Index Obfuscation for Oblivious Document Retrieval in a Trusted Execution Environment

Jinjin Shao, Shiyu Ji, Alvin Oliver Glova, Yifan Qiao, Tao Yang, Tim Sherwood
Department of Computer Science
University of California, Santa Barbara
Trusted Execution Environment for Privacy-Preserving Search

Client uploads encrypted documents and index, utilizing its massive storage and computing power.

Server is honest-but-curious: correctly executes protocols but observes/infers private data access patterns.

Challenges:

- Data access patterns leaked can lead to plaintext attacks on the encrypted index.
- Crypto-heavy techniques are too expensive.

Trusted Execution Environment (TEE, e.g. Intel SGX): an option with reasonably secure computing support.
Privacy Protection with Secure Computation in a TEE

- TEE (e.g. Intel SGX) provides a protected space where applications can run secure operations w. private data.
- However, the server can still observe data access traces which leak data-dependent access patterns and can lead to attacks on the encrypted index and queries.
- Oblivious retrieval called REARGUARD [INFOCOM18]: index matching with data access traces that do not depend the input query data, but with a high time cost.
A document retrieval scheme is oblivious over a query set if the server cannot distinguish the data access patterns of any two queries in this set.

Given a query term, REARGUARD scans through a group of posting lists, padded uniformly within a group.

The obfuscation degree of terms is the group size.

Expensive cost to achieve obliviousness when the group size is large, e.g. searching Term A scans entire Group 1.
Our Proposed Solution: Masked Inverted Index (MII)

Main ideas:

- Posting lists of terms are replicated and grouped randomly as buckets.
- Searching a term needs to access a merged posting list.
- The encrypted mask code differentiates which list is desired during the proposed oblivious query processing.
- Term replication increases obfuscation degree while incurs smaller space cost than REARGUARD.
Main steps (e.g. searching Term A):

- For a desired search term, the client sends a bucket id and an encrypted selector code (e.g. a binary code $10_2$ for Term A in the first bucket).
- The server retrieves a posting list for the bucket.
- Obliviously sort all selected & unselected documents with extracted features (e.g. Doc 1, 3, 4, 5).
- Output top-K documents (e.g. Doc 5 is dropped if $K = 3$).
Masked Inverted Index (MII): Extract Ranking Scores Obliviously

Challenges: Many empty features exist as NULL. Removing them leaks posting list structure. Including them costs significant space.

Space-optimized oblivious method with linear time:

- Only store non-empty features.
- Use a selector code to find the desired mask bit and calculate the index position of the desired feature.
- Obliviously fetch the feature (or 0, if mask bit is 0).
MII vs. REARGUARD: Privacy & Complexity

Let the lengths of posting lists follow a Zipf-like distribution.

- $g$: the group size in REARGUARD
- $k$: the term duplication degree in MII
- $b$: the bucket size in MII when $g = (k - 1) \times (b - 1)$

<table>
<thead>
<tr>
<th></th>
<th>Ratio of REGUARD/MII</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obfuscation degree of</td>
<td>$\sim 1$ with a large vocabulary size.</td>
</tr>
<tr>
<td>query and its terms</td>
<td></td>
</tr>
<tr>
<td>Index space cost</td>
<td>$\sim b$</td>
</tr>
<tr>
<td>Query processing time</td>
<td>$\sim k^2 b$</td>
</tr>
</tbody>
</table>

**Takeaway:** MII significantly outperforms REARGUARD in efficiency with competitive privacy protection.
### Experiment Results (Query Time)

<table>
<thead>
<tr>
<th></th>
<th>TREC disk4&amp;5</th>
<th>Clueweb09-Cat-B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BMW</strong></td>
<td>Ex. OR</td>
<td>REAR-GUARD</td>
</tr>
<tr>
<td><strong>2.8</strong></td>
<td>8.9</td>
<td>62.6</td>
</tr>
<tr>
<td><strong>3.2X</strong></td>
<td>22.4X</td>
<td>3.3X</td>
</tr>
</tbody>
</table>

*All times are in milliseconds. Baseline with no privacy-protection: BMW [SIGIR13], and exhaustive OR.

*Parameter settings: $g = 85, b = 18, k = 6$.

**Takeaway:**

1) MII is up-to 18.9x faster than REARGUARD, and about same speed as exhaustive OR; 2) MII vs. BMW: $\sim 4.9x$ slower (manageable) for privacy trade-off.
## Experiment Results (Space Cost)

<table>
<thead>
<tr>
<th></th>
<th>TREC disk4&amp;5</th>
<th></th>
<th>Clueweb09-Cat-B</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Encryption</td>
<td>REAR-GUARD</td>
<td>MII</td>
<td>No Encryption</td>
</tr>
<tr>
<td><strong>0.2 GB</strong></td>
<td>8.5 GB</td>
<td>3.1 GB</td>
<td>11.8 GB</td>
<td>709.1 GB</td>
</tr>
<tr>
<td><strong>-</strong></td>
<td>42.5X</td>
<td>15.5X</td>
<td><strong>-</strong></td>
<td>60.1X</td>
</tr>
</tbody>
</table>

*All sizes are after using simple-9 compression.*

**Takeaway:**

- The space cost of REARGUARD can be ~3X larger than that of MII.
- 17.6x more space than BMW for privacy trade-off. Acceptable cost.
Concluding Remarks

- **Contributions:** This work proposes a new oblivious document top-K retrieval scheme with an obfuscated inverted index to hide document-term association.
  - Avoid the pattern leakage of data access operations with oblivious index access and feature gathering.
  - Significant matching time speed-up over REARGUARD while with much smaller storage cost.
  - Slower with more space cost than BMW retrieval algorithm for privacy trade-off. The cost is acceptable.

- **Caveat and future work**
  - TEEs like SGX reside on the server machines and the risk such as physical or side-channel attacks exists
  - Integrate with ranking