A Secure Similar Document Comparison Protocol

Alan J. Fret[§], Samuel Li[†], Alexander Studor[‡], and Maalik Winters[‡]

[§]University of Puerto Rico, allanjonuel@gmail.com; [†]Princeton University, seli@princeton.edu;
[‡]Missouri University of Science & Technology, {ajs6z9, mcw8f7}@mst.edu;

NSF REU Program in Cloud Security, Missouri University of Science & Technology, Summer 2017

Contribution

- **Problem**: Let there be a client (C) and a server (S) such that C wants to compare a query document d_q with every document in the server collection and retrieve the top-k similarity scores, without revealing either party's documents to the other party.
- **Objective**: To develop protocol to achieve this comparison without a trusted third party.

Background: IR & Cryptography

- **Document Model**: *Vector Space Model.* Term weight is given by: $term\ wt. = term\ doc.\ freq. * term\ inv.\ doc.\ freq.$
- **Similarity Metric**: Naïvely, a vector dot product, similar to *Apache Lucene*:

$$\text{score}(q,d) = \cdot\ \Sigma_{t \epsilon q} \text{tf}(t,d) \cdot \text{idf}(t)^2 \qquad (1)$$

- **Cryptosystem**: Paillier - Asymmetric with homomorphic properties: If $c_1 = E(m_1)$ & $c_2 = E(m_2)$ then,

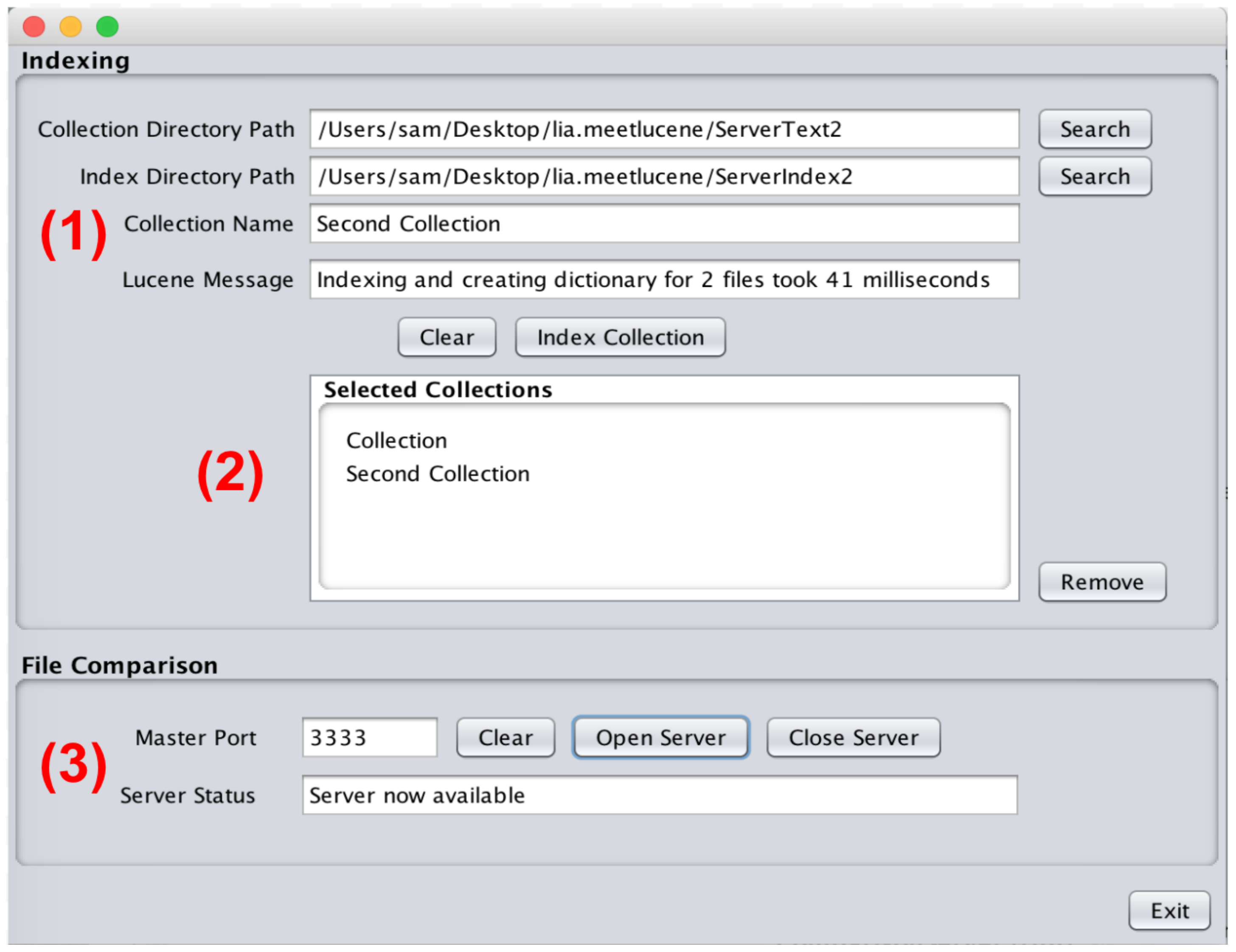
$$D(c_1 \cdot c_2) = (m_1 + m_2 \mod n) \ \ \&$$
$$D({c_1}^r) = (m_1 \cdot r \mod n)$$

- **Adversary Model**: Semi-honest as defined in Secure Multiparty Computation (SMP) protocol literatures.

Algorithm

```
Require: Server has <$d_1$,...$d_n$> and client has $d_q$.
1: Server:
   (a). for i = 1 to n do
           Index($d_i$)
   (b). Create dictionary of terms
   (c). for i = 1 to n do
           create vector $v_i$
   (d). Send dictionary to client
2: Client:
   (a). Index($d_q$)
   (b). Create vector $v_q$
   (c). PaillierEncrypt($v_q$) $\rightarrow$ E($v_q$)
   (d). Send E($v_q$) to server
3: Server:
   (a). for i = 1 to n do
           homomorphic($v_q$) $\rightarrow$ E($s_i$)
   (b). Send <E($s_i$), ...E($s_n$)> to client
4: Client:
   (a). PaillierDecrypt(<E($s_i$), ...E($s_n$)>) $\rightarrow$ <$s_1$,...$s_n$>
```

Application Interface



Collection Directory Path: /Users/sam/Desktop/lia.meetlucene/ServerText2

Index Directory Path: /Users/sam/Desktop/lia.meetlucene/ServerIndex2

Collection Name: Second Collection

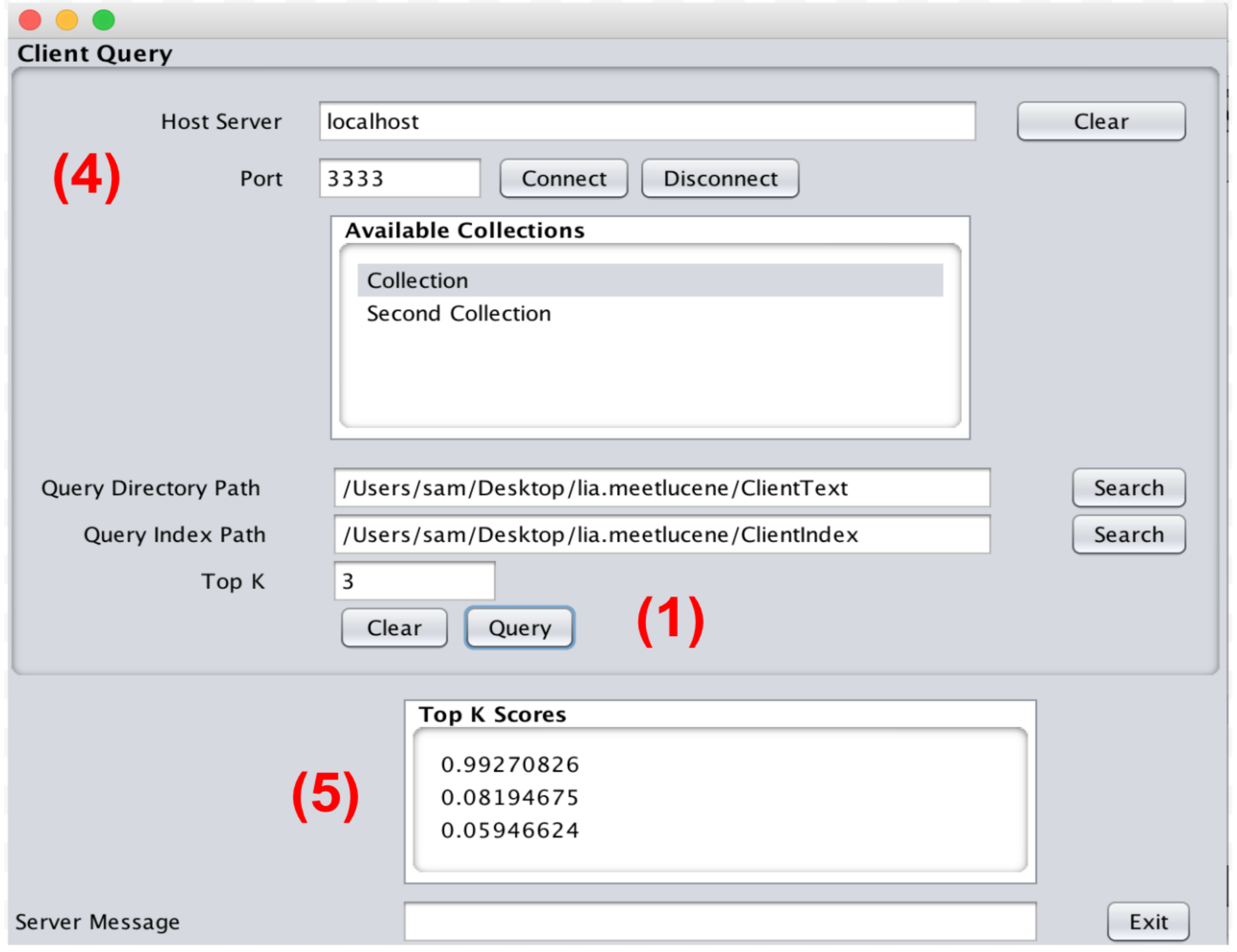
Lucene Message: Indexing and creating dictionary for 2 files took 41 milliseconds

Selected Collections:
Collection
Second Collection

Master Port: 3333

Server Status: Server now available

Figure 1: Server Application



Host Server: localhost

Port: 3333

Available Collections:
Collection
Second Collection

Query Directory Path: /Users/sam/Desktop/lia.meetlucene/ClientText

Query Index Path: /Users/sam/Desktop/lia.meetlucene/ClientIndex

Top K: 3

Top K Scores:
0.99270826
0.08194675
0.05946624

Server Message:

Figure 2: Client Application

Labeled Application Features

1. **Lucene Indexing**: The Apache Lucene library efficiently indexes the documents for vector creation
2. **Multiple Collections**: The client will be able to choose between multiple collections to query to
3. **Open master port**: All clients initially connect via the master port (3333)
4. **Connect to server**: Client must provide host name and master port
5. **Top-k scores**: Client displays scores

Time Complexity

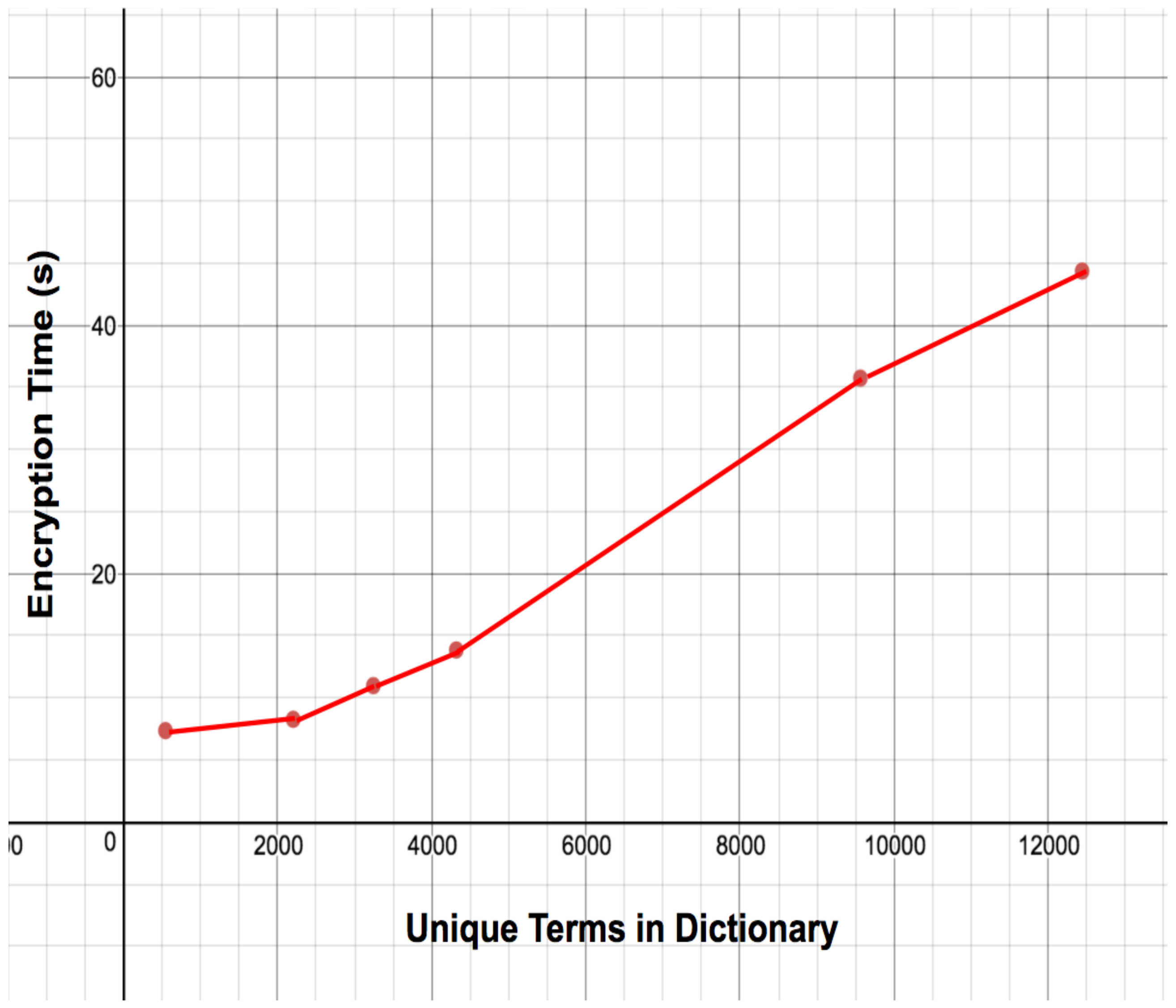


Figure 3: Encryption time versus dictionary size

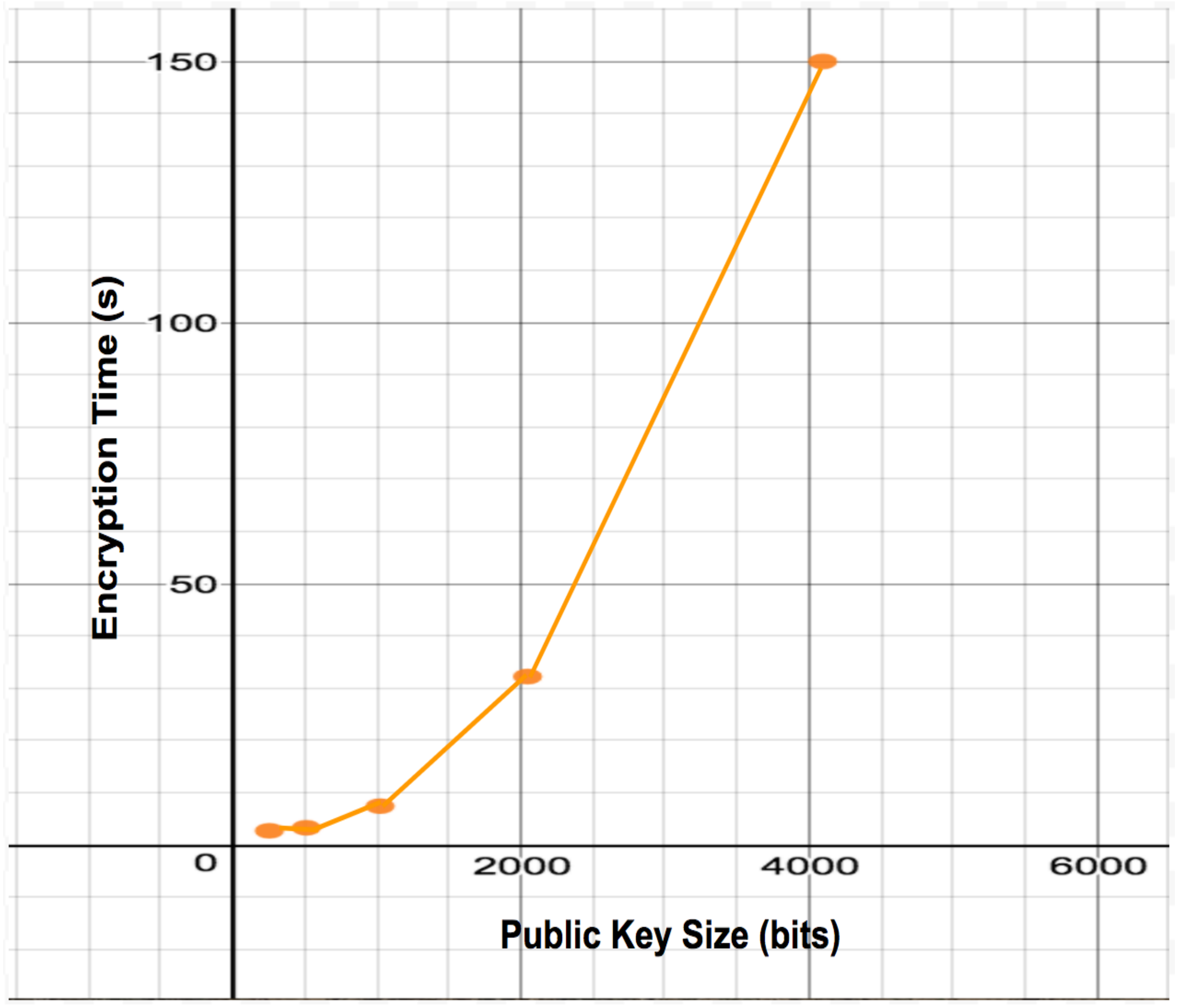


Figure 4: Encryption time versus public key size

We found that the client query encryption is the bottleneck of our protocol. Test cases carried out on varying parameters indicate a linear relationship between encryption time and dictionary size, and a quadratic relationship between encryption time and public key size.

Future Work

- **Implement SMIN**: Adopt secure SMIN protocol developed by Jiang et. al. so that client only receives top-k scores
- **Improve efficiency**: Modify our implementation to calculate comparisons faster.