International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Development of Secure Multikeyword Retrieval Methodology for Encrypted Cloud Data

Deepak I M[1], K.R. Shylaja[2], Ravinandan M E[3]

[1]M.Tech IV Sem, Dept. of CSE, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India

[2]Associate Professor, Dept. of CSE, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India

[3]Teminnova Technologies Private Limited, Bangalore, Karnataka, India

Abstract: ***Cloud computing is a technology that uses the internet and central remote servers to maintain data and application. Concerns of sensitive information on cloud potentially cause privacy problems. So data encryption protects data security to some extent. For secure purpose of sensitive information on cloud, use vector space model and hybrid encryption algorithm (RSA and AES encryption-decryption) and OTP. Vector space model helps to provide sufficient search accuracy and the hybrid of RSA and AES algorithms for encryption of information. The OTP (One Time Password) for authentication. As a result, information leakage can be eliminated and data security is ensured.***

Keywords: Cloud, ranking, vector space model, RSA and AES encryption, OTP (One Time Password).

1. Introduction

Cloud computing the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. The rapid growth in field of "cloud computing" also increases severe security concerns. Security has remained a constant issue for Open Systems and internet, when talking about security cloud really suffers.

Lack of security is the only problem in wide adoption of cloud computing. Cloud computing is surrounded by many security issues like securing data, and examining the utilization of cloud by the cloud computing vendors. A deep in cloud computing has brought lots of security challenges for the consumers and service providers [2]. There are dangers in the cloud computing jungle and cloud security measures must be put in place to eliminate and resolve them. According to the Cloud Security Alliance, three types of threats have worsened between 2010 and today, data breaches, data loss, account or service traffic hijacking.

In cloud computing, data owners may share their outsourced data with a number of users, who might want to only retrieve the data files they are interested in. One of the most popular ways to do so is through keyword-based retrieval. Tf-idf [4] (Term Frequency and Inverse Document Frequency) depicts the weight of a single keyword on a file. So, have employed a vector space model to score a file on multikeyword. A vector space model for representing a file as a vector [5]. RSA and AES encryption over encrypted cloud data[1],[3]. In this paper, introduce the concept of OTP for authentication, hybrid algorithm of RSA-AES encryption over encrypted cloud data and also using term frequency and inverse document frequency for vector space model for search accuracy.

2. Preliminaries

2.1 Scenario

In which three different entities are involved in Figure.1, such as cloud server, data owner, data user.

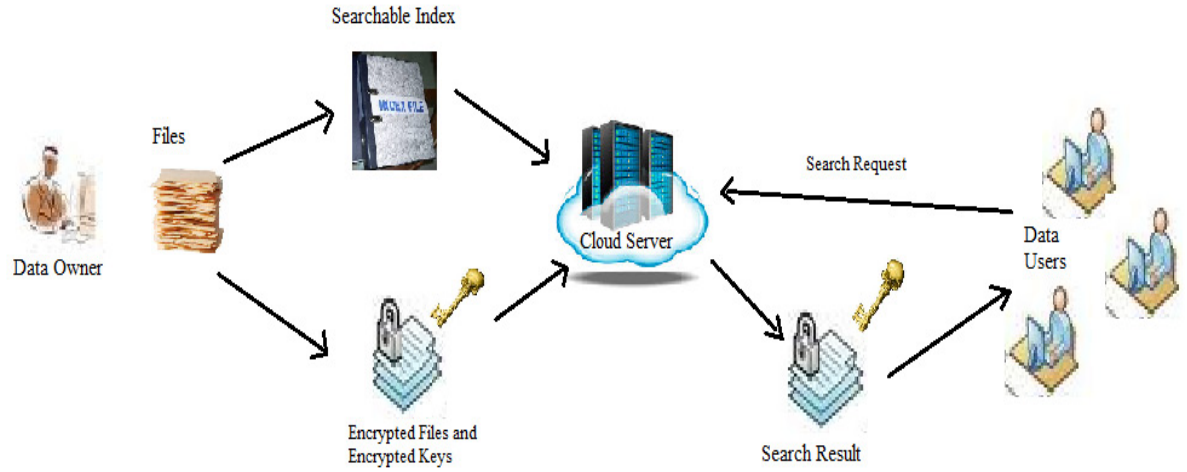

Figure 1: Scenario of retrieval of encryption cloud data

The cloud server hosts third-party data storage and retrieve services. Since data may contain sensitive information, the cloud servers cannot be fully entrusted in protecting data. For this reason, outsourced files must be encrypted. Any kind of information leakage that would affect data privacy are regarded as unacceptable.

The data owner has a collection of n files C={f1,f2,f3,…,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,w3, . . .,wl} extracted out of C, and then outsources both the encrypted index I' and encrypted files onto the cloud server[5].

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={(w'1,w'2,w'3,…,w's)|w'I ∈ 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[5]. Afterward, the data user can decrypt and make use of the files.

2.2 Vector Space Model and Scoring

Scoring is a natural way to weight the relevance. Based on the relevance score, files can then be ranked in either ascending or descending. Calculating the tf-idf weight of a term in a particular document, it is necessary to know two things: how often does it occur in the document (term frequency = tf), and in how many documents of the collection does it appear (document frequency = df). Take the inverse of the document frequency (= idf) and have both components to calculate the weight by multiplying tf by idf. In practice, the inverse document frequency is calculated as the logarithm (base 10) of the quotient of the total number of documents (N) and the document frequency in order to scale the values.

Tf–idf is the product of two statistics, term frequency and inverse document frequency [5]. Various ways for determining the exact values of both statistics exist. In the case of the **term frequency** tf(*t*,*d*), the simplest choice is to use the *raw frequency* of a term in a document, i.e. the number of times that term *t* occurs in document *d*. If denote the raw frequency of *t* by f(*t*,*d*), then the simple tf scheme is tf(*t*,*d*) = f(*t*,*d*). Other possibilities include

Boolean "frequencies": tf(*t*,*d*) = 1 if *t* occurs in *d* and 0 otherwise;
logarithmically scaled frequency: tf(*t*,*d*) = log (f(*t*,*d*) + 1);
augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document,
tf(t,d) = 0.5+((0.5xf(t,d)) / (max{f(w,d):w∈d}))
The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\mathbf{idf(t,D) = \log(N / | \{d \in D : t \in d\}|)}$$

With,
* N: total number of documents in the corpus.
* |{d ∈ D : t ∈ d}| : number of documents where the term 't' appears. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to 1+| {d ∈ D : t ∈ d}|. Mathematically the base of the log function does not matter and constitutes a constant multiplicative factor towards the overall result. Then tf–idf is calculated as tfidf(t,d,D)=tf(t,d) x idf(t,D).

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf-idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf-idf closer to 0. While tf-idf depicts the weight of a single keyword on a file, employ the vector space model to score a file on multi keyword.

The vector space model[6] [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.

3. Problem Statement

In existing, Homomorphic ciphers typically do not, in and of themselves, do not provide. In words, when encrypt the data, send it to the cloud and let the cloud compute on it Performance is often a disadvantage. Ciphertexts in the ciphers mention are much larger than the plaintexts, so communication requirements typically go up.

The computations on these large ciphertexts are typically slower than if just performed the computation on the plaintext itself. Because of this, in the outsourcing computation model, typically see a requirement that encrypting inputs and decrypting outputs should be faster than performing the computation itself. In the case of multiple parties with individual inputs this seems to be less of a concern as privacy, not efficiency is the concern.

The homomorphic encryption property also implies malleability. This means, if have some ciphertext, then can create a different ciphertext with a related plaintext, and this property can be unwanted. Malleability doesn't specify what kind of relation is implied by changing the ciphertext, while the homomorphic property usually refers to an algebraic operation.

4. Proposed System

4.1 Process of Hybrid Encryption Algorithm

RSA is the first public key algorithm used for data encryption and digital signature algorithms. It is based on the difficulty of factoring large numbers, the factoring problem. RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key[1].

To generate the two keys, choose two random large prime numbers p and q. Then randomly choose the encryption key e such that e and (p −1) (q −1) are relatively prime. Finally, use the extended Euclidean algorithm to compute the decryption key d. Firstly receiver transmits her public key (n, e) to sender and keeps the private key secret. If sender wishes to send message M to receiver. Sender change the message M in to integer m, such that $0 \leq m < n$. Then sender computes the cipher text c. Receiver can recover m from c by using her private key exponent d via computing

i. k = p * q,
ii. d = e-1 mod ((p-1) * (q-1))
iii. C≡ me (mod k)
iv. M ≡ cd (mod k)

The AES algorithm is based on permutations and substitutions. Permutations are rearrangements of data, and substitutions replace one unit of data with another. AES performs permutations and substitutions using several different techniques. AES is a block cipher with a block length of 128 bits. The number of rounds differs according to the key length as follows:
(1) 10 rounds of processing for 128-bit keys.
(2) 12 rounds of processing for 192-bit keys.
(3) 14 rounds of processing for 256-bit keys.
The first step is Add round key stage which is followed by 9 rounds of four stages and a tenth round of three stages applicable for both encryption and decryption with the exception. Each stage of a round in the decryption process is the inverse of its corresponding part in the encryption process. Substitute bytes, shift rows, mix columns and add round key are the four stages is shown in Figure. 2.
Except for the last round in each case, all other rounds are identical. The mix columns stage does not take place in the last round. The tenth round simply leaves out the Mix Columns stage. The decryption process consists of inverse shift rows, inverse substitute bytes, inverse add round key and inverse mix columns. Like the decryption process the inverse mix columns stage is omitted in the tenth round.

Considering the efficiency of both AES and RSA, AES is faster than RSA for encryption and decryption of large messages while RSA is suitable for key management as it is based on the difficulty of factoring large numbers. RSA can distribute the encryption key openly and it always keeps the decryption keys secret. While in AES, the encryption key has to be distributed secretly before communication [1].

Taking into account the advantages of both AES and RSA and avoiding their shortcomings, hybrid encryption algorithm based on AES and RSA has been proposed in which AES is used for encryption of message and RSA is used to encrypt the AES key.

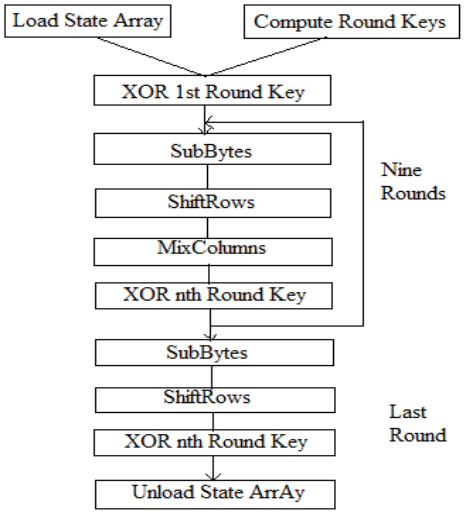

Figure 2: The AES algorithm steps

4.1.1 Process of Encryption

During the process of sending encrypted information, the random number generator produces 128-bit AES key only once, it encrypts the plaintext to produce cipher text. In RSA, p and q are randomly generated depending on the millisecond measure of the ma- chine. This is used to en-crypt the AES key.

Finally the encrypted message is send along with the encrypted AES key, private key and Big Integer n each separated by a semicolon and the order of which is known only to the sender and receiver. AES, being a block cipher divides the plaintext into number of blocks depending upon the key size for encryption. It is different from Blue-tooth stream cipher algorithm, cellular message encryption algorithm is completely safe and mathematically proven.

At first, AES algorithm encrypts original message.
(1) The 128 bit AES key is used to encrypt the data to get an encrypted content C.

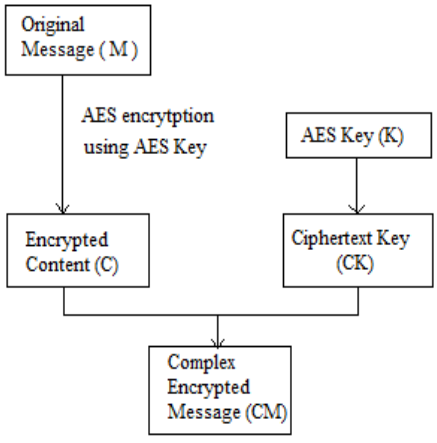

Figure 3: The Hybrid Encryption Process

The second, RSA algorithm encrypts the key of AES algorithm:

1) The pubic key of the receiver is first obtained from the server or respective sources.
2) Then the 128-bit AES is encrypted using the public key of the receiver to form the ciphertext key (CK).
3) Using the encrypted content C obtained from the AES encryption of the original message and the ciphertext key (CK) obtained from RSA encryption the complex encrypted message CM is generated which is then transmitted. The entire hybrid encryption process is shown in Figure. 3.

4.1.2 Process of Decryption

The decryption of hybrid encryption algorithm, the receiver B first divides the complex message (CM) into two parts AES encrypted ciphertext C and RSA encrypted AES session key CK. The receiver B uses its private key dB to decrypt the RSA encrypted AES session key CK to get the session key K. Using the AES key K it then decrypts the ciphertext C to generate the original message M. The entire hybrid decryption process is shown in Figure. 4.

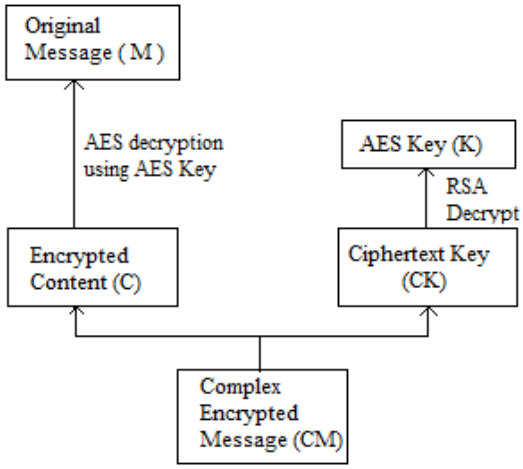

Figure 4: The Hybrid Decryption Process

4.2 Top-K Algorithm

Algorithm 1 TOPKSELECT($source, k$)

Input:
list *source* to be selected
number k

Initialization:
Set $topk \leftarrow \emptyset; topkid \leftarrow \emptyset;$

Iteration:
1: **for all** $item \in source$ **do**
2: INSERT($topk, (item, itemindex)$)
3: **end for**
4: **for all** $tuple \in topk$ **do**
5: $topkid.append(tuple[1])$
6: **end for**

Output:
topkid

(a)

Algorithm 2 INSERT($topk, (item, itemindex)$)

Input:
list *topk* to store the top-k 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** $element \in topk$ **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: **end if**
10: **end for**
11: **end if**

(b)

Figure 5: (a) Algorithm TOPKSELECT (b)Algorithm INSERT

5. Design

5.1 Initialization Phase

1. The data owner login with username and password with valid OTP for authentication.
2. The data owner generates the public key (PK) by using AES algorithm for encryption of a file.
3. The data owner extracts the collection of l keywords, W={w1,w2,w3,…,wl}, and their TF and IDF values out of the collection of n files, C={f1,f2,f3,…fn}. For each file $f_i \in$ C, the data owner builds a (l+1)-dimensional vector $v_i = \{id_i, t_{i;1_i}, t_{i;2,\ldots}, t_{i;l}\}$, where $t_{i;j} = tf\text{-}idf_{wj;fi}$ $(1 \leq j \leq l)$. The searchable index $I = \{ v_i \mid 1 \leq i \leq n \}$.
4. The data owner encrypts the searchable index I to secure searchable index $I' = \{v'_i \mid 1 \leq i \leq n\}$, where $v'_i = \{ id'_i, t'_{i,1}, t'_{i,2}, \ldots, t'_{i,l}\}$, $id'_i = Encrypt(R_{i;0}; id_i)$ and $t'_{i;j} = Encrypt(R_{i;j}; t_{i;j})(R_{i;0} \subseteq PK, R_{i;j} \subseteq PK, 1 \leq j \leq l)$.
5. The data owner encrypt the files by using AES encryption using public key.
6. The data owner encrypt a AES generated publickey using RSA, so generate publickey(PK) and secret key(SK) and assign that secretkey(SK) to users.
7. Then outsources to the cloud server.

5.2 Retrieval Phase

1. The data users first registered with data owners.
2. The data users login with valid credentials such as username and password and also enter valid OTP (One Time Password) for authenticate purpose.
3. The data user generates a set of keywords REQ={w′1, w′2, . . . ,w0 } to search, and then the query vector T_w = {m1;m2; . . .;ml} is generated in which mi = 1($1 \leq i \leq l$) if ti ∈ REQ or mi = 0 otherwise. After that, T_w is encrypted into trapdoor T_w = {c1, c2,. . . cl}, where ci = Encrypt(R, m) and S ⊆ PK, and then the user sends T_w to the cloud server.
4. For each file vector in secure searchable index, the cloud server computes the inner product. T_w with modular

reduction and then compresses and returns the result vector to the data users.

5. Decrypts the result vector and then TOPKSELECT(source , k) is invoked to get the top-k highest scoring files identifiers and sends it to the cloud.
6. The Cloud server returns the encrypted files and encrypted AES-Public key to the data users.
7. The data user first decrypt the AES public key by using secret key(SK) which is generated using RSA algorithm, sent by data owner to data users. Then data user get decrypted AES-Public key.
8. After that data user decrypt the encrypted files using decrypted AES-Public key.

6. Performance Analysis

The Initialization phase includes Setup and Index Build, and needs to be processed only once by the data owner. The Index Build stage includes building searchable index I into I'. To improve the computing efficiency, the tf-idf values are rounded to integers when building I, which does not affect the retrieve accuracy.

The Trapdoor Gen stage needs O(*l*) time to build the *l*-dimension query vector Tω from the multi keyword request. In the Score Calculate stage, the cloud server calculates the inner product of Tω with each row in I'. In the Result Decrypt stage, the data user decrypts the n-dimension result vector to get the plaintext of the scores. Since the size of the result vector depends only on the number of files in the file set and the decryption of each dimension costs constant number of modular computations, the overall complexity of decryption is O(n).

In Hybrid encryption, AES keys should be generated by a secure random number generator if used within a hybrid encryption scheme. As there are no known attacks on full AES when used as a block cipher, brute forcing the AES key is comparable to guessing a 128, 192 or 256 bit number.

All in all, both RSA with a key size of 2048 bits and AES-256 are pretty secure. There is a lot of reason to expect that other parts of the system are much more vulnerable to attack. This is especially true if communication is performed without integrity/authenticity provided by a signature or other authentication tag. Once message has integrity and authenticity protection then need to take care of side channel attacks and attacks on physical security, bugs in software (updates) etc.,.

6.1 Security Requirements

- **Confidentiality** is the assurance that data cannot be viewed by an unauthorized user.
- **Data Integrity** is the assurance that data has not been altered in an unauthorized manner. This assurance applies from the time that the data was last created, transmitted or stored by an authorized person. It is not concerned with the prevention of alteration of data, but provides a means for detecting whether data has been manipulated in an unauthorised way.
- **Data Origin Authentication** is the assurance that a given entity was the original source of received data. Data Origin Authentication is sometimes referred to as message authentication since it is primarily concerned with the authentication of the data (message) and not who we are communicating with at the time the data is received.
- **Non-Repudiation** is the assurance that an entity cannot deny a previous commitment or action. Most commonly, it is the assurance that the original source of some data cannot deny to a third party.

7. Conclusion

In this paper, for secure purpose have to be use hybrid encryption algorithm (RSA-AES encryption and decryption) before sending to cloud server. And solve the problem of secure multikeyword retrieval over encrypted cloud data. Based on invisibly leaking the sensitive information, devise a server-side ranking scheme. By security analysis, the proposed scheme guarantees confidentially, data integrity, data origin authentication and non-repudiation.

References

[1] Komal Rege, Nikita Goenka, Pooja Bhutada, Sunil Mane, "Bluetooth Communication using Hybrid Encryption Algorithm based on AES and RSA" IJCA Journal, Volume 71 - Number 22, Year of Publication: June 2013.

[2] Shaikh, F.B , "Security threats in cloud computing" Internet Technology and Secured Transactions (ICITST), 2011 International Conference, Dec. 2011, ISBN 978-1-4577-0884-8.

[3] Bhavana Agrawal, Himani Agrawal, "Implementation of AES and RSA Using Chaos System" International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May-2013, ISSN 2229-5518.

[4] ZHANG Yun-tao, GONG Ling, WANG Yong-cheng, "An improved TF-IDF approach for text classification" Journal of Zhejiang University SCIENCE, ISSN 1009-3095, http://www.zju.edu.cn/jzus, *2005 6A(1):49-55.*

[5] Jiadi Yu, Peng Lu, Yanmin Zhu, Guangtao Xue, Minglu Li, "Toward Secure MultiKeyword Top-k Retrieval over Encrypted Cloud Data" IEEE, Vol. 10, No. 4, July/Aug 2013.

[6] Stephen Robertson, "Understanding Inverse Document Frequency: On theoretical arguments for IDF" Journal of Documentation 60 no. 5, pp 503–520.

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

Author Profile

Deepak I M received the B.E. degree in Computer Science Engineer from Government Engineering College, Hassan in 2012 and pursuing MTech degree in Computer Science and Engineering from Dr. Ambedkar Institute of Technology, Bangalore in 2012-2014. He has strong passion for Computer Networks and Cloud Computing.

Mrs. K.R. Shylaja received B.E and MTech degree in Computer Science from Visvesvaraya Technological University. She had served as an Asst. Professor in Dr. Ambedkar Institute of Technology, Bangalore in 2000-2011. Currently she is Associate Professor in Dr. Ambedkar Institute of Technology, Bangalore.

Ravinandan M E received B.E degree in Computer Science Engineer from Bangalore University, M Tech degree in Computer Science & Engineering from VTU. He had served as a scientist in DRDO and GE Healthcare. Currently he is MD & CEO of Teminnova Technologies Private Limited and also a Founder Member of Organization for Rare Diseases India.