D-004

Secure Indexes for Keyword Searches in Cloud Storage

Wanasit Tanakitrungruang
The University of Tokyo
wanasit@aida.t.u-tokyo.ac.jp

Nutcha Taneepanichskul
Chulalongkorn
nutcha.tae@student.chula.ac.th

Hitoshi Aida
The University of Tokyo
aida@aida.t.u-tokyo.ac.jp

Abstract

The investigating challenge of this paper is to enable keyword queries on an encrypted cloud storage. We introduce a secure index design based on the work of Hore et al [1] and Eu-Jin Goh [2]. Our color indexing technique yields similar security level to the Hore et al's but allows users to adjust the security level dynamically on each query. A statistical analysis and experiment are also conducted to evaluate the efficiency of color indexing against real-world large datasets.

1. Introduction

Encryption technologies are the fundamental of keeping information secure outside organizations. Encrypting all data before sending it to the cloud, users can ensure the confidentiality of sensitive information. However, the encryption could disable searching and indexing features of the cloud service. If the size of data grows larger, it becomes impractical to download large portion of the data and perform search on the client side. Such a described approach would work well only with a small size or a small number of documents.

Therefore, the investigating challenge of this paper involves creating 'secure indexes' or an auxiliary data structure that enable the server to lookup documents containing a given keyword without disclosing the data inside the documents.

2. Color Indexes

We use a secure pseudo random function to select a set of S unique colors from a large set of C colors to represent word w. We call the set of colors selected for word w as w's color code or $code(w)$. If a document d consist of keywords $w_1, w_2, ..., w_n$, a color index of d or $I(d)$ will be created by combining all color codes of its keywords:

$$I(d) = code(w_1) \cup code(w_2) \cup ... \cup code(w_w)$$

The color index $I(d)$ can be stored in the cloud with encrypted content of d. Note that the index building process is one-way operation. If the pseudo random function is truly secure, it will be difficult for the server to guess which words a color represent. Therefore, the server cannot guess the original content easily just by observing the colors of $I(d)$.

In spite of being secure, $I(d)$ enables the server to do a keyword lookup on the encrypted collection. To lookup a keyword w', we simply recompute $code(w')$ and send $Q \le S$ randomly selected colors of $code(w')$ to the server. Then, let the server return all the documents of which color index contains all those Q colors.

Since all color codes of d's keywords have been added to $I(d)$ when the index was created, if w' is indeed one of d's keywords, $I(d)$ must contain all colors of $code(w')$ and d must be included in the returned results. However, there is also a chance that a combination of color code from some other words creates the set of colors that similar to $code(w')$. Thus, a number of wrong documents will be included in the returned results. We have to finalize the search on the client side by decrypting the returned documents and filtering out those false positive errors.

Those incorrect results naturally spoil the efficiency of documents lookup. They waste network bandwidth and computing resources. However, at the same time, they also provide a crucial benefit for the user security as they help hiding documents access pattern. The user can balance this trade-off between the performance and security level by adjusting value of Q.

2.1 Comparison to the Previous Works

The color indexing technique was originally from Hore et al [1]. The advantage of their design comparing to other previous works is its controllability. The users can adjust trade-off between the performance and security level by adjusting three parameters when creating an index — number of possible colors (C), number of colors computed for a keyword (S), and number of colors selected for a keyword (S').

Our design takes this concept further by allowing the users to adjust the number of colors used for a query (Q) instead of S'. This alteration enables the user to dynamically decide the security level on each query; therefore, provides even more flexible control.

Hore et al. have not described the implementation details of the indexes. We implement the color indexes by adapting the work of Eu-Jin Goh [2]. We found that, after implemented, the color indexes have the structure similar to Eu-Jin Goh's Bloom filters. Our error rate analysis also uses a statistical model resembling probabilistic efficiency of Bloom filter.

2.2 Error Rate Analysis

Consider when building an index, for each word, we randomly pick an S-size subset from C possible colors. Thus, the probability that a $code(w)$ contains a certain color is equal to S/C. We repeat this selection for all words in the document. If the document has n unique terms, the probability that the document's color index contains a certain color will be $1 - (1 - S/C)^n$. The document will match the query when its index contains all the query colors. Suppose we submit a query with Q ($\le S$) colors, the probability that a document will be returned to the submitted query is:

$$P_{hit}(C,S,Q,n) = (1-(\frac{C-S}{C})^n)^Q$$

To calculate the error rate of a given *C*, *S*, *Q* setting, the distribution of the document size in the collection need to be measured. If we know the probability that a document in the collection will have n unique keywords or $P(n)$. The expected error rate can be calculated as:

$$Error(C,S,Q) = \sum_n \left[P_{hit}(C,S,Q,n) \times P(n)\right]$$

Hore et al. and other previous works normally assume each document in the collection has equal or around the same number of keywords $n \approx \bar{n}$, therefore $Error(C,S,Q) \approx P_{hit}(C,S,Q,\bar{n})$. Unfortunately, as shown in the experiment, this assumption is not always true.

3. Experiment

Our experiment is conducted on English Wikipedia dataset. An important statistic that related to the index efficiency is the number of unique terms per each article. Thus, for each article, we count the number of unique terms and plot the histogram of the counting results in Figure 1.

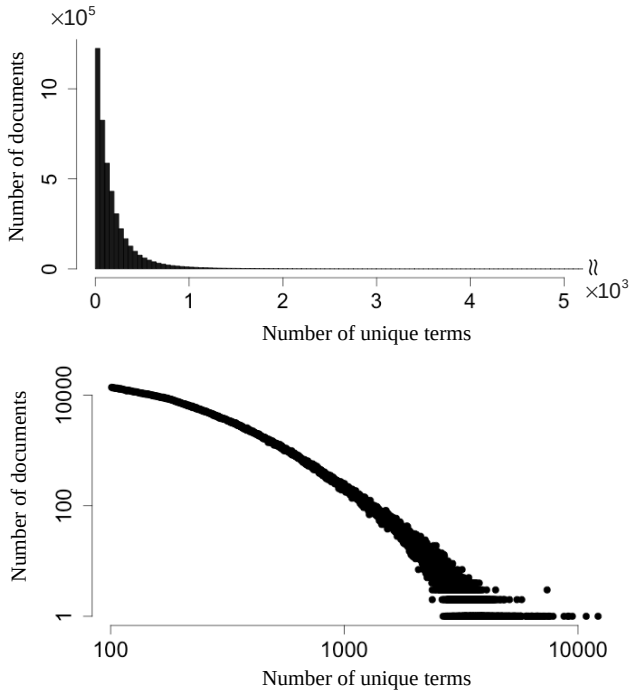

Figure 1. The histogram of unique terms count in normal scale (Top). In log-log scale (Bottom).

As described in Section 2.2, because the distribution is visibly skew, simply using the average number of terms per document ($\bar{n} \approx 197.2$) to estimate the expected error ($Error(C,S,Q) \approx P_{hit}(C,S,Q,\bar{n})$) will result in an inaccurate estimation. For example, the calculated error from $C = 8192$, $S = 8$, $Q = 8$ is less than 10^{-6}, however, real error rate from the experiment is around 0.018 (80,000 false articles from 4.4 million collection).

We measure real color indexes' error rate by querying random keywords that do not exist in the dataset. Those queries should yield empty results in non-encrypted collection, therefore, the returned documents is entirely the false positive error.

We built color indexes for Wikipedia articles using different number of colors (C = 2048, 4096 and 8192) and color code size (S = 4, 6 and 8). On each of those prepared indexes, we queried for random non-exist words with the minimum ($Q = 1$) and maximum ($Q = S$) accuracy query configuration.

	S=4		*S=6*		*S=8*	
	Q=1	*Q=S*	*Q=1*	*Q=S*	*Q=1*	*Q=S*
C=2048	**26%**	**4.5%**	**34%**	**5.7%**	**41%**	**7.1%**
C=4096	**15%**	**1.2%**	**21%**	**1.4%**	**26%**	**1.8%**
C=8192	**8.5%**	**0.2%**	**12%**	**0.2%**	**15%**	**0.3%**

Table 1: Average error rate for 5 queries on different indexes' configurations.

4. Discussion and Future Works

In the experiment, we have discovered that, practically, documents in the collection do not always have the same number of keywords. The document size distribution can have very skewed characteristic and seems to follow power law distribution which conflict with most cryptography literatures' assumption that indexing keywords are evenly distributed. For this reason, many analysis introduced in previous works produce inaccurate estimations.

To correctly predict the color indexes error on a collection, we need to find the $P(n)$ that fit the distribution of document sizes. In practice, however, we can also introduce a sampling technique to estimate the error with reasonably accurate results.

The perceived query efficiency also depends on the keyword. For example, if a user queries for a word 'world' (807,955 referred articles), having ten thousands of errors would be reasonable. In contrast, if the user lookups 'tokyo' (46810 referred articles), the user would not satisfy with the same error rate. The current design has not taken advantage of our improvement by adjusting *Q* for each query.

References:

[1] Hore, B., E.C. Chang, M. H. Diallo, S. Mehrotra. "Indexing Encrypted Documents for Supporting Efficient Keyword Search." *Secure Data Management Lecture Notes in Computer Science Volume 7482, pp 93-110*, 2012.

[2] Goh, Eu-Jin. "Secure Indexes". *IACR Cryptology ePrint Archive*, 2003.