Space-Efficient Private Search

George Danezis and Claudia Diaz
K.U.Leuven / COSIC (Belgium)

Financial Cryptography FC’07
13/02/2007
Outline

- Private Search
- Basic decoding algorithm: recursive extraction
- Extended decoding algorithm: solving equations
- Special case: one keyword
- Experimental results
- Conclusions
Private Search (Ostrovsky, Skeith)

- Alice ("searching party"): stores documents (in clear)
- Bob ("decoding party"): wants to retrieve documents matching some keywords

Properties:
- Bob gets documents containing the keywords
- Alice should not learn Bob’s keywords
- Alice should not learn the results of the search
Private search: encoding

- Homomorphic Pailler cryptosystem: $E(x) \cdot E(y) = E(x+y)$
- Dictionary:

<table>
<thead>
<tr>
<th>Term</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>“interesting”</td>
<td>$t_1=E(1)$</td>
</tr>
<tr>
<td>“boring”</td>
<td>$t_2=E(0)$</td>
</tr>
<tr>
<td>“stuff”</td>
<td>$t_3=E(0)$</td>
</tr>
</tbody>
</table>

- Document $d_1 = “interesting stuff”$
- $g_1 = t_1 \cdot t_3 = E(1) \cdot E(0) = E(1)$
- $g_1^{E(d_1)} = E(1 \cdot d_1) = E(d_1)$
Private search: encoding

- Homomorphic Pailler cryptosystem: $E(x) \cdot E(y) = E(x+y)$
- Dictionary:

<table>
<thead>
<tr>
<th>Term</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>“interesting”</td>
<td>$t_1 = E(1)$</td>
</tr>
<tr>
<td>“boring”</td>
<td>$t_2 = E(0)$</td>
</tr>
<tr>
<td>“stuff”</td>
<td>$t_3 = E(0)$</td>
</tr>
</tbody>
</table>

- Document $d_2 = “boring\ stuff”$
- $g_2 = E(0) \cdot E(0) = E(0)$
- $g_2^{E(d_2)} = E(0 \cdot d_2) = E(0)$
Private search: encoding

- Document $d_j \rightarrow$ Tuple: $T_j = (g_j, g_j^{E(d_j)})$
- $g_j = \prod_k t_k = E(m_j)$  \hspace{1cm} $g_j^{E(d_j)} = E(m_j \cdot d_j)$
- buffer of length $b$ initialized with $(E(0), E(0))$ in each position
- $T_j$ is multiplied to $L$ random locations of the buffer
Private Search: decoding

- Bob decrypts the buffer, and finds positions with one matching document
- Buffer positions with collisions are ignored
- For M expected matches: $b = 2 \cdot M \cdot L$
- Color survival game: $\Pr(d \text{ is lost}) < (\frac{1}{2})^L$
Improvement

- Buffer positions of document copies predictable to Bob
- We can still recover documents from the collisions!
Basic decoding algorithm: recursive extraction (modification)

- Agreed hash function $H$
- $p_{ij} =$ buffer position of copy $j$ of document $d_i$
- $2^{q-1} \leq b < 2^q$
Basic decoding algorithm: recursive extraction (algorithm)

\[(m_7 + m_2, m_7 \cdot d_7 + m_2 \cdot d_2)\]
Extended decoding algorithm: solving equations

- The basic decoding algorithm does not solve collisions like:

  \[
  d_1 + d_2 = b_3 \\
  d_2 + d_3 = b_5 \\
  d_1 + d_3 = b_7
  \]

- If documents’ positions are predictable \textit{a priori}, we can still recover these documents:
Extended decoding algorithm: solving equations

- **Modifications:**
  - Add serial number $i$ to documents
  - Total searched documents ($N$) known to Bob
  - $p_{ij}$ dependent on $i$ and $j$

- **Note:**
  - Applied *after* basic algorithm, in order to reduce the number of unknowns
Special case: one keyword

- In position $b_k$: $R = \sum i \cdot m_i$, $M = \sum m_i$

| $M$ | $d_i + d_j$ | $R = i + j$ |

- $m_i = 1$ if $d_i$ matched, 0 if not
- Exhaustively search all possible combinations of $M$ serial numbers that sum $R$
Special case: one keyword. Example

- \(1 \cdot m_1 + 2 \cdot m_2 + 3 \cdot m_3 = 4\)  \((m_1 + m_2 + m_3 = 2)\)
- \(2 \cdot m_2 + 4 \cdot m_4 + 5 \cdot m_5 = 7\)  \((m_2 + m_4 + m_5 = 2)\)
- \(1 \cdot m_1 + 3 \cdot m_3 + 4 \cdot m_4 + 5 \cdot m_5 = 9\)  \((m_1 + m_3 + m_4 + m_5 = 3)\)

- \(\rightarrow m_2 = m_4 = 0\)
- 3 equations and 3 unknowns remaining
Experimental results: performance

Performance of recursive extraction

Performance of solving equations

Fraction uncovered and Prob. of Success.

Matching documents (1000 buckets, 10 samples per point.)

Prob. of Success.

Matching documents (1000 buckets, 10 samples per point.)

- Recursion.
- Ostrovsky et al.

Prob. of Solving Eq.
Prob. for 1-match.
Experimental results: number of copies

Effect of number of copies

- Recursion
- Ostrovsky et al.
- Solving
- 1-match

Performance vs. Number of Copies

Number of Copies (100 buckets) vs. Performance

Number of Copies (1000 buckets) vs. Performance
Additional contributions

- Method for packing together several elements of the tuple
- Application to rateless codes with encrypted data
Conclusions

- We reduce by a significant factor the buffer size compared to Ostrovsky’s scheme.
- Basic algorithm (recursive extraction) has linear decoding complexity (Bethencourt et al. has cubic complexity).
- Extended algorithm (solving equations) is applied after recursive extraction (small number of equations/unknowns).
- For one keyword, buffer overhead is 10%.
- Technique to pack lists of values (can be applied both to Ostrovsky’s and Bethencourt’s schemes).
- Applications to rateless codes on encrypted data.
Tight packing

- $E(d_i) \cdot E(i) = E(d_i \cdot 2^k + i)$
- $E(d_j) \cdot E(j) = E(d_j \cdot 2^k + j)$
- $E(d_i \cdot 2^k + i) \cdot E(d_j \cdot 2^k + j) = E((d_i + d_j) \cdot 2^k + (i+j))$