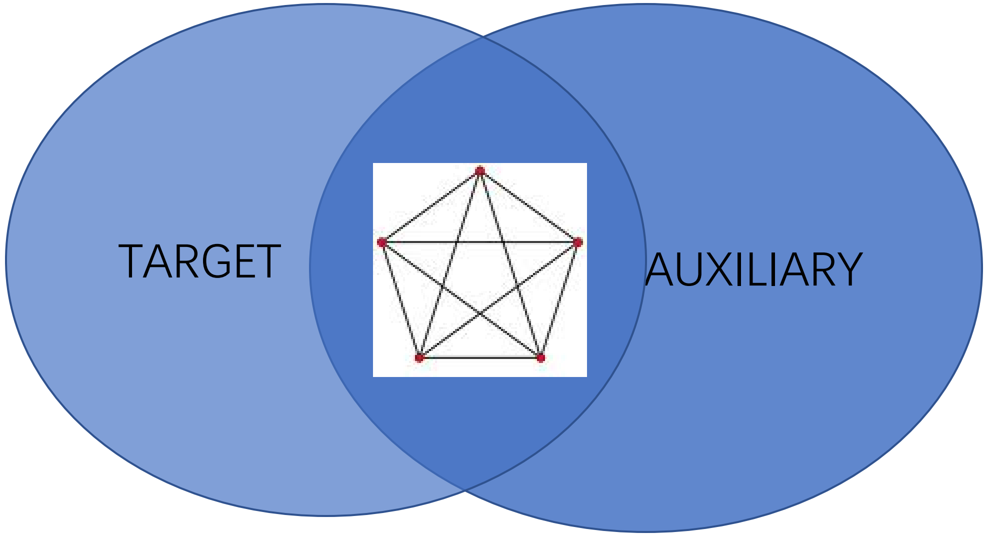
2. Propagation



De-anonymization algorithm

- Algorithm

1. Seed identification



Input:
1. A clique of k nodes known to be common to both graphs.
2. Degree of each of these nodes (AUXILIARY)
3. Number of common neighbors for each pair of nodes (AUXILIARY)

Algorithm:
Search for a unique k-clique (on TARGET) that has:
1. matching degrees (with some error factor)
2. common neighbor counts

Outputs μ_s, a partial mapping.

De-anonymization algorithm

- Algorithm

1. Seed identification

2. Propagation

Input:
Two graph and a partial seed mapping

Algorithm:
Iterate until no update:
Picks a unmapped node v in V_1 and computes a score for each unmapped node v in V_2.
Output:
deterministic 1-1 mapping

De-anonymization algorithm

- Algorithm

1. Seed identification

2. Propagation

Eccentricity measures how much an item in a set X "stands out" from the rest. Defined by: $\frac{\max(X)-\max_2(X)}{\sigma(X)}$

Edge Directionality – mapping scores for nodes *u* and *v* are computed separately for incoming and outgoing edges (and then summed).

Node Degrees – the above works in favor of high-degree nodes => divide by square root of degree.

Revisiting Nodes – as the algorithm progresses, #mapped nodes increases & errors decrease.

Reverse Match – every match is matched in both directions

De-anonymization algo

- Algorithm

1. Seed identification

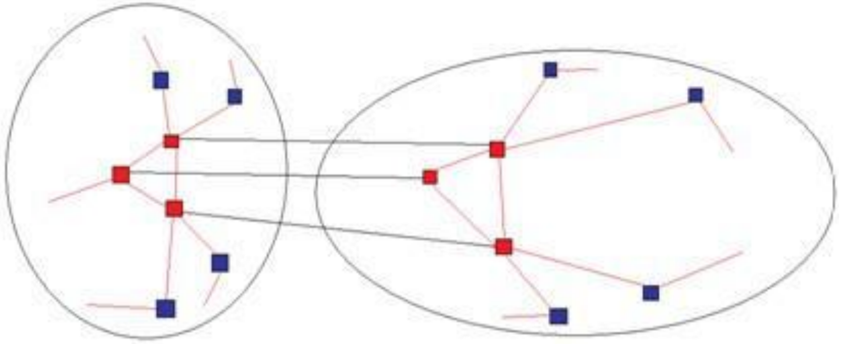
2. Propagation

```
function propagationStep(lgraph, rgraph, mapping)

  for lnode in lgraph.nodes:
    scores[lnode] = matchScores(lgraph, rgraph, mapping, lnode)
    if eccentricity(scores[lnode]) < theta: continue
    rnode = (pick node from rgraph.nodes where
          scores[lnode][node] = max(scores[lnode]))

    scores[rnode] = matchScores(rgraph, lgraph, invert(mapping), rnode)
    if eccentricity(scores[rnode]) < theta: continue
    reverse_match = (pick node from lgraph.nodes where
          scores[rnode][node] = max(scores[rnode]))
    if reverse_match != lnode:
      continue

    mapping[lnode] = rnode

function matchScores(lgraph, rgraph, mapping, lnode)

  initialize scores = [0 for rnode in rgraph.nodes]

  for (lnbr, lnode) in lgraph.edges:
    if lnbr not in mapping: continue
    rnbr = mapping[lnbr]
    for (rnbr, rnode) in rgraph.edges:
      if rnode in mapping.image: continue
      scores[rnode] += 1 / rnode.in_degree ^ 0.5

  for (lnode, lnbr) in lgraph.edges:
    if lnbr not in mapping: continue
    rnbr = mapping[lnbr]
    for (rnode, rnbr) in rgraph.edges:
      if rnode in mapping.image: continue
      scores[rnode] += 1 / rnode.out_degree ^ 0.5

  return scores

function eccentricity(items)

  return (max(items) - max2(items)) / std_dev(items)

until convergence do:
  propagationStep(lgraph, rgraph, seed_mapping)
```

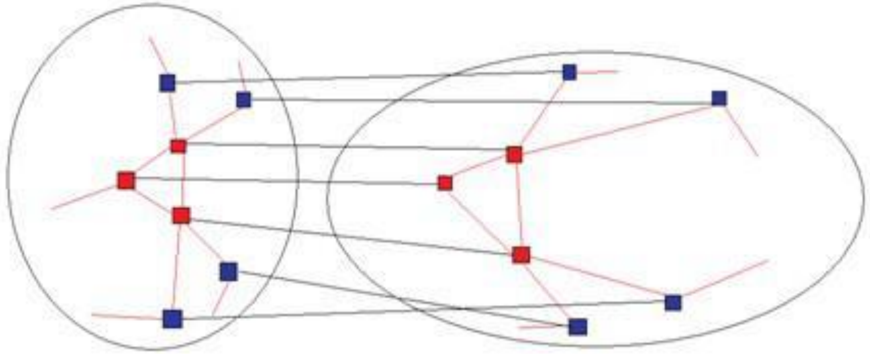
De-anonymization algorithm

- Algorithm

1. Seed identification



2. Propagation



Experimental Results

Type	Network	Relation.	Nodes	Edges	Av. Deg	Crawled
Target	Twitter	Follow	224K	8.5M	27.7	2007
Auxiliary	Flickr	Contact	3.3M	53M	32.2	2007/8

- Ground truth:
  1. mapping based on exact matches in the username or name field
  2. Calculate score based on a variety of heuristics on all username, name, location
  3. Reject the match if the score is too low
  4. Result in 27000 mappings
- Experiment setting:
  1. Seed mapping: 150 pairs of nodes selected randomly

Experimental Results

$$\frac{\sum_{v \in V_{\text{mapped}}} \mathsf{PR}[\mu(v) = \mu_G(v)]\nu(v)}{\sum_{v \in V_{\text{mapped}}} \nu(v)}$$

- 30.8% of the mappings were re-identified correctly.
- 57% were not identified
- 12.1% were identified incorrectly
  - 41% of them were mapped to distance 1 nodes from the true mapping
  - 55% of them were mapped to nodes with the same geographic location
  - 27% are completely erroneous.

Conclusion

- Social networks grow
  - Overlap between social networks increases
  - Auxiliary information is much richer

- Anonymity is not sufficient for privacy when dealing with social networks.

Conclusion

Q & A