GRECS: GRaph Encryption for Approx. Shortest Distance Queries

Xianrui Meng (Boston University)
Seny Kamara (Microsoft Research)
Kobbi Nissim (Ben-Gurion U. & CRCS Harvard U.)
George Kollios (Boston University)
Cloud Storage

Big Database

Box
Salesforce
iCloud
Amazon Web Services
Windows Azure
Graph Data

- Social Networks
- Communications
  - phone call logs
- Networks
- Web crawlers
- ... ...
Outsource Graphs
Outsourced Graph Data

Client (Data Owner)

Outsource

Query

Cloud Server
Graph Encryption

Setup Phase

Token + $L_2$

Query Phase

$\text{Enc}_K$ (token)

$\text{Enc}_K$ (image)

$+ L_1$

Leakage function

10/16/15

ACM CCS 2015
Security Definition

• Adaptive Chosen Query Attack (CQA-2)
  – Searchable Encryption
    [Curtmola-Garay-Kamara-Ostrovsky06], [CK10], [CJKRS13], etc ...
  – Simulation-based security

“No efficient adversary can learn any partial information about the data or the queries, beyond what is explicitly allowed by the leakage functions.”
“... even for queries that are adversarially-influenced and generated adaptively.”
Leakage

• Leakage function
  – Describe as *stateful* functions of the input data...
    o Size of the graph...
    o Query Pattern, i.e. whether the query has been repeated.
    o Access Pattern, i.e. pointer to the databases.
    o etc ...
State of the Art

• Searchable Encryption (SE)
  – Keyword Search [SWP01, CM05, CGKO06],
  – Boolean queries [CJJKRS13]
  – Range queries [SBCSP 07, LLWB 14]
  – Dynamic SE [KPR12, KP13, SPS 14, NPG 14]
  – Structured data [CK10]

• Oblivious RAM
  – More secure: does NOT leak access pattern
    [GO92, SDSCFRYD 13, DSS 14, WNLCSSH 14, LWNHS 15, etc...]

• Fully Homomorphic Encryption
  – [Gentry09]
GRECS: GRaph EnCryption for approx. Shortest distance queries
Querying the Encrypted Graph

$q = (u, v)$

$\text{Enc}(\text{dist}(u, v)) \xrightarrow{\text{Token}_q} \text{dist}(u, v) \approx d(u, v)^*$

* $d(u, v)$ is the *real* shortest distance between $u$ and $v$
Design a Practical Scheme

- Low Communication Complexity
- Reasonable Space Overhead
- Query Processing Overhead
  - Server: *small* computation
  - Client: *very small* computation
Shortest Distance for Graphs

G = (V, E)

O(|V| log |V| + |E|)

O(n)

O(1)

E.g. Dijkstra's algorithm

APSP \( n \times n \) matrix

Query Time

O(1)

Space

O(n^2)

Setup Time

O(n^3)

e.g., Floyd-Warshall!!

Want some **efficient** and **compact** Data Structure for Fast Shortest Distance Queries.
Practical Distance Oracle

Data Structure that is produced by some *Randomized* Algorithm...

The sketch is compact! normally $O(\log |V|)$
Most DOs have to compromise on accuracy: don't return the accurate distance but rather a constant-factor approximation of it.

DO returns $d$ s.t. $\text{dist} \leq d \leq \alpha \cdot \text{dist}$

($\text{dist}$: real shortest distance)
Our basic idea: to encrypt the DO

\[ \text{dist}(v_i, v_j) : \text{minimum}\{(2+2), (2+1)\} \]
## GRECS

<table>
<thead>
<tr>
<th></th>
<th>GraphEnc$_1$</th>
<th>GraphEnc$_2$</th>
<th>GraphEnc$_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space Complexity</strong></td>
<td>$O(n \log n)$</td>
<td>$O(n \log^2 n / \varepsilon)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>Server’s Query Complexity</strong></td>
<td>$O(1)$</td>
<td>$O(\log^2 n / \varepsilon)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td><strong>Client’s Query Complexity</strong></td>
<td>$O(\log n)$</td>
<td>$O(\text{diameter})$</td>
<td>$O(\text{diameter})$</td>
</tr>
</tbody>
</table>

- $n = |V|$ for $G = (V, E)$
- Sketch size is $\sim O(\log n)$
- GraphEnc$_3$ leaks a bit of more ...
GraphEnc₁: An Encrypted Storage Approach

Problem: Query has high Communication Complexity!!

Map to a token (PRF based)

Encrypt the distances and node & id’s

V: (b, 2) (c, 3) (e, 2) (i, 2) (m, 1) ...

Enc[(b, 2)], Enc[(c, 3)], Enc[(e, 2)], Enc[(i, 2)], Enc[(m, 1)] ...

Fₖ(v):
GraphEnc$_2$: Communication-Efficient O(1)
GraphEnc_2: our basic idea

• Setup:

1. Map v to token: F_k(v)

V: (v_1, d_1) (v_2, d_2) (v_3, d_3) (v_4, d_4)

2. Random hashing:
   h(node_id)

<table>
<thead>
<tr>
<th>0</th>
<th>h(v_4)</th>
<th>h(v_1)</th>
<th>h(v_3)</th>
<th>h(v_2)</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enc(B^{N-d_4})</td>
<td>Enc(B^{N-d_1})</td>
<td>...</td>
<td>Enc(B^{N-d_3})</td>
<td>Enc(B^{N-d_2})</td>
<td></td>
</tr>
</tbody>
</table>

3. Encode & Encrypt using SWHE (BGN encryption):

Enc(B^{N-dist})

*N is max dist in DO
*B is some positive integer

4. Encrypt the rest Enc(0)
GraphEnc_2: Query Overview

• **Query:**

\[
\text{query} = (u, v)
\]

Token: \( F_k(u), F_k(v) \)

\[
F_k(u): \quad \text{Enc}(a_1) \quad \text{Enc}(a_2) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(a_l)
\]

\[
\text{Enc}(a_1b_1 + \cdots + a_l b_l)
\]

\[
F_k(v): \quad \text{Enc}(b_1) \quad \text{Enc}(b_2) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(\cdots) \quad \text{Enc}(b_l)
\]

• **homomorphic multiplication:** bilinear pairing on ciphertext

• **homomorphic addition:** multiplication on ciphertext
Theorem:

with high probability,

\[ d(u, v) - e \leq \text{dist} \leq \alpha \cdot d(u, v) \]

\( e \): related # of common nodes in Sketch\((u)\) and Sketch\((v)\)

\( \alpha \): approximation ratio from Dist. Oracle

\[ \text{dist} = \text{Dec}(\text{Enc}(a_1b_1 + \ldots + a_lb_l)) \]

Server only returns
only one Enc(.)!
GRECS: GraphEnc$_2$

- **Theorem:**

  \[ \text{with high probability,} \]
  \[ d(u, v) - e \leq \text{dist} \leq \alpha \cdot d(u, v) \]

  \( e \): related # of common nodes in Sketch(u) and Sketch(v)

  \( \alpha \): approximation ratio from Dist. Oracle

- **Security:**
  - CQA2-secure against semi-honest adversarial server.
  - Leakage: query pattern, access pattern, |V|
GRECS: GraphEnc$_3$

- GraphEnc$_3$
  - Constant Communication Complexity
  - Much Lower Space Overhead
  - Much Faster Query Time
  - Leaks a bit of more than GraphEnc$_2$
    - Standard leakage similar to SE
Query Performance

**Query Processing (GraphcEnc₃)**

<table>
<thead>
<tr>
<th>V</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>as-skitter: 1.6M</td>
<td>11M</td>
</tr>
<tr>
<td>youtube: 1.1M</td>
<td>2.9M</td>
</tr>
<tr>
<td>gowalla: 0.20M</td>
<td>0.95M</td>
</tr>
<tr>
<td>enron: 36K</td>
<td>0.37M</td>
</tr>
<tr>
<td>condmat: 23K</td>
<td>0.19M</td>
</tr>
</tbody>
</table>

*24-core 2.9GHz Intel Xeon, 512 GB RAM*
Distance Accuracy

|           | \(|V|\) | \(|E|\)  |
|-----------|--------|---------|
| as-skitter| 1.6M   | 11M     |
| youtube   | 1.1M   | 2.9M    |
| gowalla   | 0.20M  | 0.95M   |
| enron     | 36K    | 0.37M   |
| condmat   | 23K    | 0.19M   |

![Distance Accuracy Graph]

- Frequency
- \(|\text{estimated distance} - \text{true distance}|\)
Subsequent/Ongoing work

- To support more complex graph queries/graph mining tasks ...
- More efficient searchable encrypted graph database ...

Challenge

- Leakage: how to \textit{minimize} and \textit{control} the leakage
  - Trade-off: privacy/performance/space
- Design schemes that scale to \textit{massive} data
- General Queries on Encrypted Graph Structure

More details see: \url{http://eprint.iacr.org/2015/266.pdf}
Thank you very much!

Questions?