ACHIEVING FINE GRAINED QUREY RESULT VERIFICATION OVER ENCRYPTED CLOUD DATA

[1]Abhishek Sharma, [2]Amit Das, [3]Sharath, [4]Thathi Santosh, [5]A.Shraban kumar
[5]Associate Professor, [1,2,3,4]B.Tech
[1,2,3,4,5]Department of Computer Science and Engineering,
[1,2,3,4,5]St. Martin's Engineering College, Hyderabad, India

Abstract: Secure search techniques over encrypted cloud data allow an authorized user to query data files of interest by submitting encrypted query keywords to the cloud server in a privacy-preserving manner. However, in practice, the returned query results maybe incorrect or incomplete in the dishonest cloud environment. For example, the cloud server may intentionally omit some qualified results to save computational resources and communication overhead. Thus, a well-functioning secure query system should provide a query results verification mechanism that allows the data user to verify results. In this paper, we design a secure, easily integrated, and fine-grained query results verification mechanism, by which, given an encrypted query results set, the query user not only can verify the correctness of each data file in the set but also can further check how many or which qualified data files are not returned if the set is incomplete before decryption. The verification scheme is loose-coupling to concrete secure search techniques and can be very easily integrated into any secure query scheme. We achieve the goal by constructing secure verification object for encrypted cloud data. Furthermore, a short signature technique with extremely small storage cost is proposed to guarantee the authenticity of verification object and a verification object request technique is presented to allow the query user to securely obtain the desired verification object. Performance evaluation shows that the proposed schemes are practical and efficient.

.***IndexTerms*** - **Fine grained, Query keyword, verification object**

I. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a LOUD computing is a model for enabling ubiquitous shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Driven by the abundant benefits brought by the cloud computing such as cost saving, quick deployment, flexible resource configuration, etc., more and more enterprises and individual users are taking into account migrating their private data and native applications to the cloud server. A matter of public concern is how to guarante the security of data that is outsourced to a remote cloud server and breaks away from the direct control of data owners. Encryption on private data before outsourcing is an effective measure to protect data confidentiality.

However, encrypted data make effective data retrieval a very challenging task. To address the challenge (i.e., search on encrypted data), Song et al. first introduced the concept of searchable encryption and proposed a practical technique that allows users to search over encrypted data through encrypted query keywords in. Later, many searchable encryption schemes were proposed based on symmetric key and public-key setting to strengthen security and improve query efficiency. Recently, with the growing popularity of cloud computing, how to securely and efficiently search over encrypted cloud data becomes a research focus. Some approaches have been proposed based on traditional searchable encryption schemes in, which aim to protect data security and query privacies with better query efficient for cloud computing. However, all of these schemes are based on an ideal assumption that the cloud server is an ”honest-but-curious” entity and keeps robust and secure software/hardware environments.

As a result, correct and complete query results always be unexceptionally returned from the cloud server when a query ends every time. However, in practical applications, the cloud server may return erroneous or incomplete query results once he behaves dishonestly for illegal profits such as saving computation and communication cost or due to possible software/hardware failure of the server. Therefore, the above fact usually motivates data users to verify the correctness and completeness of query results. Some researchers proposed to integrate the query results verification mechanisms to their secure search schemes, (e.g., embedding verification information into the specified secure indexes or query results). Upon receiving query results, data users use specified verification information to verify their correctness and completeness.

There are two limitations in these schemes:

1) These verification mechanisms provide a coarse grained verification, i.e., if the query result set contains all qualified and correct data files, then these schemes reply yes, otherwise reply no. Thus, if the verification algorithm outputs no, a data user has to abort the decryption for all query results despite only one query result is incorrect.

2) These verification mechanisms are generally tightly coupled to corresponding secure query constructions and have not universality. In a search process, for a returned query results set that contains multiple encrypted data files, a data user may wish to verify the correctness of each encrypted data file (thus, he can remove incorrect results and retain the correct ones as the ultima query results) or wants to check how many or which qualified data files are not returned on earth if the cloud server intentionally omits some query results. These information can be regarded as a hard evidence to punish the cloud server. This is challenging to achieve the fine-grained verifications since the query and verification are enforced in the encrypted environment. In, we proposed a secure and fine-grained query results verification scheme by constructing the verification object for encrypted outsourced data files. When a query ends, the query results set along with the corresponding verification object are returned together, by which the query user can accurately verify:

- The correctness of each encrypted data file in the results set;
- How many qualified data files are not returned and
- Which qualified data files are not returned. Furthermore, our proposed verification scheme is lightweight and loose-coupling to concrete secure query schemes and can be very easily equipped into any secure query scheme for cloud computing.

However, some necessary extensions and important works need to be further supplied to perfect our original scheme such as detailed performance evaluation and formal security definition and proof. More importantly, in the dishonest cloud environment, the scheme suffers from the following two important security problems:

1) Just as possibly tampering or deleting query results, the dishonest cloud server may also tamper or forge verification objects themselves to make the data user impossible to perform verification operation. Specially, once the cloud server knows that the query results verification scheme is provided in the secure search system, he may return in veracious verification object to escape responsibilities of misbehavior.

2) When a data user wants to obtain the desired verification object, some important information will be revealed such as which verification objects are being or have been requested before frequently, etc. This information may leak query user's privacy and expose some useful contents about data files. More importantly, this exposed information may become temptations of misbehavior for the cloud server. We will detailed describe this part content in Section.

II. LITERATURE SURVEY

Security Challenges For the Public Cloud:
Cloud computing represents today's most exciting computing paradigm shift in information technology. However, security and privacy are perceived as primary obstacles to its wide adoption. Here, the authors outline several critical security challenges and motivate further investigation of security solutions for a trustworthy public cloud environment.

Practical Techniques For Searches on Encrypted Data:
It is desirable to store data on data storage servers such as mail servers and file servers in encrypted form to reduce security and privacy risks. But this usually implies that one has to sacrifice functionality for security. For example, if a client wishes to retrieve only documents containing certain words, it was not previously known how to let the data storage server perform the search and answer the query, without loss of data confidentiality. We describe our cryptographic schemes for the problem of searching on encrypted data and provide proofs of security for the resulting crypto systems. Our techniques have a number of crucial advantages. They are provably secure: they provide provable secrecy for encryption, in the sense that the untrusted server cannot learn anything about the plaintext when only given the ciphertext; they provide query isolation for searches, meaning that the untrusted server cannot learn anything more about the plaintext than the search result; they provide controlled searching, so that the untrusted server cannot search for an arbitrary word without the user's authorization; they also support hidden queries, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server. The algorithms presented are simple, fast (for a document of length n, the encryption and search algorithms only need O(n) stream cipher and block cipher operations), and introduce almost no space and communication overhead, and hence are practical to use today.

Deterministic And Efficiently Searchable Encryption:
We present as-strong-as-possible definitions of privacy, and constructions achieving them, for public-key encryption schemes where the encryption algorithm is deterministic. We obtain as a consequence database encryption method that permit fast (i.e. sub-linear, and in fact logarithmic, time) search while provably providing privacy that is as strong as possible subject to this fast search constraint. One of our constructs, called RSA-DOAEP, has the added feature of being length preserving, so that it is the first example of a public-key cipher. We generalize this to obtain a notion of efficiently-searchable encryption schemes which

permit more flexible privacy to search-time trade-offs via a technique called bucketization. Our results answer much-asked questions in the database community and provide foundations for work done there.

Public-Key Encryption With Fuzzy Keyword Search:
A provably secure scheme under keyword guessing attack. Public-key encryption with keyword search (PEKS) is a versatile tool. It allows a third party knowing the search trapdoor of a keyword to search encrypted documents containing that keyword without decrypting the documents or knowing the keyword. However, it is shown that the keyword will be compromised by a malicious third party under a keyword guess attack (KGA) if the keyword space is in a polynomial size. We address this problem with a keyword privacy enhanced variant of PEKS referred to as public-key encryption with fuzzy keyword search (PEFKS). In PEFKS, each keyword corresponds to an exact keyword search trapdoor and a fuzzy keyword search trapdoor. Two or more keywords share the same fuzzy keyword trapdoor. To search encrypted documents containing a specific keyword, only the fuzzy keyword search trapdoor is provided to the third party, i.e., the searcher. Thus, in PEFKS, a malicious searcher can no longer learn the exact keyword to be searched even if the keyword space is small. We propose a universal transformation which converts any anonymous identity-based encryption (IBE) scheme into a secure PEFKS scheme. Following the generic construction, we instantiate the first PEFKS scheme proven to be secure under KGA in the case that the keyword space is in a polynomial size.

Parallel And Dynamic Searchable Symmetric Encryption:
Searchable symmetric encryption (SSE) enables a client to outsource a collection of encrypted documents in the cloud and retain the ability to perform keyword searches without revealing information about the contents of the documents and queries. Although efficient SSE constructions are known, previous solutions are highly sequential. This is mainly due to the fact that, currently, the only method for achieving sub-linear time search is the inverted index approach (Curtmola, Garay, Kamara and Ostrovsky, CCS ’06) which requires the search algorithm to access a sequence of memory locations, each of which is unpredictable and stored at the previous location in the sequence. Motivated by advances in multi-core architectures, we present a new method for constructing sub-linear SSE schemes. Our approach is highly parallelizable and dynamic. With roughly a logarithmic number of cores in place, searches for a keyword w in our scheme execute in o(r) parallel time, where r is the number of documents containing keyword w (with more cores, this bound can go down to O(logn), i.e., independent of the result size r). Such time complexity outperforms the optimal Θ(r) sequential search time—a similar bound holds for the updates. Our scheme also achieves the following important properties: (a) it enjoys a strong notion of security, namely security against adaptive chosen-keyword attacks; (b) compared to existing sub-linear dynamic SSE schemes (e.g., Kamara, Papamanthou, Roeder, CCS ’12), updates in our scheme do not leak any information, apart from information that can be inferred from previous search tokens; (c) it can be implemented efficiently in external memory (with logarithmic I/O overhead). Our technique is simple and uses a red-black tree data structure; its security is proven in the random oracle model.

2.1 Advantages

- In a search process, for a returned query results set that contains multiple encrypted data files, a data user may wish to verify the correctness of each encrypted data file (thus, he can remove incorrect results and retain the correct ones as the ultima query results) or wants to check how many or which qualified data files are not returned on earth if the cloud server intentionally omits some query results. These information can be regarded as a hard evidence to punish the cloud server.
- A secure and fine-grained query results verification scheme by constructing the verification object for encrypted outsourced data files. When a query ends, the query results set along with the corresponding verification object are returned together, by which the query user can accurately verify:
  1) the correctness of each encrypted data file in the results set

III. OVERVIEW

3.1 Module Description

- Owner
- Cloud Server
- User

Owner:

 File owner will register with application and registration details are sent to Clod server for verification after verification is successful user can login with username and password and upload files to cloud Server and data is stored in Drive HQ.

 File owner can view all uploaded files in the cloud and check requests received form users to download files and owner will send keys to user for downloading file.

Cloud Server:

 Cloud Server can login with valid user name and password. The cloud storage server provides storage services to the registered clients for storing outsourced files and he can activate or de active user and owner accounts. Cloud server can view encrypted files and he can modify data which are uploaded by file owner.

User:

 User will register with application and registration details are sent to Clod server for verification after verification is successful user can login with username and password. User can search files with keywords and based on matched keyword files will be received to user form cloud. User can select files from the list and request for file owner to get key for download. User will Verify Files which are uploaded by Owner, verify with file verification object and cloud storage verification object. If two objects are match there is no change in user data if verification objects are not match data is change by cloud server.

3.2 Architecture

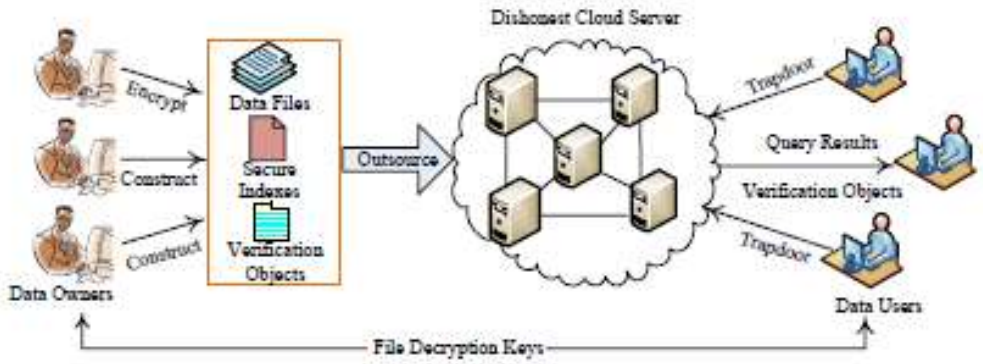

Fig. 1 Architecture of Proposed System

Above architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

3-Tier Architecture:

The three-tier software architecture (a three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems

.

Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

IV. METHODOLOGY

Coming to the method we used in this project is AES algorithm method. The Advanced Encryption Standard, or AES, is a symmetric block cipher. Before storing the data into cloud encryption process is done that convert the readable format into an unreadable format so that only the true person can download the data with the encryption and decryption key. The first step of the ciphertext after which the cipher transformations are repeated over a number of encryption rounds. The number of rounds is determined by the key length.

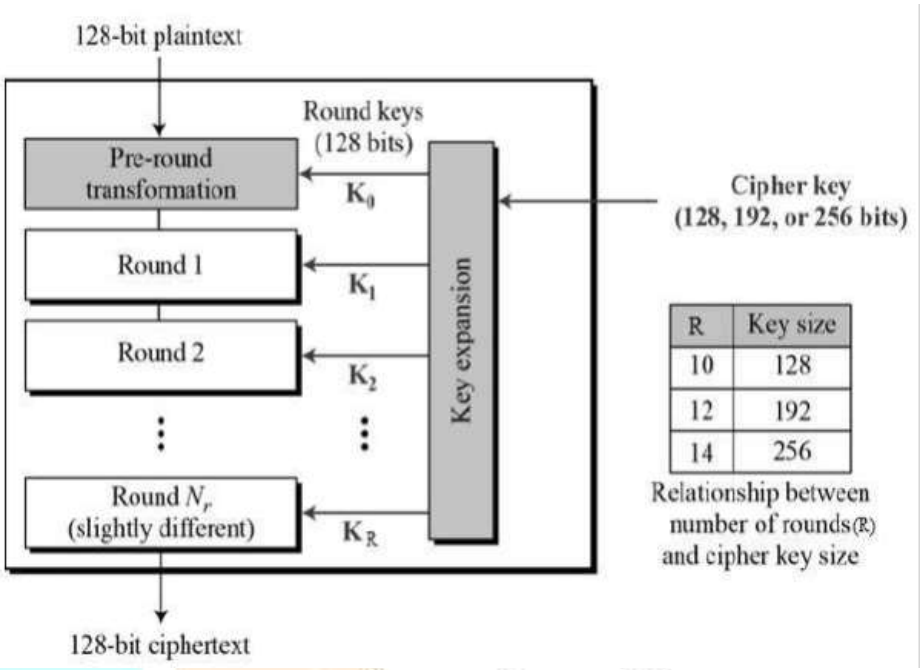

Fig. 3 AES algorithm

ENCRYPTION PROCESS

The encryption process is done in round process. Each round consists of 4 steps. The below are the 4 steps for each round.

1) Byte Substitution
2) Shift Rows
3) Mix Columns
4) Add round key

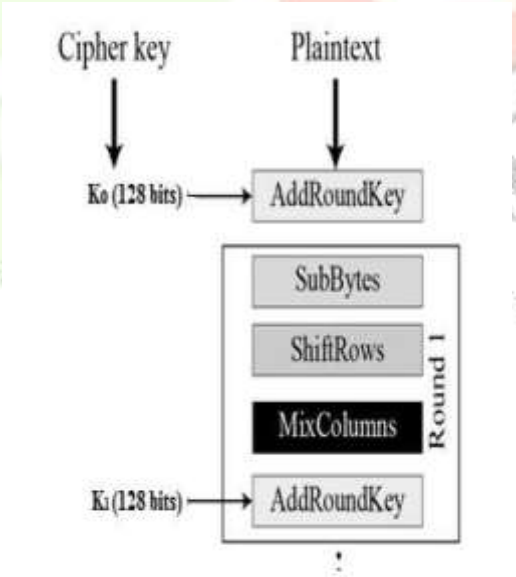

Fig. 4 Rounds in AES algorithm

Substitution (Sub Bytes):

The input is given. The result is in a matrix form

Shift rows:

The shift is carried out as follows −

- The first row will be the same.
- The second row is moved by one (byte) position to the left side.

- The third row is moved by two positions to the left side.
- The fourth row is moved by three positions to the left side.
- The output is a new matrix.

Mix Columns:

Mix column is done by performing a mathematical function. The will takes input as four bytes of one column and outputs as four bytes which replace the original column. The result is 16 new bytes.

Add round key:

A combination of 128bits of round key and 128 bits of a matrix (that is 16 bytes) is considered. If this is the final round then the output is the ciphertext. Otherwise, the 16 bytes are considered and we begin another similar round.

Decryption Process::

The decryption process is exactly similar to the encryption process but in the reverse order. There are four rounds which are performed in reversed order. The rounds are given below:

- Add round key
- Mix columns
- Shift rows
- Byte substitution

We can also use an AESBase64 encoding and decoding technique. For key generation also we can use an AESBase64 technique.

V. RESULTS

The Verification Object Construction:

To maximize reduce storage and communication cost and achieve privacy guarantee of the verification objects, in this paper, we will utilize Counting Bloom Filters and the pseudo-random function to construct our verification objects, on which the authorized data user can efficiently perform query results verification.

An Enhanced Completeness Verification Construction:

For completeness verification, a significant advantage of our constructed verification object is that the query user can quickly count the number of data files omitted by the cloud server for an incomplete query result set. As a more strict completeness verification requirement (i.e., which qualified data files are not returned in a query), an enhanced verification object based on the Counting Bloom Filter was proposed in our work, which further allows the data user to definitely obtain the file identifier of each data file that satisfies the query yet is omitted by the cloud server, by reasonably designing the identifiers of data files and secretly preserving them in the corresponding verification object. Here, we are not intend to elaborate on the verification construction to avoid unnecessary repetition.

VI. SCREENSHOTS

Fig. 5 Screenshot of the proposed system

VII. CONCLUSION AND FUTURE SCOPE

In this paper, we propose a secure, easily integrated, and fine-grained query results verification scheme for secure search over encrypted cloud data. Different from previous works, our scheme can verify the correctness of each encrypted query result or further accurately find out how many or which qualified data files are returned by the dishonest cloud server. A short signature technique is designed to guarantee the authenticity of verification object itself. Moreover, we design a secure verification object request technique, by which the cloud server knows nothing about which verification object is requested by the data user and actually returned by the cloud server. Performance and accuracy experiments demonstrate the validity and efficiency of our proposed scheme.

7.1 OBSERVATIONS

Verifying the Correctness and Completeness of Query Results:
When a query ends, the query results set and corre-sponding verification object are together returned to the query user, who verifies the correctness and complete-ness of query results based on the verification object. Our proposed query results verification scheme not only allows the query user to easily verify the correctness of each encrypted data file in the query results set, but also enables the data user to efficiently perform com-pleteness verification before decrypting query results. More importantly, for an incomplete query results set, our verification objects can definitely tell the data user that the cloud server has omitted how many qualified data files for a query, which is a significant advantage compared with the previous related works.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing,"http://dx.doi.org/10.602/NIST.SP.800-145.

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69–73, 2012.

[3] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Springer RLCPS, January 2010.

[4] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in IEEE Symposiumon Security and Privacy, vol. 8, 2000, pp. 44–55.

[5] E.-J.Goh, "Secure indexes," IACR ePrint Cryptography Archive, http://eprint.iacr.org/2003/216, Tech. Rep., 2003

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public-key encryption with keyword search," in EUROCRYPR, 2004, pp. 506–522.

[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetricencryption:improve definitions and efficient constructions," in ACM CCS, vol. 19, 2006, pp. 79–88.

[8] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in Springer CRYPTO, 2007.

[9] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," Lecture Notes in Computer Science, vol. 7397, pp. 258– 274, 2012.

[10] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," IEEE Transactions on Computers, vol. 62, no. 11, pp. 2266–2277, 2013.

[11] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2013, pp. 258–274.

[12] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in IEEE S&P, May 2014, pp. 639–654.

[13] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in IEEE ICDCS, 2010, pp. 253–262.