
# Demonstration of Doc$^\star$: Access Control for Information-Theoretically Secure Data

Komal Kumari
NJIT, USA
kk675@njit.edu

Yin Li
DGUT, China
yunfeiyangli@gmail.com

Sharad Mehrotra
UC Irvine, USA
sharad@ics.uci.edu

Shantanu Sharma
NJIT, USA
shantanu.sharma@njit.edu

Venkata Sreeramakavacham
NJIT, USA
vs787@njit.edu

## ABSTRACT

This demonstration presents Doc$^\star$, a system for secure keyword-based access control over document stores. To ensure data security, Doc$^\star$ adopts Shamir's secret-sharing that provides information-theoretic security, i.e., security independent of the adversary's computational power. The database owner outsources documents in secret-shared form, along with keyword-based access control rules for clients, to the public cloud. Clients submit keyword queries, also in secret-shared form, and the cloud returns documents containing the queried keywords only if the client has access rights to all keywords associated with each document. The system further includes mechanisms to detect malicious behavior, such as clients attempting unauthorized access or servers tampering with data.

The demonstration is designed to abstract away all cryptographic complexities, offering end-users an experience akin to Google Drive or Dropbox, but with keyword-based access control. The demo showcases two scenarios highlighting the system's full capabilities: secure document and access-right outsourcing, secure query execution, dynamic operations (e.g., granting/revoking access rights, adding or deleting clients, documents, and keywords), resilience against malicious clients and servers, and scalability with respect to the number of clients and documents.

**PVLDB Reference Format:**
Komal Kumari, Yin Li, Sharad Mehrotra, Shantanu Sharma, and Venkata Sreeramakavacham. Demonstration of Doc$^\star$: Access Control for Information-Theoretically Secure Data . PVLDB, 19(12): 4810 - 4813, 2026.
doi:10.14778/3827998.3828128

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/SecretDeB/DocStar_Demo_VLDB_2026.

## 1 INTRODUCTION

This demonstration presents a system for secure outsourcing of access control over ciphertext document stores, Doc$^\star$ (where $\star$ denotes *STorage with Access control and secuRity*), published in VLDB 2025 under the title "Access Control for Information-Theoretically Secure Data." Three main contributions of Doc$^\star$ are as follows:


Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828128

***(i) Realization of keyword-based access control.*** Doc$^\star$ is the first system to implement keyword-based access control for key–document stores, where keys correspond to keywords extracted from documents and values correspond to the documents that contain those keywords. A document may be associated with multiple keywords, which serve as indices for retrieval, and access to document is governed by the access rights defined for these keywords. For example, a document may be indexed by keywords such as "project" and "confidential." A client with access to the keyword "project" can retrieve documents containing "project;" only if the client is authorized for all keywords associated with the document.
***(ii) Outsourcing-based access control over ciphertext data.*** Doc$^\star$ supports outsourcing of both documents and keyword-based access control rules to an untrusted cloud in secure ciphertext form. Clients perform keyword-based queries, which are also in ciphertext form, directly over ciphertext data. During query execution, the cloud learns neither the queried keyword nor whether the client has access to the requested data. For each query, cloud always returns a fixed number of files, ensuring that the result size does not leak information. If the client has access to all keywords associated with a file, the correct file is obtained after decryption; otherwise, the client receives indistinguishable garbage content. To achieve this, Doc$^\star$ uses secret-sharing techniques, specifically Shamir's secret-sharing that provides *information-theoretic security*, i.e., security, regardless of the computational power of the adversary.[1]
***(iii) Robustness against malicious clients and servers.*** Doc$^\star$ handles adversarial behavior from both clients and servers. Malicious clients, even when colluding with a minority of servers, cannot infer any information about stored data or other clients' queries. Clients can verify whether access denial is legitimate and ensure both correctness and completeness of the retrieved files.

### 1.1 Demonstration Scenarios

The user interface is implemented in React, while the system backend uses Java with Spring Boot. During the demo, the system will be deployed across public cloud providers in different geographic regions, emulating a realistic setting with non-colluding cloud servers. The demonstration will illustrate the following two scenarios for interacting with Doc$^\star$ — all scenarios will allow the user to experience the full capabilities of the system, resembling Google Drive or Dropbox with fine-grained keyword-based access control.

[1] Informally, a secret-sharing algorithm takes a secret $s$ and creates partitions of the secret $s$ using mathematical functions; these partitions are called secret-shares. Each of the secret-shares is stored at a non-colluding cloud server. To reconstruct the secret, a fixed number of shares are required.

**(*i*) Preinstalled databases and access rights.** The attendee interacts with a predefined database owner (DBO) and clients. The DBO outsources 0.5M Enron emails, indexed by 5000 keywords, and defines access control policies for three clients. The three clients submit queries in parallel, demonstrating multi-client interaction.
**(*ii*) User-defined databases and access rights.** Attendees can create their own DBO, register clients, and outsource a dataset of their choice. Registered clients can then fetch files using keywords, allowing attendees to explore the entire system as they want.

## 1.2 What Attendees Will Learn

The demo showcases a fully functional system that enforces keyword-based access control over ciphertext documents while hiding all cryptographic complexities. The demo highlights the three main contributions of Doc$^\star$, as mentioned below:
**Power of keyword-based access over coarse-grained file-level access control.** While coarse-grained access control at the file/doc-id level is popular, key-based access can often be much easier to specify and implement, and scales well with large numbers of documents and clients. E.g., a DBO managing hundreds of thousands of documents can easily define access rights based on keywords – permit a client to access all files containing "Urgent" but not those containing "Finance." From the system perspective, checking access rights for a keyword is more efficient than verifying access for many documents individually. The first scenario with preinstalled database will demonstrate how keyword-based access control efficiently scales well compared to file-id-based access control.
**Full capabilities of keyword-based access control and search, and scalability of the system.** Attendees will experience the system as both DBO and client. As a DBO, they can upload documents, extract keywords, define access rights for multiple clients, and outsource both documents and access rights to the cloud in secret-shared form — Figure 2 shows the screenshots of all such operations performed by DBO in our implementation, and detailed descriptions of these figures are given in §2.1. As clients, they can retrieve files via keyword queries — Figure 4 shows screenshots of how clients can retrieve files based on keyword searches and what they see when access is denied or the keyword is absent. The demo will highlight the system's scalability in terms of the number of clients, documents, access rights, and concurrent operations, including queries and dynamic updates, e.g., granting/revoking access rights, outsourcing new documents, and deleting files or keywords — Figure 3 shows the screenshots of all the dynamic operations.
**Resilience to adversarial environments.** Doc$^\star$ deals with both malicious servers and malicious clients. During the demo, we will simulate a malicious server while the attendee acts as an honest client. Attendees can verify that the server correctly returns all documents containing the queried keyword without altering content. Conversely, the system can detect and abort queries from malicious clients. Figure 5 shows the screenshots of such verification scenarios performed by the client, also detailed in §2.2.

## 1.3 Who Will Benefit from the Demo

This demo will be relevant for academic and industry audiences. Academic researchers focusing on access control, secure data processing, encrypted query execution, and secret-sharing techniques will gain a hands-on understanding of system design and performance tradeoffs. Industry professionals and engineers developing

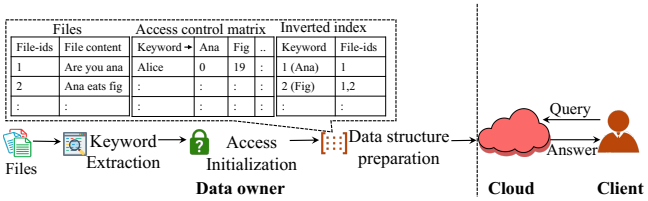



**Figure 1: Dataflow in Doc$^\star$.**

document stores, cloud-based data services, or fine-grained access control solutions will benefit from seeing how keyword-based access policies can be efficiently enforced over encrypted data at scale. Overall, the demo bridges the gap between theoretical advances and practical deployment in secure document management.

## 2 DESCRIPTION OF DOC$^\star$

The demo offers an interactive interface to facilitate secure keyword-based document retrieval under keyword-based access control, based on Doc$^\star$ that assumes three entities: a trusted database owner (DBO), the non-colluding public cloud servers, and clients. Figure 1 provides an overview of the workflow, organized into two stages: data outsourcing by DBO to the cloud server (see §2.1) and keyword-based document retrieval by the clients (see §2.2).

## 2.1 Database Owner Operations

*2.1.1* ***System Initialization*** DBO holds a set of documents that they want to outsource to the cloud. DBO initializes the system by (*i*) uploading a set of documents, Figure 2a, (*ii*) extracting keywords from the documents, Figure 2b, (*iii*) finding the desired keyword that will be used for search operation, Figure 2c, (*iv*) defining keyword-based access rights for each client/group (e.g., provosts, deans, and department chairs) in an organization; Figure 2d, and (*iv*) outsourcing both access rights and documents in secret-shares to the cloud; Figure 2e. Below, we discuss these steps.

**Interface-level Functioning.**
***Document upload and keyword extraction (Figure 2a, 2b, and 2c).*** DBO starts with uploading documents via the interface, and then the system finds a set of keywords (which will be used for defining access control policies by DBO and/or for retrieving documents by clients) – currently, nothing is outsourced to cloud. Keywords of length from 3 to 19 are selected, while short words (e.g., 'the' or 'is'), long words, and similar terms (found through stemming) are discarded. Figure 2a shows DBO selecting files from Enron email dataset for outsourcing. Figure 2b shows extracted keywords, e.g., "subject" and "enron," and their file occurrences, which are used to define keyword-based access control, see Figure 2c.
***Access right initialization (Figure 2d).*** Now, DBO grants access to clients on the selected keywords, assuming clients are already registered in the system. Figure 2d shows that for a client, namely Alice, DBO grants access to keywords such as "trade" and "discuss," and denies access to keywords "philip" and "forward"; all unselected keywords are denied access by default. New clients or revocation of existing access are handled via update operations detailed in §2.1.2.
***Data outsourcing (Figure 2e).*** After defining access rights to the registered clients/groups, DBO simply outsources the access rights and documents in ciphertext. Figure 2e shows the number of files, keywords, and access rights that will be outsourced to the cloud.

**Table 1: Cleartext files.**

| File-ids | File content |
|---|---|
| 1 | How are you |
| 2 | Are you Ana |
| 3 | Fig is a fruit |

**Table 2: Access control (AC) matrix.**

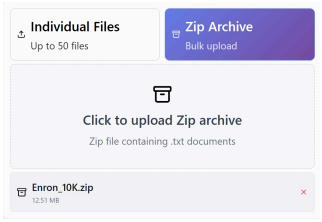
| Keywords → | Are | Ana | Fig |
|---|---|---|---|
| Starting address in inverted index | 1 | 2 | 3 |
| Clients' information ↓ | | | |
| Lisa | **0** | **1** | **2** |
| Ava | **3** | **0** | **0** |

**Table 3: Inverted index.**

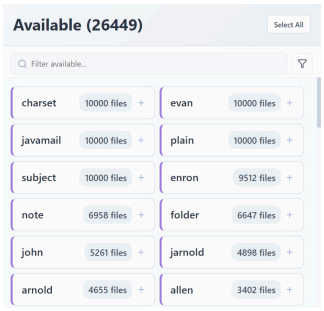
| Positions | File-ids |
|---|---|
| 1 (are) | 1, 2, 0, H(1, 2, 0, are) |
| 2 (Ana) | 2, 0, 0, H(0, H(2, 0, 0ana) |
| 3 (Fig) | 3, 0, 0, H(3, 0, 0, Fig) |

**Table 4: Files.**

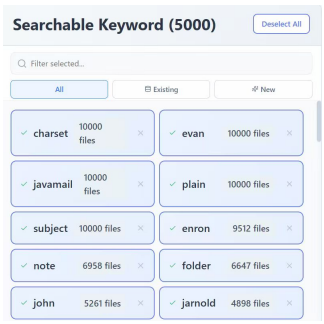
| file_ids | Special tag | File content with digest |
|---|---|---|
| 1 | 1,0,H(1) | How are you, H(how are you, H(1)) |
| 2 | 1,2,H(1)+H(2) | Are you Ana, H(are you Ana, H(2)) |
| 3 | 3,0,H(3) | Fig is a fruit, H(Fig is a fruit, H(3)) |
| 0 | 0,0,H(0) | Dummy, H(Dummy, H(0)) |



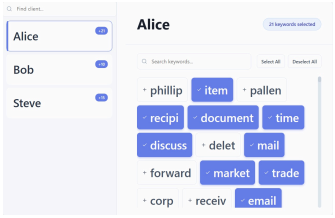

**(a) Document upload interface to upload files individually or in bulk.**



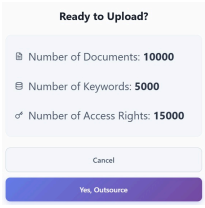

**(b) Keyword extraction interface to show keywords extracted from uploaded files.**




**(c) Keyword selection interface to allow selection of keywords from the extracted set.**

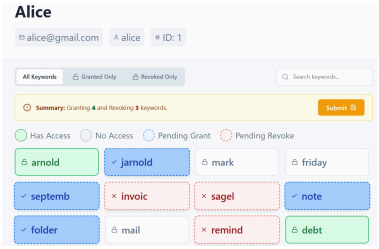



**(d) Access rights initialization interface to define access rights over keywords for each client.**

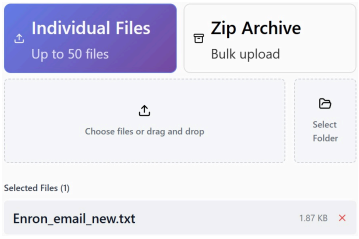



**(e) Data outsourcing interface to outsource file and access rights to the server.**

**Figure 2: Database owner operations.**




**(a) Update access rights interface to update access rights on existing keywords for a client.**




**(b) Adding a new file to the system.**


**(c) Addition of new keywords on adding a new file.**

**Figure 3: Dynamic Operations for Database Owners.**

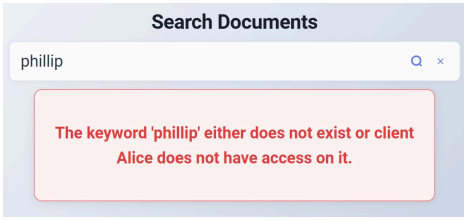
**System Realization of DBO Operations.**

To implement the above-mentioned functionality, the system implements three data structures: an access control matrix, an inverted index, and a file data structure, as briefly discussed below:

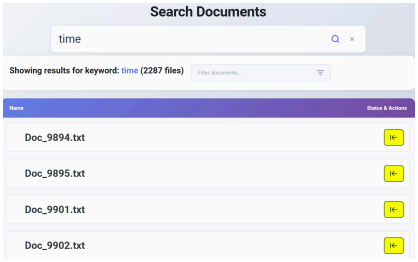
***Access control matrix.*** Once DBO selects keywords, access rights are stored as an $\alpha \times \beta$ access control matrix, where $\alpha$ is the number of clients/groups and $\beta$ the number of keywords; each entry is 0 or a random value, denoting denied or allowed access. Table 1 shows three cleartext files. For keywords `are`, `ana`, `fig`, Table 2 builds this matrix for clients `Lisa` and `Ava`: row 1 holds the keywords, row 2 (gray, ***not*** outsourced) holds their row-ids in the inverted index, and the yellow part gives each client's capability list, e.g., $\langle 0, 1, 2 \rangle$ means `Lisa` can access files with `are`. Values 1, 2, 3 serve as the random numbers.

***Inverted index.*** An inverted index is created for each keyword of the access control matrix, containing the doc-ids/file-ids where that keyword appears. Each row also includes a hash digest over these doc-ids, letting clients verify correctness and completeness. Table 3 shows this index for the three keywords along with the hash digests. A fake file-id 0 is added to `Ana` and `Fig` so all keywords have the same number of file-ids. The gray-colored part is ***not*** outsourced.

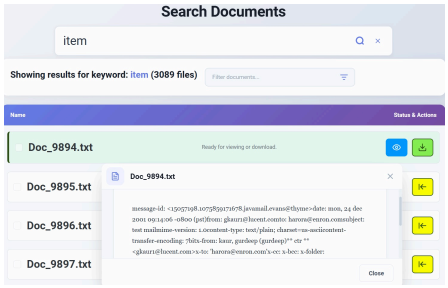
***File data structure.*** DBO appends each file with its id, a hash digest over the file-id and content (enabling client-side file content verification), and a special tag that contains the information about the keywords present in both the access control matrix and file. This tag prevents servers from sending a file containing at least one keyword the client lacks access to. DBO also adds a dummy file to hide the volume (i.e., number of files returned for a keyword) from servers. Table 4 shows file-ids, file content, the hash digest,




(a) Client search interface shows denied access on search keyword.

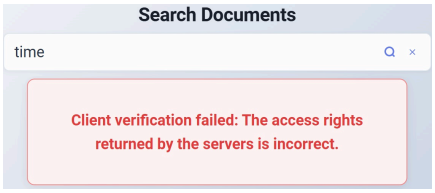



(b) Client retrieves file-ids for a keyword with allowed access.

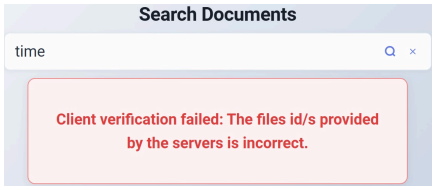



(c) Client retrieves a document with allowed access to all associated keywords.

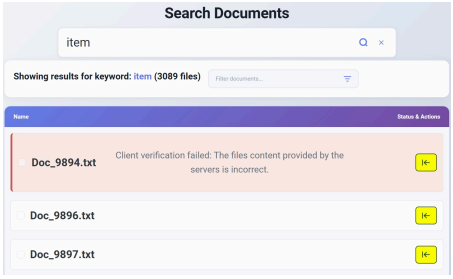
**Figure 4: Client operations.**




(a) Client receives incorrect access right information from the server.




(b) Client receives incorrect file-ids for a keyword.




(c) Client retrieves incorrect file content.

**Figure 5: Client verification under malicious server settings.**

and the tag, with the last row showing a dummy file-id and its content. DBO then creates four Shamir's secret-shares of the three data structures, outsourcing the $i^{th}$ share to server $\mathcal{S}_i$.

*2.1.2* ***Dynamic Operations.*** The system allows DBO to update existing access rights (add or revoke), add new documents/keywords, and delete existing documents/keywords.

**Interface-level Functioning.**

***Updating access rights (Figure 3a):*** DBO can grant or revoke client access rights at any time: new clients are granted access as in §2.1.1, while for existing clients, DBO fetches, updates locally, and then outsources the updated infromation. Figure 3a shows the four states a keyword can be in during this update: (*i*) client has existing allowed access (e.g., "remind," "arnold"), (*ii*) client has existing denied access (e.g., "mark," "note," "folder"), (*iii*) DBO chooses to grant access to keywords (e.g., "note," "folder"), and (*iv*) DBO chooses to revoke access from keywords (e.g., "remind").

***Add/remove new/existing documents (Figure 3b):*** DBO can add new documents that can be indexed by existing or new keywords. In Figure 3b, DBO outsources the new file "Enron-email-new," which may introduce new searchable keywords and/or reuse existing ones (Figure 3c), with access to any new keyword granted as in Figure 3a.

***Add/delete new/existing keywords (Figure 3c):*** DBO can add new keywords to search outsourced files. New keywords are introduced when new files are added; e.g., adding file "Enron-email-new" in Figure 3c, introduces keywords such as "attorney" and "charset".

**System Realization of DBO Operations.**

To implement these functionalities, the system updates the three data structures discussed in §2.1.1: the access control matrix for granting/revoking rights of existing/new clients; the inverted index and file structures for adding/deleting files; the access control matrix alone for deleting keywords; and both the access control matrix and inverted index for adding new keywords from new documents.

## 2.2 Client Operation — Retrieving Documents

**Interface-level Functioning.**

***Querying keyword (Figure 4):*** Client functioning is straightforward: the client searches for a keyword, and if access is denied, sees a message indicating that access is not allowed (see Figure 4a). Note that regardless of whether the keyword is present or not, or if access is denied, the client cannot distinguish. In contrast, if access is allowed, the client instead sees a random list of document-ids and the total number of matching documents (Figure 4b), and can then fetch the document.

***Verification of the results (Figure 5):*** The client can verify the correctness of files returned by the server, across three scenarios: (*i*) If a keyword search does not return the corresponding file-ids, the client may verify whether server provided incorrect results. Figure 4b shows successful verification for "time" when the client has access to the keyword; otherwise, sees the message, as in Figure 5a, indicating incorrect answers by the server. (*ii*) Upon receiving file-ids, the client verifies their correctness for the queried keyword. Figure 4b shows the file-ids for "time" when the client has access and verification succeeds; otherwise, sees the message, as in Figure 5b. (*iii*) Upon receiving the file content, the client verifies its correctness and completeness. Figure 4c shows the file content when the client has access to all its keywords, and verification succeeds; otherwise, sees the message, as in Figure 5c.

## ACKNOWLEDGMENTS

Y. Li was funded by National Natural Science Foundation of China Grant 62372107; S. Mehrotra by NSF Grants 1952247, 2133391, 2032525, and 2008993; and S. Sharma by NSF Grant 2245374.