SEARCHABLE SYMMETRIC ENCRYPTION: IMPROVED DEFINITIONS AND EFFICIENT CONSTRUCTIONS
Contents

- Background
- New SSE Constructions
  - Efficient Non-adaptive Security scheme
  - Adaptive Security Scheme
Background (i)

- Gen, Enc, Dec
- PRF & PRP (Pseudo-Random Function, Pseudo-Random Permutation)
  - Both map an input to a “random” output
Background (ii)

- SSE Schemes based on secure indices
  - Client generates secure index from documents
  - Sends index to server
  - Performs searches based on keywords on the index
  - Access Patterns
Background (iii)

- **Non-adaptive Security**
  - Adversary is not allowed to see previous query history to the server in order to guess which documents belong to which words

- **Adaptive Security**
  - Adversary is allowed to see previous query history to the server to guess which documents belong to which word
New SSE constructions

- $\Delta = (w_1, w_2, w_3, \ldots, w_d)$ be a dictionary of words
- id(D) = identifier of document D
- D(w) = list consisting of the identifiers of all documents containing word w.
- $\delta(D)$ be the set of all distinct words in document collection D
- $\delta(d)$ be the set of all distinct words in document d
- let n be the number of documents
Efficient Non-adaptive secure construction (i)

- Construct index I

  - Array A: which has for all w in δ(D), we store D(w)
  - Table T: contains information for word w which locates and decrypts the element in A
Efficient Non-adaptive secure construction (ii)

- For each $w$ in $\delta(D)$, construct a linked list
  - Each node contains identifier $D(w_i)$ & location of next node
  - Encrypt each node with a random key
- Compile all linked lists, permute and store into array
Efficient Non-adaptive secure construction (iii)

- Each entry in table T consists of \(<\text{address}, \text{value}>\) pair corresponding to \(w_i\).
- Value contains address of first node as well as the key to decrypt that node.
- Value is encrypted by output of pseudorandom function.
- Client sends the array, lookup table, and encrypted documents to server.
Efficient Non-adaptive secure construction (iv)

- On lookup, user generates address and value decryption key pair from the word.
- If no padding, reveals whether or not documents contain more than one keyword.

- \#A = size of array = \# of distinct keywords found in each document
- What happens if \#A < \#D (number of all words)?
Efficient Non-adaptive secure construction (v)

- Adversary can “guess” #D,
- Let $s'$ be the min size of keyword, and $s$ be the max size of keyword
- $s' \leq #D \leq s$
- Pad array A so that #A = $s$
- Pad table T so that it has $\Delta$ entries
Efficient Non-adaptive secure construction (v)

- Reveals only:
  1. access pattern
  2. search pattern
  3. total size of document collection
  4. and number of documents
Adaptive Secure Construction (i)

- Derive a label for \( w \)
  - Concatenate \( w \) and \( j \) where \( j \) represents the number that the word \( w \) appears in document collection \( D \)
    - i.e., if coin appears in three documents, the family for coin is \{coin1, coin2, coin3\}
- Make a lookup table \( T \) with entries \([\text{address}, \text{value}]\)
  - Address is the encrypted version of \( w||j \)
  - Value is the document identifier of the document that contains \( (w||j) \)
Adaptive Secure Construction (ii)

- Max family size is $n$, the number of documents
- Find the most amount of times ($Q$) a document appears for a keyword
- Pad all keywords so that each keyword has that many entries in the table
Adaptive Secure Construction (iii)

- Search by querying for the word $Q$ times
- Can be improved by adding one more round of communication
Adaptive Secure Construction (iv)

- Contain information about the number of words in the family in the value field
- When $w_i$ is queried, the size of the family is realized
- Make another query round knowing the size of the family
Adaptive Secure Construction (v)

<table>
<thead>
<tr>
<th>Properties</th>
<th>[35, 25]</th>
<th>[35, 25]-light</th>
<th>[40]</th>
<th>[23]</th>
<th>[18]</th>
<th>SSE-1</th>
<th>SSE-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>hides access pattern</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>server computation</td>
<td>$O(\log^3 n)$</td>
<td>$O(\sqrt{n})$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>server storage</td>
<td>$O(n \cdot \log n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>number of rounds</td>
<td>$\log n$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>communication</td>
<td>$O(\log^3 n)$</td>
<td>$O(\sqrt{n})$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>adaptive adversaries</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

- [40] Practical Techniques for searching on encrypted data
- [18] Privacy preserving keyword searches on remote encrypted data
- SSE - 1 is the first scheme presented
- SSE - 2 is the second scheme presented