Towards Graph Watermarks

Xiaohan Zhao, Qingyun Liu, Haitao Zheng, Ben Y. Zhao

SAND Lab, UCSB
qingyun_liu@cs.ucsb.edu
Sensitive Datasets Captured in Graphs

• Graphs are everywhere
  – e.g. Internet networks, Social networks, Biological networks

• Many of today’s sensitive datasets are captured in large graphs
  – e.g. maps of autonomous system, friendships in social networks, interaction of proteins in personal health care
Desires to Securely Share Graphs

• Data owners: often want to share data with selected parties, without data leakage into public domain
  – ISP vs networking equipment vendor
  – Facebook vs trusted academic collaborators

• Research community: need real graphs for progress in many areas
  – Understand underlying structure and process
  – Validate models and theories
Current Solutions: Far From Ideal

• Option 1: Build strong access control mechanisms
  – Have limited control once the data is shared
  – Attacks on human elements
    o e.g., phishing, baiting

• Option 2: modify data to reduce the impact of potential leakages
  – Usually significant modification, make data noisy
    o Subsampling, summarization, synthetic graphs
  – Significantly reduces “utility” of graph dataset

A new alternative: graph watermarks
Watermark: Data Signature

• Watermark: *signature* in data as ownership identifier
  – Data owner embeds a signature in the data
  – If the data is leaked, announce ownership by the signature

• Widely used in digital files to limit piracy

• **Graph watermark**: signature in a graph
  – Watermarks applied to graphs
  – Serve as a deterrent against graph leakage
Our Goals

• Design an **effective** graph watermark system
  – **Low distortion**
    o Small impact on graph structure
    o Difficult to detect
  – **Uniqueness**
    o Not occur naturally nor easily faked
    o Existence securely associated with an authorized party
  – **High robustness**
    o Watermarks remain after attacks
  – **Efficient** to embed and extract watermarks
Scenario: Share Graph With Multiple Users

- Each user uniquely associated with a watermark
- Once find a leaked version, identify the source by watermark

Acts as “deterrent” against data leakage
Outline

• Motivation

• Graph Watermark System
  – Watermark Embedding
  – Watermark Extraction

• Key Properties

• Experimental Evaluation Summary

• Conclusion
Graph Watermark System Overview

• **Embedding:**
  add watermark into the original graph
  - Generate watermark with $G$’s secret key $K^G$ + user $i$’s signature $S^i$
  - Require joint efforts from data owner and user $i$

• **Extraction:**
  search in a leaked graph for any watermark
Graph Watermark System Overview

- **Embedding:**
  - Add watermark into the original graph
    - Generate watermark with $G$’s secret key $K^G$ + user $i$’s signature $S_i$

- **Extraction:**
  - Search in a leaked graph for any watermark

**Challenges:**
Rely on only *graph structure*, not meta data $\rightarrow$
Subgraph isomorphism problem (NP-complete)

**Our Solution:** Efficient Pruning Algorithm
Watermarking Embedding

**Step 1:** verify user $i$’s signature $S^i$

- A random generator seed $\Omega^i = K^G + S^i$

```
Data Owner

Pub^i

Pub^i(S^i) == T^i?

User i

Prv^i(T^i)

S^i = Prv^i(T^i)

Public-private key pair <Pub^i, Prv^i>
```
Watermarking Embedding

**Step 1:** verify user $i$’s signature $S^i$
- A random generator seed $\Omega^i = K^G + S^i$

**Step 2:** generate the watermark
- A random graph of $k$ ordered nodes, seeded by $\Omega^i$

**Step 3:** select embedding location
- A subgraph of $k$ ordered nodes in $G$, seeded by $\Omega^i$

**Step 4:** embed the watermark (XOR)
Watermarking Embedding

Step 1: verify user $i$’s signature $S^i$
  – A random generator seed $\Omega^i = K^G + S^i$

Step 2: generate the watermark
  – A random graph of $k$ ordered nodes, seeded by $\Omega^i$

Step 3: select embedding location
  – A subgraph of $k$ ordered nodes in $G$, seeded by $\Omega^i$

Step 4: embed the watermark (XOR)
Node Naming Algorithm

• Generate “label” for nodes
  – Regenerate “meta data” from only graph structure
  – Label = an array of sorted neighboring degrees

• Efficient in practice
  – Real graphs often have high node heterogeneity
  – Small # of nodes share the same label
Watermark Extraction

Data owner: a leaked graph $G^{\text{leak}}$, original graph $G$, random generator seed $\Omega^i$ for each user (i=1,2,…)

- **Step 1**: regenerate embedded watermark
  - Repeat watermark embedding for each user

- **Step 2**: search if any embedded watermark in $G^{\text{leak}}$
  - Pruning algorithm

\[ \Omega^1 \]
\[ \Omega^2 \]
\[ \Omega^3 \]
\[ \ldots \]
Pruning Algorithm

• Exhaustive search
  – Efficient by restricting to small # of nodes

• For each embedded watermark $E^i$ (i=1,2,…)
  – Find candidates in $G^{\text{leak}}$ by matching node label
  – Enumerate combinations and check graph structure

• Stop until matching or exhausting all combinations
Watermark Uniqueness

• Watermark uniqueness: an embedded watermark not isomorphic to
  – Any subgraph of the original graph (naturally occurring)
  – Any other embedded watermarks (watermark collision)

• Proof sketch
  – Given original graph $G$, users $x \neq y$, the embedded watermark of user $x$ $E_x$, the watermarked graph of user $y$ $G_y'$
  – Step 1: with high probability, $E_x$ is not isomorphic to a given subgraph in $G$ nor $G_y'$
  – Step 2: with high probability, $E_x$ is not isomorphic to any subgraph in $G$ nor $G_y'$

Details
Watermark Applicability

• Graphs suitable for watermarking
  – Can “well hide” embedded watermark
  – Judging criteria
    o Node degree
    o Subgraph density

• Test on 48 real network graphs
  – Represent 10 types of networks
    o e.g. OSNs (Facebook, Youtube …), communication networks
  – Sizes: thousands to millions nodes/edges

• Most (35) graphs are suitable
  – Unsuitable: only 3 types
    o e.g. Road networks
  – Sparse graphs
Experimental Evaluation Summary

• Low distortion
  – Node/edge modification < 0.04%

• High efficiency
  – e.g. graph with 2M nodes, 16M edges
    o Embedding: < 2 mins
    o Extraction: < 4 mins when parallelized across 10 machines

• Robust to attacks
  – Single attack model: have one watermarked graph
  – Collusion attack model: have multiple watermarked graphs
  – Multiple defense techniques (details in paper)
Conclusion

• Graph watermarks useful in many applications
  – e.g. tracking data leaks, data authentication

• Our work: a first step
  – Identify the problem
  – Initial implementation: an efficient system with unique, robust watermarked graphs in low distortion

• Future work: improve robustness against many other attacks
Thank you
Any questions?
Watermark Embedding

- **Select embedding location**
  - Sort node labels of $G$
    - e.g. use secure one-way hash like SHA-1
  - Use $\Omega$ to randomly pick labels as selected nodes
    - If multiple nodes have the same labels, sort them in any deterministic order and use $\Omega$ to pick one

- **Embed watermark**
  - Match both subgraphs by node order
  - Apply XOR on each pair of nodes
Fast Pruning Algorithm

• Complexity is bounded: \( O\left(\sum_{m=2}^{k} \left( \prod_{i=1}^{m} |C_i| \right) \cdot \frac{m}{2} \right) \)
  – \(|C_i|\): # of candidates for \(i\)-th node in embedded watermark

• In practice, far from the worst case scenarios
  – Real graphs have high node heterogeneity \(\rightarrow\) small \(|C_i|\)

• Repeated empirical experiments show efficiency

<table>
<thead>
<tr>
<th>Graph</th>
<th># of Nodes</th>
<th># of Edges</th>
<th>Embedding</th>
<th>Extraction*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook (LA)</td>
<td>603,834</td>
<td>7,676,486</td>
<td>&lt; 1min</td>
<td>&lt; 1min</td>
</tr>
<tr>
<td>Flicker</td>
<td>1,715,255</td>
<td>15,555,041</td>
<td>&lt; 2min</td>
<td>&lt; 4min</td>
</tr>
</tbody>
</table>

*: Extraction parallelized across 10 machines, each with 192 GB RAM
Watermark Uniqueness

- Intuition: embedded watermark is a special graph, when large enough difficult to find isomorphism in G
  - Erdos-Renyi random graph with edge probability $\frac{1}{2}$
  - Size $k \geq (2+\delta) \log_2 n$
    - $n$: size of $G$, $\delta > 0$
- We prove when $k \geq (2+\delta) \log_2 n$
  - Prob. of embedded watermark isomorphic to any subgraph in $G$
    - $P < \frac{1}{2} \left( \frac{\delta k^2}{2(2+\delta)} \right)^{-\frac{3k}{2}+1}$
    - Reduces exponentially to 0 as $k$ increases
      - e.g., for $G$ with 5M nodes, $k = 52$, $P < 10^{-30}$
Watermark Applicability

• 48 real graphs: 35 suitable

<table>
<thead>
<tr>
<th>OSNs</th>
<th>Collaboration networks</th>
<th>✔</th>
<th>Citation Networks</th>
<th>Communication networks</th>
<th>✔</th>
<th>Web graphs</th>
<th>Location based OSNs</th>
<th>✔</th>
<th>AS graphs</th>
<th>Amazon Co-purchasing Networks</th>
<th>✗</th>
<th>P2P networks</th>
<th>Road Networks</th>
<th>✗</th>
</tr>
</thead>
</table>

• Judging criteria for a suitable graph G
  – Have expected **node degree** for embedded watermark between
    o [min degree in G, max degree in G]
  – Have expected **graph density** for embedded watermark
    o [min density in k-size subgraph in G, max density in k-size subgraph in G]
**Attack Models**

- **Single Attack:** have *one* watermarked graph
  - Best strategy: randomly adding or deleting edges
  - Defense: additional features in system
    - e.g. add randomness in node labeling and matching, embed a watermark multiple times

- **Collusion Attack:** have *multiple* watermarked graphs
  - Best strategy: compare graphs to remove watermarks
  - Defense: hierarchical watermark embedding
    - Embed watermarks with potions of overlap
Hierarchical Embedding