Achieve Efficient and Verifiable Conjunctive and Fuzzy Queries over Encrypted Data in Cloud

Jun Shao (Visiting Professor), Rongxing Lu*, Yunguo Guan, and Guiyi Wei

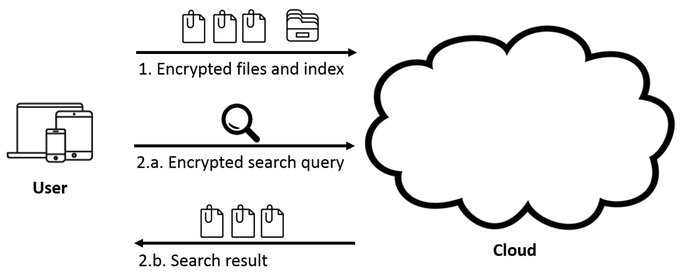
Contact Email: rlu1@unb.ca

Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB)

ABSTRACT

Due to the high demands of searchability over encrypted data, searchable encryption (SE) has recently received considerable attention and been widely suggested in encrypted cloud storage. Typically, the cloud server is assumed to be honest-but-curious in most SE-based cloud storage systems, i.e., the cloud server should follow the protocol to return valid and complete search results to users. However, this trust assumption is not always true due to some unanticipated situations, such as misconfigurations and malfunctions. Therefore, the function of verifiability of search results becomes crucial for the success of SE-based cloud storage systems. For this reason, many verifiable SE schemes have been proposed; however, they either fail to support query operators "OR", "AND", "*" and "?" simultaneously, or require many time-consuming operations. Aiming at addressing this problem, in this paper, we propose a new verifiable SE scheme for encrypted cloud storage. The proposed scheme is characterized by integrating various techniques, i.e., bitmap index, radix tree, format preserving encryption, keyed-hash message authentication code and symmetric key encryption, for achieving efficient and verifiable conjunctive and fuzzy queries over encrypted data in cloud. Detailed security analysis shows that our proposed scheme holds the confidentiality of data and verifiability of search results at the same time. In addition, extensive experiments are conducted, and the results demonstrate our proposed scheme is efficient and suitable for users to retrieve their data from the cloud to their mobile devices.

System Model



Search

- Encrypted search query: The symbols in the query are encrypted by FPE
- Search: Search files according to the radix tree by using the encrypted search query, and return the files with the necessary verifiable values

Encrypted Files and Index

- Encrypted files: $f_i' = E(k_1, f_i)$
- Encrypted index: Extract keywords from files, and use the obtained keywords to build a radix tree. After that, encrypt the keywords by using format preserving encryption (FPE), and add some necessary values.

Ancestor Chain $\mathcal{A}_{p_r} = \emptyset$	
Root \$	$h_{cp_r} = \mathrm{HMAC}(k, \emptyset \| \| \$, \mathcal{S}'_{c_r})$
Children Set $\mathcal{S}'_{c_r}$	

a. root node

Ancestor Chain $\mathcal{A}_{p_i}$	
Node s_i'	$h_{cp_i} = \mathrm{HMAC}(k, \mathcal{A}_{p_i} \| \| s_i', \mathcal{S}'_{c_i})$
Children Set $\mathcal{S}'_{c_i}$	

b. inner node

Leaf #$_i$	$h_{\mathcal{F}'_{w_i}} = \mathrm{HMAC}(k, \mathcal{F}'_{w_i}, w_i)$
Descriptor $\mathrm{Des}_{\mathcal{F}'_{w_i}}$	
Bitmap B_{w_i}	$h_{B_{w_i}} = \mathrm{HMAC}(k, B_{w_i}, w_i)$
Hash set $\mathcal{H}_{\mathcal{F}'_{w_i}} = \{h_{j,w_i} = \mathrm{HMAC}(k, f_j', w_i), \dots\}$	

c. leaf node

Experimental Results

- The underlying dataset is composed by 15,000 random documents larger than 100 bytes from Wikivoyage, and the underlying radix tree is constructed based on 8,000 keywords parsed from the dataset.
- Cloud side: Intel Xeon E5-2603v4 1.7 GHz and 32 GB memory running Ubuntu 16.04 LTS
- User side: Laptop (Intel Core i5-4210U 1.7 GHz and 8 GB memory running Windows 10 Professional), small phone (Kirin 960 and 6 GB memory running Android 7.0)
- The source code and dataset can be downloaded from http://github.com/guanyg/vsse

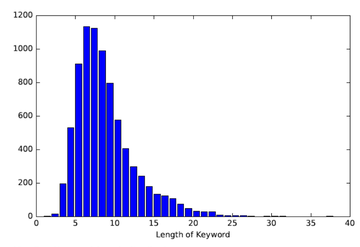


Fig. 7. Keyword length distribution.

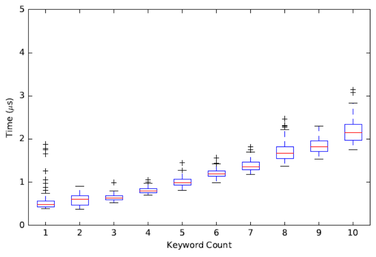
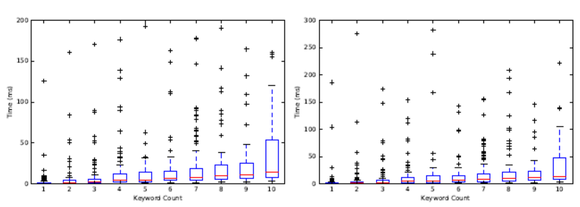


Fig. 8. Search efficiency for queries with operator "OR"



(a) Verifying in a laptop (b) Verifying in a smart phone

Fig. 9. Verification efficiency for queries with operator "OR"

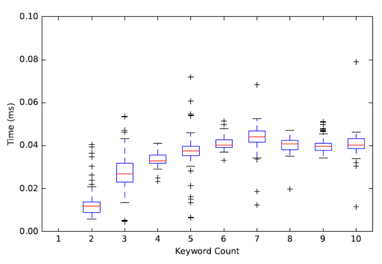
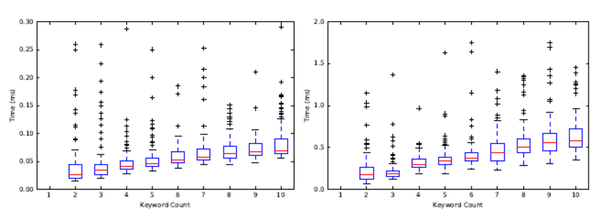


Fig. 10. Search efficiency for queries with operator "AND"



(a) Verifying in a laptop (b) Verifying in a smart phone

Fig. 11. Verification efficiency for queries with operator "AND"

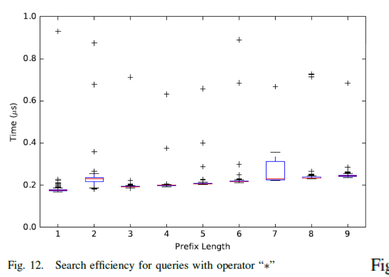
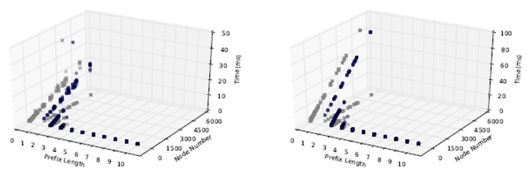


Fig. 12. Search efficiency for queries with operator "∗"



(a) Verifying in a laptop (b) Verifying in a smart phone

Fig. 13. Verification efficiency for queries with operator "∗"

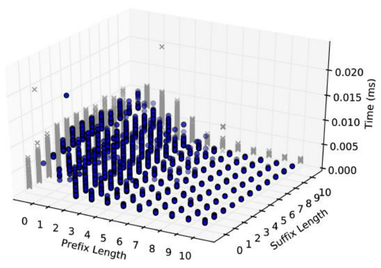


Fig. 14. Search efficiency for queries with operator "?"

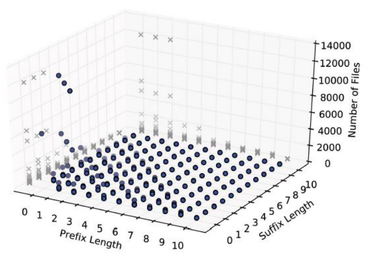
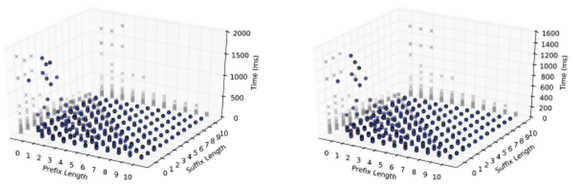


Fig. 15. Number of returned files for queries with operator "?"



(a) Verifying in a laptop (b) Verifying in a smart phone

Fig. 16. Verification efficiency for queries with operator "?"