Concept-based Semantic Search over Encrypted Cloud Data

Fateh Boucenna[1,2], Omar Nouali[1] and Samir Kechid[2]
[1]*Security Division, CERIST: Research Center on Scientific and Technical Information, Algiers, Algeria*
[2]*LRIA, USTHB: University of Sciences and Technology Houari Boumediene, Algiers, Algeria*

Keywords: Cloud Computing, Searchable Encryption, Data Privacy, Weighting Formula, Concept-based Retrieval, Semantic Search, Ontology.

Abstract: Cloud computing is a technology that allows companies and individuals to outsource their data and their applications. The aim is to take advantage from the power of storage and processing offered by such technology. However, in order to preserve data privacy, it is crucial that all data must be encrypted before being outsourced into the cloud. Moreover, authorized users should be able to recover their outsourced data. This process can be complicated due to the fact that data are encrypted. The traditional information retrieval systems only work over data in the clear. Therefore, dedicated information retrieval systems were developed to deal with the encrypted cloud data. Several kinds of search over cloud data have been proposed in the literature such as Boolean search, multi-keyword ranked search and fuzzy search. However, the semantic search is little addressed in the literature. In this paper, we propose an approach called SSE-S that take into account the semantic search in the cloud by using Wikipedia ontology to understand the meaning of documents and queries with maintaining the security and the privacy issues.

1 INTRODUCTION

Cloud computing is a technology that allows companies and individuals to outsource their data to a remote server. This technology is increasingly used since its appearance. This is justified by the large storage space and the enormous computational power offered to users.

The data outsourced to the cloud are usually sensitive and confidential (photos, emails, financial documents, etc.). The outsourced data must be protected against possible external attacks and the cloud server itself. For that, it is necessary to encrypt them by the data owner before sending them to the cloud server.

Users tend to take advantage of the large storage space offered by the cloud to store a huge number of documents. However, This can complicate the user's task to retrieve a specific document. To overcome this problem, the use of an information retrieval system (IRS) becomes necessary into a cloud server.

Considering that the data hosted in the cloud server are encrypted, therefore the classical information retrieval is not feasible. For this reason that many searchable encryption schemes have been proposed in the literature (Song et al., 2000), (Curtmola et al., 2006).

The common point between these approaches is that the user sends an encrypted query (trapdoor) to the cloud server, upon receiving this query, the server searches into a collection of encrypted documents (represented by an encrypted index) and returns to the user a subset of relevant documents. However, it is crucial that the search should not cause any information leakage.

The first works that have been proposed in the literature only support single keyword search (Song et al., 2000). The downside is that a user cannot properly express his information need. Consequently, the precision of the search is reduced.

To improve the search accuracy, Boolean search over encrypted data have been proposed in the literature (Ballard et al., 2005). However, this improvement still insufficient, given that building Boolean queries by an inexperienced user is a difficult task.

After that, other works (Xu et al., 2012), (Li et al., 2013), (Yu et al., 2013), (Wang et al., 2014), (Cao et al., 2014) have turned to the use of several techniques known in the information retrieval (IR) area as the weighting formulas, similarity scores, vector representation, etc.

It is noticed that the vast majority of the schemes proposed in the literature is merely a syntactic search.

Boucenna, F., Nouali, O. and Kechid, S.
Concept-based Semantic Search over Encrypted Cloud Data.
In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 2*, pages 235-242
ISBN: 978-989-758-186-1

These schemes are based only on the keywords of the query sent by the user and returning documents containing the query terms. However, this is not always the best way to perform a search. The downside is that if a user does not select the appropriate keywords of the query, the server would not return the most pertinent documents. Indeed, the server ignores every document not containing at least one query term, even if it has a meaning close to that of the query. Consequently, the search is not optimal. To overcome this problem, it is necessary to introduce a semantic search over encrypted cloud data.

There are few works in the literature that have tried to address this problem by proposing semantic search approaches. (Sun et al., 2013), (Yang, 2015) have proposed approaches that exploit the technique of expansion of the query (a single term query) by inserting synonyms of the query term. These approaches have not solved the problems previously posed. Their limit is that they do not use external resources such as ontologies and thesauri. In addition, except synonymy, they do not exploit relationships between terms (associative relation, homonym, instance-of relation, related term, etc.).

In this paper, we present our proposed scheme. The goal is to solve the problems mentioned above by performing a semantic search over encrypted cloud data in which an external resource (Wikipedia ontology) is exploited. In addition, we will introduce an improved version of our approach by proposing a new weighting formula. Furthermore, an experimental study validates our proposed approach.

2 PROBLEM FORMULATION

2.1 Toward a Semantic Search

The majority of encryption searchable schemes over cloud data proposed in the literature performs a keyword-based search. Indeed, during the search process, when the server receives a query, it tries to find documents containing the query terms. Documents not containing any query term will not be returned despite they can be relevant.

Therefore, to get the more relevant documents, the user is obliged to choose the right keywords when formulating his query. However, this is not always easy, especially for an inexperienced user. Consequently, the search may become a tedious task for the system users. In addition, many relevant documents not containing any query term will not be returned to the user.

To illustrate the problem, let us take the following example: Assuming we have two short documents[1], the first document deals with the London Stock Exchange[2]; whereas, the second one is about the England football team[3].

Document 1. *The London Stock Exchange is a stock exchange located in the City of London in the United Kingdom. As of December 2014, the Exchange had a market capitalization of US$6.06 trillion, making it the third-largest stock exchange in the world by this measurement.*

Document 2. *The England national football team represents England and the Crown Dependencies of Jersey, Guernsey and the Isle of Man for football matches as part of FIFA-authorised events, and is controlled by The Football Association, the governing body for football in England.*

If a user sends the query *Economy of England*, the server will search for documents containing the terms *Economy* and / or *England* in the documents collection. The server will surely find that the first document does not contain any of these terms, so it ignores this document. Contrariwise, it will find that the second document contains the term *England*, so it will return it. However, if we analyze the content of the two documents, we will notice that the first document is relevant, since its meaning is close to that of the query, given that it talks about the London stock exchange which is strongly related to the economy of England. Contrary to the second document that talks about football in England and has no relationship with economy. Therefore, this document is not supposed to be relevant even if it has terms in common with the query.

In order to solve the problem that we have faced in the syntactic search. IR community has turned to the use of techniques exploited in natural language processing. Indeed, they have exploited external resources such as thesauri and ontologies in order to understand the meaning of the queries sent by the users. The goal is to improve the precision and recall of the search by returning documents that have a meaning close to that of the query rather than relying on the syntax. This area of research is called semantic information retrieval.

To the best of our knowledge, very few studies (Sun et al., 2013), (Yang, 2015) have exploited the semantic information retrieval over encrypted cloud data. These works are based on the query expansion technique by adding the synonyms of the query term. The drawback of these schemes is that except the syn-

[1] Extracted from Wikipedia

[2] https://en.wikipedia.org/wiki/London_Stock_Exchange

[3] https://en.wikipedia.org/wiki/England_national_football_team

onymy, other relationships between terms such as the associative relationship, homonym, instance-of relationship, related term, etc. are not exploited. In summary, those techniques allow to improve recall, but they are still far from the real semantic search.

Among the fields of semantic IR, we find, contextual IR, personalized IR, conceptual IR etc. Contrary to other fields of semantic search, machine learning and user profile are not used in conceptual IR. Therefore, the server can learn nothing neither about the user's interest nor about the documents collection. Consequently, the conceptual IR is the most appropriate for the realization of an encrypted semantic search scheme in cloud computing.

A concept is an idea grouping in the same category, objects semantically close to each other (e.g. stock market, economy, finance, currency). Conceptual IR is based on concepts rather than keywords in the indexing and matching process. Therefore, it is necessary to use external resources such as ontologies to achieve a mapping between keywords and concepts. Conceptual IR allows to detach from the syntactic aspect and go near the natural language. Consequently, it is possible to perform a semantic guided search rather than relying on the syntax of the query.

2.2 Threat Model

Security is a crucial aspect in cloud computing given that the outsourced data are often personal or professional. The cloud server is exposed to all kinds of external attacks. Hence, it is necessary that every data (document, Index, query) will be encrypted before sending it to the server.

In addition to that, the cloud server itself is curious and it can collect information about the content of documents by statistical analyzes. Hence, the search process should be secure and have to protect the data privacy.

When designing a search scheme over encrypted data, it is important to take into account the threats discussed below. For this reason, security constraints were elaborated by the IR community (Cao et al., 2014), (Li et al., 2013).

Protected Content. It is necessary that all data flowing through the cloud server will be encrypted.

Keyword Privacy. The proposed scheme must be able to hide to the server *the term distribution* (The frequency of a given term in each document of the collection) and *the inter-distribution* (The distribution of scores of terms in a given document). This is in order to prevent the server to make a link between a set of terms and a document.

Trapdoor Unlinkability. The proposed scheme must be able to prevent the server to deduce the relationship between a given set of encrypted queries. Therefore, it is necessary that the encryption of a query will be random enough.

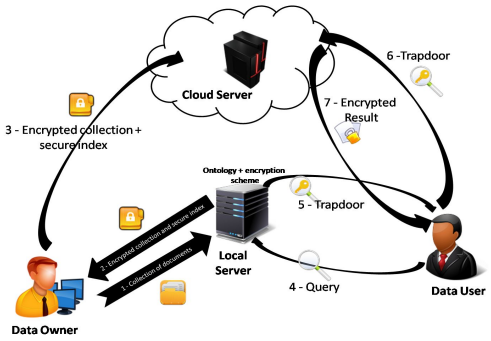

Figure 1: System model.

Search Pattern. The proposed scheme must be able to hide to the server the sequence of results returned to a user during the search.

2.3 System Model

Our proposed scheme uses an ontology during the indexing process. Indeed, after the creation of the index, each document will be represented by a vector of terms. From those vectors, the data owner can construct a concepts vector for each document using the ontology. The whole concepts vectors corresponds to the conceptual index of the collection. After creating the conceptual index, both the collection and the index will be encrypted and sent to the cloud. During the search, an authorized user have to formulate a query. Then, the concepts vector of the query will be created using the ontology. After that, it will be encrypted before sending it to the server. Upon receiving the encrypted query, the server calculates the scalar product between each document vector and the query vector. Finally, the server returns the most relevant documents to the user (Figure 1).

2.4 Design Goals

Our goal is to propose a semantic searchable scheme over encrypted cloud data. For that, an ontology has been exploited during the indexing process of the documents and the queries.

Two majors contributions have been proposed in our work:

1. Exploiting the semantic search over encrypted cloud data.
2. Proposal of a new weighting formula to solve the problem posed in (Egozi et al., 2011) (see section 3.2).

3 THE PROPOSED SCHEME (SSE-S)

In this section, we present the Semantic Searchable Encryption Scheme (SSE-S) that we have proposed. For this, we first explain the ontology used in our scheme. Then, we present the new weighting formula that we have proposed. After that, we present the encryption method exploited in the SSE-S approach. Finally, we present the details of the proposed scheme.

3.1 Wikipedia as Ontology

In order to understand the meaning of queries and documents many researchers have operated external resources such as dictionaries, thesauri, semantic graphs and ontologies. In our work, we opted for the use of an ontology due to its robustness and reliability.

More precisely, we decided to use Wikipedia as ontology. The choice of Wikipedia was guided by its great richness of information given that it contains more than four (4) million pages, in addition it contains articles in all areas and most languages.

Lot of works have exploited Wikipedia as ontology in order to calculate the semantic similarity between two given texts (Gabrilovich and Markovitch, 2006), (Egozi et al., 2011). Our scheme is based on Gabrilovich's approach (Gabrilovich and Markovitch, 2006) where the Wikipedia ontology is constructed as follows:

1. Each Wikipedia page P_i corresponds to a concept C_i (e.g. Data mining, Financial crisis).
2. Each concept C_i is represented by a vector of terms $V_i = \{(T_1, W_{i1}), (T_2, W_{i2}), ..., (T_n, W_{in})\}$ extracted from the corresponding Wikipedia article. These terms are weighted using the TFIDF formula.

   The weight W_{ij} of a term T_j in the vector V_i corresponds to the association degree between the term T_j and the concept C_i.
3. In order to accelerate the similarity calculation process, an inverted index I_{wiki} is constructed where each term T_i is represented by a set of concepts V'_i to which it belongs, $V'_i = \{(C_1, W_{1i}), (C_2, W_{2i}), ..., (C_m, W_{mi})\}$.
4. The inverted index $I_{wiki} = \{V'_1, V'_2, V'_3, ..., V'_n\}$ which is constructed of the set of concepts vectors corresponds to Wikipedia ontology.

Before calculating the similarity between two documents, each of them must first be represented by a vector of concepts as follows:

1. At first, a vector of terms $D_i = \{(T_1, W'_{i1}), (T_2, W'_{i2}), ..., (T_n, W'_{in})\}$ must be constructed for each document d_i using the TFIDF formula.
2. Then, from the vector D_i, a vector of concepts $D'_i = \{(C_1, S'_{i1}), (C_2, S'_{i2}), ..., (C_m, S'_{im})\}$ will be calculated by mapping between terms and concepts through the Wikipedia ontology.
3. The score S'_{ij} assigned to a concept C_j in the concepts vector D'_i is calculated by the following formula:

$$S'_{ij} = \sum_{T_k \in d_i} W'_{ik}.W_{jk} \quad (1)$$

   where W'_{ik} is the weight of a term T_k belonging to the document d_i and W_{jk} is the association degree between the term T_k and the concept C_j.
4. After that, Each document will be represented by the top X ($X = 100$ is a good value) concepts that have the highest scores.
5. Finally, the similarity between the two documents is calculated by applying the scalar product between the two concepts vectors.

To implement our proposed scheme, We have constructed an ontology based on a version of Wikipedia dated 12-Mar-2015 containing 4,828,395 pages.

3.2 Double Score Weighting Formula

Conceptual IR allows users to find relevant documents even if they do not contain query terms or their synonyms. This is explained by the fact that the search is guided by the meaning through the use of an ontology.

Let us take the example given in (Egozi et al., 2011): suppose that a user sends the query *shipwreck salvaging treasure* and that the collection contains the document entitled *Ancient Artifacts Found* below:

Ancient Artifacts Found. *Divers have recovered artifacts lying underwater for more than 2,000 years in the wreck of a Roman ship that sank in the Gulf of Baratti, 12 miles off the island of Elba, newspapers reported Saturday.*

A keyword-based search cannot find the document above given that it has not any term in common with the query. However, with the conceptual IR, this document will be returned to the user given that the document vector has concepts in common with the query vector.

Nevertheless, it happens that a concept based search returns documents containing terms in common with a query despite they are not relevant. To illustrate that, an example was given in (Egozi et al., 2011): if a user sends the query *Estonia economy* and

the collection contains the document entitled *Olympic News In Brief* below:

Olympic News in Brief. *Cycling win for Estonia. Erika Salumae won Estonia's first Olympic gold when retaining the women's cycling individual sprint title she won four years ago in Seoul as a Soviet athlete*

As keyword-based search, concept-based search cannot ignore this document even if it is not relevant. That is justified by the high frequency of the term *Estonia* in the document and thus the vector representing the document *Olympic News In Brief* contains many concepts associated with the term *Estonia*. Similarly, more than half of the concepts of the vector representing the query are associated with the term *Estonia*. Therefore, there is many common concepts (34 concepts were found in our experimentation) between the document vector and the query vector. Consequently, concept-based search returns the document *Olympic News In Brief* in response to the query *Estonia economy* even if it is assumed not to be relevant.

In order to understand the origin of this problem, we have analysed the concepts representing the document *Olympic News In Brief* and the concepts representing the query *Estonia economy*. We have also analysed the concepts associated with the terms *Economy* and *Estonia* separately.

On the one hand, we have noticed that eight (8) of the top ten concepts representing the document *Olympic News In Brief* are part of the top 10 concepts associated with the term *Estonia*. That is justified by the high frequency of the term *Estonia* in the document *Olympic News In Brief* which increases the scores of the concepts associated with this term when applying the formula (1). On the other hand, even if the frequencies of the query terms are similar, we noticed that the majority of the concepts representing the query are associated with the term *Estonia* rather than the term *Economy*. That is due to the fact that the concepts associated with the term *Estonia* have greater weights, and thus most of these concepts will be selected to represent the query.

In order to represent documents and queries by the most appropriate concepts, we have proposed a new weighting formula that we called *Double Score Weighting Formula* which allows to represent a document (or a query) by a set of concepts strongly associated with the general meaning of the document rather than representing it by concepts associated with terms that have the highest frequencies.

In Wikipedia ontology, each term is associated with a set of concepts. Thus, to represent a document by the most appropriate concepts, our idea is to select the concepts that are associated with the greatest number of terms of the document. For example, if we have the query *Estonia Economy*, for representing this query, it is more advantageous to choose a concept associated with both *Estonia* and *Economy* than choosing a concept only associated with the term *Estonia*, even if the second concept has a greater score.

We have proposed a new weighting formula to be able to represent a document by concepts that are associated with its general meaning. Below we present the steps needed to represent a document by the most appropriate concepts:

1. Construct a weighted terms vector for the document by applying the TFIDF formula.
2. Get all concepts associated with each term of the document vector constructed above by using the Wikipedia ontology.
3. For each of these concepts, attribute two scores as follows:
   (a) The first score is the number of terms (without redundancy) of the document associated with this concept, this score is called the primary score (S_p).
   (b) The second score is the TFIDF weight of the concept in the document, this score is called the secondary score (S_s) and is calculated by the formula 1.
4. Sort the concepts with regard to their primary scores then based on their secondary scores in the case of equality.

$$(S_{p1}, S_{s1}) > (S_{p2}, S_{s2}) \Rightarrow$$
$$(S_{p1} > S_{p2}) \vee ((S_{p1} = S_{p2}) \wedge (S_{s1} > S_{s2})) \quad (2)$$

5. Keep the top Y ($Y = 100$ is a good value) concepts with their scores to represent the document.

We applied our method on the first example to calculate the similarity between the document *Ancient Artifacts Found* and the query *shipwreck salvaging treasure*. We have found that there are thirteen (13) common concepts between the top 100 concepts representing the document and the top 100 concepts representing the query rather than one (1) concept when applying Gabrilovich's method. Thus, as Gabrilovich's method, our method is able to retrieve relevant documents even if they have no term in common with the query. Besides, our method is more efficient than the Gabrilovich's method concerning such documents (13 concepts in our method versus 1 concept in Gabrilovich's method).

Similarly, we applied our method on the second example to calculate the similarity between the document *Olympic News In Brief* and the query *Estonia Economy*. We have not found any common concept

between the top 100 concepts representing the document and the top 100 concepts representing the query rather than thirty-four (34) concepts when applying Gabrilovich's method. Thus, our weighting method has corrected the problem encountered when applying Gabrilovich's method. More precisely, Our method is able to ignore irrelevant documents even if they have terms in common with the query.

3.3 The Encryption Method Used

It is necessary to encrypt the index of the collection (set of concepts vectors representing the documents) built by the data owner as well as users' queries before sending them to the cloud server. The SSE-S scheme that we have proposed uses the same encryption method proposed in (Cao et al., 2014). Our choice was guided by the reliability and the robustness of this encryption method. in addition, the data structure used in our scheme to represent documents and queries (concepts vector) is compatible with this encryption method.

The encryption key proposed in the *MRSE* scheme (Cao et al., 2014) which we used in our SSE-S scheme is composed of one vector S of size $(m+U+1)$ and two $(m+U+1)*(m+U+1)$ invertible matrices $(\{M1,M2\})$, with m is the total number of concepts.

The encryption process is done in three (3) steps (extension, splitting and multiplication) as follows:

1. At first, $U+1$ dimensions are added to each document vector D_i of size m. The value 1 is assigned to the $(m+1)^{th}$ dimension. Whereas, a random value ε_i^j is assigned to the $(m+j+1)^{th}$ dimension (where $j \in [1,U]$). The U last dimensions correspond to dummy keywords.

$$\vec{D_i} = \{D_i, 1, \varepsilon_i^1, \varepsilon_i^2, \varepsilon_i^3, ..., \varepsilon_i^U\}$$

   Moreover, a query vector (which is also of size m) is multiplied by a random parameter r. Then, a dimension with a random value t is added to the obtained vector. After that, U dimensions are added to this vector. a value α^j is assigned to the $(m+j+1)^{th}$ dimension (with $\alpha^j \in \{0,1\}$).

$$\vec{Q} = \{r.Q, t, \alpha^1, \alpha^2, \alpha^3, ..., \alpha^U\} / \alpha^j \in \{0,1\}$$

2. After that, each document vector $\vec{D_i}$ is split into two vectors $\{\vec{D'}_i, \vec{D''}_i\}$, and each query vector $\vec{Q}$ is split into two vectors $\{\vec{Q'}, \vec{Q''}\}$. The vector S is used as a splitting indicator. Indeed, if the j^{th} element of S is equal to 0 then $\vec{D'}_i[j]$ and $\vec{D''}_i[j]$ will have the same value as $\vec{D_i}[j]$ and each of the two elements $\vec{Q'}[j]$ and $\vec{Q''}[j]$ will have a random value such that their sum is equal to $\vec{Q}[j]$. In the case where the j^{th} element of S is equal to 1, we follow the same principle, except that the document vector and the query vector are switched.
3. Finally, both M_1 and M_2 matrices are used to finalize the encryption of each document vector as follows: $I_i = \{M_1^T.\vec{D'}_i, M_2^T.\vec{D''}_i\}$ and for the encryption of each query vector as follows: $T_q = \{M_1^{-1}.\vec{Q'}, M_2^{-1}.\vec{Q''}\}$

When applying the scalar product between a document vector and a query vector we obtain:

$$\begin{aligned} I_i.T_q &= \{M_1^T.\vec{D'}_i, M_2^T.\vec{D''}_i\} \times \{M_1^{-1}.\vec{Q'}, M_2^{-1}.\vec{Q''}\} \\ &= \vec{D'}_i \times \vec{Q'} + \vec{D''}_i \times \vec{Q''} \\ &= \{D_i, 1, \varepsilon_i^1, \varepsilon_i^2, ..., \varepsilon_i^U\} \times \{r.Q, t, \alpha^1, \alpha^2, ..., \alpha^U\} \\ &= r.D_i.Q + \sum_{j=1}^{U} \varepsilon_i^j.\alpha^j + t \end{aligned}$$

The random parameters $\{\varepsilon_i^j, \alpha^j, t, r\}$ are used to hide the real similarity score between a document and a query. However, the alternative similarity scores are useful to sort documents by relevance as has been proved in (Cao et al., 2014).

In our scheme, each document or query is represented by a concepts vector of size m (where m is the total number of concepts). The j^{th} field of the vector is a couple of scores (SP_i^j, SS_i^j) where the first one is the primary score of the concept C_j in the document d_i and the second one represents its secondary score. Thus, in order to the encryption method presented below becomes operational in our approach, it is necessary that the parameters $\varepsilon_i^j, \alpha^j, t$ will be as couple of values. Namely $\varepsilon_i^j = (\varepsilon_i'^j, \varepsilon_i''^j)$, $\alpha^j = (\alpha'^j, \alpha''^j)$ and $t = (t', t'')$ where $\alpha'^j = \alpha''^j$. Whereas, the parameter r still as a single value.

3.4 Semantic Searchable Encryption Scheme (SSE-S)

Our proposed scheme is composed of five (5) functions and two main phases. We start by presenting the five functions of our scheme:

- **KeyGen.** The data owner randomly generates a secret key $SK = \{S, M_1, M_2\}$, where S is a vector of size $(m+U+1)$ and (M_1, M_2) are two invertible matrices of size $(m+U+1) \times (m+U+1)$ (see section 3.3).

- **BuiltOnto.** The ontology is built from Wikipedia. For that, English Wikipedia pages are indexed, where each page is represented by a vector of weighted terms by applying the TFIDF formula; each page corresponds to a concept; an inverted index of Wikipedia I_{wiki} is created where each term is represented by a vector of weighted concepts (see section 3.1).
- **BuiltIndex (F, SK).** At first, a vector of terms is constructed for each document of the collection F by applying the TFIDF formula; then, using the Wikipedia ontology, a vector of concepts is built for each document by applying the double score formula (see section 3.2); finally, each vector of concepts is encrypted by the secret key SK (see Section 3.3). The set of the encrypted vectors constitutes the index I' of the collection F.
- **Trapdoor (W, SK).** At first, a vector of terms is constructed from the query keywords, where the i^{th} field of the vector is set to 1 if the query contains the corresponding term, otherwise it is set to 0; after that, a vector of concepts is constructed to represent the query, by using the Wikipedia ontology and applying the double score formula (see section 3.2); finally, the vector of concepts is encrypted by the secret key SK (see Section 3.3).
- **Search (T, I', K).** Upon receipt of the encrypted query T (represented by a vector of concepts), the cloud server calculates the scalar product between each document vector and the query vector (the result is a couple of scores). Then, it sorts the documents on the basis of primary scores and possibly secondary scores in case of equality (by using the formula 2). Finally, the server returns to the user the Ids of top k relevant documents.

The search process consists of two main steps:

- **Initialization Phase.** In this phase, the data owner prepares the search environment as follows:
  1. At first, he calls *KeyGen* to generate a secret key SK that is shared with authorized users by using a secure communication protocol.
  2. Then, he calls *BuiltOnto* to construct an ontology from Wikipedia. This ontology will be stored in a local server and will be accessible by the authorized users.
  3. Finally, the data owner calls *BuiltIndex* to construct a secure index from a collection of documents. The secure index as well as the collection of documents (encrypted by another encryption algorithm like *AES*) will be outsourced in the cloud server.
- **Retrieval Phase.** This is the phase where an authorized user performs a search as follows:
  1. At first, an authorized user calls *Trapdoor* to build an encrypted query.
  2. Upon the server receives the encrypted query, it launches the *search* process, and returns to the user the Ids of top k relevant documents.

4 RESULT AND COMPARISON

Yahoo! Answers[4] is a website that allows users to ask questions or answer to questions asked by other users. A data collection was collected from the *Yahoo! Answers* corpus. This collection is composed of 142,627 questions and 962,232 answers. We have performed our experiments on the collection *Yahoo! Answers* where questions represent the queries and answers represent the documents.

We have tested 1150 random selected queries to compare our proposed scheme (SSE-S) with two other schemes. Namely, we have compared the SSE-S scheme with the MRSE scheme (Cao et al., 2014) which uses a conventional search and with Gabrilovich's scheme (Gabrilovich and Markovitch, 2006) adapted for an encrypted search.

Each scheme returns one hundred (100) documents in response to a received query. we calculated the sum of relevant documents retrieved in each scheme according to the number of queries. Figure 2 shows that our proposed scheme (SSE-S) gives better results than the MRSE scheme (60% of improvement) due to a concept-based search, and it gives better results than the Gabrilovich's scheme (36% of improvement) due to the use of double score formula. This clearly demonstrates that conceptual search (GS, SSE-S) increases the recall compared to conventional search (MRSE). Moreover, our experiments confirm that the proposed double score formula is more efficient than TFIDF formula used in Gabrilovich's scheme.

Then, in order to test the quality of the results returned by each scheme, we assume that the Detailed answers are better than the short ones. Thus, to measure the quality of the retrieved documents, we added a filter that ignores documents having a size less than a certain threshold α. We have gradually increased the value of this threshold as, $\alpha = 0$ in the first fifty (50) queries, then $\alpha = 10$ at the fifty (50) queries that follow, then $\alpha = 20$ in the third group of the fifty (50) queries and so on. Figure 3 shows that the results returned in SSE-S scheme are better quality

[4]https://answers.yahoo.com/

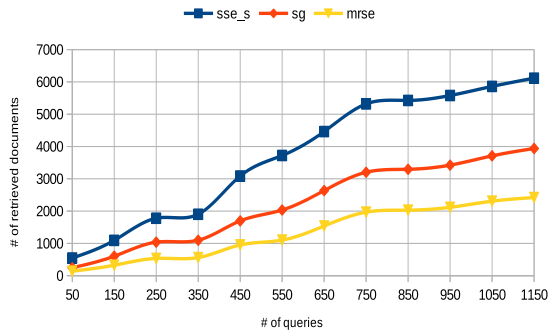

Figure 2: Number of retrieved documents according to the number of queries in three different approaches.

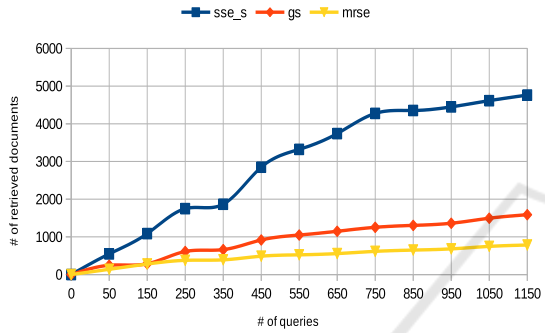

Figure 3: Number of retrieved documents according to the number of queries when applying a filter.

than the results returned in both MRSE scheme and Gabrilovich's scheme. Indeed, our proposed scheme improves the quality of the returned results by 67% compared to the Gabrilovich's scheme and 84% compared to MRSE scheme.

5 CONCLUSIONS

In this paper, we identified the problems of conventional information retrieval that is exploited in most of the search approaches over encrypted cloud data. To fix these problems, we have proposed a searchable encryption scheme called SSE-S. Indeed, the use of a concept-based search allows a significant enhancement of the recall by retrieving pertinent documents even if they do not have any common term with the query. Moreover, the use of the proposed double score formula rather than TFIDF formula allows to ignore irrelevant documents that contain terms in common with the query. Finally, We validated our scheme by an experimental study, where we have compared our scheme with other schemes proposed in the literature.

REFERENCES

Ballard, L., Kamara, S., and Monrose, F. (2005). Achieving efficient conjunctive keyword searches over encrypted data. In *Information and Communications Security*, pages 414–426. Springer.

Cao, N., Wang, C., Li, M., Ren, K., and Lou, W. (2014). Privacy-preserving multi-keyword ranked search over encrypted cloud data. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):222–233.

Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM.

Egozi, O., Markovitch, S., and Gabrilovich, E. (2011). Concept-based information retrieval using explicit semantic analysis. *ACM Transactions on Information Systems (TOIS)*, 29(2):8.

Gabrilovich, E. and Markovitch, S. (2006). Computing semantic relatedness of words and texts in wikipedia-derived semantic space. In *IJCAI*, volume 7, pages 1606–1611. Citeseer.

Li, K., Zhang, W., Tian, K., Liu, R., and Yu, N. (2013). An efficient multi-keyword ranked retrieval scheme with johnson-lindenstrauss transform over encrypted cloud data. In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 320–327. IEEE.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE.

Sun, X., Zhu, Y., Xia, Z., Wang, J., and Chen, L. (2013). Secure keyword-based ranked semantic search over encrypted cloud data.

Wang, B., Yu, S., Lou, W., and Hou, Y. T. (2014). Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *INFOCOM, 2014 Proceedings IEEE*, pages 2112–2120. IEEE.

Xu, J., Zhang, W., Yang, C., Xu, J., and Yu, N. (2012). Two-step-ranking secure multi-keyword search over encrypted cloud data. In *Cloud and Service Computing (CSC), 2012 International Conference on*, pages 124–130. IEEE.

Yang, Y. (2015). Attribute-based data retrieval with semantic keyword search for e-health cloud. *Journal of Cloud Computing*, 4(1):1–6.

Yu, J., Lu, P., Zhu, Y., Xue, G., and Li, M. (2013). Toward secure multikeyword top-k retrieval over encrypted cloud data. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):239–250.