A Cloud Access Security Broker Based Approach for Encrypted Data Search and Sharing

Chuanyi Liu[1] *, Guofeng Wang[2], Peiyi Han[2], Hezhong Pan[2], Binxing Fang[1,3]

[1] Harbin Institute of Technology(Shenzhen), Shenzhen, 518055, China
[2]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China
[3]Dongguan University of Electronic Science and Technology Institute, Dongguan 523000, China

(* Corresponding Author: cy-liu04@mails.tsinghua.edu.cn)

***Abstract*— Encryption will play a more important role as more computation and storage are outsourced. Achieving good balance between strong security and application functionality preservation becomes a cutting-edge research problem. SE algorithms are well studied, which delegate search capabilities to the cloud provider without decrypting the documents. But this approach imposes extra constraints on cloud API and loses search query expressiveness. This paper proposes a CASB based framework for encrypted search and data sharing, which builds the search index locally and only points the identifiers of ciphertext into cloud, promisingly expresses full set search functionalities while using keys that the user can control. Experimental results are taken to quantitatively analyze the performance overhead and throughput of the broker.**

Keywords*—Encrypted Search; Data Privacy; Data Share; Cloud Access Security Broker (CASB)*

I. INTRODUCTION

According to a recent study by the Cloud Security Alliance (CSA) [1], the security of data in the cloud is now a board-level concern for 61% of organizations. The significant challenge with cloud applications is the customers lost the control over the data stored in the cloud, especially with mission-critical data or privacy-sensitive data.

An effective solution of data protection is encryption, uploading only ciphertext to the cloud. However, it brings a conflict to be weighed and solved between taking advantage of application's full functionality and ensuring sensitive data under customers' control. Data search is one of the most common functions, so the arisen problem is whether customers could still search the ciphertext for the relevant content?

Searchable Encryption (SE) has been well studied in the recent decade [2, 3, 4], allowing the cloud provider to search the encrypted documents directly without decryption. However, this approach has two infeasible problems from practical system's perspective: firstly, application programming interfaces on the cloud side should be modified to invoke the SE implementation library, so as to support search on encrypted documents and corresponding metadata; Secondly, depending on a trapdoor function to transform the plaintext queries into cipher correspondents, and then search action executed by the cloud, the expressiveness of SE is largely degraded, limited to key-word granularity.

Gartner advocates Cloud Access Security Broker (CASB) [5] as an security enforcement point, siting between the cloud application and its users, intercepting critical data before it is passed into the cloud, and replacing it with a random token or encryption value that is meaningless for the cloud. A CASB works by mediating connections between cloud apps and the outside world, typically via a combination of proxies and API connectors to applications. With the CASB architecture introduced, encryption becomes transparent and automatic, and users can preserve fullest possible search functionality. CASB also provides a central point for monitoring and managing access to cloud resources.

Unfortunately, as CASB is more like a business terminology, most of CASB solutions are proposed by commercial companies, and the key techniques and critical details are not public available. On the other hand, some academic works [6, 7, 8, 9, 10] are almost the same with CASB model, but without obvious equivalent key-words, so it's hard for the beginners to build a foundation for further research, neither allow practitioners to find suitable schemes for various application scenarios.

This paper proposes a CASB based framework for encrypted data search and sharing to uncover the hood how CASB works, and analyses connections between the related schemes in a systematic way. It also

This paper is supported by the National High Technology Research and Development Program of China (863 Program) under Grant No.2015AA016001, Production-Study-Research Cooperation Project in Guangdong Province under Grant No.2016B090921001, Innovation projects in Shandong Province under Grant No.2014ZZCX03411

978-1-5090-4588-4/17/$31.00 ©2017 IEEE

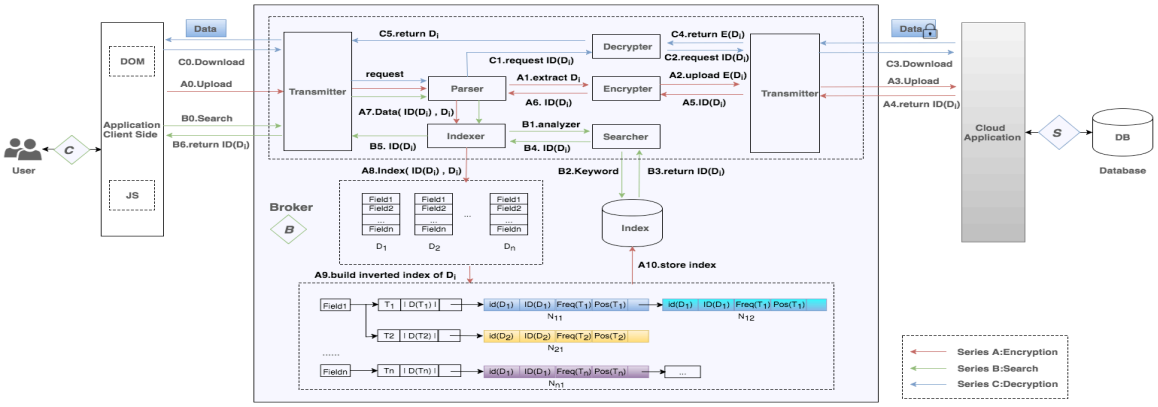

Fig. 1. Architecture for CASB based Encrypted Search

identifies typical technical challenges in this approach and indicates open research problems. Furthermore, this paper gives a quantitative analysis and performance evaluation on real Internet cloud applications as well as the broker itself.

II. Related Work

According to the position of the broker in I/O path and the integration model with cloud services, we classify current state-of-art into the following schemes:

A. Between user and application's client-side

ShadowCrypt [6] works at the chokepoint *C* in Figure 1. ShadowCrypt runs as a browser extension and encrypts data before the application code accesses it. It replaces input elements in a page with secure, isolated shadow inputs and encrypted text with secure, isolated cleartext. However, ShadowCrypt is unable to achieve encryption for non-textual data such as documents, images etc. and can not support any mobile applications.

Mimesis Aegis [7] proposed a privacy-preserving solution for mobile platforms. M-Aegis not only provides isolation but also preserves the user experience through the creation of a conceptual layer called Layer 7.5 (L-7.5), which is interposed between the application (OSI Layer 7) and the user (Layer 8). And M-Aegis only supports encryption for textual data.

B. Between application's client-side and cloud application

Mylar [8] is based on the Meteor JavaScript framework and builds applications that encrypt all their data sent to the server. Developers of Mylar need to tell Mylar what data needs to be encrypted, and the program must be developed in Meteor JavaScript framework affecting backwards compatibility. What's more, data statistics and sorting can not be used in cloud applications with Mylar.

Virtru [9] performs email encryption to keep webmail providers like Gmail from viewing users' data in the clear. Unfortunately, Virtru can not be extended to other web applications and mobile applications and only provides a point solution for a handful of webmail providers.

C. Between cloud applications and database

CryptDB [10] encrypts user's confidential data between server-side and database server. Then it stores encrypted data in the database. CryptDB includes a proxy that interposes on user's operation and translates normal queries into queries on encrypted data. When receiving encrypted results, the proxy decrypts it and returns corresponding plaintext to the user. CryptDB effectively protects confidential data against the database administrator and allows user to query transparently on encrypted data. However, it exists shortcomings and limitations. First, to ensure data confidentiality, user needs to trust both the client-side and server-side application, any compromised party may leak privacy. Second, the proxy requires program logic, which brings backward compatibility issues. Finally, CryptDB cannot compute over data encrypted with different keys. Some operations require multiple encryption and decryption increasing execution time.

III. CASB Based Encrypted Search and Sharing

A. Architecture

As depicted in Figure 1, the broker is responsible for data encryption and decryption, search index building and update, execution of search queries on behalf of the end uses, and management of security keys as well as metadata.

When data are uploaded to the cloud, the transmitter module pass the data stream to a corresponding application parser, where the data structure and semantic are understudied, and only sensitive or pre-configured data fields are encrypted, at the same time corresponding meatadata and encryption keys are stored in the broker. After the ciphertext are successfully uploaded to the cloud, it will return back the data ID in the cloud, and broker should maintain the mapping table

TABLE I. SEMANTIC LIBRARY FOR TENCENT QQ WEBMAIL AS AN EXAMPLE OF APPLICATION PARSER

Application	Function	Request Method	Request URI	Content Type	Encryption Field
QQmail	Send Mail	POST	set3.mail.qq.com/cgi-bin/compose_send?sid=JIq-6XB24WTmf0ke	Key-Vaule format	Subject content
QQmail	Receive Mail	GET	set3.mail.qq.com/cgi-bin/readmail?folderid=1&folderkey=1&t=readmail&mailid=ZC0010-3nAORN1KF5ITb1kOXm	HTML format	Response data

between metadata of the plaintext and ciphertext ID in the cloud.

When end user initiates a search query, the broker will firstly pass the query to the local index, where the query is parsed into keywords, and by searching the reverted keyword index maintained locally, the very ciphertext ID can be got. Broker request the cloud with ciphertext ID to obtain the ciphertext, after retrieving the encryption key and metadata against that ciphertext, the broker decrypts the ciphertext and returns the plaintext to the end user.

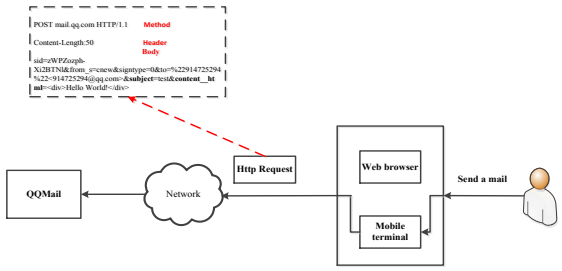

Fig. 2. The protocol of Tencent webmail in http request format

The broker can be deployed as a physical proxy within customer premises; or as a virtual appliance even hosted in the cloud, using specific client tools (such as VPN tunnel, or endpoint agent) to direct the data stream to the broker.

B. *Application Protocol Recognition and Semantic Analysis*

There are two technical challenges for cloud application protocol recognition and semantic analysis: firstly, as cloud applications are various and enormous, it works best if protocol analysis can be automatic and ubiquitous; secondly, as cloud applications always evolve and update, it is important for the broker to adapt the application changes, especially in a timely fashion.

Take webmail for example. Figure 2 depicts the protocol format when sending a web email. The attributes and contents of data stream can be obtained by parsing the Key-value pairs in JSON or XML format embedded into http body. E.g., "subject" means mail subject, "content_html" means mail content, and attachments in multipart format. Each category of

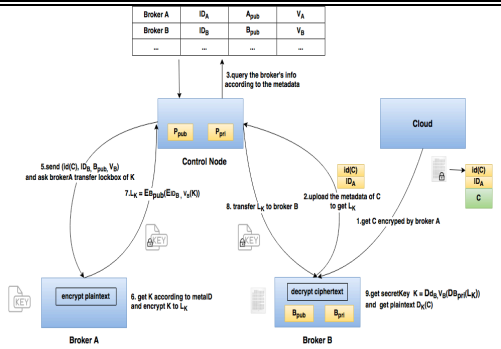

Fig. 3. Procedure of encryption key exchange between different brokers. Where C is ciphertext, id(C) is identity of C, ID_A is identity of broker A, ID_B is id of broker B, K is symmetric key of C, L_K is the lockbox of K, P_{pub} is master public key of control node, P_{pri} is mater private key of control node, B_{pub} is PKE public key of broker B, B_{pri} is PKE private key of broker B, d_B is IBE private key of broker B

applications has a specific parser plugin, which is shown in Figure 1, and semantic of the very kind of applications is abstracted as a finite-state machine, exposing a unified API for invoking. A typical semantic library for Tencent QQ web mail is listed in Table 1.

C. *Search Index Building in CASB*

A local search index is generated in the broker, with pointers to the encrypted data associated with the relevant keywords in the index. When a user searches for data, the search query is executed against this local index, returning all of the associated ciphertext IDs pointing to the cloud, where the encrypted files or records are searched and retrieved, and the broker decrypting the data for end users on the fly. The above procedure is detailed in Figure 1.

It is important to note, with CASB based encrypted search scheme, current modern search engines, such as Elastic Search[11], can be seamlessly integrated without any modification. Compared with other SE schemes, it will provide full set search functionalities at the same time preserving strong cryptography. So search index building in this scheme is almost the same as ordinary search engines [12], only with mapping from keywords to actual files or records are pointing to the encrypted data hosted in the cloud.

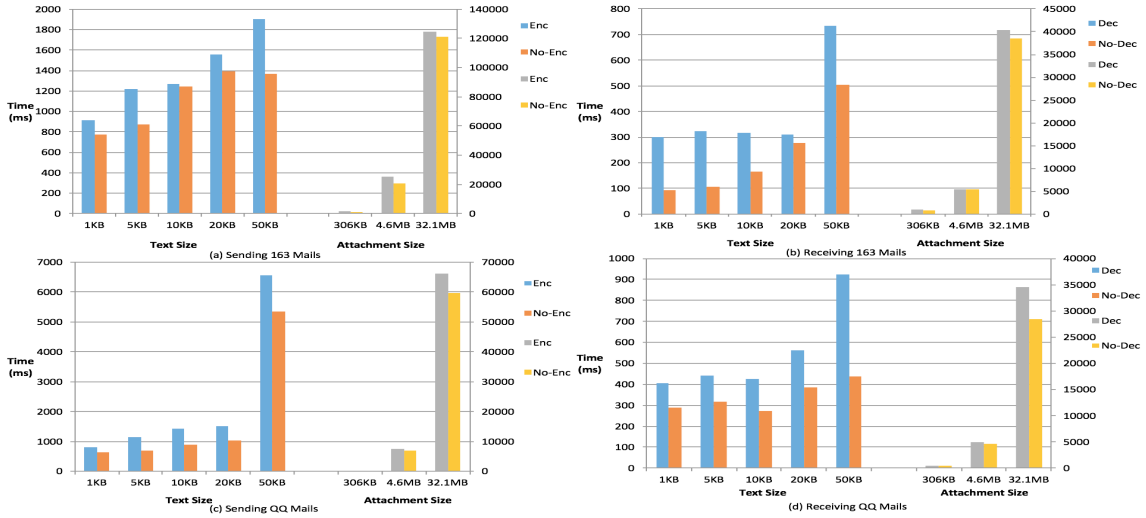

Fig. 4. Performance Evaluation when sending and receiving web emails.
(1) Enc denotes connecting to a broker with sensitive data encrypted and
(2) No-Enc denotes connecting to a broker without parser and encryption

D. Search on Shared Encryption Data

A typical scenario for shared encryption search is that user A, on premise of broker A, wants to share a file with user B who is under another broker, say broker B. So the problem becomes how to share search index and security keys between different brokers. As the search index maintained by the broker are completely compatible with ordinary search engine, there is nothing new than dissemination of distributed search index which is well-established technology, E.g. in Elastic Search[11].

What counts is sharing and exchange of encryption keys. In order to do so, we need a third party, named Control Node (CN for short), responsible for initializing and updating the basic information, such as broker ID, broker public key, key version, etc., as well as orchestrating the exchange of data encryption keys. The procedure is depicted in Figure 3.

When broker B downloads a ciphertext C, encrypted by broker A, from the cloud, it firstly extracts the metadata ($id(C)$, ID_A) appended in ciphertext C, and transmits ($id(C)$, ID_A) to the CN to request the encryption key K for C. Then CN queries broker B's related info initialized and updated maintained and sends the tuple ($id(C)$, ID_B, B_{pub} , V_B) to broker A and ask for key K. When broker A receives the request, it look up the key table locally according to ciphertext ID, and encrypts K using broker B's identifier by Identity Based Encryption [13], expressed as $T = E_{ID_B, V_B}(K)$. After that, broker A further encrypts T with broker B's public key, as $L_K = E_{B_{pub}}(E_{ID_B}, V_B(K))$, and delivers L_K to broker B directly or relayed by CN. Lastly, broker B decrypts the K, $K = D_{d_B, V_B}(D_{B_{pri}}(L_K))$, and decrypts the corresponding plaintext.

IV. Evaluation

The main objectives of evaluation include: search query expressiveness comparisons of SE algorithm based schemes and CASB based scheme in metric of supported query types, and extra overhead introduced by the broker and its throuphput.

As shown in Table II, compared with SE algorithm based schemes, where SEKS denotes the schemes with symmetric key encrypted keyword search, PEKS denotes the schemes with public key encrypted keyword search, prefix indexed denotes the corresponding schemes with search index per document or per keyword, CASB based schme retains full functionalities of modern search engines. And it can be seamlessly integrated with commodiy search engines. Morevoer, while most SE schemes only work on text formd documents in contrast with the variety forms of data streams or records in real cloud applications, CASB based sheme supports more transparently and adaptively.

We implemented a prototype to evaluate the performance. It consists of an endpoint agent on users' computer to filter and direct related data flow to the broker, which is deployed as a virutal appliance hosted in the cloud, with configuration of Intel 2.5GHz two cores equiped 2GB of RAM, and inbound/outbound bandwidth to Internet of 5Mbps.

Two typical web email service, Tencent QQmail[14] and Netease 163 email [15], are chosen as performance evaluation candidates. We simply sent and received email letters with different body size and different formats of the attachment data. Figure 4 shows that the broker incurs throughtput loss of 30%-40% on sending mails. In fact, it costs much time that CASB based

TABLE II. COMPARISONS BETWEEN SE SCHEMES AND CASB BASED SCHEME

Category	Query Expressiveness					
	Single Keyword	Multi-Keyword	Fuzzy Search	Ranked Search	Complex Data Structure Support	Dynamic search
SEKS scheme[2]	✓	✗	✗	✗	✗	✗
PEKS scheme[16]	✓	✗	✓	✗	✗	✗
Indexed SEKS scheme[17]	✓	✓	✗	✗	✗	✗
Indexed PEKS scheme[18]	✓	✓	✗	✗	✗	✗
CASB based system	✓	✓	✓	✓	✓	✓

scheme recognizes protocols and analyzes the semantics of request to mail services. To test big file, we sent and received different attachment data respectively, includes 306KB image, 4.6MB document and 32.1MB executable program. The broker adds just 10%-20% overhead for the encryption of big attachments, because the broker is able to process segments of the attachment data and transform the segment data to the application instead of encrypting the whole content of attachment. As we can see, this performance overhead induced is minor and acceptable according to user's perceptual experience in the millisecond level.

V. CONCLUSIONS

Compared with SE Algorithms, which delegate search capabilities to the cloud provider, CASB based approach, proposed by this paper, builds the search index locally when data stream to upload are parsed and encrypted, at the same time stores corresponding meatadata, encryption keys, and a pointer mapping to ciphertext ID in the cloud. When the user initiates a search query, the broker will firstly pass the query to the local reverted index, with which commodity search engines can be seamlessly integrated without any modification. After looking up the index record locally, the very ciphertext ID can be got, and then to retrieve corresponding encrypted data in the cloud finally. So it achieves both strong security and full set search functionalities.

Certainly, as there is no free lunch, CASB also have technical challenges and research gaps to become practical, such as application recognition and adaption, key management and exchange. This paper also takes experiments to evaluate the performance overhead and throughput of the broker.

REFERENCES

[1] Cloud Security Alliance (CSA) Report: Cloud Adoption Practices & Priorities Survey Report. January, 2015

[2] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC, 44–55.

[3] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: Improved definitions and efficient constructions. In CCS. ACM, New York, NY, 79–88.

[4] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004b. Public key encryption with keyword search. In EUROCRYPT (LNCS), Vol. 3027. 506–522.

[5] Gartner Report: How to Evaluate and Operate a Cloud Access Security Broker. December 8,2015.

[6] He W, Akhawe D, Jain S, et al. Shadowcrypt: Encrypted web applications for everyone[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1028-1039.

[7] Lau B, Chung S, Song C, et al. Mimesis aegis: A mimicry privacy shield–a system's approach to data privacy on public cloud[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 33-48.

[8] Popa R A, Stark E, Valdez S, et al. Building web applications on top of encrypted data using Mylar[C]//11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). 2014: 157-172.

[9] Virtru. http://www.virtru.com

[10] Popa R A, Redfield C, Zeldovich N, et al. CryptDB: protecting confidentiality with encrypted query processing[C] //Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011: 85-100.

[11] ElasticSearch. https://www.elastic.co/products/elasticsearch

[12] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press. 2008.

[13] Dan B, Franklin M. Identity-Based Encryption from the Weil Pairing[J]. Siam Journal on Computing, 2003, 32(3):213-229.

[14] https://mail.qq.com.

[15] http://mail.163.com.

[16] Sedghi S, Van Liesdonk P, Nikova S, et al. Searching keywords with wildcards on encrypted data[C]//International Conference on Security and Cryptography for Networks. Springer Berlin Heidelberg, 2010: 138-153.

[17] Golle P, Staddon J, Waters B. Secure conjunctive keyword search over encrypted data[C]//International Conference on Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2004: 31-45.

[18] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data[C]//Theory of Cryptography Conference. Springer Berlin Heidelberg, 2007:535-554