International Journal of Engineering Research & Technology (IJERT)
NCITSF`14 Conference Proceedings
ISSN: 2278-0181

Secure Multikeyword Top-k Retrieval of Encrypted Cloud Data

Rachitha B S
4th Sem M.Tech, Department of CSE
AMC Engineering College
Bangalore, India.
rachitha152@gmail.com

Bharathi R
Associate Professor, Department of CSE
AMC Engineering College
Bangalore, India.

Abstract— **With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to the commercial public cloud for great flexibility and economic savings. But for protecting data privacy, sensitive data has to be encrypted before outsourcing, which obsoletes traditional data utilization based on a plaintext keyword search. Searchable symmetric encryption (SSE) allows retrieval of encrypted data over cloud. Addressing data privacy issues using SSE are done. The privacy issue is formulated from the aspect of similarity relevance and scheme robustness. Server-side ranking leaks data privacy. To eliminate the leakage, a two-round searchable encryption (TRSE) scheme is proposed that supports top-k multikeyword retrieval. In TRSE, a vector space model and homomorphic encryption are employed. The vector space model helps to provide sufficient search accuracy, and the homomorphic encryption enables users to involve in the ranking while the majority of computing work is done on the server side, by operations only on the ciphertext. As a result, information leakage can be eliminated and data security is ensured.**

IndexTerms*— *Cloud, data privacy, homomorphic encryption, ranking, similarity relevance, vector space model.

I. Introduction

Cloud computing [1], enables cloud customers to remotely store their data into the cloud so as to get the on-demand high quality applications and services from a shared pool of configurable computing resources. With the prevalence of cloud services, more and more sensitive information are being centralized into the cloud servers, such as emails, personal health records, private videos and photos, company financial data, government documents, etc. Reports of data loss and privacy breaches in cloud computing systems appear from time to time [2], [3]. The main threat on data privacy roots in the cloud itself [4]. When users outsource their private data into the cloud, the cloud service providers are able to control and monitor the data and the communication between users and the cloud at will, lawfully or unlawfully. To protect data privacy, sensitive data has to be encrypted before outsourcing [5] so as to provide end-to-end data confidentiality assurance in the cloud. However, data encryption makes effective data utilization a very challenging task given that there could be a large amount of outsourced data files. Besides, in Cloud Computing, data owners may share their outsourced data with a large number of users, who might want to only retrieve certain specific data files they are interested in during a given session. One of the most popular ways to do so is through keyword-based search. Such keyword search technique allows users to selectively retrieve files of interest and has been widely applied in plaintext search scenarios [6], in which users retrieve the relevant files in a file set based on keywords. However, it turns out to be a difficult task in ciphertext scenario due to limited operations on encrypted data. Besides, to improve feasibility and save on the expense in the cloud paradigm, it is preferred to get the retrieval result with the most relevant files that match users' interest instead of all the files, which indicates that the files should be ranked in the order of relevance by users' interest and only the files with the highest relevances are sent back to users

Although a series of searchable symmetric encryption (SSE) schemes have been proposed to enable search on ciphertext. Traditional SSE schemes enable users to securely retrieve the ciphertext, but these schemes support only Boolean keyword search, i.e., whether a keyword exists in a file or not, without considering the difference of relevance with the queried keyword of these files in the result. These schemes, however, suffer from two problems—Boolean representation and how to strike a balance between security and efficiency. In the former, files are ranked only by the number of retrieved keywords, which impairs search accuracy. In the latter, security is implicitly compromised to tradeoff for efficiency, which is particularly undesirable in security-oriented applications. Preventing the cloud from involving in ranking and entrusting all the work to the user is a natural way to avoid information leakage. However, the limited computational power on the user side and the high computational overhead precludes information security. The issue of secure multikeyword top-k retrieval over encrypted cloud data, thus, is: How to make the cloud do more work during the process of retrieval without information leakage. The concepts of similarity relevance and scheme robustness is introduced to formulate the privacy issue in searchable encryption schemes, and then solve the insecurity problem by proposing a two-round searchable encryption (TRSE) scheme. Novel technologies in the cryptography community and information retrieval (IR) community are employed, including homomorphic encryption and the vector space model. In the proposed scheme, the majority of computing work is done on the cloud while the user takes part in the ranking, which guarantees top-k multikeyword retrieval over encrypted cloud data with high security and practical efficiency.

II. Preliminaries

A. Scenario

Consider a cloud computing system hosting data service, as illustrated in Fig. 1, in which three different entities are involved: cloud server, data owner, and data user. The cloud server hosts third-party data storage and retrieve services. Since data may contain sensitive information, the cloud servers cannot be fully entrusted with protecting data. For this reason, outsourced files must be encrypted. Any kind of information leakage that would affect data privacy are regarded as unacceptable.

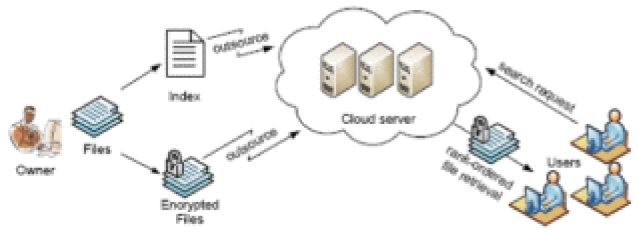

Fig.1 Architecture of search over encrypted cloud data

The data owner has a collection of n files *C*=$\{f1,f2,\dots fn\}$ to outsource onto the cloud server in encrypted form and expects the cloud server to provide keyword retrieval service to data owner himself or other authorized users. To achieve this, the data owner needs to build a searchable index I from a collection of *l* keywords $W=\{w1,w2,\dots wl\}$ extracted out of *C*, and then outsources both the encrypted index *I'* and encrypted files onto the cloud server.

The data user is authorized to process multikeyword retrieval over the outsourced data. The computing power on the user side is limited, which means that operations on the user side should be simplified. The authorized data user at first generates a query $REQ = \{(w1', w2', \dots ws')|wi' \varepsilon W, 1< i< s< l\}$. For privacy consideration, which keywords the data user has searched must be concealed. Thus, the data user encrypts the query and sends it to the cloud server that returns the relevant files to the data user. Afterward, the data user can decrypt and make use of the files.

B. Relevance Scoring

Some of the multikeyword SSE schemes support only Boolean queries, i.e., a file either matches or does not match a query. Considering the large number of data users and documents in the cloud, it is necessary to allow multikeyword in the search query and return the documents in the order of their relevancy to the queried keywords. Scoring is a natural way to weight the relevance. Based on the relevance score, files can then be ranked in either ascendingly or descendingly. Several models have been proposed to score and rank files in IR community. Among these schemes, the most widely used one tf-idf weighting, which involves two attributes :*term frequency* and *inverse document frequency*.

Term frequency (tf t,f) denotes the number of occurrences of term t in file f. Document frequency (df t) refers to the number of files that contains term t, and the inverse document frequency (idf t) is defined as: idft = log N/dft, where N denotes the total number of files. Then, the tf-idf weighting scheme assigns to term t a weight in file f given by tf-idf t,f = tf t,f *idf t. By introducing the IDF factor, the weights of terms that occur very frequently in the collection are diminished and the weights of terms that occur rarely are increased.

C. Vector Space Model

While tf-idf depicts the weight of a single keyword on a file, a vector space model is employed to score a file on multikeyword. The vector space model [7] is an algebraic model for representing a file as a vector. Each dimension of the vector corresponds to a separate term, i.e., if a term occurs in the file, its value in the vector is nonzero, otherwise is zero. The vector space model supports multiterm and nonbinary presentation.

Moreover, it allows computing a continuous degree of similarity between queries and files, and then ranking files according to their relevance. It meets the need of top-k retrieval. A query is also represented as a vector q, while each dimension of the vector is assigned with 0 or 1 according to whether this term is queried. The score of file f on query q(score f,q) is deduced by the inner product of the two vectors: score f,q = vf ,q. Given the scores, files can be ranked in order and, therefore, the most relevant files can be found.

III. Problem Statement

When users outsource their private data on cloud, they encrypt the data to ensure privacy. This results in great challenges to effective data utilization. A series of searchable symmetric encryption (SSE) schemes have been proposed to enable search on the ciphertext. Traditional SSE schemes enable users to securely retrieve the ciphertext, but these schemes support only Boolean keyword search, i.e. whether a keyword exists in a file or not, without considering the difference of relevance to the queried keyword of these files in the result. To improve security without sacrificing efficiency, some schemes support top-k single keyword retrieval under various scenarios. These schemes, however, suffer from two problems—Boolean representation and how to strike a balance between security and efficiency. In the former, files are ranked only by the number of retrieved keywords, which impairs search accuracy. In the latter, security is implicitly compromised to tradeoff for efficiency, which is particularly undesirable in security-oriented applications.

A. Statistic Leakage

Although all data files, indices, and requests are in encrypted form before being outsourced onto the cloud, the cloud server can still obtain additional information through statistical analysis. The possible information leakage is denoted by statistic leakage. There are two possible statistic leakages, including term distribution and interdistribution. The term distribution of term t is t's frequency distribution of scores on each file *i(i* Ɛ *C).* The interdistribution of file f is file f's frequency distribution of scores of each term *j(j* Ɛ *f).* Term distribution and interdistribution are specific [8]. They can be deduced either directly from the ciphertext or indirectly via statistical analysis over access and search pattern [9].

Here, access pattern refers to which keywords and the corresponding files have been retrieved during each search request, and search pattern refers to whether the keywords retrieved between two requests are the same. Distribution information implies a similar relationship between terms or files. On the one hand, terms with similar term distribution always have simultaneous occurrence. For instance, obviously, the term “states” are very likely to co-occur with “united” in an official document from the White House, and their term distribution, are very same in a series of such a kind of document. The document is encrypted but term distribution is not concealed, once an adversary somehow cracks out the plaintext of “united,” then the adversary can reasonably guess the term that shares a similar term distribution with “united” may be “states.” On the other hand, files with similar interdistribution are always the same category, eg. two medical records from a dental office surely are the same category, and they are very likely to share a similar interdistribution (such as the titles of each entry is the same). Therefore, this specificity should be hidden from an untrusted cloud server.

IV. TRSE DESIGN

Existing SSE schemes employ server-side ranking to improve the efficiency of retrieval over encrypted cloud data. However, server-side ranking violates the privacy of sensitive information, which is considered uncompromisable in the security-oriented third party cloud computing scenario, i.e., security cannot be tradeoff for efficiency. To achieve data privacy, ranking has to be left to the user side. Traditional user-side schemes, however, load heavy computational burden and high communication overhead on the user side, due to the interaction between the server and the user including searchable index return and ranking score calculation. Thus, the user-side ranking schemes are challenged by practical use. A more server-siding scheme might be a better solution to privacy issues.

A new searchable encryption scheme is proposed, in which novel technologies in cryptography community and IR community are employed, including homomorphic encryption and the vector space model. In the proposed scheme, the data owner encrypts the searchable index with homomorphic encryption. When the cloud server receives a query consisting of multikeywords, it computes the scores from the encrypted index stored on the cloud and then returns the encrypted scores of files to the data user. Next, the data user decrypts the scores and picks out the top-k highest scoring files’ identifers to request to the cloud server. The retrieval takes a two-round communication between the cloud server and the data user. Thus the name Two Round Searchable Encryption scheme, in which ranking is done at the user side while scoring calculation is done at the server side.

A. Practical Homomorphic Encryption Scheme

To alleviate the computational burden on the user side, computing work should be on the server side, so there is a need of an encryption scheme to guarantee the operability and security at the same time on server side. Homomorphic encryption allows specific types of computations to be carried out on the corresponding ciphertext. The result is the ciphertext of the result of the same operations performed on the plaintext. That is, homomorphic encryption allows computation of ciphertext without knowing anything about the plaintext to get the correct encrypted result. Although it has such a fine property, the original fully homomorphic encryption scheme, which employs ideal lattices over a polynomial ring [10], is too complicated and inefficient for practical utilization. Fortunately, as a result of employing the vector space model to top-k retrieval, only addition and multiplication operations over integers are needed to compute the relevance scores from the encrypted searchable index. Therefore, the original homomorphism can be reduced in a full form to a simplified form that only supports integer operations, which allows more efficiency than the full form does.

In the fully homomorphic encryption over the integers(FHEI) scheme [11], the approximate integer greatest common divisor (GCD) is used to provide sufficient security, i.e., given a list of integers $I = \{i1, i2, \ldots, in\}$ that are approximate multiples of a hidden integer j, to find the hidden integer j. The approximate GCD problem has been proven hard by Howgrave-Graham [12]. Let m and c denote the plaintext and ciphertext of the integer, respectively. The encryption scheme can be expressed as the following formulation: $c = pq + 2r + m$, where p denotes the secret key, q denotes the multiple parameter, and r denotes the noise to achieve proximity against brute-force attacks. The public key is $pq + r$.

However, as the scores of items in file vector of searchable index Ip is multibit, the total size of Ic and the computed results will be very large due to the FHEI scheme encrypts 1 bit to $||p|| + ||q||$ bit (here $||p||$ refers to bit length of p, i.e., $||p|| = [\log p]$). To downsize the ciphertext and, thus, mitigate the communication overhead, the original FHEI scheme is modified more flexible to meet the needs: $c = pq + xr + m$, where $x = 2^{2||m||}$, $p >> r$, and $r >> x$ to ensure the correctness of the decryption. Since the size of the result will be doubled after multiplication, the noise parameter x is, thus, required to be at least $2^{2||m||}$. Therefore, multibit is considered as a unit for encryption, and the size of the ciphertext is significantly reduced, i.e., the size of the ciphertext can be reduced down to $1/||m||$ of that in original FHEI scheme. For example, assume the value of scores is up to 2^{10}, then the size of ciphertext will be $10(||p|| + ||q||)$ for encryption of each bit of m if applying the original FHEI scheme, while only $||p|| + ||q||$ in the modified FHEI scheme. The modified FHEI scheme guarantees homomorphism property according to the following theorem 1:

Theorem 1: *The modified FHEI scheme is homomorphic for addition and multiplication.*

Proof: Given two plaintext *m1,m2* and their corresponding ciphertext *c1,c2* by employing the modified FHEI scheme, where $ci = pqi + xri + mi$ $(i = 1, 2)$. Then,

$$c1+c2=(pq1+xr1+m1)+(pq2+xr2+m2)$$
$$=p(q1+q2)+x(r1+r2)+(m1+m2)) \quad (1)$$

$$c1.c2=(pq1+xr1+m1)+(pq2+xr2+m2)$$
$$=p\,q1q2+px\,(q1r2+q2r1)+p(q1m2+q2m1)$$
$$+x\,r1r2+x(r1m2+r2m1)+m1m2. \quad (2)$$

Therefore (1) and (2) can be deduced as

$$((c1+c2)\ mod\ p)\quad mod\ x=m1+m2$$
$$((c1.c2)\ mod\ p)\quad mod\ x=m1.m2$$

Hence, theorem 1 is true.

On the basis of homomorphism property, the encryption scheme can be described as four stages: KeyGen, Encrypt, Evaluate, and Decrypt.

- *KeyGen*(λ): The secret key SK is an odd n-bit number randomly selected from the interval [2n-1,2n]. The set of public keys PK = *{k0,k1,... kn}*. Note that the secret key is used for encryption and the public keys are used for decryption, which are different from the concepts of keys in public-key cryptography.

- *Encrypt*(*PK ,m*): Randomly choose a subset R = *{1, 2, . . t}* and an integer and then return ciphertext *C = m + xr.*

- *Evaluate*(*C1, C2, . . . Ct):* Apply the binary addition and multiplication gates to the *t* ciphertext *Ci*, perform all necessary operations, and then return the resulting integer.

- *Decrypt*(*p,x*):The modified FHEI scheme is relatively time consuming so employ it to encrypt the searchable index *I*, while the file set *C* can be encrypted with other symmetric encryption scheme. Note that the Evaluate stage sets no limit to how many addition or multiplication operations can be executed without decryption. In fact, the ciphertext of an integer, which is another integer, can be applied to as many evaluations as needed.

B. Framework of TRSE

The framework of TRSE includes four algorithms: *Setup, IndexBuild, TrapdoorGen, ScoreCalculate,* and *Rank.*.

- *Setup*(λ): The data owner generates the secret key and public keys for the homomorphic encryption scheme. The security parameter λ is taken as the input, the output is a secret key SK, and a public key set *PK*.

- *IndexBuild*(*C, PK*):The data owner builds the secure, searchable index from the file collection *C*. Technologies from IR community like stemming are employed to build searchable index *I* from *C*, and then *I* is encrypted into *I'* with *PK*, output the secure searchable index *I'*.

- *TrapdoorGen*(*REQ, PK*): The data user generates secure trapdoor from his request *REQ*. Vector *T* is built from user's multikeyword request *REQ* and then encrypted into secure trapdoor *T'* with public key from *PK*, output the secure trapdoor *T'*.

- *ScoreCalculate*(*T',I'*): When receives secure trapdoor *T'*, the cloud server computes the scores of each files in *I'* with *T'* and returns the encrypted result vector *N* back to the data user.

- *Rank* (*N,SK*,k): The data user decrypts the vector *N* with secret key *SK* and then requests and gets the files with top-k scores.

λ is only involved in the Setup algorithm, and the Setup algorithm needs to be processed only once by the data owner, λ, thus, is a constant integer for one individual application instance. The whole framework can be divided into two phases: *Initialization* and *Retrieval*. The Initialization phase includes *Setup* and *IndexBuild*. The Setup stage involves the secure initialization, while the IndexBuild stage involves operations on plaintext. For security concerns, the vast majority of work should only be done by the data owner. Moreover, for convenience of retrieval, the original vector space model is modified by adding each vector *vi*, a head node *idi* at the first dimension of *vi* to store the identifier of *fi*. In this way, the correspondence between scores and files is established. The details of the Initialization phase are as follows:

Initialization Phase:

- The data owner calls KeyGen(λ) to generate the secret key SK and public key set PK for the homomorphic encryption scheme. Then the data owner assigns *SK* to the authorized data users.
- The data owner extracts the collection of *l* keywords, *W={w1, w2, . . . wl}*, and their TF and IDF values out of the collection of n files, *C={f1,f2,...fn}*. For each file *fi* in *C*, the data owner builds (*l*+1)-dimensional vector .
- The data owner encrypts the searchable index I to secure, searchable index *I'*.
- The data owner encrypts *C={f1,f2,...fn}* into *C'={f1',f2',...fn'}* with other cryptology schemes, and then outsources *C'* and *I'* to the cloud server.

The Retrieval phase involves *TrapdoorGen, ScoreCalculate,* and *Rank*, in which the data user and the cloud server are involved. As a result of the limited computing power on the user side, the computing work should be left to server side as much as possible. Meanwhile, the confidentiality, privacy of sensitive information cannot be violated. The ranking should be left to the user side while the cloud server still does most of the work without learning any sensitive information.

The details of the Retrieval phase are as follows:

Retrieval Phase:

- The data user generates a set of keywords *REQ={w1',w2',... wn'}*to search.
- For each file vector, the cloud server computes the inner product with modular reduction and then compresses and returns the result vector to the data user.
- The data user decrypts *N'* into *N*. The TOPKSELECT(*N*,k) is invoked to get the top-k highest-scoring files' identifiers *{i1,i2,....ik}*and sends it to the cloud server.
- The cloud server returns the encrypted k files *{f1,f2,...,fk}*to the data user.

```
Algorithm 1: TOPKSELECT(source,k)
Input:
 List source to be selected
 Number k
Initialization:
 Set topk ←null; topkid ← null;
Iteration:
 1: for all item Ɛ source do
 2: INSERT(topk,(item,itemindex))
 3: end for
 4: for all tuple Ɛ topk do
 5: topkid.append(tuple[1])
 6: end for
Output:
 Topkid
```

Fig 2(a) Algorithm TOPKSELECT

```
Algorithm 2: INSERT(topk,(item,itemindex))
Input:
  List topk to store the topk scoring item
  Tuple(item,itemindex)
Iteration:
 1: if len(topk)< k then
 2: insert (item,itemindex) into topk in nondecreasing order
of item
 3: else
 4: for all element Ɛ topk do
 5: if item < element[0] then
 6: continue
 7: else
 8: discard topk[0], insert (item,itemindex) into topk in
nondecreasing order of item
 9: endif
 10: end for
 11: end if
```

Fig 2(b) Algorithm INSERT

As a result of the limited computing power on the user side, complexity of ranking is considered. Since the decryption of *N* can be accomplished in O(n) time, the only function that could influence the time complexity of ranking is the top-k select algorithm, i.e., TOPKSELECT algorithm. The details of TOPKSELECT algorithm are shown in Fig. 2(a). Since the complexity of the INSERT algorithm is O(k), as illustrated in Fig. 2(b), the overall complexity of TOPKSELECT algorithm is O(nk). Note that k, which denotes the number of files that are most relevant to the user's interest, is generally very small compared to the total number of files. In case of large value of k, the complexity of the TOPKSELECT algorithm can be easily reduced to O(n log k) by introducing a fixed-size min-heap.

V. Related Work

Traditional searchable encryption are investigated in [8], [22], [23] focusing on security definitions and encryption efficiency, and these works support only Boolean keyword retrieval without ranking. Swaminathan et al. [24] explored secure rank-ordered retrieval with improved searchable encryption in the scenario of the data center. They built a framework for privacy-preserving top-k retrieval, including secure indexing and ranking. Zerr et al. [10] proposed a ranking model to guarantee privacy-preserving document exchange among collaboration groups, which allows for privacy-preserving top-k retrieval from an outsourced inverted index. They proposed a relevance score transformation function to make relevance scores of different terms indistinguishable and such that improves the security of the indexed data. Wang et al. [9] explored top-k retrieval over encrypted data in cloud computing. On the basis of SSE, they proposed the one-to-many OPM to further improve the efficiency while security guarantee and retrieval accuracy are slightly weakened. However, these schemes support only single keyword retrieval. Considering the large number of data users and documents in the cloud, it is necessary to allow multikeyword in the search request and return the most relevant documents in the order of their relevance with these keywords. Some existing works [20], [21] proposed several schemes supporting Boolean multikeyword retrieval. Cao et al. made the first attempt to define and solve the problem of top-k multikeyword retrieval over encrypted cloud data. They employed the concepts of coordinate matching and inner product similarity to measure and evaluate the relevance scoring. Hu et al. employed homomorphism to preserve the data privacy. They devised a secure protocol for processing k-nearest-neighbor (kNN) index query, thus preserving both the data privacy of the owner and the query privacy of the client. These two schemes employed Boolean representation in their searchable index, i.e., 1 denotes the corresponding term exists in the file and 0 otherwise. Thus, files that share queried keywords have the same score, a situation that is far from precise, thus, weakens the effectiveness of data utilization. Since all these server-side schemes employ server-side ranking, the security is compromised.

VI. Conclusion

The problem of secure multikeyword top-k retrieval over encrypted cloud data is solved. A server-side ranking SSE scheme is devised. A TRSE scheme is then proposed employing the fully homomorphic encryption, which fulfills the security requirements of multikeyword top-k retrieval over the encrypted cloud data. By security analysis, it can be shown that the proposed scheme guarantees data privacy.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58 2010.

[2] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmail-disasterreportsof- mass-email-deletions/, Dec. 2006.

[3] Amazon.com, "Amazon s3 Availability Event: July 20, 2008,"http://status.aws.amazon.com/s3-20080720.html, 2008.

[4] RAWA News, "Massive Information Leak Shakes Washington over Afghan War," http://www.rawa.org/temp/runews/2010/08/20/massive-information-leak-shakes-washington-overafghan-war.html, 2010.

[5] AHN, "Romney Hits Obama for Security Information Leakage," http://gantdaily.com/2012/07/25/romney-hits-obama-forsecurity-information-leakage/, 2012.

[6] Cloud Security Alliance, "Top Threats to Cloud Computing," http://www.cloudsecurity alliance.org, 2010.

[7] C. Leslie, "NSA Has Massive Database of Americans' PhoneCalls,"http://usatoday30.usatoday.com/news/washington/2006-05-10/, 2013.

[8] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," Proc. ACM 13th Conf. Computer and Comm. Security (CCS), 2006.

[9] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS), 2010.

[10] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," Proc. 12th Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT), 2009.

[11] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," Proc. 29th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques, H. Gilbert, pp. 24-43, 2010.

[12] M. Perc, "Evolution of the Most Common English Words and Phrases over the Centuries," J. Royal Soc. Interface, 2012.

[13] O. Regev, "New Lattice-Based Cryptographic Constructions," J. ACM, vol. 51, no. 6, pp. 899-942, 2004.

[14] N. Howgrave-Graham, "Approximate Integer Common Divisors," Proc. Revised Papers from Int'l Conf. Cryptography and Lattices (CaLC' 01), pp. 51-66, 2001.

[15] "NSF Research Awards Abstracts 1990-2003," http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html, 2013.

[16] "20 Newsgroups," http://kdd.ics.uci.edu/databases/ 20newsgroups/20newsgroups.html, 2013.

[17] S. Gries, "Useful Statistics for Corpus Linguistics," A Mosaic of Corpus Linguistics: Selected Approaches, Aquilino Sanchez Moises Almela, eds., pp. 269-291, Peter Lang, 2010.

[18] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proc. 41st Ann. ACM Symp. Theory of computing (STOC), pp. 169- 178, 2009.

[19] D. Dubin, "The Most Influential Paper Gerard Salton Never Wrote," Library Trends, vol. 52, no. 4, pp. 748-764, 2004.

[20] A. Cuyt, V. Brevik Petersen, B. Verdonk, H. Waadeland, and W.B. Jones, Handbook of Continued Fractions for Special Functions.

[21] Springer Verlag, 2008.

[22] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, pp. 856-887. MIT Press and McGraw-Hill, 2001.

[23] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," Proc. IEEE Symp. Security and Privacy, 2000.

[24] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persian, "Public- Key Encryption with Keyword Search," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (Euro crypt), 2004.