A SUMMER INTERNSHIP PROJECT REPORT

on

Enabling Privacy Preservation of Encrypted Bank Data in Cloud using Multi-Keyword Ranked Search

Submitted By

Sajid Hussain
Admn No: 2012 JE 0742
B.Tech CSE

Under the Guidance of

Dr. P. Syam Kumar
Assistant Professor, IDRBT, Hyderabad

July 2015
CERTIFICATE

This is to certify that the summer internship project report entitled “Enabling Privacy Preservation of Encrypted Bank Data in Cloud using Multi-Keyword Ranked Search” submitted to Institute for Development & Research in Banking Technology [IDRBT], Hyderabad is a bonafide record of work done by “Sajid Hussain”, Admn no. – 2012JE0742, B.tech Computer Science and Engineering, Indian School of Mines, Dhanbad” under my supervision from “6th May, 2015” to “7th July, 2015”

Dr. P. Syam Kumar,
Assistant Professor
IDRBT, Hyderabad.
Acknowledgement

I would like to express my sincere gratitude to the Institute for Development and Research in Banking Technology (IDRBT) and particularly Dr. P. Syam Kumar who was my guide for this project. This opportunity of learning about Cloud Computing and various services like Saas, Iaas, Paas was a boon to me as one rarely gets such exposure. I would like to add that this short period in IDRBT has added a different facet to my life as this is a unique organisation being a combination of academics, research, technology, communication services, crucial applications etc.

I am extremely grateful to Dr. P. Syam Kumar for his advice, innovative suggestions and supervision. I thank him for introducing me to this excellent area of Cloud Computing for research. I am thankful to IDRBT for providing such an amazing platform for students, like me, to work in real application oriented research. I thank one and all who made this project successful either directly or indirectly.

Sajid Hussain
ISM Dhanbad
Project Trainee
IDRBT Hyderabad.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>1</th>
<th>Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Definition of Cloud Computing</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Benefits of using cloud computing in Bank</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>The main challenges</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Motivation</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Literature Survey</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Problem Formulation</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>Threat Model</td>
<td>7</td>
</tr>
<tr>
<td>3.2</td>
<td>Design Goals</td>
<td>8</td>
</tr>
<tr>
<td>3.3</td>
<td>Preliminary</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Framework for MRSE</td>
<td>9</td>
</tr>
<tr>
<td>4.1</td>
<td>Privacy Requirements</td>
<td>9</td>
</tr>
<tr>
<td>4.2</td>
<td>Algorithms</td>
<td>10</td>
</tr>
<tr>
<td>4.3</td>
<td>Secure inner product computation</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Architecture and Implementation</td>
<td>13</td>
</tr>
<tr>
<td>5.1</td>
<td>Four Phases</td>
<td>14</td>
</tr>
<tr>
<td>5.2</td>
<td>Hardware and Software requirements</td>
<td>15</td>
</tr>
<tr>
<td>5.3</td>
<td>Working Prototype Demo</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Bibliography</td>
<td>25</td>
</tr>
</tbody>
</table>
Abstract

With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to the commercial public cloud for great flexibility and economic savings. But for protecting data privacy, sensitive data have to be encrypted before outsourcing, which obsoletes traditional data utilization based on plaintext keyword search. Thus, enabling an encrypted cloud data search service is of paramount importance. Considering the large number of data users and documents in the cloud, it is necessary to allow multiple keywords in the search request and return documents in the order of their relevance to these keywords. Related works on searchable encryption focus on single keyword search or Boolean keyword search, and rarely sort the search results. In this report, we implement challenging problem of privacy-preserving multi-keyword ranked search over encrypted data in cloud computing (MRSE). Among various multi-keyword semantics, we choose the efficient similarity measure of “coordinate matching,” i.e., as many matches as possible, to capture the relevance of data documents to the search query. We further use “inner product similarity” to quantitatively evaluate such similarity measure.
1. Introduction:

The IT revolution has had a great impact on the banking sector in India and banking in general. Introduction of computers lead to online banking, which tremendously improved the quality of customer service offered by the banks.

To drive growth and innovation in banking it is increasingly necessary to dramatically leapfrog the competition using IT and transforming the business model. Cloud computing the revolutionising the ecosystems in many industries and banks are no exception. Cloud technology offers secure deployment option that can help banks develop new customer experiences, enabling effective collaborations, and improving speed to market. However in developing or updating the bank's cloud strategy and infrastructure it is important to keep security in mind.

1.1 Definition of cloud computing

Cloud computing is a model for enabling ubiquitous, convenient and on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

It has got the following five characteristics:

1. On demand self service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction.
2. Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platform.
3. Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the
customer generally has no control or knowledge over the exact location of the provided resources.

4. **Rapid elasticity**: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.

5. **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

It has the following service models:

1. **Software as a Service** (Saas): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

2. **Platform as a Service** (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

3. **Infrastructure as a Service** (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

Cloud Can be deployed in following ways:

1. **Private cloud**: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It
may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

2. **Community cloud.** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

3. **Public cloud.** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

4. **Hybrid cloud.** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

### 1.2 Benefits of using cloud computing in bank.

1. **It cuts costs:** It is getting pretty clear that an individual company is unlikely to do as good of a job securing data than a reputable cloud provider. There are governmental regulations to be followed which can be daunting for some institutions but the major cloud vendors have been certified and will provide the infrastructure that fits their requirements.

2. **Improving the flexibility and scalability:** The cloud gives banks the ability to respond quickly to changing markets, customers and technological needs.

3. **Increased Efficiency:** Banks will enjoy improved efficiency ratios and operating leverage.
1.3 The main challenges:

1. **Security and compliance:** As the data is outsourced outside the bank it has to make sure that the data in the cloud is secured and protected and does not fall in wrong hands.

2. **Reliability:** The bank also have to make sure the data outsources remains reliable and corrected without any corruption and damage.

3. **Cloud management:** In the classical model the management of the servers are in the hands of bank, but here the management are looked by CSP. The bank has to make sure that the data is stored, without the need to manage it regularly.

4. **Interoperability:** As there are different cloud service providers, when bank decides to choose another CSP for another project, it should be interoperable.

5. **Regulations:** There are some regulations regarding the locations of the CSP, and privacy norms set by government. The cloud has to make sure that these regulations are implemented.

1.4 Motivation:

*Privacy Preserving through MRSE:*

This project aim at solving the security issues in the cloud. The cloud’s great flexibility and economic savings are motivating banks to outsource their local complex data management system into the cloud. To protect data privacy and combat unsolicited accesses in the cloud and beyond, sensitive data, tax documents, financial transactions, and so on, may have to be encrypted by data owners before outsourcing to the commercial public cloud. However, this obsoletes the traditional data utilization service based on plaintext keyword search. The trivial solution of downloading all the data and decrypting locally is clearly impractical, due to the huge amount of bandwidth cost in cloud scale systems aside from eliminating the local storage management, storing data into the cloud serves no purpose unless they can be easily searched and utilized. Thus, exploring privacy-preserving and effective search service over encrypted cloud data is of paramount importance. On the one hand, to meet the effective data retrieval need, the large amount of documents demand the cloud server to perform result relevance ranking, instead of returning undifferentiated results. Such ranked search system enables data users to find the most relevant information quickly, rather than burdensome sorting through every match in the content collection.
The literature survey can be divided into two types:

1. **Single keyword searchable encryption**

   Traditional single keyword searchable encryption schemes [7], [8], [9], [10], [11], [12], [13], [14], [15], [25], [26] usually build an encrypted searchable index such that its content is hidden to the server unless it is given appropriate trapdoors generated via secret key(s) [4]. It is first studied by Song et al. [7] in the symmetric key setting, and improvements and advanced security definitions are given in Goh [8], Chang et al. [9], and Curtmola et al. [10]. Our early works [25], [26] solve secure ranked keyword search which utilizes keyword frequency to rank results instead of returning undifferentiated results. However, they only support single keyword search. In the public key setting, Boneh et al. [11] present the first searchable encryption construction, where anyone with public key can write to the data stored on server but only authorized users with private key can search. Public key solutions are usually very computationally expensive however. Furthermore, the keyword privacy could not be protected in the public key setting since server could encrypt any keyword with public key and then use the received trapdoor to evaluate this ciphertext.

2. **Boolean Keyword Searchable Encryption**:

   To enrich search functionalities, conjunctive keyword search [16], [17], [18], [19], [20] over encrypted data have been proposed. These schemes incur large overhead caused by their fundamental primitives, such as computation cost by bilinear map, for example, [18], or communication cost by secret sharing, for example, [17]. As a more general search approach, predicate encryption schemes [21], [22], [23] are recently proposed to support both conjunctive and disjunctive search. Conjunctive keyword search returns “all-or-nothing,” which means it only returns those documents in which all the keywords specified by the search query appear; disjunctive keyword search returns undifferentiated results, which means it returns every document that contains a subset of the specific keywords, even only one keyword of interest. In short, none of existing Boolean keyword searchable encryption schemes support multiple keywords ranked search over encrypted cloud data while preserving privacy as we propose to explore in this paper. Note that, inner product queries in predicate encryption only predicates whether two vectors are orthogonal or not i.e., the inner product value is concealed except when it
equals zero. Without providing the capability to compare concealed inner products, predicate encryption is not qualified for performing ranked search. Furthermore, most of these schemes are built upon the expensive evaluation of pairing operations on elliptic curves. Such inefficiency disadvantage also limits their practical performance when deployed in the cloud. Our early work [1] has been aware of this problem, and provides solutions to the multi-keyword ranked search over encrypted data problem. In this paper, we extend and improve more technical details as compared to [1]. We propose two new schemes to support more search semantics which improve the search experience of the MRSE scheme, and also study the dynamic operation on the data set and index which addresses some important yet practical considerations for the MRSE design. On a different front, the research on top-k retrieval [31] in database community is also loosely connected to our problem. Besides, Cao et. al. proposed a privacy-preserving graph containment query scheme [36] which solves the search problem with graph semantics.
3. Problem Formulation:

Considering a cloud data hosting service involving three different entities
1. The data owner (The Bank)
2. The cloud server (Third party cloud service provider)
3. The data user (customer)

The data owner has a collection of data documents \( F \) to be outsourced to the cloud server in the encrypted form \( C \). To enable the searching capability over \( C \) for effective data utilization, the data owner, before outsourcing, will first build an encrypted searchable index \( I \) from \( F \), and then outsource both the index \( I \) and the encrypted document collection \( C \) to the cloud server. To search the document collection for given keywords, an authorized user acquires a corresponding trapdoor \( T \) through search control mechanisms. Upon receiving \( T \) from a data user, the cloud server is responsible to search the index \( I \) and return the corresponding set of encrypted documents. To improve the document retrieval accuracy, the search result should be ranked by the cloud server according to some ranking criteria (e.g., coordinate matching, as will be introduced shortly).

3.1 Threat Model:

The cloud server is considered as “honest-but-curious” in our model, which is consistent with related works on cloud security. Specifically, the cloud server acts in an “honest” fashion and correctly follows the designated protocol specification. However, it is “curious” to infer and analyze data (including index) in its storage and message flows received during the protocol so as to learn additional information. Based on what information the cloud server knows, we consider two threat models with different attack capabilities as follows.

1. **Known ciphertext model**: In this model, the cloud server is supposed to only know encrypted data set \( C \) and searchable index \( I \), both of which are outsourced from the data owner.

2. **Known background model**: In this stronger model, the cloud server is supposed to possess more knowledge than what can be accessed in the known ciphertext model. Such information may include the correlation relationship of given search requests (trapdoors), as well as the data set related statistical information. As an instance of possible attacks in this case, the cloud server could use the known trapdoor information combined with document/keyword frequency to deduce/identify certain keywords in the query.
3.2 Design Goals:

To enable ranked search for effective utilization of outsourced cloud data under the aforementioned model, our system design should simultaneously achieve security and performance guarantees as follows.

a. **Multi-keyword ranked search.** To design search schemes which allow multi-keyword query and provide result similarity ranking for effective data retrieval, instead of returning undifferentiated results.

b. **Privacy-preserving.** To prevent the cloud server from learning additional information from the data set and the index, and to meet privacy requirements.

c. **Efficiency.** Above goals on functionality and privacy should be achieved with low communication and computation overhead.

3.3 Preliminary:

*Coordinate Matching:* As a hybrid of conjunctive search and disjunctive search, “coordinate matching” is an intermediate similarity measure which uses the number of query keywords appearing in the document to quantify the relevance of that document to the query. When users know the exact subset of the data set to be retrieved, Boolean queries perform well with the precise search requirement specified by the user. In cloud computing, however, this is not the practical case, given the huge amount of outsourced data. Therefore, it is more flexible for users to specify a list of keywords indicating their interest and retrieve the most relevant documents with a rank order.
4. Framework for MRSE:

The MRSE system consists of four algorithms as follows.

1. Setup: Taking a security parameter as input, the data owner outputs a symmetric key.

2. Build Index: Based on the data set $F$, the data owner builds a searchable index $I$ which is encrypted by the symmetric key and then outsourced to the cloud server. After the index construction, the document collection can be independently encrypted and outsourced.

3. Trapdoor: With $t$ keywords of interest as input, this algorithm generates a corresponding trapdoor.

4. Query: When the cloud server receives a query request as, it performs the ranked search on the index $I$ with the help of trapdoor, and finally returns, the ranked id list of top-$k$ documents sorted by their similarity.

4.1 Privacy Requirements:

The representative privacy guarantee in the related literature, such as searchable encryption, is that the server should learn nothing but search results. With this general privacy description, we explore and establish a set of strict privacy requirements specifically for the MRSE framework. As for the data privacy, the data owner can resort to the traditional symmetric key cryptography to encrypt the data before outsourcing, and successfully prevent the cloud server from prying into the outsourced data. With respect to the index privacy, if the cloud server deduces any association between keywords and encrypted documents from index, it may learn the major subject of a document, even the content of a short document. Therefore, the searchable index should be constructed to prevent the cloud server from performing such kind of association attack. While data and index privacy guarantees are demanded by default in the related literature, various search privacy requirements involved in the query procedure are more complex and difficult to tackle as follows.

*Keyword privacy.* As users usually prefer to keep their search from being exposed to others like the cloud server, the most important concern is to hide what they
are searching, i.e., the keywords indicated by the corresponding trapdoor. Although the trapdoor can be generated in a cryptographic way to protect the query keywords, the cloud server could do some statistical analysis over the search result to make an estimate. As a kind of statistical information, document frequency (i.e., the number of documents containing the keyword) is sufficient to identify the keyword with high probability. When the cloud server knows some background information of the data set, this keyword specific information may be utilized to reverse engineer the keyword.

**Trapdoor unlinkability.** The trapdoor generation function should be a randomized one instead of being deterministic. In particular, the cloud server should not be able to deduce the relationship of any given trapdoors, for example, to determine whether the two trapdoors are formed by the same search request. Otherwise, the deterministic trapdoor generation would give the cloud server advantage to accumulate frequencies of different search requests regarding different keyword(s), which may further violate the aforementioned keyword privacy requirement. So the fundamental protection for trapdoor unlinkability is to introduce sufficient nondeterminacy into the trapdoor generation procedure.

4.2 Algorithms:

To efficiently achieve multi-keyword ranked search, we propose to employ “inner product similarity” to quantitatively evaluate the efficient similarity measure “coordinate matching.” Specifically, D is a binary data vector for document where each bit of D is 0, 1 represents the existence of the corresponding keyword in that document, and Q is a binary query vector indicating the keywords of interest where each bit of Q is 0, 1 represents the existence of the corresponding key. The similarity score of document word in the query is therefore expressed as the inner product of their binary column vectors. For the purpose of ranking, the cloud server must be given the capability to compare the similarity of different documents to the query. But, to preserve strict system wise privacy, data vector, query vector Q and their inner product should not be exposed to the cloud server. In this section, we first propose a basic idea for the MRSE using secure inner product computation, which is adapted from a secure kNN technique, and then show how to significantly improve it to be privacy-preserving against different threat models in the MRSE framework in a step-by-step manner. We further discuss supporting more search semantics and dynamic operation.
4.3 Secure inner product computation:

In the secure kNN scheme, euclidean distance between a data record \( p \) and a query vector \( q \) is used to select \( k \) nearest database records. The secret key is composed of one 1-bit vector as \( S \) and two invertible matrices as \( M_1 \) and \( M_2 \). The query vector \( q \) is scaled by a random number \( r > 0 \). Then, data record is split into two random vectors and \( q \) is also split into two random vectors. Note here that vector \( S \) functions as a splitting indicator. Namely, if the \( j \)th bit of \( S \) is 0, \( j \)th bit of split vectors are set as the same as \( j \)th bit of \( p \) while \( j \)th bits of split vectors of \( q \) are set to two random numbers so that their sum is equal to the \( j \)th bit of \( q \). If the \( j \)th bit of \( S \) is 1, the splitting process is similar except that \( p \) and \( q \) are switched. The split data vector pair\((p1, p2)\) is encrypted as \( M_1^T * p_1 \) and \( M_2^T * p_2 \) and query vector pair are encrypted as similar to data vector pairs but multiplied to inverse of \( M_1 \) and \( M_2 \). In the query step, the product of data vector pair and query vector pair is serving as the indicator of euclidean distance to select \( k \) nearest neighbors.

The whole scheme to achieve ranked search with multiple keywords over encrypted data is as follows:

1. **Setup.** The data owner randomly generates a vector as \( S \) and two invertible matrices \( M_1, M_2 \). The secret key is in the form of a 3-tuple as \( S, M_1, M_2 \).

2. **BuildIndex:** The data owner generates a binary data vector \( D \) for every document \( F \), where each binary bit of \( D \) represents whether the corresponding keyword appears in the document \( F \). These index files are encrypted by the above knn secure product computation.

3. **Trapdoor:** With \( t \) keywords of interest, a one binary vector \( Q \) is generated where each bit indicates the existence of that word in dictionary. This binary vector is then scaled by a random vector and then it is splitted and encrypted by the secure knn computation procedure.

4. **Query:** With the trapdoor the cloud server computes the similarity scores of each document as follows:
\[ I_i \cdot T_{W_i} = \{ M_1^T \bar{D}_i', M_2^T \bar{D}_i'' \} \cdot \{ M_1^{-1} \bar{Q}', M_2^{-1} \bar{Q}'' \} \]
\[ = \bar{D}_i' \cdot \bar{Q}' + \bar{D}_i'' \cdot \bar{Q}'' \]
\[ = \bar{D}_i \cdot \bar{Q} \]
\[ = (D_i, \varepsilon_i, 1) \cdot (rQ, r, t) \]
\[ = r(D_i \cdot Q + \varepsilon_i) + t. \]
5. Architecture and Implementation:

MRSE architecture

cloud-IDRBT

Bank

lib

crypting_vector.so

generating_keys.so

Preprocessing

crypting_vector.c

generate_keys.c

setup.py

Server

Django Files

q_vector.py

Cloud

cloud_server

Django Files

multiply.c

lib

multiply.so
5.1 Four Phases:

There are four phases during the searching. The source code of this implementation can be found at https://github.com/sajjusajuu/cloud-IDRBT. The four phases are as follows:

Note: The first two steps are carried for one time to set up the bank server and cloud server. For the later purpose of searching the third and fourth phases are carried for every keyword to be searched.

1. Setup: During this phase all the files of the database are listed and a collection of words present in the files are listed in a file called dictionary.json. Let the total number of keywords in dictionary be \( n \), then we generate the keys consisting of a random binary vector \( s\text{.key} \), and two random invertible matrices \( m1\text{.key} \) and \( m2\text{.key} \). From the generated keys additional components like inverse of \( s \), \( i_s\text{.key} \) and inverse and transpose of \( m1 \) and \( m2 \) which are \((m1\_t\text{.key}, m1\_i\text{.key}),(m2\_t\text{.key}, m2\_i\text{.key})\) are computed.

2. Building Index: During this phase keys, for each file in the database a binary index vector is created corresponding to the keywords in the dictionary. This index vector is then encrypted into \((\text{crypt}_1\text{.key}, \text{crypt}_2\text{.key})\). Thus for every file in the database we have compute the index files and encrypt them. These encrypted files are stored in crypted_index_files. After computing the index vectors the files in the database are themselves encrypted and stored in crypted_documents, which are later uploaded to the cloud server.

3. Trapdoor generation: For this purpose the bank hosts a website which will serve the trapdoor files to the customers which are then passed to cloud to search the documents. The trapdoor consists of two files \((\text{crypt}_1\text{.key}, \text{crypt}_2\text{.key})\) which are generated as the index vector but with a slight different procedure. The trapdoor is sent back to the customer along with the passkeys by which the documents were encrypted.

4. Query: The cloud hosts its website as an interface for the searching of documents. The customer uploads the trapdoor generated in the above phase. The trapdoor is given to cloud server. It takes the trapdoor and multiplies it\((\text{multiply.c})\) with all the encrypted index vectors and calculates
the score for every document. The top non-zero documents up to five are then returned to the customer, in the encrypted format. The customer downloads the documents and decrypts it with passkeys.

5.2 Hardware and Software Requirements:

Required applications/libraries for the projects:

1. gnu gsl libraries.
2. Python (2.7)
3. gcc compiler
4. Natural Language ToolKit package for python
5. Django
6. Gnu Privacy Guard
7. Apache web server

Hardware Requirements for the project.

1. A cloud server, with minimum of 1GHz processor with a memory of 100MB running Linux.

2. A cloud server with minimum of 1GHz processor with a memory of 200MB running Linux.

3. A client computer which can run a browser application running any OS.
5.3 Working Prototype Demo:

1. Set the environment variable LD_LIBRARY_PATH to the library
   
   ```
   export LD_LIBRARY_PATH=~/cloud-IDRBT/Bank/lib/
   ```

2. Running the setup.py:

Results before executing:

![Image of file directory with setup.py and related files]
The processing step of setup.py:

After the preprocessing these files are generated.
3. After that, start the bank server

4. Now in the browser type the ip address of the server, as in my case it is 192.168.1.20

5. Login with the given username and password.
6. That should open a keyword search page as shown. In the given form enter the key-words and press submit.

7. That will generate the trapdoor (you will get query.zip file download)

Thus we have trapdoor for our given keywords.
8. Now start the cloud-server. For the cloud server I have chosen an amazon instance as a sample cloud-server. So let us connect to the cloud instance:

![Cloud server connection](image1)

9. Start the server in cloud:

![Server started](image2)
10. Now open the browser and type the ip-address of the instance:

We yet need to upload the encrypted index files and encrypted documents. For that click on the Upload files.

Now select the zip-files which were generated in step 2.
11. It takes some time to get upload and after upload we have as below:

13. Now click submit. It will give the files, with top similarity scores.

We can download the files by clicking them.

14. After downloading file 169 and opening it shows:

It is a binary file and encrypted.
15. After decrypting it we have original file as shown:
7. Bibliography:

37. Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper