**Privacy-Preserving and Trusted Keyword Search for Multi-Tenancy Cloud**

**ABSTRACT**:

Cloud service models intrinsically cater to multiple tenants. In current multi-tenancy model, cloud service providers isolate data within a single tenant boundary with no or minimum cross-tenant interaction. With the booming of cloud applications, allowing a user to search across tenants is crucial to utilize stored data more effectively. However, conducting such a search operation is inherently risky, primarily due to privacy concerns. Moreover, existing schemes typically focus on a single tenant and are not well suited to extend support to a multi-tenancy cloud, where each tenant operates independently. In this article, to address the above issue, we provide a privacy preserving, verifiable, accountable, and parallelizable solution for “privacy-preserving keyword search problem” among multiple independent data owners. We consider a scenario in which each tenant is a data owner and a user’s goal is to efficiently search for granted documents that contain the target keyword among all the data owners. We first propose a verifiable yet accountable keyword searchable encryption (VAKSE) scheme through symmetric bilinear mapping. For verifiability, a message authentication code (MAC) is computed for each associated piece of data. To maintain a consistent size of MAC, the computed MACs undergo an exclusive OR operation. For accountability, we propose a keyword-based accountable token mechanism where the client’s identity is seamlessly embedded without compromising privacy. Furthermore, we introduce the parallel VAKSE scheme, in which the inverted index is partitioned into small segments and all of them can be processed synchronously. We also conduct formal security analysis and comprehensive experiments to demonstrate the data privacy preservation and efficiency of the proposed schemes, respectively.

|  |  |
| --- | --- |
| **EXISTING SYSTEM** | **PROPOSED SYSTEM** |
| * In early research, most works on SSE focused on the honestbut-curious cloud service provider (CSP). * In such a model, the search result is fully trusted and the CSP is assumed to honestly follow the protocol specification. * Search results in practice may contain corrupted data due to underlying hardware/software failures. * In addition, for self-interest, the CSP may deviate from the protocol specification. For example, to reduce computational costs, CSP may randomly choose data as a search result. | * This paper proposed an authenticated Merkle hash tree to verify the search result. The proposed privacy-aware attribute-based encryption with user accountability and applied it to the file storage system. * However, their scheme relied on attribute-based encryption, which is different from SSE because SSE uses symmetric encryption. * In the proposed scheme, each data owner encrypts its own data (with its unique key) and outsources the storage and processing tasks for operations to the CSP. * In their scheme, each keyword is encrypted in two layers. The first layer is implemented via pseudo-random permutation and used to hide the keyword. The second layer is realized by the pseudo-random function and used to support encrypted keyword searches. To improve search efficiency |
| **EXISTING ALGORITHM**  Attribute-Based Keyword Search | **PROPOSED ALGORITHM**  Verifiable yet accountable keyword searchable encryption (VAKSE) scheme through symmetric. For verifiability, a message authentication code (MAC) |
| **EXISTING ALGORITHM**  **Description:**  the first attribute-based keyword search (ABKS) scheme that combined public key encryption with keyword search (PEKS) .Moreover, in the above works, no accountability and verifiability mechanisms are available. If a client abuses his/her right (e.g., by sharing search tokens with unauthorized clients), there is no way to identify the client. In addition, the search result returned by the CSP may be incorrect due to deliberate modifications or unpredicted data loss. For example, instead of returning target documents, the CSP may send the client an advertisement. Moreover, the hardware and software are not stable, which may result in data loss due to hardware or software failure. Additionally, packet loss during communication. | **PROPOSED ALGORITHM**  **Description:**  To support the search result verification, we propose a novel construction of message authentication code (MAC), in which all the associated data is required to compute MAC, and then exclusive or (XOR) operation is conducted between all the computed MACs. Finally, the result is encrypted and embedded into the index. In addition, to enable fine-grained access control and user accountability, we propose a keyword-based accountable token mechanism. Each user needs to obtain a token for a specific keyword before they are able to launch a query. The token is utilized to transform a user’s query to support cipher text search. Moreover, with the construction of a token, the identity of the client is properly embedded into the token without privacy violation. Because of the way the token is constructed, it can be used to trace back to the client. Finally, to enhance the search efficiency, we propose a parallelism mechanism, where the inverted index is partitioned into segments and all of them are executed in parallel. Finally, MAC is concatenated with the document identity and encrypted by the symmetric encryption algorithm |
| **DRAWBACKS: -**   * Low security of data processing. * This is a critical feature that allows a flexible number of data owners in the system, who have complete control of their multiple data owners, and allows a legitimate client to search across parties. | **ADVANTAGES: -**   * High security for data Sharing * In this work, our goal is to present a privacy-preserving keyword search scheme that has multiple independent data owner |

**SYSTEM ARCHITECTURE**

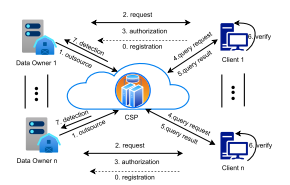
****

Fig 1: System Architecture

**MINIMUMSYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS**

* Processor : Dual core 2 duo.
* Ram : 8 Gb DD RAM
* Hard disk : 250 Gb

**SOFTWARE REQUIREMENTS**

* Front End : J2ee (jsp, servlet)
* Back End : My Sql 5.5
* Operating System : Windows 10
* Ide : Eclipse