COS2: Contextual Oblivious Similarity Searching for encrypted data over cloud storage services

SNEHA UMESH LAVNIS
x17101026
MSc in Cloud Computing

12th September 2018

Abstract

With the development of collaborative storage services, public cloud has picked up force because of their pay-as-you-go billing structure. For the most part, the information archived on the cloud is secured through encryption before outsourcing which makes it had to recover via searching. The search over encrypted cloud approaches flourish to tackle this issue by utilizing cryptographic and indexing procedures to decide the best outcomes. In any case, the vast majority of these approaches utilize exact matching to fulfil the search criteria. In due course, this exact matching conception was expanded by incorporating similarity ranking algorithms. However, this expansion could not succeed in the practical world due to its dependence on third parties to evaluate the search, thus bargaining the privacy of the stored information. Another drawback is that they require extra computational assets and storage resources for search execution, which sets them far behind from a perfect Search Engine. Hence through this research we implement an advance version of similarity search model by (Pervez et al.; 2016)(Pervez et al.; 2017), known as Contextual Oblivious Similarity based Search (COS2). With the proposed system, authorized users can categorize searches resilient to typing errors. Dissimilar to primitive approaches, COS2 introduces browsing caches to better the client experience. Wordnets and Lexicons with dual encryption mechanisms help realizing relevance searches without revealing confidential data on untrusted cloud domains. Finally, this contextual search thrives to reduce the computational overhead of the overall search procedure.

Contents

1 Introduction

The applications accessible in todays digital era are mostly data intensive and are aggregating double the information each year (Chang et al.; 2014). This period of data boom has prompted generation of enormous stores of information ranging from open shareable information to very private data (Kambatla et al.; 2014) (Esposito et al.; 2015). As of late, business ventures have investigated this collected data to compose Big Data (Manyika et al.; 2011). With the presentation of Big Data, businesses are battling with issues related with their storage and administration. Cloud computing with its versatile pool of assets has provided businesses a chance to use the pay as you go and on request data monitoring services given by cloud to dealing with their Big Data activities. Further, the information usage on the cloud has incredibly profited by the reception of Search over encrypted data schemes. These search schemes empower clients to rapidly recover data of interest from enormous information bulks and therefore has turned into a subject of considerable intrigue recently. The search methods should thrive to enhance the client experience and furthermore upgrade the information recovery technique. When crafting a reasonable encoded information retrieval system, two fundamental viewpoints must be considered: Firstly, the system should bolster relevant keyword search and secondly the information ought to be protected from inquisitive cloud providers and unapproved clients (Sun et al.; 2014). The similarity search model introduced by (Pervez et al.; 2016) (Pervez et al.; 2017) meets the previously mentioned criterion. Additionally, it directs to contextual searches as a subsequent adoption. Hence, this paper attempts to address the following question:

> *"Can the COS2 system implemented in this research overpower the similarity search model given by (Pervez et al.; 2016) (Pervez et al.; 2017) and in addition also enhance the search proficiency for encrypted cloud data?"*

So, what precisely is the issue with enduring encrypted data on cloud? As indicated by the Statista expectations for 2017 public cloud storage, around 1.8 billion individuals overall utilized the cloud for individual use and the demand for the same was around 14,800 exabytes (Statista; 2017) (Cisco; 2016). Also, numerous organizations are moving their information to untrusted cloud domains. Confirmation to this claim can be found in the 2017 cloud adoption report by Gartner which proposes that there has been 18% expansion in usage of public cloud services (Pettey and Goasduff; 2017). However, data protection over cloud has always been questionable. Every now and then, there have been consistent reports of information leaks. European Network and Information Security Agency (ENISA) claims Loss of Governance as a prime factor for data over cloud (Rev; 2012). Furthermore, a study led by Cloud Security Alliance (CSA) defines cloud storage to be high-level risk (Luciano and John; 2014).

Think about Dropbox as a perfect illustration. Dropbox sends substantial encryption to encode client information, yet the private key administration is agreed by Dropbox and not the client. This can prompt misconfiguration issues causing security breaks, such as the blackout experienced in 2014 permitting auto login to 25 million arbitrary accounts at Dropbox (Dropbox; 2014). Some other illustrations include the network intrusion of Sony PlayStation and the celebrity photograph leaks by Apple iCloud (Sanchez; 2015) (Lawson; 2015). The other worry that the clients often face is how the cloud providers handle their sensitive data. For instance, Google's ongoing security policy demonstrates that the data shared by the client utilizing Google platform can be utilized, recreated and conveyed to improve and advance their Service (e.g., Our Gmail inbox is examined to give custom advertisements) (Google; 2017). Likewise, information is gathered and conveyed to business accomplices by Yahoo! (Yahoo!; 2018). This information utilized by cloud suppliers would be advantageous for offering customized benefits yet from an alternate point of view they can raise security issues. For example, a cloud provider can find that a client has diabetes from their search on sugar free items and further dispatch this data to insurance agencies which can list the client as high-risk candidate.

The best answer to relieve the security risks is to encrypt the outsourced data and distribute the decrypting rights to authorized consumers (Yu et al.; 2010) (Kamara and Lauter; 2010). Nevertheless, encryption does not warrant 100% security as users communicate with the cloud for data utilization which enables them to work on the data posing risks for security breaches (Kandukuri et al.; 2009). Usually, the clients are charged for network and storage services by the cloud providers (Park et al.; 2013). Hence, for best usage of cloud assets at reduced costs, clients ought to be given the relevant data without the confidentiality being compromised. Likewise, putting away encrypted information over cloud led to the trouble of recovering important data using

conventional search (Kuzu et al.; 2012). This prompted the proposition of various search algorithms for effective recovery of encoded information, known as the search over encrypted data (Tang; 2013).

The vast majority of search over encrypted data make utilization of trapdoors and pre-defined catch-phrases to execute the search for authorized clients (Song et al.; 2000) (D. Boneh and Persiano; 2004). Some of these algorithms incorporate multi-keyword search (Li et al.; 2011).While some others use encrypted indexes stored on cloud for retrieving outsourced data (Pervez et al.; 2016) (Pervez et al.; 2017). Further, majority of these techniques precisely concentrate on ensuring data and search privacy. The main disadvantage of these schemes is that they rely on exact matches to materialize the search which also indicate that they are prone to typographical errors. Another drawback is that they do not consider the honest but curious nature of the cloud (Sun et al.; 2014).In the honest but curious model, information can be deduced by the cloud provider by distinguishing patterns from search phrases and the indices persisted virtually in their shared reserve. Thus, a new evolved algorithm that could accomplish both semantic information recovery and oblivious search calculation is of need. This requirement is satisfied by the COS2 system implemented in this research. COS2 refrains intruders from accessing the sensitive information stored on cloud by encoding both the index and the search. Apart from this, the system allows access to only authorized users by applying standard login procedures. Wordnet and browsing cache form the basis of the COS2 system. With Wordnet, synonyms for search phrases are identified enabling relevant data retrieval. Lastly, caching provides the users with an interactive search experience in minimal response time.

The remaining paper is distributed as: Section 2 covers the related work in encrypted search area, Section 3 explains the approach and the design considered to build the system, Section 4 discusses the development standards followed to implement the system, Section 5 verifies the outcomes of the system in terms of performance and complexity and finally Section 6 concludes the research

2 Literature Review

In this segment, we cover the existing approaches for performing cloud based search over encrypted information.

The searchable symmetric key cryptography (SKC) was the first searchable cryptosystem proposed by (Song et al.; 2000).This system uses the concept of trapdoors, which are functions that can be computed in one direction only. The SKC technique returns positive results only if exact occurrences of the keywords are identified and is also known to return false positive results as the exact matches are evaluated through probabilistic manipulations. Further, the rectification of these manipulations cost communication overheads in between the server and the client. The public key encryption with keyword search (PKES) is a similar approach proposed by (D. Boneh and Persiano; 2004). In contrast to SKC, asymmetric encryption is used in PKES. The authors of this system have put forth two variations of PKES. First, a highly efficient Diffie Hellman bilinear map-based system and second, a less efficient random trapdoor permutation-based system. However, both SKC and PKES utilize the generated trapdoors to carry out the search based on exact matching. The transmission of the trapdoors on public cloud poses privacy risks to the data and can be susceptible to server-side dictionary attacks. In addition, separate trapdoors are required for each keyword which impacts the search capability and limits the user search to only particular keywords.

(Li et al.; 2011) put forth the Authorized private keyword search (APKS) for encrypted cloud data. This mechanism exploited the online public health reserves to identify the necessity of authorized searches. The authorization was achieved through access matrix and the trapdoors were hierarchically delegated with the help of trusted third parties. The search is accomplished by identifying exact matches between the reserves and the trapdoors. Moreover, this mechanism eliminates the potential risks identified by SKC and PKES systems since it deploys proxy servers to improve the security. Additionally, similar to SKC and PKES, APKS also restricts the users search capability by defining trapdoors only for particular keywords. (Wang et al.; 2010) came up with a relevance search scheme for encrypted data over cloud. Symmetric encryption with order preserving technique is employed to derive ranked based search. The ranking is calculated using relevance scores defined by the TF (Term Frequency) X IDF (Inverse document frequency) rule. But, this scheme lacks the facility to manage multiple keywords and attain relevance from massive data stacks.

Another accessible and distributed cryptographic storage framework (CS2) was introduced by (Kamara et al.; 2011). This framework seeks encrypted data over cloud by gathering dynamic revisions from the outsourced data which assures that the data is not reprocessed in case of any updates or deletes. It fulfils this by deriving a coordinated match across the encrypted index and the search. Nevertheless, CS2 was designed for just enjoying cloud capacity benefits and not for collaborative administration and sharing information. Correspondingly, a more secure and proficient platform for encoded search was proposed by (Hahn and Kerschbaum; 2014). Equivalent to CS2, this platform also operated by searching over encoded indices. But unlike CS2, no additional storage was required to manage the index by this platform on the client machine. The client just stores the shared symmetric key. Moreover, since this platform relies on predetermined access sequence to discover the keywords, it cannot be extended to comprehend relevant results.

The privacy preserving multi-keyword searching was presented by (Sun et al.; 2013). A similarity-based rank algorithm is utilized in this system to obtain relevant results. An Index tree is constructed using a multi-dimensional algorithm to perform the search. Further, this tree structure and the data are encrypted before cloud transmission owing to the honest-but-curious nature of the cloud. But the system fails to recognize authorized subscribers providing malicious attackers an opportunity to deduce sensitive information. Besides, the search criterion retains its plaintext nature which constitutes to additional security flaw. An alternative multi-keyword search was introduced by (Cao et al.; 2014). Another frontier algorithm known as Oblivious Term Matching (OTM) to conduct oblivious searches was presented by (Pervez et al.; 2013). As the name suggests, this search does not reveal any confidential data through the index or the search query. OTM maintains this privacy by encrypting both data and indices and also incorporates the mechanism to accommodate conjunctive keywords. Thus, complex search queries can be easily compiled by the data consumers of this system. Although it supports multiple querying, the retrieval still depends on exact matching of individual keywords. (Cash et al.; 2013) proposed a more scalable search mechanism for untrusted cloud domains. Alike OTM, the researchers of this paper applied indexing for conjunctive searches. However, this mechanism was unsuccessful in realizing relevance-based search.

In order to overcome the challenges posed by exact matching across the data and the search, an efficient similarity searching algorithm was put forth by (Kuzu et al.; 2012). This algorithm applies relevance searching by using Locality sensitive hashing to determine the closest neighbor of the search (Indyk and Motwani; 1998). Secured Indices in form of variable size buckets are constructed and entries that are nearly identical are set in the same bucket whereas the remaining entries are discarded. Numerous security checks were carried out by the authors in order to validate this algorithm. Regardless, this algorithm resorts to dual servers to execute the search. Furthermore, it assumes that the two servers do not communicate with each other which obliviously is not possible in the real world. Over that, if the algorithm is deployed with a single server it can lead to data leaks via the plain text search.

Another class of searching technique defined as Fuzzy keyword search was presented by (Meharwade and Patil; 2016). This technique strives to enhance the user search experience by conducting searches that are resilient to typographic errors. Like most of the methodologies discussed above, this technique also uses trapdoor-based indices. The Index incorporates all forms of typographic errors so that the search can be computed by exact matching. The major drawback of this approach is that certain variants for a particular keyword are likely to be bypassed while creating the index. Also, due to the immense size of the generated index, this system lags in performance. A comparatively new search as a service scheme was implemented by (Li et al.; 2014). The authors of this research make use of hybrid clouds with the fuzzy search logic to improve the efficacy of the search process. However, the utilization of private cloud demands continuous monitoring which in turn hinders future migrations to public cloud. One more challenge of this scheme is precomputing misspelled keywords which indicates the same defects as in the prior fuzzy search schemes. (Boldyreva and Chenette; 2015) introduced a better revised fuzzy search logic to capture noisy data. A closeness measure (close, near and far) is calculated to locate similarity between keywords. Yet this logic yields unsatisfactory results and provides a below par experience to the user.

There are various products accessible in the business sector that empower cloud-based search. Google search appliance and Microsoft Search Server are two such products that enable querying within the public and the private data centers owned by an enterprise (Microsoft; 2001) (Google; 2002). These products are mostly useful to intermediate and large organizations who possess their own data centers and thus does not apply for personal usage. The Index is centrally located inside the in-house server of the enterprise. Although, the index is outfitted to handle authorized access, the central nature of the index could lead to single point of failure. Moreover, as the search process is completely managed by the enterprise, future movements to the cloud are interfered. Some defects pertaining to the access control privileges were recognized by the research conducted by (Singh et al.; 2009). The research also shows that the system can be easily intruded by slight modifications in the search queries to gain valuable insights.

The domain of encrypted search has made tremendous progress in regards to refining the read-write efficiency, authentic searching and storage capabilities over the years. Based on the concept of nearest locality combined with symmetric encryption, a search mechanism was put forth by (Asharov et al.; 2016). The purpose of this mechanism was to better the read efficiency and the storage capacity. Apart from this mechanism, a distributed approach on similar grounds was suggested by (Ishai et al.; 2016). This approach could accommodate large datasets by constructing the index as B-tress. Yet, the system relied on two servers to perform the search execution. This mechanism was advanced with the introduction of verifiable search by (Cheng et al.; 2015). Verifiable results were returned by measuring the search correctness and availing inclusion of conjunctive and Boolean queries.

To overcome the obstacles of data security but at the same time offer support for relevance based search, a semantic similarity searching technique was proposed by (Pervez et al.; 2016). Their presented system eliminates trusted third parties to handle trapdoors. Rather the system depends on the availability of the cloud service.

The search queries are carefully formulated to actualize a methodical privacy-aware system. The authors of this research made advancements in their old system to define the Oblivious similarity search (OS2) (Pervez et al.; 2017). Moreover, in finishing up their work, they demonstrated the possibility of improvement in their current model by incorporating contextual searches. Incorporation of the context of the subscribers will increase the accuracy of the relevant results returned to the users. Hence, by integrating the rewards of the OS2 model, the proposed contextual search (COS2) optimizes the search productivity and in turn upgrades the user search experience.

With this paper, we introduce the following commitments in the sector of search over encrypted cloud data:

- The search is resilient to typographical errors.
- Wordnet based semantic querying evaluates a similarity function to gauge the relevance between outsourced data and entered search.
- Use of Lexicon structure to conduct oblivious assessment without relying on any semi-trusted or trusted third parties.
- Caching to support contextual processing and successively improve subscriber experience.

Table 1 gives a correlation of the existing methodologies with the implemented COS2 system.

Search mechanism	Availability Requirement		Query Execution		User defined search criteria	Honest but curious	Relevance based Search	Contextual Search
	Storage Service	Third Party services	Storage Service	Third Party services				
Wang et al. (2010)	Y	Y	Y					
Li et al. (2011)	Y	Y		Y				
Kuzu et al. (2012)	Y	Y		Y		Y	Y	
Cash et al. (2013)	Y		Y			Y		
Pervez et al. (2013)	Y		Y		Y			
Sun et al. (2013)	Y		Y			Y	Y	
Cao et al. (2014)	Y		Y			Y	Y	
Hahn and Kerschbaum (2014)	Y		Y					
Li et al. (2014)	Y	Y		Y			Y	
Boldyreva and Chenette (2015)	Y		Y				P	
Cheng et al. (2015)	Y		Y					
Ishai et al. (2016)	Y	Y		Y		Y		
Pervez et al. (2017)	Y		Y		P		Y	
Proposed System	Y		Y		P		Y	Y

Table 1: Chronological comparison between various cloud based encrypted data search methodologies (Pervez et al.; 2017). (Y represents supported features whereas P represents features that can be supported in the extended version of the system.)

3 Methodology

3.1 Notations

The notations cited with their detailed description is given in Table 2.

Notation	Description
F	File to be outsourced
I	Inverted index for F with n keywords
H	Hashing function for SHA256
E_S, D_S	Symmetric key encryption and decryption (AES)
k	shared key for symmetric cryptography
Q	Search Query
Q_E	Enriched search Query
E_H, D_H	Homomorphic encryption and decryption
h_p, h_s	Public and Private key for homomorphic encryption
$\alpha_{0 \cdots n}$	Coefficients of Polynomial modelling which define the search
β	Oblivious Search results

Table 2: Notations used in the system

3.2 Approach

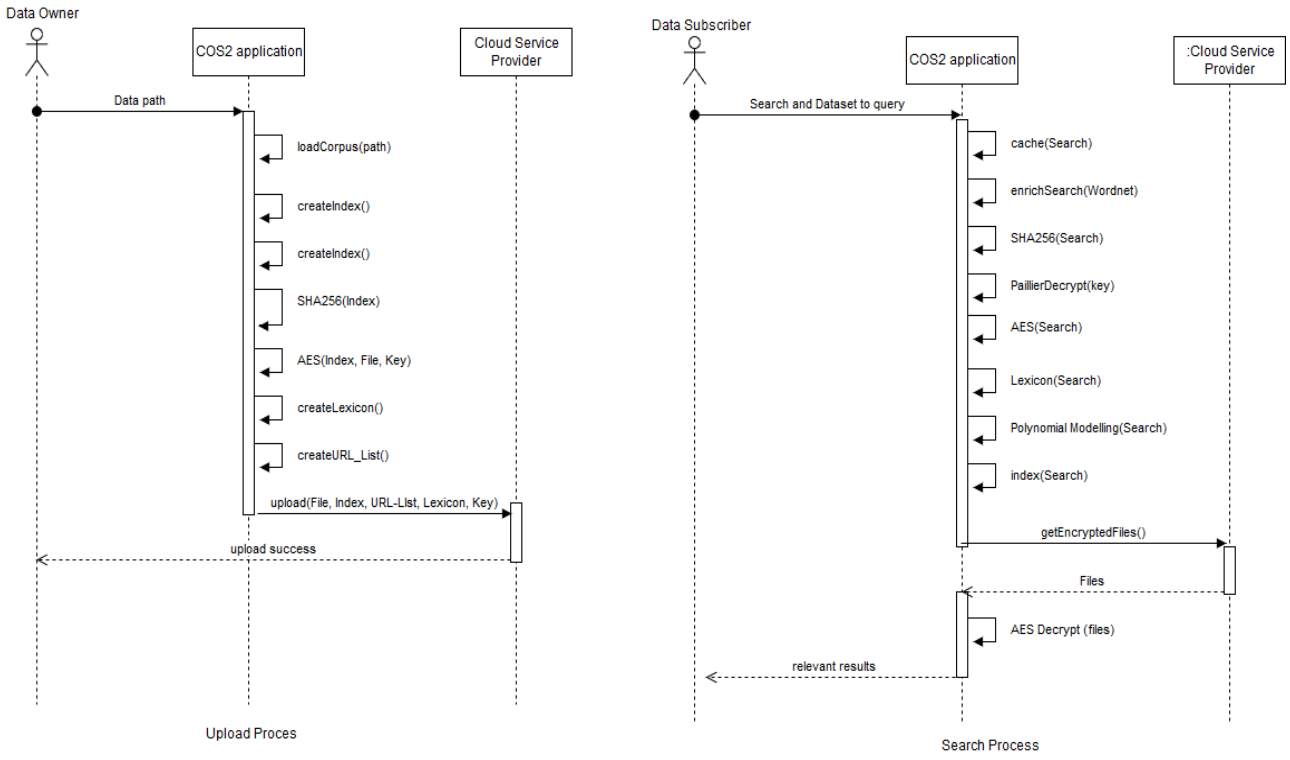


Figure 1: Sequence Diagram

The users (Data owner/Data subscriber), COS2 application and the cloud service provider (CSP) are the three principle entities embracing the whole process. The storage service is maintained and owned by the CSP. The Data owner is responsible for uploading data and key needed for the search process. The subscriber consumes the data by querying for desired results. The user can assume any role (owner or subscriber) at any time depending on the requirement. The processing of files, key management, initiating searches and building complex data structures is solely under the control of COS2 application. The CSP manages User authentication and storage resources for COS2. Such hybrid architectures has applications in areas of Genome studies, Medical research facilities, News organizations and much more where confidential information is shared on a daily basis

(Pervez et al.; 2016) (Pervez et al.; 2017). The sequence diagram shows interaction between the three entities of COS2 1.

3.3 Architectural goals and constraints

Providing users with an interactive search experience just similar to discovering information on the Internet is the motive of the COS2 system. In COS2, the cloud supplier is considered as honest-but-curious and thus, can conspire with hackers to attack the system (Sun et al.; 2014). Therefore, the cloud service is utilized for just storage purposes. The search criteria and the outsourced data are both encrypted which assures there are no information leaks during network transmission between the application and the cloud. Further, successful distribution of the keys is beyond the scope of this project. To overcome this issue of key management, the secret key (k) is also encoded and stored along with the data and the indexes on the cloud storage. As search history differs from consumer to consumer, caches for contextual searching is built within the COS2 application locally. Besides, Wordnet dictionary is embedded within the COS2 application to aid with relevance searches (Christiane; 2005).

3.4 System Design

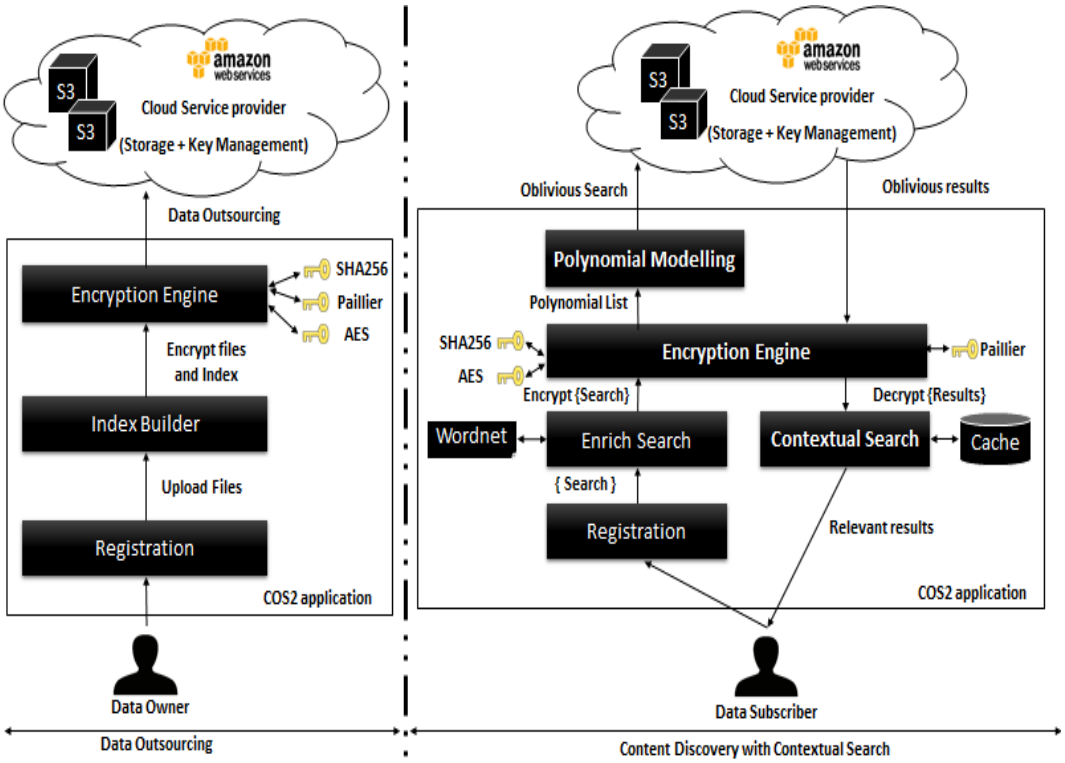


Figure 2: System Architecture

The system architecture is shown in figure 2. For the first time, the data owner should register with the COS2 application using a username and password. The login credentials are safely stored using relational databases in the cloud. To upload files to the cloud storage, the data owner has to sign in to the COS2 application using the web interface. On successful login, the data owner needs to specify the path of the file to be outsourced and a secret key (k) to be used for AES encryption (NIST; 2001). The COS2 application navigates to the data path and encrypts all the files in it. Then it creates an Index to account for all the words present in the outsourced files. Likewise, the index is hashed and encrypted. Additionally, two more structures URL List and Lexicon are created. These structures are also encrypted. The key entered by the data owner is encoded with Paillier cryptography for security purposes (Paillier; 1999). The encrypted files and indexes along with the additional structures are transmitted to the cloud storage using APIs provided by the cloud provider (Amazon; 2006, 2009). Similarly, the data subscribers have to be registered to initiate the search. The data subscribers specify the data store they want to query. The search is enriched with synonyms by exploiting the Wordnet dictionary integrated with COS2 application. The key file stored in cloud is accessed and decrypted using Paillier cryptosystem to obtain secret key (k). The enriched search criteria are hashed and encrypted using (k). Later, a list of search keywords is created with polynomial modelling (Bai et al.; 2009). The COS2

system constructs a cache to record subscribers search history (Massimo; 2012). The polynomial list is first evaluated against the cache to achieve results. If no matches are found the list is sent over to the cloud for processing. The cloud provider computes oblivious results. Based on the oblivious value received, the COS2 application decrypts the matching files and displays them in a user-friendly manner. Since, the upload and search processes are wholly managed by the COS2 system, the cloud provider remains clueless regarding the data stored in its virtual pool. The cloud provider is merely responsible for offering storage services. In this way, COS2 preserves privacy and recovers relevant results.

3.5 Main Modules

The COS2 system is made out of four essential modules: User Registration, Index Building, Data Outsourcing and Query Processing, which are talked about in detail in the accompanying sub sections. The class diagram for the main modules is depicted in figure 3.

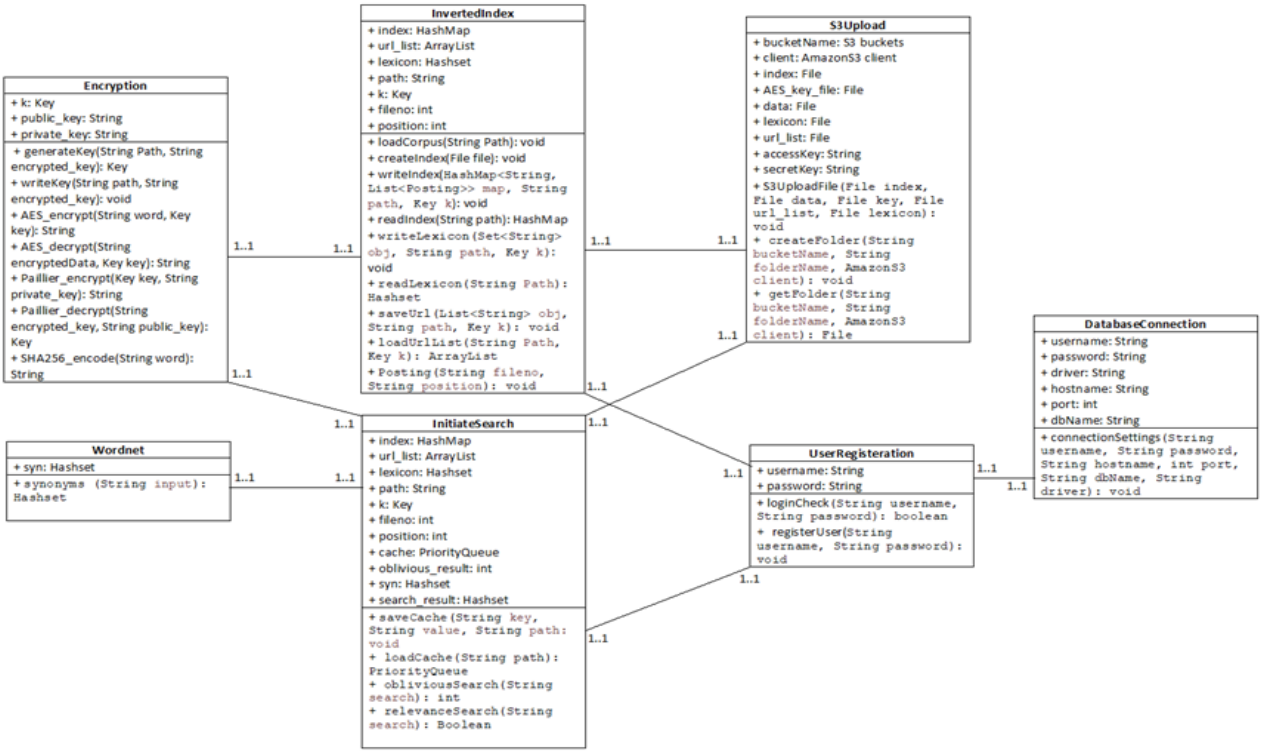


Figure 3: Class Diagram

3.5.1 User Registration

This module verifies the identity of the user (Stallings; 2010). The identity is provided as a combination of username and password. This module also allows first time users to sign up with the COS2 application. The user is required to select a unique username in order to complete the registration phase. The password is a secret known only to the user and is stored as a hash in the cloud relational database. The login also employs a CAPTCHA to determine whether or not the user is indeed human (Luis et al.; 2007).

3.5.2 Index Building

An Index data structure is used to perform the similarity searching in the COS2 system. An Index creates a posting that maps a list of documents to a particular keyword (Zobel and Moffat; 2006). Typically, an Index can be considered as a database where processed information is stored to assist in fast information retrieval. The COS2 follows a more traditional index building approach as opposed to the OS2 system implemented by (Pervez et al.; 2016, 2017). The authors of OS2 have incorporated Apache Lucene for their information retrieval and index construction. However, Apache Lucene has known to have update issues with their Indices which limits extension of the existing Index entries (Biaecki et al.; 2012). Furthermore, numerous bugs and defects have been identified in their Index merging procedure. Rather than utilizing Apache Lucene, the COS2 system creates the Index from scratch. The Document-at-a-time (DAAT) procedure is followed which ensures that one page is extracted at a time and send to the parser for processing. The parser identifies meaningful terms in

the page and creates a posting for each such term. A global file number is assigned to every document that is parsed. A combination of this file number and the position of occurrence of the keyword make a single posting entry, as depicted in figure 4.

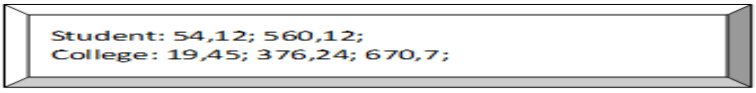


Figure 4: Inverted index sample in plaintext

Two additional data structures, namely the URL List and the Lexicon aid in the search process. The URL list keeps track of all the page visits in order to eliminate duplicates while parsing whereas the Lexicon stores an extensive dictionary of all the words that were indexed. The relationship between the URL List, Lexicon and Index is shown in figure 5. An important distinction from the OS2 system is that the posting does not include synonyms to perform relevance searches. This index enriching step to compute relevance is intentionally discarded as it prolongs the index building time. Besides the relevance search is accomplished alternatively while processing the search as described in the Query Processing module.

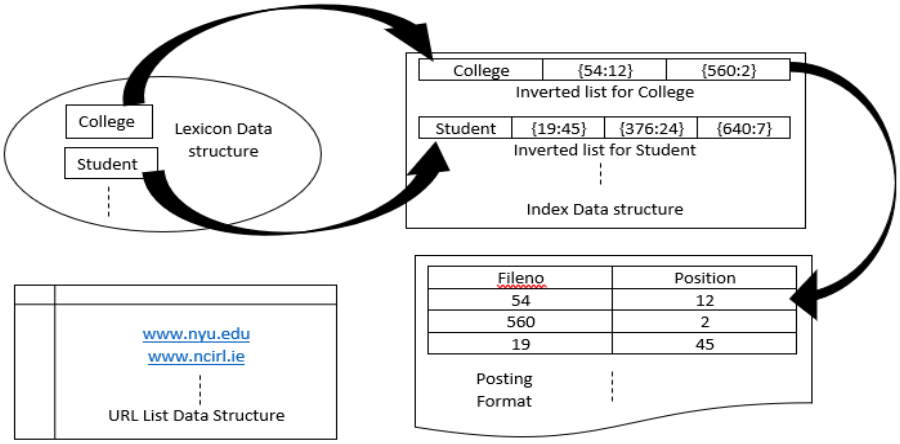


Figure 5: The relation between the Index, URL list and the Lexicon Structure

Every posting in the index is hashed. Mathematical operations are performed to convert a sequence of text into values of fixed sizes via a hash function (Stallings; 2010). The hash is a one-way function and its inverse cannot be calculated. COS2 uses Secure Hash Algorithm (SHA) to encode the keywords. SHA is defined as the information processing standard by the National Institute of Standards and Technology (NIST) (NIST; 2012). Its time efficiency and robustness make SHA quite popular. We apply the SHA256 variant in COS2 which generates a 32-byte hash.

3.5.3 Data Outsourcing

In this stage, the COS2 application establishes a connection to a public cloud storage in order to preserve the Data and the Index needed for the search process. As both the file uploaded by the Data owner and its corresponding Index generated by the Index building module include sensitive information, these files must be encrypted before they are transferred to the cloud. Failing to do so may give malicious hackers and conspiring cloud providers an opportunity to access confidential data. The symmetric key encryption is used as the encryption mechanism (Stallings; 2010). This technique uses a secret key (k) to accomplish the encryption and this key (k) is further stored along with the data and the index on the cloud. As the COS2 deploys strict authentication and authorization, secret key (k) is accessible only by registered users. To preserve confidentiality and achieve data outsourcing, the uploaded Data files and the Index are encrypted as E_S(F, k) $\rightarrow$ (F) ^{k} and E_S(I,k)$\rightarrow$(I)^{k} respectively and later both (F) ^{k} and (I) ^{k} are distributed over cloud. The COS2 system particularly employs Advanced Encryption standard (AES) algorithm for cryptography. AES has a worldwide acceptance and is mainly used by prime organizations such as the US government (NIST; 2001). AES has replaced its predecessor DES algorithm to gain recognition as the new encryption standard as defined by NIST. In AES, symmetric blocks aid with the encryption and the generated ciphertext can be reverted back to plaintext through decryption. The typical block size for the cipher is 128 bits and it works well with key sizes of 128, 192 and 256 bits. COS2 uses the AES-128-bit encryption with a permutation and substitution framework. The compact design and fast speed of the AES algorithm are very much ideal for the system.

The second part of the encryption process concentrates on securing the AES key file. To preserve the privacy of the AES key, we apply Homomorphic encryption. COS2 opts for the minimal time, partial homomorphic algorithm introduced by Pascal Paillier in the year 1999 (Paillier; 1999). Paillier cryptosystem is based on zero-knowledge proof which ensures that the key can be transmitted without being decoded. This additive homomorphic algorithm uses two separate keys similar to asymmetric encryption. In the proposed COS2 system, a key set (h_p, h_s) is defined where h_p is the public key whereas h_s is the private key. The encryption is framed as $E_H(\mathrm{k}, h_s) \rightarrow (\mathrm{k})^{h_s}$. Further, this encrypted key is decrypted with the h_p at the time when the search results are obtained from the cloud provider for processing. (Refer the notations in section 3.1.)

3.5.4 Query Processing

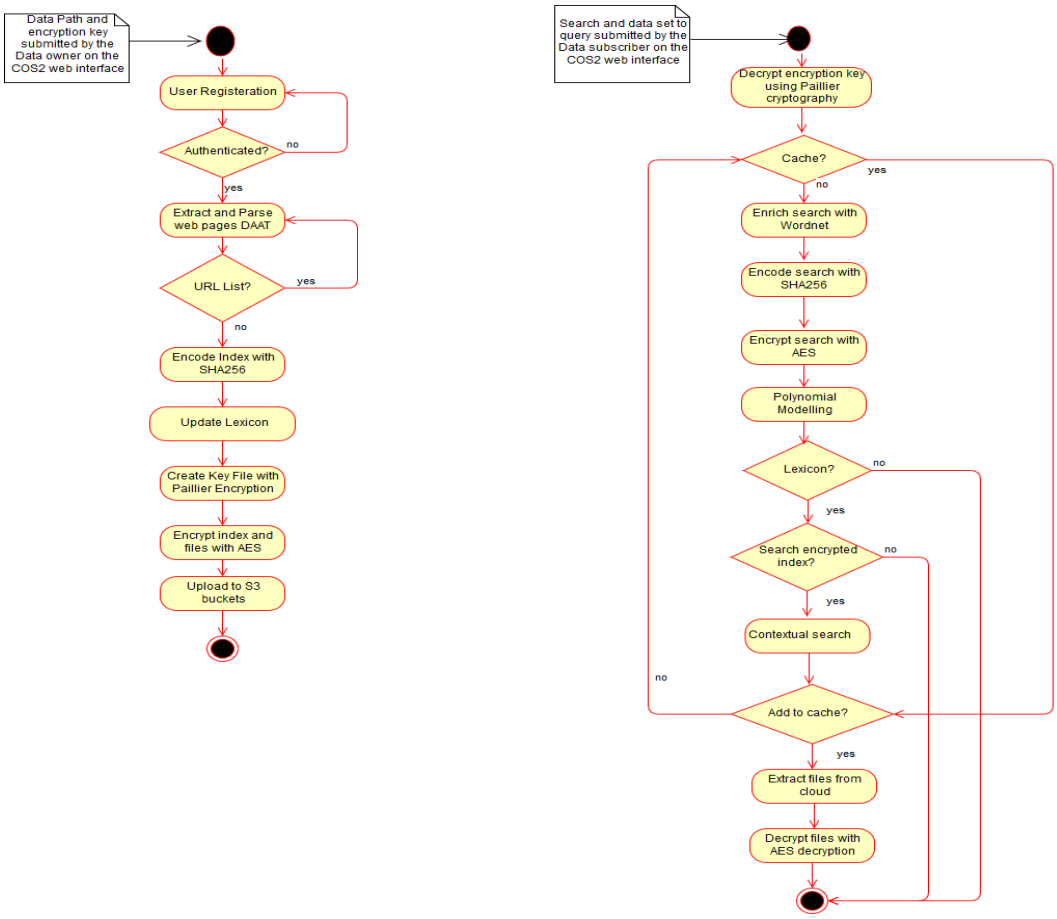


Figure 6: Activity Diagram

The activity diagram is given by the figure 6. A search query (Q) is defined by the authorized user. The COS2 system extracts the synonyms for this query (Q) to assist in relevance searching. To do so, the open source Wordnet library issued by Princeton University is utilized. Wordnet basically comprises of large data store of English lexicas. As opposed to thesaurus, it includes both the meaning and the sense for the words to pair them together. Identifying words in a dictionary is time consuming. Besides, there can be reiterations with thesaurus. Thus, Wordnet recognizes nouns, verbs, adjectives and adverbs and sets them in a matrix for simple access. It is critical to note that we do not utilize sliding windows with Bloom filters for finding the relevance as suggested in the literature review (Pervez et al.; 2017). This is because bloom filters can return false positives and choosing a well-suited sliding window size requires testing against numerous data sets which would be difficult in the short time span of this research. Once the synonyms for the search are identified, an enriched search query (Q_E) is formulated. Further, this enriched query (Q_E) is encoded with the same hash function as used in the Index Building module (SHA256). The AES key file is decrypted using the public key of Paillier cryptosystem to obtain the secret key (k) as $D_H(\mathrm{k}, h_p) \rightarrow (\mathrm{k})$. This key (k) is used to encrypt the search using AES cryptography. Finally, a polynomial model is defined as a list with $keyword_0 : keyword_n \in (\mathrm{C})^{\mathrm{k}}$ where the root of $\mathrm{P(x)} = \mathrm{P}(\mathrm{x} \in (\mathrm{C})^{\mathrm{k}}) = \sum_{i=0}^{\mathrm{n}} \alpha_i = 0$ and $\alpha_{0 \ldots n}$ are the coefficients of P(x). This polynomial list helps in catching the association between words and is comparatively a new phenomenon. This entire query formulation process is carried out within the COS2 application. This list is then compared with Cache stored in

the client machine. The Cache maintains the search history of the client and is basically employed to perform Contextual searching. If a match between the coefficients ($\alpha_{0...n}$) of the list and the Cache are identified, a Cache hit is recorded and results are retrieved in minimal response time. If no matches are identified, a Cache Miss is recorded and the list is transmitted to the cloud provider for processing. Also, the search query is recorded in the cache for further references. The Cache is maintained in Least Recently Used (LRU) format which stores the recently accessed page right at the top. Another attempt to reduce response time is made by iterating the Lexicon to find matches. The Lexicon search is evaluated obliviously as shown by the equation 1. (Refer the notations in section 3.1.)

$$\beta = \begin{cases} 1 & \text{if search exists in the index} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

If the value of β is 0, the process exits with no results. Otherwise, with β as 1, a deep search on the index is conducted to retrieve the encrypted inverted list for Q_E. This list is returned to the COS2 application where it is decrypted as D_S(I, k) $\rightarrow$ I. I provide the URL of the page with the corresponding position for the search. The file F containing the search is retrieved as D_S(F, k) $\rightarrow$ F. The oblivious execution and processing on the application side guarantees that neither the cloud provider nor hackers can deduce any useful information.

4 Implementation

4.1 Coding

COS2 has no special computation and processing requirements. Hence, it is developed on a low-level Laptop with Windows operating system and other specifications as: Intel core i5 processing with 1Tb Ram and 8Gb main memory. Furthermore, the successful implementation of COS2 relies on availability of cloud storage and the interface handling upload/search mechanism. Checking the service level agreement is a good way to ensure availability of the cloud service provider (Sakr and Liu; 2012). The Gartner Magic quadrant ranks Amazon Web Services (AWS) to be the frontier cloud computing platform of 2017 (Leong et al.; 2017). Besides, AWS delivers service availability of 99.999% (Amazon; 2018). Thus, we will use AWS S3 buckets to store the outsourced data and the encrypted index. Also, AWS RDS will be utilized for user registration management. The previous work in privacy-aware searches suggest Java as the best programming language to design such systems. Moreover, being a object oriented language, Java offers added benefit of reusability, platform independence, security, dynamicity and robustness (Horstmann and Cornell; 2002). Also, debugging and testing becomes easy with the stable Java platform. COS2 uses third party libraries such as Wordnet and javallier that are compatible with Java (Christiane; 2005; Thorne; 2017). The Eclipse IDE for Java comes as an ideal development tool as it is well equipped with AWS integrations.

4.1.1 User Registration

To store login credentials, AWS RDS service is utilized. RDS stands for Relational database service (Amazon; 2009). This service is easy to connect and monitor and is highly secure. COS2 uses MySQL variant of the database. To use the service, a db instance is created by specifying the database name, username and password on the AWS RDS dashboard. Later, a connection is made from the Java code by configuring the MySQL-Java-connector and embedding the connection strings as shown in code snippet below.

Listing 1: Java-MySQL configuration

```
public class DatabaseConnection {
        public String connectionSettings(){
                // connectionString format = jdbc:driver://hostname:port/dbName?
                    user=userName&password=password
                String connectionString = "jdbc:mysql://cos2-connection.
                    cegx3klf57zi.eu-west-1.rds.amazonaws.com:3306/COS2_db?user=
                    root&password=rootroot";
                return connectionString;
        }
}
```

The web interface of COS2 allows users to sign up with their username and password. A corresponding login table is created in RDS with username as the primary key (figure 7). Also, for security reasons, the password entered by the user is stored as SHA256 hash.

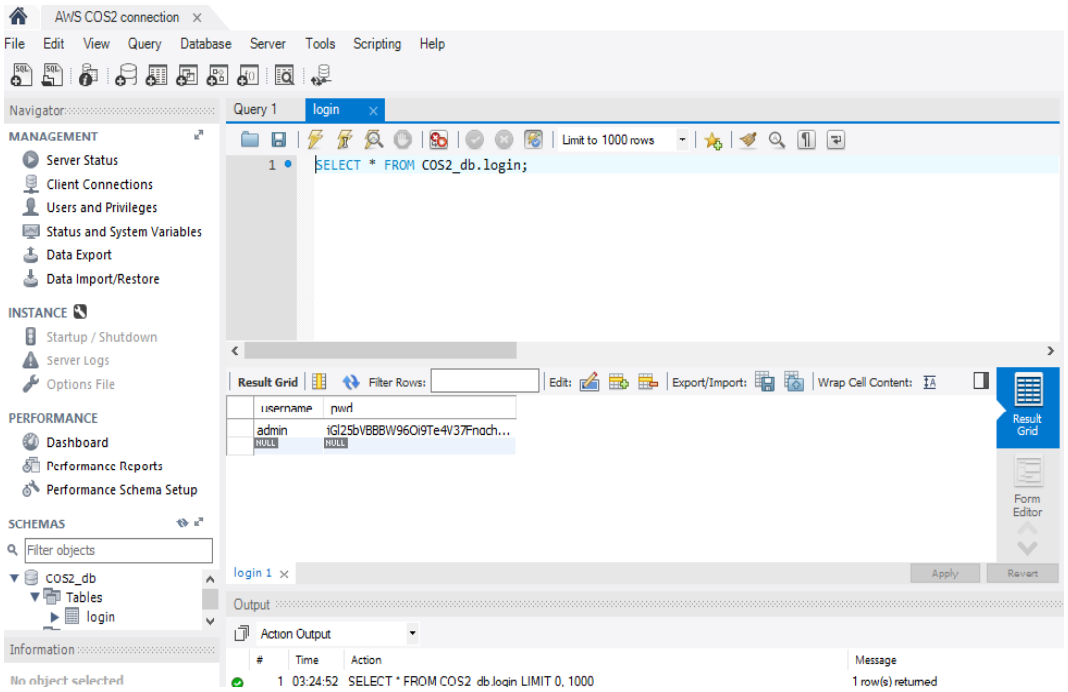
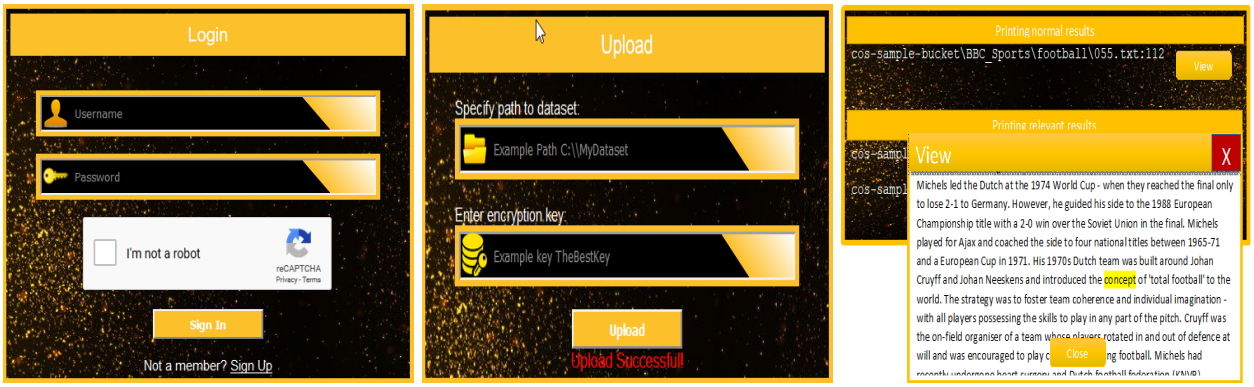


Figure 7: AWS RDS



(a) Login Page (b) Upload Page (c) Display Page

Figure 8: COS2 application screenshots

For logging into COS2 system, the web interface requires username, password and reCAPCTHA as illustrated in figure 8a. reCAPTCHA is used to verify whether the user is in fact human and also protect the application from spams and abuse (Luis et al.; 2007). reCAPTCHA is a free Google API that can be accessed using certain token keys.

4.1.2 Index Building

HashMap from Java collections is used to define the Index (Horstmann and Cornell; 2002). Index file will contain the root word along with a posting list. The posting is created as a user defined class with fileno and position as the data entities. Java Server Pages (JSP) form is used to get input for the dataset and the encryption key from registered data owner. The JSP is processed using servlets. The key obtained from the servlet is encoded with open source javallier library and stored in AES_key_file.txt (Thorne; 2017). Every data file is iterated through isDirectory() method of java.io.File class. A posting is created for each word using the following code listing.

Listing 2: InvertedIndex Class

```
//write index to file
public void writeIndex(HashMap<String,List<Posting>> map, String path, Key k){
    try{
                Encryption e = new Encryption();
```

```
                File file = new File(path);
                FileOutputStream fos=new FileOutputStream(file);
            PrintWriter pw=new PrintWriter(fos);
            System.out.println("Encrypting index.....");
            for(Entry<String, List<Posting>> m :map.entrySet()){
                pw.print(e.AES_encrypt(m.getKey(), k)+":");
                for(Posting l: m.getValue()){
                    String fileno = l.getFileno();
                        String pos = l.getPosition();
                        pw.print(e.AES_encrypt(fileno, k)+ ","+e.AES_encrypt(pos
                            , k)+";");
                }
                pw.println();
            }
            pw.flush();
            pw.close();
            fos.close();
  }catch(Exception e){}
}
```

An encryption class is created to include AES, SHA256 and Paillier cryptosystems. We use MessageDigest and Key class from java.security to return SHA256 and AES (Horstmann and Cornell; 2002). The index file that is created is encoded and encrypted using the Encryption class. Besides a URL list is maintained as an ArrayList to record all the parsed files. Lexicon data structure is implemented as a Hashset to curate a dictionary for all the words in the dataset. Index Builder finally generates index.txt, lexicon.txt and url_list.txt.

4.1.3 Data Outsourcing

Once the three files - index.txt, lexicon.txt and url_list.txt are formulated, the AWS API for S3 buckets is used to upload the files. S3 stands for simple storage service (Amazon; 2006). To access S3 buckets, the access and secret keys have to be instantiated on the AWS IAM console. These keys are then included in the Java code along with the bucket name and region of access. The aws-sdk dependency for S3 should be specified in Maven plugins. Appropriate Try-catch blocks are added to handle exception in case of connection failures. On successful upload of encrypted files and indexes,Upload Successful! message is displayed to the data owner (figure 8b). The following code helps with the S3 upload.

Listing 3: S3Upload Class

```
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonS3 s3client = AmazonS3ClientBuilder.standard().withRegion(Regions.
    EU_WEST_1).withCredentials(new AWSStaticCredentialsProvider(credentials)).
    build();
String bucketName = "cos2-example-bucket";
String folderName = null;
if (s3client.doesBucketExistV2(bucketName)) {
        folderName = "MyData\\index";
        createFolder(bucketName, folderName, s3client);
}
```

4.1.4 Query Processing

The Data subscribers should login into the COS2 system to initiate the search process. The Search JSP takes in the search criteria and the dataset to query as input. The entered search is sent to the Wordnet library to retrieve synonyms for the same (Christiane; 2005). The Wordnet uses a predefined dictionary to obtain synsets of a particular word based on its part of speech (noun). A polynomial model is constructed using ArrayList of root word and the synsets of Wordnet. The SHA256 function in the Encryption class translates this ArrayList to a Hashed List (NIST; 2012). The Aes_key_file.txt is then extracted from the S3 buckets to decode the shared key(k) via Paillier Decryption (Paillier; 1999). The shared key(k) is then stored as MessageDigest key and used for AES encryption of the search (NIST; 2001). This search is compared with the local Cache to find a hit or a miss. The Cache uses PriorityQueue data structure and records every search henceforth performed by the subscriber. Cache hit returns relevant results in minimal time whereas a miss requires comparison with the Lexicon data structure. The results returned by Lexicon comparison are oblivious (0 or 1). A 0 indicates the

search query does not exist in the dataset and 1 requires deep search of the index to find the path and position. Once a path and position for the search are received, they are decrypted and the normal results and relevant results are displayed on the Display JSP. This JSP is equipped with a view button, clicking on which shows the highlighted search in a dialog (figure 8c. The typing errors in the search are identified using the open source Java spell-checker known as LanguageTool. The cache is implemented as shown in listing below.

Listing 4: S3Upload Class

```
CACHE = c.loadCache(CACHE_NAME);
System.out.println("Enter string to search: ");
String search = scanner.nextLine();
if(CACHE.containsKey(search)){
        System.out.println("Cache Hit!");
        LOGGER.info("Cache Hit!");
        CACHE_value = CACHE.get(search);
        normal= results[0].split(",");
    relevant= results[1].split(",");
        System.out.println("Printing normal search results....");
    for (int i=0; i<normal.length; i++){
                        System.out.println(normal[i]);
        }
        System.out.println("Printing relevant search results....");
        for (int i=0; i<relevant.length; i++){
                        System.out.println(relevant[i]);
        }
}
```

4.2 Datasets

Three main datasets of varying sizes are used to test the COS2 system. All of these datasets include txt files. The first dataset is BBC_Sports which contains original articles related to cricket, football, rugby, tennis and athletics (Greene and Cunningham; 2006). The articles date to the year 2004-2005 and in total there are 737 files in this dataset. Another original dataset from BBC contains actual news contents (Greene and Cunningham; 2006). This dataset labelled as BBC_News has around 2225 documents and covers business, politics, sports, tech and entertainment news. This too dates back to the year 2004-2005. The third dataset contains 129,000 documents with abstracts from National Science Foundation from the year 1990 to 2003 (UCI; 2003).

4.3 Unit Testing

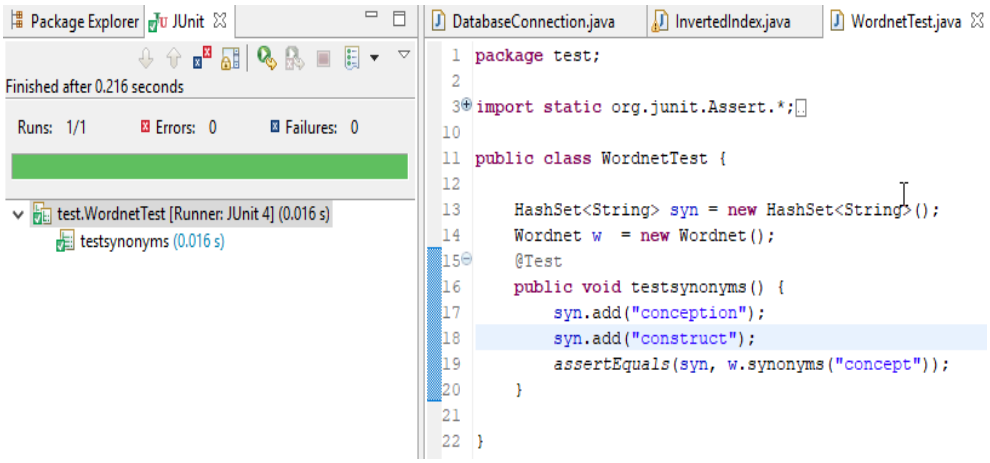


Figure 9: JUnit Testing

The COS2 system is designed using Test-driven development (TDD) approach. This approach helps to manage the quality of the code. We have tested most of the essential libraries in COS2 to provide a full-test coverage and eliminate manual testing during continuous deployments of added features. To examine our code, we have applied JUnit test tool in Java. COS2 strictly follows the TDD best practices given by (Langr et al.; 2015):

- Write tests.
- Run these tests, they fail as there is still no implementation.
- Write minimal code just for the test to pass.
- Run tests again, they pass.
- Revise and repeat the process.

The JUnit test cases are verified by reflection. We use the name convention with Test suffix for all the testing classes and place them under the test package for instant access. The test cases are labelled with @Test annotations. Assertions such as assertSame() and assertEquals() are employed to compare expected results with actual output. JUnit test cases can be developed quickly giving instance feedback on the code. A basic code snippet of the Test cases for the Wordnet Library is illustrated in figure 9.

5 Evaluation

5.1 Challenges and Key Findings

The COS2 application runs successfully with datasets of varying sizes. It is also possible to create indexes with datasets in languages other than English. However, the query processing module fails to evaluate non-English searches and gives no results. Another point to note here is all the datasets used for testing include .txt files with plaintext words. But an ideal search engines would work with a combination of _index and _data inputs. The _index consists of all crawled webpages and _data stores information regarding them. It was difficult to find free source datasets in these formats. One such corpus was identified in the common crawler dataset (Nagel and Crouse; 2018). Common crawler requires an overhead implementation to import data from their WART files which would be difficult to develop in this short timespan. Yet a plaintext search engine was developed based on similar grounds. Constructing a LRU cache was tough as the recently accessed pages were to be maintained on the top of the queue. A normal write to the file would not suffice and hence a PriorityQueue was used to surpass this challenge. Another limitation is providing continuous access to the data and the other files. The availability of the system solely depends on that provided by the cloud supplier. In case of provider downtime or loss of data, appropriate backup and recovery mechanisms should be put in place. When handling terabytes or petabytes of data, compression and external merge sort algorithms should be deployed so that the system does not run out of memory and the index is assembled in chunks. Additionally, instead of opting for Relational database login, JSON based login structure should be put in place with subscription and uploads lists. Also with the introduction of GDPR necessary policy measures should be evaluated.

5.2 Complexity Analysis

Most of the datastructures used to implement the COS2 system belong to the Java Collections class. The Index is a HashMap which has O(1) get and contains complexity in the worst case scenario. URL Lists and Lexicons are computed through Hashset which again has constant complexity. The polynomial modelling generates an ArrayList which has complexity of O(N). The encryption techniques used in the COS2 system are fixed sized cipher blocks, hence they evaluate to O(1). PriorityQueue for caching has search complexity of O(n). The Wordnet defines dictionary in form of Hashtable which has constant complexity. The table 3 shows the complexity comparison of COS2 with the previous similarity algorithm. Clearly, COS2 is better as opposed to prior search techniques as it creates the index as well as evaluates the search in O(N) time.

Module	Input Size	OS2 (Pervez et al.; 2016)	COS2
User Registration	N	-	O(N)
Index Building	N	O(N)	O(N)
Data Outsourcing	N	O(N)	O(N)
Query Processing	N	$O(n^2.N)$	O(N)

Table 3: Complexity Analysis

5.3 Performance analysis

The search response time of COS2 application with and without cache is utilized to evaluate the performance. 60 random searches are generated by using the three datasets with 20 keywords from each set. The time logs

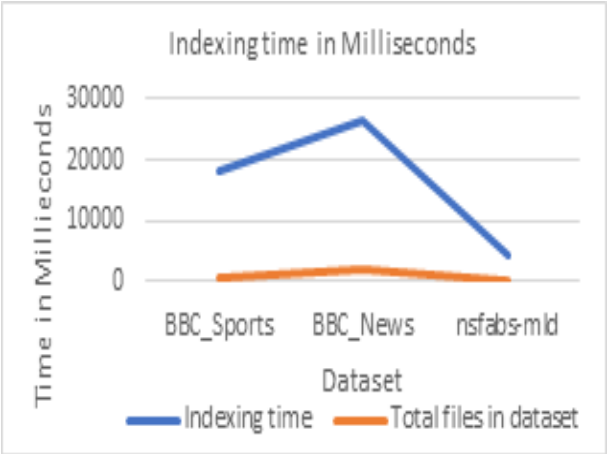


Figure 10: Performance of Index Builder

Figure 11: Search response time in Milliseconds

Dataset	Without cache	With cache
BBC_Sports	12875.90	4092.11
BBC_News	71958.65	5108.75
nsfabs-mld	3768.7	2799.6

Table 4: Average search performance

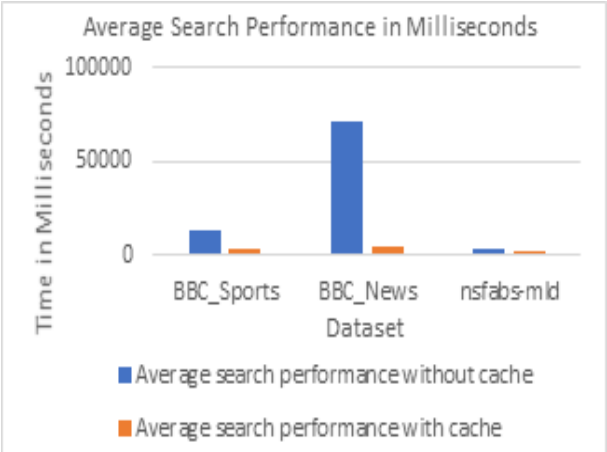


Figure 12: Average search performance in Milliseconds

for search processing and index building are noted using Java time logger. The three datasets are of varying sizes. The impact of total number of documents on the index building time is illustrated in graph 10. We

can see that the indexing time largely depends on the total number of files in the dataset. Further, the search time noted for the 60 queries show that cache reduces the response time considerably 11. These 60 searches can be considered to be equivalent to 60 users querying at the same time. The average search performance for the three datasets are shown in the table 4 and the corresponding graph is depicted in 12. Caching makes the COS2 systems 7 times faster in comparison to the old system and an overall performance improvement of 86% is achieved.

6 Conclusion and Future work

The accessibility of relevant contents through query processing and information retrieval is essential for data dependent services. The implemented COS2 system proves to be feasible search engine for encrypted data over cloud storage. The proposed COS2 system can derive the best results based on the user search query. It was successful in leveraging the benefits of the previous similarity search algorithm presented by Pervez et al. (2016). COS2 provides both normal as well as relevant results for a query in minimum response time. The initiation of cache to conclude contextual search increases the performance of the system tremendously. An 86% improvement in terms of search efficiency is prominent from testing and evaluation of the system. Intensive encryption techniques utilized assist in ensuring the privacy of the data and commencing the oblivious search. Since COS2 does not use trusted third parties to conclude the search, it thereby reduces the overall computational requirements of the system. Besides, typographical errors in search is handled effectively which improves the search experience by few notches. As future enhancements, conjunctive and disjunctive searches with multiple keywords can be incorporated. Also, the search results can be ranked using (TF)X(IDF) to consider frequency of the search in the retrieved documents. In revised versions of COS2, the current caching mechanism can be evolved to consider the subscriber location. Further, the COS2 application can be deployed in hybrid cloud environment with the application acting as middleware-as-a-service or as a deployment on top of the in-house private cloud. Thus, in coming years, COS2 system has the potential of acting as an ideal search engine for encrypted data by satisfying the user demands with the best possible outcomes.

References

Amazon (2006). Amazon s3, `https://aws.amazon.com/s3/`.

Amazon (2009). Amazon rds, `https://aws.amazon.com/rds/`.

Amazon (2018). Aws service level agreement, `https://aws.amazon.com/de/legal/service-level-agreements/`.

Asharov, G., Naor, M., Segev, G. and Shahaf, I. (2016). Searchable symmetric encryption: Optimal locality in linear space via two-dimensional balanced allocations, *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, ACM, Cambridge, MA, USA, pp. 1101–1114. Core Rank: A*, doi: 10.1145/2897518.2897562.

Bai, B., Sadamasa, K., Weston, J., Qi, Y., Grangier, D., Cortes, C., Collobert, R. and Mohri, M. (2009). Polynomial semantic indexing, *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, NIPS'09, Curran Associates Inc., Vancouver, British Columbia, Canada, pp. 64–72. Core Rank:A*.

Biaecki, A., Muir, R. and Ingersoll, G. (2012). Apache lucene 4, *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*, ACM, Portland, Oregon, USA, pp. 17–24. Core Rank: A*, doi: 10.1145/2422256.2422269.

Boldyreva, A. and Chenette, N. (2015). Efficient fuzzy search on encrypted data, *in* C. Cid and C. Rechberger (eds), *Fast Software Encryption*, Springer, Istanbul, Turkey, pp. 613–633. Core Rank: B, doi: 10.1007/978-3-662-46706-0_31.

Cao, N., Wang, C., Li, M., Ren, K. and Lou, W. (2014). Privacy-preserving multi-keyword ranked search over encrypted cloud data, *IEEE Transactions on Parallel and Distributed Systems* **25**(1): 222–233. Core Rank: B, doi: 10.1109/TPDS.2013.45.

Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C. and Steiner, M. (2013). Highly-scalable searchable symmetric encryption with support for boolean queries, *in* R. Canetti and J. A. Garay (eds), *Advances in Cryptology - CRYPTO 2013*, Springer Berlin Heidelberg, Santa Barbara, California, USA, pp. 353–373. Core Rank: A*, doi: 10.1007/978-3-642-40041-4_20.

Chang, R. M., Kauffman, R. J. and Kwon, Y. (2014). Understanding the paradigm shift to computational social science in the presence of big data, *Decision Support Systems* **63**: 67–80. Core Rank: A*, doi: 10.1016/j.dss.2013.08.008.

Cheng, R., Yan, J., Guan, C., Zhang, F. and Ren, K. (2015). Verifiable searchable symmetric encryption from indistinguishability obfuscation, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, ACM, Singapore, Republic of Singapore, pp. 621–626. Core Rank: B, doi: 10.1145/2714576.2714623.

Christiane, F. (2005). *WordNet: An Electronic Lexical Database.*, 2.1 edn, Princeton University, Princeton, NJ, USA. Retrieved from: `https://wordnet.princeton.edu/`.

Cisco (2016). Forecast number of personal cloud storage consumers/users worldwide from 2014 to 2020 (in millions), `https://www.statista.com/statistics/499558/worldwide-personal-cloud-storage-users/`.

D. Boneh, G. Di Crescenzo, R. O. and Persiano, G. (2004). Public key encryption with keyword search, Vol. 3024, Springer, Copenhagen, Denmark, pp. 506–522. Core Rank: C, doi: 10.1007/978-3-540-24676-3_30.

Dropbox (2014). Dropbox status update, `https://blogs.dropbox.com/tech/2014/01/dropbox-status-update/`.

Esposito, C., Ficco, M., Palmieri, F. and Castiglione, A. (2015). A knowledge-based platform for big data analytics based on publish/subscribe services and stream processing, *Knowledge-Based Systems* **79**: 3–17. Core Rank: B , doi: 10.1016/j.knosys.2014.05.003.

Google (2002). Google search appliance, `https://enterprise.google.com/search/products/gsa.html`.

Google (2017). Privacy and terms, `https://policies.google.com/privacy?hl=en&gl=IE`.

Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering, *Proc. 23rd International Conference on Machine learning (ICML'06)*, ACM Press, Dublin, Ireland, pp. 377–384. Core Rank: A*, doi: 10.1145/1143844.1143892.

Hahn, F. and Kerschbaum, F. (2014). Searchable encryption with secure and efficient updates, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, ACM, Scottsdale, Arizona, USA, pp. 310–320. Core Rank: A*, doi: 10.1145/2660267.2660297.

Horstmann, C. S. and Cornell, G. (2002). *Core Java 2: Volume I, Fundamentals, Sixth Edition*, 6th edn, Pearson Education. This is a book published by Pearson Education with total of 633 citations.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality, *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, ACM, Dallas, Texas, USA, pp. 604–613. Core Rank: A*, doi: 10.1145/276698.276876.

Ishai, Y., Kushilevitz, E., Lu, S. and Ostrovsky, R. (2016). Private large-scale databases with distributed searchable symmetric encryption, *Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016*, Vol. 9610, Springer, San Francisco, CA, USA, pp. 90–107. Core Rank: B, doi: 10.1007/978-3-319-29485-8_6.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage, *Proceedings of the 14th International Conference on Financial Cryptograpy and Data Security*, FC'10, Springer, Tenerife, Canary Islands, Spain, pp. 136–149. Core Rank: B, doi:10.1007/978-3-642-14992-4_13.

Kamara, S., Papamanthou, C. and Roeder, T. (2011). Cs2: A searchable cryptographic cloud storage system, *Technical report*, Microsoft. Retrieved from: `https://www.microsoft.com/en-us/research/publication/cs2-a-searchable-cryptographic-cloud-storage-system/`.

Kambatla, K., Kollias, G., Kumar, V. and Grama, A. (2014). Trends in big data analytics, *Journal of Parallel and Distributed Computing* **74**(7): 2561–2573. Core Rank: A*, doi: 10.1016/j.jpdc.2014.01.003.

Kandukuri, B. R., Paturi, V. R. and Rakshit, A. (2009). Cloud security issues, *Proceedings of the 2009 IEEE International Conference on Services Computing*, SCC'09, IEEE Press, Washington DC, USA, pp. 517–520. Core Rank: A, doi: 10.1109/SCC.2009.84.

Kuzu, M., Islam, M. S. and Kantarcioglu, M. (2012). Efficient similarity search over encrypted data, *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, IEEE Computer Society, Washington, DC, USA, pp. 1156–1167. Core Rank: A*, doi: 10.1109/ICDE.2012.23.

Langr, J., Hunt, A. and Thomas, D. (2015). *Pragmatic Unit Testing in Java 8 with JUnit*, Pragmatic Bookshelf.

Lawson, C. E. (2015). Cascading exploitations: The celebrity nude photo hack as digital phenomenon, *The 16th Annual Meeting of the Association of Internet Researchers*, Vol. 5, Phoenix, AZ, USA, pp. 21–24. Core Rank: C.

Leong, L., Bala, R., Lowery, C. and Smith, D. (2017). Magic quadrant for cloud infrastructure as a service, worldwide, `https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519&st=sb`.

Li, J., Li, J., Chen, X., Liu, Z. and Jia, C. (2014). Privacy-preserving data utilization in hybrid clouds, *Future Generation Computer Systems* **30**: 98–106. Core Rank: A, doi: 10.1016/j.future.2013.06.011.

Li, M., Yu, S., Cao, N. and Lou, W. (2011). Authorized private keyword search over encrypted data in cloud computing, *2011 31st International Conference on Distributed Computing Systems*, Minneapolis, Minnesota, USA, pp. 383–392. Core Rank: A, doi: 10.1109/ICDCS.2011.55.

Luciano, S. and John, Y. (2014). Cloud usage: Risks and opportunities, *Technical report*, Cloud Security Alliance. Retrieved from: `https://cloudsecurityalliance.org/download/cloud-usage-risks-and-opportunities-survey-report/`.

Luis, v. A., Ben, M., Colin, M., Harshad, B. and Manuel, B. (2007). *WordNet: An Electronic Lexical Database.*, 3.0 edn, Google, Menlo Park, California, USA. Retrieved from: `https://www.google.com/recaptcha/intro/v3beta.html`.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R. and Roxburgh, C. (2011). Big data: The next frontier for innovation, competition,and productivity, *Technical report*, McKinsey Global Institute. Retrieved from: `https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation`.

Massimo, M. (2012). Contextual search: A computational framework, *Foundations and Trends in Information Retrieval*, Vol. 6, NOW Publishers Inc., Hanover, MA, USA, pp. 257–405. Core Rank:C, doi: 10.1561/1500000023.

Meharwade, A. and Patil, G. (2016). Efficient keyword search over encrypted cloud data, Vol. 78, Procedia Computer Science, Nagpur, India, pp. 139–145. Core Rank: C, doi: 10.1016/j.procs.2016.02.023.

Microsoft (2001). Enterprise search server solutions, `https://products.office.com/en-IE/sharepoint/collaboration?ms.officeurl=sharepoint`.

Nagel, S. and Crouse, S. (2018). *So youre ready to get started.*, cc-main-2018-13 edn, Common Crawl, Sydney, Australia. Retrieved from: `http://commoncrawl.org/the-data/get-started/`.

NIST (2001). Announcing the advanced encryption standard (aes), *Federal Information Processing Standards Publication 197*, pp. 46–53. Core Rank: A, doi: FIPS/NIST.FIPS.197.

NIST (2012). Secure hash standard (shs), *Federal Information Processing Standards Publication 180-4*. Core Rank: A, doi: 10.6028/NIST.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes, *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, Springer, Prague, Czech Republic, pp. 223–238. Core Rank: A*, doi: 10.1007/3-540-48910-X_16.

Park, K. W., Han, J., Chung, J. and Park, K. H. (2013). Themis: A mutually verifiable billing system for the cloud computing environment, *IEEE Transactions on Services Computing* **6**(3): 300–313. Core Rank: A, doi: 10.1109/TSC.2012.1.

Pervez, Z., Ahmad, M., Khattak, A. M., Lee, S. and Chung, T. C. (2016). Privacy aware relevant data access with semantically enriched search queries for untrusted cloud storage services, *PLOS ONE* **11**(8): 1–20. Google Scholar Rank: 25, doi: 10.1371/journal.pone.0161440.

Pervez, Z., Ahmad, M., Khattak, A. M., Ramzan, N. and Khan, W. A. (2017). Os2: Oblivious similarity based searching for encrypted data outsourced to an untrusted domain, *PLOS ONE* **12**(7): 1–22. Google Scholar Rank: 25, doi: 10.1371/journal.pone.0179720.

Pervez, Z., Awan, A. A., Khattak, A. M., Lee, S. and Huh, E. N. (2013). Privacy-aware searching with oblivious term matching for cloud storage, *The Journal of Supercomputing* **63**(2): 538–560. Core Rank: B, doi: 10.1007/s11227-012-0829-z.

Pettey, C. and Goasduff, L. (2017). Gartner says worldwide public cloud services market to grow 18 percent in 2017, https://www.statista.com/statistics/499558/worldwide-personal-cloud-storage-users/.

Rev, B. (2012). Cloud computing benefits, risks and recommendations for information security, *Technical report*, European Network and Information Security Agency. Retrieved from: https://resilience.enisa.europa.eu/cloud-security-and-resilience/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security.

Sakr, S. and Liu, A. (2012). Sla-based and consumer-centric dynamic provisioning for cloud databases, *2012 IEEE Fifth International Conference on Cloud Computing*, IEEE Computer Society, Washington, DC, USA, pp. 360–367. Core Rank: B, doi: 10.1109/CLOUD.2012.11.

Sanchez, G. (2015). Case study: Critical controls that sony should have implemented, *Whitepaper*, SANS Institute. Retrieved from: https://www.sans.org/reading-room/whitepapers/casestudies/case-study-critical-controls-sony-implemented-36022.

Singh, A., Srivatsa, M. and Liu, L. (2009). Search-as-a-service: Outsourced search over outsourced storage, *ACM Trans. Web* **3**(4): 1–33. Core Rank: B, doi: 10.1145/1594173.1594175.

Song, D. X., Wagner, D. and Perrig, A. (2000). Practical techniques for searches on encrypted data, *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, Berkeley, CA, USA, pp. 44–55. Core Rank: Core A*, doi: 10.1109/SECPRI.2000.848445.

Stallings, W. (2010). *Cryptography and Network Security Principles and Practices*, Prentice Hall Press, Upper Saddle River, NJ, USA, chapter 5,12, pp. 134–160,353–358. This is a book published by Prentice Hall with total of 9694 citations.

Statista (2017). Data storage supply and demand worldwide, from 2009 to 2020 (in exabytes), https://www.statista.com/statistics/751749/worldwide-data-storage-capacity-and-demand/.

Sun, W., Lou, W., Thomas Hou, Y. and Li, H. (2014). *Secure Cloud Computing*, Springer, New York, NY, chapter 9, pp. 189–212. This is a book published by Springer with total of 19 citations.

Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y. T. and Li, H. (2013). Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, ACM, Hangzhou, China, pp. 71–82. Core Rank: B, doi: 10.1145/2484313.2484322.

Tang, Q. (2013). Search in encrypted data: Theoretical models and, *Theory and Practice of Cryptography Solutions for Secure Information Systems* pp. 84–108.

Thorne, B. (2017). *A Java library for Paillier partially homomorphic encryption.*, 0.6.0 edn, Data61, Sydney, Australia. Retrieved from: https://github.com/n1analytics/javallier.

UCI (2003). Nsf research award abstracts 1990-2003 data set, https://archive.ics.uci.edu/ml/datasets/NSF+Research+Award+Abstracts+1990-2003.

Wang, C., Cao, N., Li, J., Ren, K. and Lou, W. (2010). Secure ranked keyword search over encrypted cloud data, *2010 IEEE 30th International Conference on Distributed Computing Systems*, Genoa, Italy, pp. 253–262. Core Rank: A, doi: 10.1109/ICDCS.2010.34.

Yahoo! (2018). Yahoo privacy center, https://policies.yahoo.com/us/en/yahoo/privacy/index.htm?redirect=no.

Yu, S., Wang, C., Ren, K. and Lou, W. (2010). Achieving secure, scalable, and fine-grained data access control in cloud computing, *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, IEEE Press, San Diego, California, USA, pp. 534–542. Core Rank: A*, doi: 10.1109/INFCOM.2010.5462174.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines, *ACM Comput. Surv.* **38**(2). Core Rank: A*, doi: 10.1145/1132956.1132959.