2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress

A Combined Solution for Conjunctive Keyword Search, Phrase Search and Auditing for Encrypted Cloud Storage

Hoi Ting Poon and Ali Miri
Department of Computer Science
Ryerson University
Toronto, Ontario, Canada
hoiting.poon@ryerson.ca, samiri@scs.ryerson.ca

***Abstract*—Cloud computing has garnered much interest in recent years for its many advantages, but also for its security and privacy concerns. The storage and access of confidential documents has been identified as one of the central problems in the area. Many researchers investigated solutions to search over encrypted documents stored on remote cloud servers. Others proposed schemes for ensuring data integrity or reduce overhead through deduplication while keeping the data encrypted and inaccessible by the cloud operator. While many schemes have been proposed to perform the individual functionalities, less attention has been made to more complete solutions featuring multiple desired functionalities. In this paper, we present a solution that incorporates search, phrase search and auditing where resources are reused for enabling each functionality, achieving an overall smaller storage cost and complexity than implementing each of the functionalities separately. The solution performs search over encrypted documents as efficiently as the leading phrase search scheme in the literature while also enabling unlimited number of audit queries.**

***Index Terms*—Conjunctive keyword search, Auditing, Privacy, Security, Encryption.**

I. Introduction

Cloud services provide a low cost and flexible solution to companies for managing computing resources. However, the out-sourcing of data also raises various security and privacy concerns. In recent years, many researchers have investigated solutions to various issues surrounding cloud storage services and improvements are continually being made. While there are significant efforts in addressing the individual problems in the area, few have looked at more complete solutions, combining the various desired functionalities. In this paper, we propose a solution for storing, auditing and searching through encrypted document sets on cloud storage. Instead of including resources that enable the functionalities separately, our setup uses the same pool of resources, leading to a smaller overhead than simply using two separate solutions. Our solution maintains many desirable features such as privacy of documents and search terms, proof of retrievability with theoretically unbounded number of audits and public verifiability.

The problem of search over encrypted data has been recognized as one of the most important issues towards the development of a secure and privacy-protected cloud storage system. Much of the challenge lies in the fact that the data is encrypted and not possessed by the data owner. To ensure privacy, the cloud service provider should not be able to learn any information on the data stored. However, mechanisms should also be in place to enable the cloud service provider to search the content at the user's request. Boneh et al. [1] proposed one of the earliest works on keyword searching. Their scheme uses public key encryption to allow keywords to be searchable without revealing data content. Waters et al. [2] investigated the problem for searching over encrypted audit logs. Many of the early works focused on single keyword searches. Recently, researchers have proposed solutions on conjunctive keyword search, which involves multiple keywords [3], [4]. Other interesting problems, such as the ranking of search results [5], [6] and searching with keywords that might contain errors [7], [8] termed fuzzy keyword search, have also been considered. The ability to search for phrases was also recently investigated [9], [10].

It's also generally agreed that auditing is needed for cloud storage services, in particular where they are used as data archives and backups, where they may not be accessed for long periods of time. Since the data is not in the owner's possession, verifying that the data is indeed available is not a simple task. Where privacy and security is concerned, the cloud service provider should also not learn the content of the encrypted data as a result of the auditing process. Ateniese et al. were among the first researchers to consider the problem of auditing encrypted cloud storage. They defined the notion of provable data possession (PDP) and proposed various solutions over the past decade [11], [12]. Their solution generally involves the addition of homomorphic tags, used for data verification. One of the common shortfalls of exisiting schemes such as in [11] is that there's often a limit on the number of audits that can be performed before the data have to be reprocessed. Shacham et al. [13] argued that the previous definition is also insufficient, in that it allows the verification of a subset of blocks without guaranteeing the availability of the data as a whole, and proposed the use of a stronger proof of retrievability (POR). Researchers have, over the years, devised

978-1-5090-2771-2/16 $31.00 © 2016 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.161

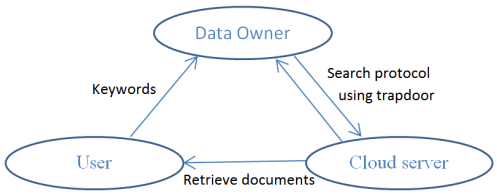

Fig. 1. Communication model for keyword search over encrypted data

various POR schemes, generally requiring the use of an erasure code to achieve the retrievability criterion without verifying all stored data. The resulting data expansion from the use of erasure code is the cost of achieving this stronger guarantee [14]. Wang et al. [15] investigated the possibility of a public auditing service, where a third party may perform the auditing task for the data owner. The challenge, however, is that the auditing service provider should not be considered any more trusted than the cloud service provider. As such, the auditing service provider must perform auditing without knowledge of the data content. To this end, Wang et al. proposed a scheme using public key encryption that could meet these requirements. Another recent work noted the vulnerability of verifying data ownership by knowledge of its signature that was identified in many cloud service at the time [16] and proposed two-way auditing where the user must also prove to the cloud its knowledge and, thus, ownership of the data.

II. Communication Model

For keyword search, our communication model involves up to three parties: The data owner, the cloud server and the user. Our discussions will assume the public cloud scenario involving all three parties. The algorithms can easily be adapted to the private cloud scenario where the user is simply the data owner. A standard keyword search protocol is shown in figure 1. The user begins by sending a search request containing the queried keywords to the data owner. To prevent the cloud from learning the keywords, the data owner computes and sends a trapdoor to the cloud to initiate a protocol to search for the requested keywords in the corpus. Finally, the cloud responds to the user with the indexes to the requested documents.

For auditing, the private model on which our solution is based on is shown in figure 2. The data owner begins by selecting and querying a set of keywords or rows from the indexes. Upon retrieving the results, the data owner randomly selects a number of bits to audit. The cloud must respond by providing the hash of the bit stream queried. Our solution can be used in a public auditing model, shown in figure 3, but a set of pre-computed challenge and signature response must be sent to the auditor.

In each scenario, the cloud operator is assumed to be semi-honest, following our protocol without deviation, but is interested in learning about stored user data. The auditor is also assumed to be semi-honest in the public auditing model.

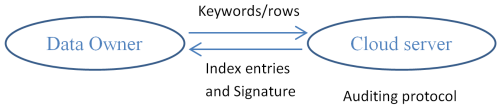

Fig. 2. Communication model for private auditing over encrypted data

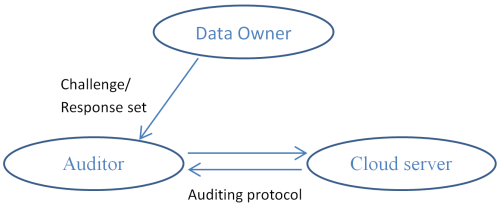

Fig. 3. Communication model for public auditing over encrypted data

III. Keyword and Phrase Search Schemes

Our combined audit and search solution works with any keyword and phrase search scheme that has the ability to query a keyword's location within a document and uses a symmetric encryption algorithm in counter mode. Examples include [17], [18], [9]. Without loss of generality, we'll describe a basic phrase search scheme here.

Given a document set, $D = \{D_1, D_2, \ldots, D_n\}$, each document, D_i, is parsed for a list of keywords, kw_j. An index, I, is generated mapping keywords to documents such that $I(kw_j) = \{d_{1,j}, d_{2,j}, \ldots, d_{n,j}\}$, where $d_i = 1$ if kw_j is found in the document. The index is then encrypted:

$$I(H_K(kw_j)) = \{E_K(d_{1,j}, d_{2,j}, \ldots, d_{n,j})\}. \quad (1)$$

where $H_K()$ is a cryptographically secure hash function. For each document, D_i, a keyword location index, I_l, is also generated containing entries:

$$\{H_K(D_i|kw_i), E_K(j_1, j_2, \ldots, j_n)\} \quad (2)$$

where j_x are the byte locations of kw_i within D_i. The documents are then encrypted using any symmetric encryption, such as AES, under counter mode and uploaded along with the encrypted indexes to the cloud server. The initialization vector is set to the hash of the document index, $H(D_i)$

To perform a standard keyword search, the user sends the queried keywords $kw' = \{kw_1, kw_2, \ldots, kw_q\}$ to the data owner. The data owner computes $H_K(kw')$ and sends to the cloud server. The cloud server returns the encrypted index entries to the data owner, who then finds the documents matching the keywords from the intersection of index entries:

$$D_K(I(H_K(kw_1))) \,\&\, D_K(I(H_K(kw_2))) \cdots \,\&\, D_K(I(H_K(kw_q))), \quad (3)$$

where $\&$ denotes a bitwise *and* operation.

To perform a phrase search, the user begins with a standard keyword search to identify candidate documents. Then,

the locations of the keywords in the candidate documents are queried by sending $H_K(D_i|kw_i)$ to the cloud. Upon decrypting the location information, the location of the first keyword in the phrase, kw_1, is extracted. For each location, $Loc(kw_1) = \{j_1, j_2 \ldots, j_m\}$, the data owner verifies if the following keyword is kw_2. For every positive match, the process repeats for the following keyword until the last keyword in the phrase is verified or until no candidates remain.

IV. Auditing protocol

To enable auditing, we add a row-based query mechanism to the basic phrase scheme described in previous section so that each row of the index, I_l, corresponding to a keyword, also includes $E_K(row_{no}|kw)$, where row_{no} is the row number of the keyword, kw, and, optionally, the row number can also be added to the keyword locations as follows: $E_K(row_{no}|j_1|j_2 \ldots)$. The latter provides earlier fault detection should a malicious cloud returns a different set of entries than asked. The data owner must also maintain a record of the number of rows available for each file's index. An audit query proceeds as follows: The auditor performs a keyword location or phrase search for a set of keywords, $kw_a = \{kw_{a1}, kw_{a2}, kw_{a3}, \ldots kw_{an}\}$, or randomly chooses a set of documents and performs a row-based query by sending a set of row numbers $row_a = \{r_{a1}, r_{a2}, r_{a3} \ldots, r_{an}\}$. The former has the advantage that the first step is simply a conjunctive keyword search and can be gathered based on user queries over time, without the added complexity of a row-based query mechanism. The latter has the advantage of not requiring knowledge over what keywords may be present in the document set and increasing the randomness of audits. In both cases, the server would return the encrypted locations for the set of keywords or the rows requested. Upon receiving an entry, $\{E_K(row_{no}|kw), E_K(j_1, j_2, \ldots, j_n)\}$, the first term is decrypted to verify that the row number or the keyword matches the one being queried. If there are discrepancies, the audit fails. Otherwise, the data owner is now aware of the locations of the keywords and can verify them by specifying a bit sequence belonging to a subset of the locations queried and requesting the cloud server to compute a hash signature of the bit sequence.

While it is possible to audit any bit sequence using this approach, it is useful to have a simple description of the locations to avoid having to specify the locations of every bit under audit. To this end, we chose to fix the number of bits to audit per keyword location. The following audit message is sent to the cloud server, describing the bits under audit: $\{run\_size, \{Doc_{ID_1}, byte_{loc_1}, byte_{loc_2}, \ldots byte_{loc_n}\}, \ldots \{Doc_{ID_n}, byte_{loc_1}, byte_{loc_2}, \ldots byte_{loc_n}\}\}$. Upon receipt, the server locates the document, Doc_{ID_1}, and retrieves run_size bytes at each byte location, $byte_{loc_i}$ for $i = 1$ to n, and place them in a buffer in order. Document Doc_{ID_2} is then processed in the same manner and the requested bytes added to the buffer until all bytes under audit are placed in the buffer. Aside from the bits under audit, the data owner also sends a salt value, $Salt$, which the cloud must also placed in the buffer. Finally, a hash signature of the bit sequence in the buffer is computed and returned. By decrypting $E_K(row_{no}|kw)$, the data owner knows the keywords at all the specified locations and can compute their corresponding ciphertexts. Because the auditor can recompute the ciphertexts, he can reproduce the buffer that the cloud server should obtain and verify that the signatures match. The inclusion of the salt serves to defend against replay attack should the same set of bit sequence gets audited at a later time.

To make use of a public auditor, the data owner must precompute sets of bit sequence, salt and signature and provide them to the auditor. An audit would be performed by sending the message describing the run_size and bit locations, then verifying the signature the cloud server responds with.

A. Modes of operation

The choice of symmetric encryption was natural due to its efficiency and well-studied security. For the purpose of our auditing protocol, it is also important that the data owner can systematically reproduce the ciphertext at any byte location.

While cipher-block chaining (CBC) is the most commonly used mode of operation for symmetric encryption, the data owner would require the preceding ciphertext block in order to compute the ciphertext at any location. This could significantly increase communication cost depending on the number of disjoint bit sequences under audit.

The choice of counter mode (CTR) is more suitable for our application as each ciphertext can be computed independent of other blocks, averting the communication cost of performing encryption in CBC mode. The initialization vector can be stored in plain with the document or use $H(D_i)$ to further reduce storage cost.

Furthermore, it is also possible to perform non-indexed keyword search under CTR mode. Detailed discussion can be found in [17].

B. Parameters for row and keyword selections

On average, English word length is approximately 5 characters, which translates to 40 bits. For a sample document set obtained from Project Gutenberg [19], keywords appear on average 15 times per document. Therefore, 600 bits on average would be available per row returned. Auditing 1024 kbits would require, on average, a minimum of 1707 rows. Suppose we request 3500 rows, we can then select $run\_size = 20$ bits at each keyword location. Then, for each keyword location, the auditor randomly chooses a starting bit location with an offset between 1 and $keyword\_length - 20$. The bits under audit are the 20 bits following the specified start location.

V. Analysis

In terms of storage, our scheme requires only that the encryption for each distinct keyword and the row number, $E_K(row_{no}|kw)$, be added in the keyword location indexes. The communication and computational complexity of keyword and phrase search is identical to the underlying phrase search schemes.

When compared to audit schemes, our solution uses encrypted indexes in place of tags and erasure codes. The storage cost of proof of retrievabiity schemes heavily depend on the recoverability of the erasure codes used and grows proportionally to the document set size. Our storage cost also increases as the document set size increases. However, as noted in [17], the probability distribution of natural languages is far from uniform. The number of distinct keywords, and so the size of the indexes used in our scheme, grows much slower than the document sizes do. The communication cost is similar to existing audit schemes in that the bits under audit must be specified in some way. The main computational cost during setup is that of generating the indexes for the entire document set. During audit, the data owner must decrypt the retrieved keywords and encrypt the audit bit sequence while the cloud hashes the requested bits.

Note that the number of audits allowed in our scheme is theoretically upper bounded only by the information available in the document set and the size of the salt chosen. Any set of bits in the document can be audited, provided that the keyword at the location is indexed. As such, indexing all words in the document set would allow our scheme to achieve proof of retrievability, provided the index itself is retrievable. This can be achieved by storing the index locally or using existing proof of retrievability schemes on the indexes. Note that the indexes are much smaller than the document set. Therefore, the data expansion and computational cost resulting from the use of erasure code would also be much lower.

In terms of security, the stored documents are symmetrically encrypted and the private key remains with the data owner at all times. The indexes are also symmetrically encrypted for privacy. In the public auditing model, the auditor is given a set of byte locations and the cryptographic hash of the bit sequence it should obtain. Since it is computationally infeasible to compute the pre-image of a cryptographic hash, the auditor would not be able to manipulate bit sequences to obtain different valid signatures. Bit locations also do not reveal information on the stored data. Attempts to store and replay audit queries by a malicious cloud operator would only be successful if exactly the same bit sequence and salt were chosen. Attempts to suggest the requested row do not exist would be caught since the data owner is aware of the number of rows available. Should the cloud operator send a set of locations for a different keyword than the one requested, it would also result in an audit failure since a different keyword would result in a different bit sequence or a mismatched row_{no} should it be included with the locations.

VI. Conclusion

We presented a keyword and phrase search scheme with auditing for encrypted documents stored on cloud servers. Our solution is based on encrypted indexes combined with the use of symmetric encryption in counter mode, along with fault detection mechanism. The main advantage of the proposed scheme is the reuse of encrypted indexes for both search and audit, leading to a smaller storage cost than implementing separate search and audit functionalities. Other than a small increase in storage cost, the scheme exhibits similar performance to the leading phrase search scheme in literature. Our solution also allows for auditing of any bits in a document set and, thus, the number of possible audits is bounded only by the amount of data in the stored documents and the chosen size of the salt.

References

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *In proceedings of Eurocrypt*, 2004, pp. 506–522.

[2] B. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *Network and Distributed System Security Symposium*, 2004.

[3] M. Ding, F. Gao, Z. Jin, and H. Zhang, "An efficient public key encryption with conjunctive keyword search scheme based on pairings," in *IEEE International Conference onNetwork Infrastructure and Digital Content*, 2012, pp. 526–530.

[4] F. Kerschbaum, "Secure conjunctive keyword searches for unstructured text," in *International Conference on Network and System Security*, 2011, pp. 285–289.

[5] C. Hu and P. Liu, "Public key encryption with ranked multi-keyword search," in *International Conference on Intelligent Networking and Collaborative Systems*, 2013, pp. 109–113.

[6] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Transactions on Consumer Electronics*, vol. 60, pp. 164–172, 2014.

[7] H. Tuo and M. Wenping, "An effective fuzzy keyword search scheme in cloud computing," in *International Conference on Intelligent Networking and Collaborative Systems*, 2013, pp. 786–789.

[8] M. Zheng and H. Zhou, "An efficient attack on a fuzzy keyword search scheme over encrypted data," in *International Conference on High Performance Computing and Communications and Embedded and Ubiquitous Computing*, 2013, pp. 1647–1651.

[9] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in *IEEE Global Communications Conference*, 2012, pp. 764–770.

[10] Y. Tang, D. Gu, N. Ding, and H. Lu, "Phrase search over encrypted data with symmetric encryption scheme," in *International Conference on Distributed Computing Systems Workshops*, 2012, pp. 471–480.

[11] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 598–609.

[12] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*

[13] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, 2008, pp. 90–107.

[14] ——, "Compact proofs of retrievability," *Journal of Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.

[15] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proceedings of the 14th European Conference on Research in Computer Security*, 2009, pp. 355–370.

[16] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011, pp. 491–500.

[17] H. Poon and A. Miri, "An efficient conjunctive keyword and phase search scheme for encrypted cloud storage systems," in *IEEE International Conference on Cloud Computing*, 2015.

[18] ——, "A low storage phase search scheme based on bloom filters for encrypted cloud services," to appear in *IEEE International Conference on Cyber Security and Cloud Computing*, 2015.

[19] "Project Gutenberg," https://www.gutenberg.org/wiki/Main_Page, accessed: 2014.