Privacy Preserving Multi Keyword Ranked Search with Context Sensitive Synonyms over the Encrypted Cloud Data

Anu Khurana[1(✉)], Rama Krishna Challa[2], and Navdeep Kaur[3]

[1] I.K. Gujral PTU, Kapurthala, Punjab, India
annu_khurana@yahoo.com
[2] Department of Computer Science and Engineering, NITTTR, Chandigarh, India
rkc_97@yahoo.com
[3] Department of Computer Science and Engineering, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India
drnavdeep.iitr@gmail.com

Abstract. The increase in the number of people and organizations relying on cloud for storing their data has led to significant increase in the volume of data on cloud. Wherever the Cloud Service Provider (CSP) allows encryption of documents and metadata, keyword search becomes necessary for quick, easy and effective retrieval of outsourced encrypted cloud data. As the cloud users, over the period of time may tend to forget the exact keywords for issuing a search query, the keyword search therefore should support synonyms. But the synonym discovery is always context sensitive, as not all synonyms can simply replace a word in all occurrences of a query. It is therefore, important to keep the connotation of the word under consideration. As some synonyms can infuse a different meaning, than the one user actually intends for and can cause drift to user's search. In this paper, we propose a scheme Context sensitive Multi keyword Ranked Search (CSMRS) to this issue by analyzing the encrypted query click logs. The results achieved show that the synonyms selected with CSMRS are more appropriate and as per the context as intended by the user.

Keywords: Multi-keyword · Synonyms · Context sensitive
Privacy preserving · Cloud · Encrypted query click logs

1 Introduction

With cloud computing, computing now sees no border. A number of users are getting hooked up with cloud because of the enormous benefits it provides, with a good percentage, using storage as a service [1]. Since the volumes of data stored on the cloud may contain user's sensitive information as well, therefore the data is stored in encrypted format on the cloud. However, at the time of locating and retrieval of the desired documents, a user would prefer a search mechanism for easy and quick retrieval of the needed documents. Encryption though required, makes the operations like searching difficult. The notion of searchable encryption (SE) without the loss of data

S. Verma et al. (Eds.): CNC 2018, CCIS 839, pp. 165–180, 2019.
https://doi.org/10.1007/978-981-13-2372-0_15

confidentiality has been introduced for the first time by Song et al. [2]. Thereafter many researchers, have worked towards increasing efficiency and improving ways of searching over the encrypted data [3–9]. But, these searchable encryption schemes do not directly fit into the cloud environment as they lack the effective mechanism to ensure file retrieval accuracy.

Many researchers [10–19] gave keyword search schemes for searching over encrypted cloud data. Later, for enhancing the search capability, some researchers [20–22] worked in the domain of keyword ranked search allowing queries that support synonyms.

Fu et al. in [20] used searchable index tree with the document index vectors for searching related documents. They built a common synonym thesaurus on the foundation of the New American Roget's College Thesaurus. They extend the extracted keywords for building index with common synonyms in order to allow synonym based search.

Mittal et al. in [21] created a customized synonym dictionary that is updated as new keywords from the files to be outsourced are added. On finding a new keyword, synonyms for every relevant keyword are added to the synonym dictionary. They later expand their query with synonyms from this synonym dictionary and perform search for locating relevant documents.

Saini et al. in [22] builds a synonym dictionary during setup phase. For the query issued, the cloud instead of replacing the query keywords with synonym suggests some synonyms for the user. If the user selects the synonym then the query returns ranked search results for the same.

As per the literature survey the existing schemes of search over encrypted cloud data supporting synonyms either keep waiting for user input or use a fixed set of synonyms. But the synonyms are context sensitive. Synonym for a word simply cannot replace all occurrences of a word in all the queries. For e.g. the words "powerful" and "muscular" are synonyms for the word "strong" but when we use "strong coffee" we do not equally say "powerful coffee" or "muscular coffee". This means the word "powerful" is not used as a synonym for the word "strong" over here. However, in the search query "powerful man" synonym "strong" or "muscular" may replace the keyword "powerful" and we can use "strong man" or "muscular man" as parallel queries in order to get more relevant results as per user's intent.

In this paper, we propose a scheme Context Sensitive Multi keyword Ranked Search (CSMRS) that supports privacy preserving multi keyword ranked search over encrypted cloud data supporting context sensitive synonyms. CSMRS uses encrypted query click logs for locating the relevant synonyms as per the context. CSMRS limits the synonym replacement in the multi keyword queries according to relevance as per the co-occurring words in the query. The encrypted query click logs are recorded whenever a data user downloads any document against the issued search query.

2 Problem Formulation

2.1 Problem

To find the context sensitive synonyms, we need to extract the context sensitive synonyms from the full set of synonyms picked from $\mathcal{SD}$ for a word. Neighbouring words in a search query can help to understand the context for a word, hence the need for query click logs. However, for multi keyword ranked search with context sensitive synonyms over the encrypted cloud data we need to take care that it should be privacy preserving. It means that the search query and the search results returned to the user should be hidden from the public cloud server, despite it performing the search operation.

2.2 System Model

CSMRS considers the cloud data hosting service to involve four different entities namely the data owner, the data user, public cloud server and a trusted third party wherein the private cloud server is considered as the trusted third party. The details of the responsibilities of these entities is given in Sect. 5. The architecture of the proposed CSMRS is shown in Fig. 1.

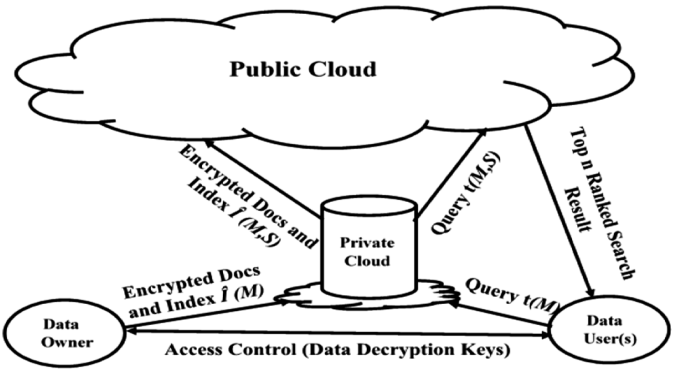

Fig. 1. Architecture of CSMRS (Context Sensitive Multi Keyword Ranked Search)

2.3 Threat Model

In this paper, we consider public cloud server to be "honest-but-curious". It is assumed that it carries out the assigned duties honestly, but is curious to examine data for additional insight. Regarding private cloud we consider it as a trusted third party.

2.4 Notations and Preliminaries

Notations

- $\hat{I}(M)$: Document Index encrypted with key M
- $\hat{I}(M, S)$: Document Index encrypted with key M further encrypted with key S
- $t(M)$: Query unique words encrypted with key M
- $t(M, S)$: Query unique words encrypted with key M further encrypted with key S
- $\mathbb{WV}$: Table containing all ciphers of the word vectors
- $\mathbb{SD}$: Synonym Dictionary
- $\mathbb{CD}$: Collocation Dictionary
- $\bar{Q}$: Master Encrypted Query Click Log.

Preliminaries

Paillier Encryption

It is a probabilistic public key algorithm with additive homomorphic properties. This means that given $E(a1)$ and $E(a2)$ one can get $E(a1 + a2)$. It can also be used to find $E(a1) * a3$, where $a3$ is not encrypted. We use the ability of the Paillier encryption to increment the count (number of times a query has been issued against which documents are downloaded) in the encrypted query click logs. We also use Paillier encryption for encrypting the term frequency of the keywords and their squares in the Index.

3 Encrypted Query Click Logs in CSMRS

Query logs are extremely valuable for information acquisition. Ranging from simple statistics to deeper mining, query logs can be used to get a variety of information like user's search preferences, automatic generation of natural language resources, semantic analysis etc. [23]. To achieve all this, one of the most important steps of query processing is segmentation of the query into terms. If the query logs contains the URL or reference of the document opened or clicked by the user then they are referred to as query click logs.

The proposed scheme CSMRS uses query click logs, but because it is deployed in a public cloud environment the query click logs used are encrypted. Further, because we need to segment the query into terms, the query encryption is done word-wise/term-wise. The encrypted query click logs are maintained on the public cloud server, which can cause a threat in spite of encryption of the query terms. Also, encryption of the query term-wise can allow statistical attacks reveal the pattern of search. Even the most commonly and frequently issued queries by an organization/user may be guessed. Therefore, simply encrypting them with a deterministic encryption algorithm could result in failure in preserving the privacy of the search terms in a cloud environment. Also, if we use non-deterministic encryption algorithms, we would not be able to align the queries against each other. So, we need a mechanism which allows aligning the

query terms without actually making the pattern reveal the plaintext. To overcome, the problem of retrieving the plain text through crypto attacks, CSMRS introduces and uses word vectors. It further encrypts the obtained word vectors with ciphers from the table $\mathbb{W}V$ by lookup method. The word vectors obtained for every word is different even if they have the same set of alphabets as shown in the Sect. 4.1. This uniqueness provided in the formation of word vectors makes the guessing of the plain text very difficult, as it will be needed to be done for every word. CSMRS makes the encryption of plain text even stronger as after obtaining the word vectors they are further encrypted with corresponding cipher from table $\mathbb{W}V$. CSMRS provides protection against guessing of plaintext by infusing dummy words [24] in the query and in the index.

4 Important Notion

4.1 Word Vectors

Word vector is a bit string build up with key *M*. The key *M* has the random English alphabets. It is ensured that the key has the complete alphabet set, the alphabets are repeated many times and the alphabets like 'i', 'e', 'a' and others which are more in use in the formation of English words have the repetition number higher than rest of the alphabets.

If we consider, English alphabets to have letters 'a to f' with 'a' being used in most of the English words constructed over the set {'a', 'f'} then a snippet of column matrix could be as shown in Fig. 2.

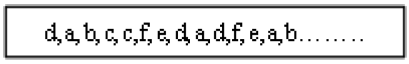

Fig. 2. Snippet of the *1D* matrix (key *M*) on which the word vectors are built

Example: Over the set {'a', 'f'} with key *M* as in Fig. 2, for getting the word vector for the word "bed" check the *1D* matrix (key *M*), pick first character of the word "bed" i.e. 'b' and compare with *1D* matrix since first character is 'd' in *1D* matrix set it to 0, next is 'a' set it to 0, next is 'b' set it to 1. Once the bit is set 1, pick next character from the given word i.e. 'e' here. From the point last left, start checking the *1D* matrix. Keep setting all bits to 0 till a match is found so after setting three bits to 0, next bit is set to 1 for character 'e'. Now the bit is set to 1, so pick next character i.e. 'd' from the given word and start checking from the point last left. Next character is 'd' in *1D* matrix set it to 1. Since the bit has been set to 1, so pick next character from the word. We do not have any more characters left, so set remaining all bits from the point last left to 0. Hence the obtained word vector for the word "bed" is 00100011000000.

Similarly the word vectors for the words "bad", "dad", "bed", and "bead" based on the snippet of this matrix will be

$$\text{bad} = 00100000110000,\ \text{dad} = 11000001000000,$$
$$\text{bed} = 00100011000000,\ \text{bead} = 00100010110000$$

Obtained word vectors are further treated to ensure cryptographic strength by shifting it around by a shift factor $\check{S}$. In CSMRS, mid-point is taken as the shift factor. So, the word vector 00100011000000 for the word "bed" after shifting becomes 10000000010001.

4.2 $\not{W}V$ Table

During setup phase (offline phase) we build a $\not{W}V$ table, wherein, we record prebuilt word vectors (of words and dummy words) and their equivalent ciphers with a non-deterministic cipher (AES in our case) with a random key S. We limit the use of non-deterministic cipher in a manner, that we encrypt it and record only one of the random cipher obtained for a word vector in the $\not{W}V$ table. The entire table is built during the setup phase and uploaded on the private cloud server. The presence of key M and $\not{W}V$ table at different locations provides security to the final cipher obtained and its link to the plain text. Therefore, having the key M alone does not reveal the final cipher being uploaded on the public cloud.

4.3 Encrypted Query Click Logs

Query logs are widely used in web search. As Wei et al. in [25] refers that search with synonyms is a challenging task in web as it could lead to intent drift to user's search. Wei et al. in [25] gives a solution to this issue in web search by the analysis of co-clicked queries and further alignment of these queries against each other, where the co-clicked queries are the queries leading to clicking the same documents. So, with a different methodology but similar idea, we have used encrypted query click logs. These encrypted query click logs need to be aligned word by word against each other. Therefore, their encryption as discussed, need to be word-wise or term-wise. To align the queries while preserving the privacy of the search terms, CSMRS uses word vectors for every unique word of index file $\hat{I}(M)$ and queries $t(M)$. These word vectors are created at the user's end with the help of a unique key M shared with him by the data owner. Once generated $\hat{I}(M)$ is sent to the private cloud where they are further encrypted by key S to form $\hat{I}(M, S)$. The key M gives a unique fingerprint to each word. The queries are also encrypted by keys M and S to generate $t(M, S)$.

We store this information, without feeling a threat because otherwise also, a cloud server is capable of having a history of all the search process and collect this information. To break the direct linkage, between the issued query and its recorded entry in $\bar{Q}$ in the public cloud server, the log for a file download is recorded in the private cloud server. At a suitable periodic interval i.e. depending on the number of queries being issued, these logs are written to the public cloud server.

5 Proposed CSMRS Overview

CSMRS architecture comprises of four entities data owner, data user, trusted third party (Private cloud in this case) and public cloud.

- ***Data Owner***: Data owner is responsible for outsourcing a collection of documents $D = (D1, D2, D3,\ldots)$ in an encrypted form $C = (C1, C2, C3,\ldots.)$ to the cloud server. He encrypts the document collection D with a symmetric key encryption AES with key K. In order to perform multi-keyword ranked search over the outsourced encrypted data, the data owner builds searchable index $\hat{I}(M)$. $\hat{I}(M)$ is then sent to the private cloud where all word vectors $\hat{I}(M)$ are replaced with ciphers from $\not{W}V$ to form $\hat{I}(M, S)$ by lookup method. This is done for every document to be uploaded.
- ***Data User***: Data users are authorized users who can securely search the document collection for keyword(s). They use the secret key which is shared with them by the data owner and can issue a search request. The search query $t(M)$ issued by the data user is encrypted with the same key M with which index keywords were encrypted.
- ***Private Cloud***: The private cloud server on receiving the search query $t(M)$ assigns weight of 2 to the query keywords. It further checks the synonym dictionary $\S Đ$ and add all synonyms of the query keywords with weight 1. Then it extends the encrypted query with dummy keywords with weight 0 and further encrypts it with the key S to get $t(M, S)$. The dummy keywords are randomly added to the query to protect against statistical attacks of guessing the more frequently used terms. It then sends the search $t(M, S)$ to the public cloud server.
- ***Public Cloud***: Public cloud in the offline stage constructs the collocation dictionary $€Đ$ by calling the AlignQuery() procedure that works on the master encrypted query click log $\bar{Q}$. In online stage the public cloud server on receiving the search query $t(M, S)$, refines the synonyms of $t(M, S)$ with procedure SynSelection(), find the cosine similarity and returns a ranked list of top n encrypted documents to the data user. The data user can download the files of his intent and then decrypt it with the key K secretly shared with him by the data owner.

6 Proposed CSMRS Framework

This section gives a detailed description of CSMRS with integrated security and privacy mechanisms.

6.1 Build_Index()

Key_Gen(m)
With the size parameter m, generate the secret key M where M is $1D$ matrix of size m, $M \in \{a, z\}$. M contains many non-uniform occurrences of the characters 'a–z' and it is not in a collating sequence but a random order.

Algorithm 1: Build Word Vector.

Procedure Build_Word_Vector(M,WD).

Initialization: Extract the distinct words from the file, file name and the file descriptor (these are the words that acts as tag for the document, and are entered by the data owner while uploading the document) as WD.
for each WD as wd
  w=splitword(wd)
  for each w as wi
    if first occurrence
    set wi =1
    else
    set 0
    end if
  endfor
  $\hat{I}$(M) = $\hat{I}$(M) + Paillier Encrypted(Term Frequency, square (Term Frequency))
endfor
return Î(M)
end procedure

Algorithm 2: Build_Vec(Î(M),₩V).

Procedure: Build_Vec(Î(M), ₩V).

Initialization: extract the distinct word vectors from $\hat{I}$(M) as *EW*
for each *EW* as *ew*
  f= Paillier Encrypted(Term Frequency, square (Term Frequency))
  Lookup ₩V
  Find corresponding cipher CR
  Î(M,S) =CR , f
endfor
Randomly add into Î(M,S) dummy keywords with Paillier encrypted (term frequency 0)
return Î(M,S)
end procedure

6.2 Dictionaries

Synonym Dictionary (*SD*)
SD is built during offline stage and kept on the private cloud. It contains commonly used synonyms for a word in all grammatical usages of a word. We used Moby Thesaurus [26] for picking the synonyms. All entries in *SD* are encrypted with key *M* and *S*.

Collocation Dictionary (€Đ)

€Đ is an *M*, *S* keys encrypted dictionary having the aligned words and the words co-occurring with these aligned words (neighbour words). AlignQuery() procedure is used to align the queries available in the co-clicked query clusters. The detailed process is as explained below:

Master Query Click Log $\bar{Q}$

Every time a user download a document for a particular query it is recorded in a log on the private cloud. All these logs from private cloud server are appended into the master query click log $\bar{Q}$ after a periodic interval. The appropriate period for update of $\bar{Q}$ depends upon the usage of the system. However, regular updating of $\bar{Q}$ gives an efficient execution of the scheme. $\bar{Q}$ contains the encrypted query *t(M, S)* issued by the user, doc id of the document that was downloaded and the count n (number of times the same document was downloaded for the given query). *n* is encrypted with Paillier encryption that is additively homomorphic so that n is not revealed to the cloud server however n could be incremented on appending more entries to $\bar{Q}$.

Co-clicked Query Clustering

Co-clicked query clusters are greedily formed by first reserving all the doc-ids that are linked to a particular query and then putting into cluster all the queries that were issued with which these doc ids are linked.

Query Pair Alignment

CSMRS aligns the query pairs in a manner similar to [25]. It selects the co-clicked query pairs with similar length i.e. same number of terms. These pairs have same terms at all positions except one in order to extract the words that are replacement for each other. These extracted words become the aligned words that are recorded in the collocation dictionary along with the neighbouring keywords.

Algorithm 3: FormCluster($\bar{Q}$).

Procedure: FormCluster($\bar{Q}$).

```
c=1
A:
for each docid in Q̄ as d
   Store into cluster c,(t(M,S), docid) for d
   for each t(M,S)in c extract docid
      goto A
   endfor
endfor
increment c
end procedure
```

Algorithm 4: AlignQuery().

Procedure: AlignQuery().

```
call cluster()
for each c
   pick t(M,S)with equal number of keywords as c
   n=count (t)
   for i = 1 to n
      for j = (i+1) to n
      if weight not 0 and only one keyword is different in corresponding t
         add into €Ð, differing keywords as main_word and aligned_word
         add the similar words of t as neighbor words.
      end if
      endfor
   endfor
endfor
return €Ð
end procedure
```

// €Ð composition= (main_word, aligned_word, neigbour words)

Algorithm 5: SynSelection(t(M,S), €Ð).

Procedure: SynSelection(t(M,S), €Ð).

```
Initialization: extract the distinct word with weight 2 as MW and weight 1 as SW
for each MW as mw
   Lookup €Ð
   Store into aw aligned words for mw
endfor
for each SW as sw
   if sw in aw then
   keep sw in t(M,S)
   else
   drop sw
endfor
return t(M,S)
end procedure
```

6.3 Search

$\mathbb{WV}$ and $\mathbb{SD}$ in the private cloud requires one time build up during the setup phase of CSMRS. Figure 6 shows the total time required to build cipher dictionary and synonym dictionary.

$\mathbb{CD}$ requires time to time update to acquire more suitable synonyms. During experiments it was seen that if collocation dictionary is not updated it resulted in less number of matched synonyms. The only periodic computation cost incurred by CSMRS is aligning of queries and updating of collocation dictionary.

The search process requires users' encrypted query *t(M, S)* to be sent to the public cloud server. Here it looks the collocation dictionary and matches the aligned words using procedure SynSelection().This process filters out all the excess synonyms added in the query. The filtered out synonym terms are those which actually were never used in the users' query simply because the use of those words together is inappropriate. Now finally we are left with the main keywords, shortlisted synonyms and dummy keywords in the *t(M, S)*. We calculate the cosine similarity using Eq. 1, and find the relevant documents. In Eq. 1, Q refers to the final query *t(M, S)* after synonym selection phase and D refers to the stored documents. The top n relevant documents returned are ranked as per their similarity scores and returned to the user. The presence of dummy words in the query and in the index file does not affect the final scores as their term frequency was set to 0 in index file. In the entire search process only the final score are known to the cloud server. CSMRS preserves the privacy of entire data and process from the cloud server.

$$similarity = \cos(\theta) = \frac{D \cdot Q}{||D||_2 \cdot ||Q||_2} = \frac{\sum_{i=1}^{n} D_i Q_i}{\sqrt{\sum_{i=1}^{n} D_i^2} \cdot \sqrt{\sum_{i=1}^{n} Q_i^2}} \quad (1)$$

7 Experimental Results

The experiments are done in python on a Linux machine with Intel i7 processor with 8 GB RAM. For alignment of encrypted query click logs the encrypted queries were built for the search queries in accordance with selected queries from AOL-user-ct-collection [27] a corpus of query logs. We ran the word alignment algorithm on both the selected queries in plain and then on the encrypted queries of the same queries to check the similarity in the aligned words returned by them. We found that the words aligned are almost similar with both. Though with encrypted queries we only had some extra dummy keywords almost everywhere in order to conceal the actual data. The dataset of documents was specially built to make the documents suitable to run and use the queries corpus from AOL-user-ct-collection.

8 Functionality and Efficiency

8.1 Index Generation

Index generation step in CSMRS is one time computation comprising of two main steps i.e. forming $\hat{I}(M)$ and its further encryption to $\hat{I}(M,S)$. In CMSRS we include word vectors of the unique words from the document to be uploaded, unique keywords of its filename and file descriptors (these acts as document tags and are entered by the data owner while uploading the file/document). We include the term frequency of these words and their squares and encrypt them with Paillier encryption.

For constructing $\hat{I}(M)$ key M is built with English alphabets randomly placed, they are repeated many times and the frequency of most appearing alphabets in formation of words in English language is more than other alphabets. The length for M used in our scheme is 167. The randomness and repetition of alphabets is good enough to construct many English words. We experimented creating word vectors and careful examination of 10,000 English words picked from the /usr/share/ dict/british-english of Ubuntu 14.04.

Figure 3 shows the index construction time. It shows the word vector creation time and time taken for final encryption of the index by lookup method into the ~~HV~~ table. It shows that the time taken for final encryption of the word vector in private cloud is almost negligible with respect to the total time taken for index construction. The generation time of the index is increasing linearly with respect to the number of keywords.

Fig. 3. Index construction *time (in secs.)*

The total index construction time in Fig. 3 includes time taken for extracting the unique keywords for the relevant document along with its term frequency and its square, word vector creation time and time taken for encryption by lookup method in $\mathcal{WV}$ table.

9 Search over Encrypted Data

9.1 Search Efficiency

The public cloud server computes the similarity scores with Eq. 1, and return the top n ranked list of the relevant documents during the search process. Figure 4 shows the time taken by CSMRS for searching the relevant files. It clearly shows assuming that having fixed number of keywords in the query, the search time is dependent on the number of documents in the dataset.

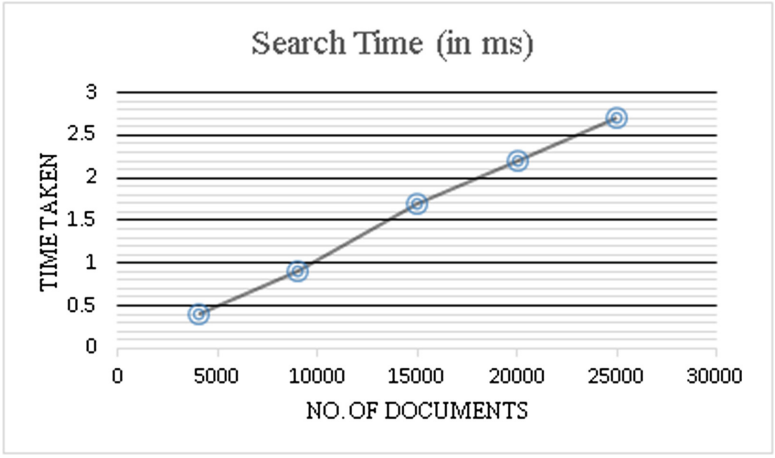

Fig. 4. Search *time (in ms)*

9.2 Synonym Quality

We experimented with some selected search queries and saw that our scheme did not give weight to "powerful" as synonym when it was listed with the word "coffee". Similarly we tried another query "ship goods to Bombay" over here the word "boat" was not given weight as synonym for the word "ship". One detailed example of the query pair alignment and reduction of synonyms from the full list of synonyms is shown in the Table 1.

Table 1. Example of reducing synonym terms by CSMRS with given queries to be aligned for the word sharp

Query	Woman with sharp tongue
Example word	Sharp
Complete synonym list	Able, acerbic, acidic, acidulous, acrid, adept, adroit, agile, annoying, artful, astringent, biting, brainy, brilliant, capable, caustic, clever, consummate, cool, cultivated, cutting, deft, dexterous, experienced, expert, effectual, effective, efficient, facile, galling, gifted, harsh, hateful, hurtful, ingenious, intelligent, keen, learned, masterful, masterly, nasty, practiced, piquant, polished, powerful, prepared, proficient, pungent, qualified, responsible, rough, sharp, savvy, skilled, skillful, smart, spiky, talented, tart, accomplished
Aligned queries in master query click log	Woman with loose tongue, woman with red tongue, woman with cut tongue, woman with pierced tongue, woman with cursing tongue, woman with sarcastic tongue, woman with rough tongue, woman with pleasant tongue, woman with pungent tongue, woman with biting tongue
Synonyms filtered and selected for example word	Sarcastic, rough, pungent, biting

10 Conclusion

In this paper, we addressed the problem of privacy preserving context sensitive synonym support over the multi-keyword ranked search over the encrypted cloud data by exploiting the encrypted query click logs. The proposed Context sensitive Multi-keyword Ranked Search (CSMRS) takes care of preserving the privacy of the user data. CSMRS uses and stores encrypted query click logs but does not reveal the exact number of keywords in the query to the public cloud server and befools it by extending the users' encrypted query with dummy keywords. This makes CSMRS secure against mapping of the cipher with a particular keyword. Final encryption of users' data (either index keywords or query keywords) is possible with cipher table in private cloud server after getting word vectors from the data user. This prevents unauthorized users to make search requests. Also the word vector design of each word provides a strong safety feature against crypto attacks. Though encrypted query click logs gives some sense of presence of similar keywords but it leaks nothing more than that, as the cloud server cannot reach to the exact plain keywords from it. Also the cloud servers are curious enough to analyze user's queries and can even get the history so our query click logs stores nothing beyond the capability of the public cloud server. Rather we are using it for our benefit. We also take care that these query click logs are first stored on the private cloud server and later it updates the public cloud server master query click logs. It does so to hide the exact relevance of the encrypted queries issued with its encrypted

query click log entry. CSMRS shows improvement in the quality of the synonym selection for the expansion of search query and hence the ranked search results contains relevant documents. The search time taken by CSMRS is similar to the existing schemes.

References

1. Rightscale 2016 State of the Cloud Report - Hybrid Cloud Adoption Ramps as Cloud Users and Cloud Providers Mature (2016). http://assets.rightscale.com/uploads/pdfs/RightScale-2016-State-of-the-Cloud-Report.Pdf
2. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy (2000)
3. Goh, E.-J.: Secure Indexes*. Cryptology ePrint Archive: Report 2003/216 (2004)
4. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
6. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
7. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
8. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: ACM CC 2006 (2006)
9. Brinkman, R: Searching in encrypted data. Ph.D. thesis. University of Twente (2007)
10. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: Proceedings of ICDCS 2010 (2010)
11. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of IEEE INFOCOM, pp. 829–837 (2011)
12. Ahsan, M.M., Chowdhury, F.Z., Sabilah, M., Wahab, A.W.B.A., Idris, M.Y.I.B.: An efficient fuzzy keyword matching technique for searching through encrypted cloud data. In: International Conference on Research and Innovation in Information Systems (ICRIIS) (2017)
13. Yang, C., Zhang, W., Xu, J., Xu, J., Yu, N.: A fast privacy-preserving multi-keyword search scheme on cloud data. In: International Conference on Computing & Processing (Hardware/Software), pp. 104–110 (2012)
14. Xu, Z., Kang, W., Li, R., Yow, K., Xu, C.Z.: Efficient multi-keyword ranked query on encrypted data in the cloud. In: ICPADS 2012, pp. 244–251 (2012)
15. Xu, J., Zhang, W., Yang, C., Xu, J., Yu, N.: Two-step-ranking secure multi-keyword search over encrypted cloud data. In: International Conference on Computing & Processing (Hardware/Software), pp. 124–130 (2012)
16. Yang, C., Zhang, W., Xu, J., Xu, J., Yu, N.: A fast privacy-preserving multi-keyword search scheme on cloud data. In: International Conference on Computing & Processing (Hardware/Software), pp. 104–110 (2012)

17. Handa, R., Challa, R.K.: A cluster based multi-keyword search on outsourced encrypted cloud data. In: 2nd IEEE International Conference on Computing for Sustainable Global Development, pp. 115–120 (2015)
18. Krishna, C.R., Handa, R.: Dynamic cluster based privacy-preserving multi-keyword search over encrypted cloud data. In: 6th IEEE International Conference on Cloud System and Big Data Engineering, pp. 146–151 (2016)
19. Khan, N.S., Krishna, C.R., Khurana, A.: Secure ranked fuzzy multi-keyword search over outsourced encrypted cloud data. In: 5th IEEE International Conference on Computer and Communication Technology, pp. 241–249 (2014)
20. Fu, Z., Sun, X., Linge, N., Zhou, L.: Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Trans. Consum. Electron. **60**(1), 164–172 (2014)
21. Krishna, C.R., Mittal, S.A.: Privacy preserving synonym based fuzzy multi-keyword ranked search over encrypted cloud data. In: International Conference on Computing, Communication and Automation (ICCCA2016), pp. 1187–1194 (2016)
22. Saini, V., Challa, R.K., Khan, N.S.: An efficient multi-keyword synonym-based fuzzy ranked search over outsourced encrypted cloud data. In: Choudhary, R.K., Mandal, J.K., Auluck, N., Nagarajaram, H.A. (eds.) Advanced Computing and Communication Technologies. AISC, vol. 452, pp. 433–441. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-1023-1_43
23. Medelyan, O.: Why not use query logs as corpora? In: Ninth ESSLLI Student Session, pp. 1–10 (2004)
24. Liu, C., Zhu, L., Wang, M., Tan, Y.: Search pattern leakage in searchable encryption: attacks and new construction. J. Inf. Sci.: Int. J. **265**, 176–188 (2014)
25. Wei, X., Peng, F., Tseng, H., Lu, Y., Wang, X., Dumoulin, B.: Search with synonyms: problems and solutions. In: Coling 2010, Poster Volume, Beijing, pp. 1318–1326 (2010)
26. (2017). http://moby-thesaurus.org/
27. (2017). http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection