Encrypted Edit Distance for Real-World Applications

Wouter Legiest, Jan-Pieter D’Anvers, and Ingrid Verbauwhede
COSIC KU Leuven, Belgium

Edit Distance

- Check the similarity of two strings \(a\) and \(b\)
- Count number of operations to transform \(a\) into \(b\)
- Common Operations in academic research:
  - **Insertion** of a character
  - **Deletion** of a character
  - **Substitution** of a character by another
  - **Transposition** of two adjacent characters
- Each edit distance has its allowed operations

<table>
<thead>
<tr>
<th>Distance</th>
<th>Ins/Del</th>
<th>Subs</th>
<th>Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming</td>
<td>•</td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>Longest Common Subsequence</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Levenshtein</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Damerau-Levenshtein</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

- Ex.: Levenshtein distance of ‘kitten’ and ‘sitting’ is 3

Edit Distance in the FHE World

- Lots of the optimised plaintext edit distance algorithm uses and/or
  - data-depended branching
  - data-depended preprocessing
  - small-alphabet specific optimisations
  - optimised for unit-costs

⇒ Two candidates:
- Wagner-Fisher algorithm
- bit-vector algorithm of Myers [1]
  - Only for unit costs operations; has \(O(n)\) time complexity

Wagner-Fisher Algorithm

- Textbook method to calculate edit distance: build up a matrix \(D\)
- Each element in the matrix is calculated as:
  \[ D_{i,j} = \min(D_{i-1,j} + 1; D_{i,j-1} + 1; D_{i-1,j-1} + (a_i = b_j)) \]
- Hard to parallelise; uses dynamic programming has \(O(n^2)\) time complexity

Real-World Requirements

- Non-unit cost of operations:
  - Insertion and Deletion: 2
  - Substitution: 0/1/2
- Alphabet using all numbers, lower and uppercase letters, and special characters, i.e. \(|\Sigma| > 64\)
- Strings length of \(|a| = |b| = 132\)
- Additional (symmetrical) operations, with reduced or zero cost
  - Substitution of vowels \(a \leftrightarrow e\)
  - Substitution of ‘close letters’ \(m \leftrightarrow n\)
  - Transposition of 2chars to 2chars \(cc \leftrightarrow ch\)
  - Transposition of 2chars to char \(dd \leftrightarrow d\)

Unit Cost Implementation Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
<th>Params</th>
<th>System</th>
<th>Algorithm TFHE-rs API</th>
<th>Time per (D_{i,j}) [ms]</th>
<th>Tot time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>WF Integer - <em>smart</em></td>
<td>TFHE-rs v0.5.3 [2]</td>
<td>Integer: 4 × PARAM_MESSAGE_2_CARRY_2_KS_PBS</td>
<td>Dual AMD EPYC™ 73F3 16-Core @ 3.5GHz</td>
<td></td>
<td>1067.6</td>
<td>5h 10m</td>
</tr>
<tr>
<td>Integer - <em>smart_par</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>421.0</td>
<td>2h 3m</td>
</tr>
<tr>
<td>High Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>495.5</td>
<td>2h 24m</td>
</tr>
<tr>
<td>Myers High Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>10m 55s</td>
</tr>
</tbody>
</table>

Non-unit Implementation Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scheme</th>
<th>Params</th>
<th>System</th>
<th>Algorithm TFHE-rs API</th>
<th>Time per (D_{i,j}) [ms]</th>
<th>Tot time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>WF Integer - <em>smart</em></td>
<td>TFHE-rs v0.5.3 [2]</td>
<td>Integer: 4 × PARAM_MESSAGE_2_CARRY_2_KS_PBS</td>
<td>Dual AMD EPYC™ 73F3 16-Core @ 3.5GHz</td>
<td></td>
<td>1226.2</td>
<td>5h 56m</td>
</tr>
<tr>
<td>Integer - <em>smart_par</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>480.6</td>
<td>2h 20m</td>
</tr>
<tr>
<td>High Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>608.0</td>
<td>2h 56m</td>
</tr>
</tbody>
</table>

References & Funding


ERC Adv. Grant No. 101020005; CSF No. VR20192203; FWO Post-doc No. 133185

wouter.legiest@esat.kuleuven.be