A New Bloom Filter Structure for Searchable Encryption Schemes

Chi Sing Chum
Computer Science Dept., Graduate Center
City Univ. of New York
365 5th Ave, New York, NY 10016 USA
cchum@gradcenter.cuny.edu

Xiaowen Zhang
Computer Science Dept., Graduate Center &
College of Staten Island, City Univ. of New York
2800 Victory Blvd, Staten Island, NY 10314 USA
xiaowen.zhang@csi.cuny.edu

ABSTRACT

We propose a new Bloom filter structure for searchable encryption schemes in which a large Bloom filter is treated as (replaced with) two smaller ones for the search index. False positive is one inherent drawback of Bloom filter. We formulate the false positive rates for one regular large Bloom filter, and then derive the false positive rate for the two smaller ones. With examples, we show how the new scheme cuts down the false positive rate and the size of Bloom filter to a balanced point that fulfills the user requirements and increases the efficiency of the structure.

Keywords

Searchable encryption; cryptographic hash function; Bloom filter; false positive rate;

1. INTRODUCTION

We are dealing with a large volume of data of different forms nowadays. As data grow exponentially, many companies start to store their data outside their premises in cloud storage provided by various cloud providers, e.g., Amazon, Google, etc. The advantages to use cloud storage mainly include shorter setup time, lower implementation cost, "easier up/down scaling," cheaper ongoing cost (pay-as-you-go). However, as data are off the premises, security of the data is a big concern. Encrypting the data before sending to the cloud servers is a solution. But the next question would be how to search on these encrypted data effectively when the user wants to only retrieve a subset of the data without leaking any information to the server. Searchable encryption (SE) provides an answer.

Research on SE has been ongoing for over a decade; there have been many works done in the area [3, 9]. In particular, there are SE schemes that use Bloom filters, for example in [1, 5, 6, 7, 8]. Three entities are involved in SE: querier who queries the data, data owner who initially owns the data, and server who stores and processes the data. A querier issues a query to a server to directly search over encrypted stored data. The query itself also is encrypted, and the server returns the encrypted version of matching results to the user. Searchable encryption scheme provides strong security and privacy guarantees. It does not leak any information to the server; other than returned matching results the querier does not get any information about the rest of the data stored on the server.

The rest of the paper is organized as follows. In Section 2, we describe how Bloom filter works; most importantly, we derive the false positive rate that closely depends on the size of Bloom filter, number of hash functions, and number of keywords stored in Bloom filter. In Section 3, we propose a new structure that splits a Bloom filter into two smaller ones. We use an example to illustrate that the proposed structure balances the Bloom filter size, false positive rate, and user requirements, and ultimately increases the efficiency of an SE. We conclude the paper in Section 4.

2. BLOOM FILTER

2.1 Background

A Bloom filter [2] is a dynamic data structure which can be used effectively to add elements into a set and test whether an element is in a given set. The Bloom filter B consists of a bit array of m bits which are denoted by $B[0], \ldots, B[m-1]$ and initially set to 0. The filter uses r independent uniformly distributed hash functions $h_1, \ldots, h_r$. The hash functions have hash length of s bits, $2^s = m$, and $h_i : \{0,1\}^* \rightarrow \{0,1\}^s$ for $i \in \{1, r\}$. Suppose we have a list of words $D = \{w_1, \ldots, w_l\}$ and want to set up a set for D using a Bloom filter. For each word $w_j \in D$, $j \in [1, \ldots, l]$, the array bits at the positions $h_1(w_j), \ldots, h_r(w_j)$ are set to 1. To see if a word w belongs to D, we check the bits at positions $h_1(w), \ldots, h_r(w)$. If all checked bits are 1's, then w is (possibly) contained in D. However, there is a small possibility of false positive. Even if all the bits corresponding to w are equal to 1, there is no guarantee that w is in D as these bits could be set to 1's by other words. That is, the bits at positions $h_1(w), \ldots, h_r(w)$ in B are 1's and w appears to be in D, but actually is not. But if any one of the checked bits is 0, then w is definitely not in D. It means that there is no false negative.

False positive is one drawback of Bloom filter. However, a Bloom filter does provide an effective tool for searching in searchable encryption. It does not store the word but only those bits corresponding to the hashes of the word are set to 1. This saves space and the server does not learn anything about the words stored in the filter. Adding words

CODASPY'17 March 22-24, 2017, Scottsdale, AZ, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4523-1/17/03.
DOI: http://dx.doi.org/10.1145/3029806.3029839

and testing any given word in a Bloom filter are efficient due to the fast calculation of the hash functions.

For each encrypted document, we create a Bloom filter to store (represent) all its keywords. Therefore, the Bloom filter acts as an index to the document. When the user supplies a keyword to the server, all the Bloom filters will be checked to see if the associated documents need to be selected. This avoids the time-consuming process to scan all the documents. We can even improve the security by storing the encrypted keywords in the Bloom filters and sending the encrypted keyword to the server for searching, like in [5].

2.2 False positive rate

Suppose we want to test if a word w is in the Bloom filter B, which stores l keywords $w_1, \ldots, w_l$ for the document D.

i) w is equal to one of w_j's ($1 \leq j \leq l$). Then, we will get a "Yes" answer and that will be correct.

ii) w is not equal to any one of w_j's. Let $h_1(w) = x_1, \ldots, h_r(w) = x_r$. The probability that x_1th bit of B will not be set to 1 by h_1 on w_1 (assuming hash function h_i is uniformly distributed) is

$$Pr[h_1(w_1) \neq x_1] = \frac{m-1}{m} = 1 - \frac{1}{m}. \quad (1)$$

The probability that x_1th bit of B will not be set to 1 by all r hash functions (assuming all hash functions are independent) is

$$Pr[h_i(w_1) \neq x_1] = \left(1 - \frac{1}{m}\right)^r, 1 \leq i \leq r. \quad (2)$$

Since there are l words in B, the probability that x_1th bit of B will not be set to 1 by all r hash functions on l words:

$$Pr[h_i(w_j) \neq x_1] = \left(1 - \frac{1}{m}\right)^{rl}, 1 \leq i \leq r, 1 \leq j \leq l. \quad (3)$$

So, the probability that x_1th bit of B will be set to 1 by r hash functions on these l words is

$$1 - \left(1 - \frac{1}{m}\right)^{rl}. \quad (4)$$

Assuming that the probabilities of each bit $x_1, \ldots, x_r$ being set to 1 are independent, the probability that all the bits will be set to 1 by r hash functions on these l words (false positive rate) is

$$R = Pr[B[x_1] = 1, \ldots, B[x_r] = 1] = \left(1 - \left(1 - \frac{1}{m}\right)^{rl}\right)^r. \quad (5)$$

However, Bose et al [4] did a detailed analysis and showed that the false positive rate of a Bloom filter B is actually higher than the result stated in Equation (5). This is because from Equation (3) to Equation (5) we assume that all the bits being set to 1 are independent to each other. But this is not the case as collisions may occur.

3. NEWLY PROPOSED STRUCTURE

3.1 Description

The general idea of using a Bloom filter appears in the literature. However, there is no rigorous treatment about what the size of the filter should be. While it is obvious that increasing the size of the filter will decrease the false positive rate, as shown in Equation (5), the relation is not discussed in details. It is through this motivation that we propose how to use/apply Bloom filters more effectively to cut down the false positive rate and the size of Bloom filters to a balance point which fulfills the user requirements.

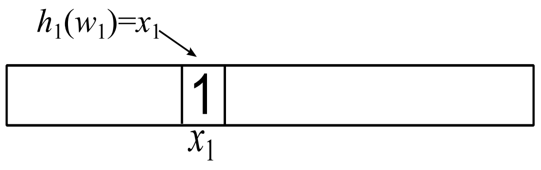

Figure 1: Bloom filter B of size $m = 2^s$.

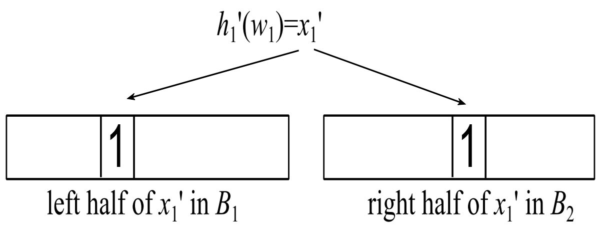

Figure 2: Bloom filters B_1 and B_2 of size $m' = 2^{s'}$ each.

Now suppose we replace these r hash functions h_i (here $1 \leq i \leq r$) with another set of r hash functions h_i' with the length of the hash equaling to $2 \times s'$ and the Bloom filter B with two Bloom filters B_1 and B_2 with size of $m' = 2^{s'}$ each. In essence, we treat one Bloom filter as two smaller Bloom filters.

The procedure remains the same with the exception that we set those bits in $B_1(B_2)$ that correspond to the left (right) halve of the hashes from h_i' to 1. Note that the length of x_i' now equals $2 \times s'$ in bits (see Figure 1 and Figure 2).

Following Equation (5), the false positive rate $R^{'}$

$$\begin{aligned} R^{'} &= \left(1 - \left(1 - \frac{1}{m'}\right)^{rl}\right)^r \left(1 - \left(1 - \frac{1}{m'}\right)^{rl}\right)^r \\ &= \left(1 - \left(1 - \frac{1}{m'}\right)^{rl}\right)^{2r}. \end{aligned} \quad (6)$$

We need to find s' to fulfill the following condition:

$$2^{s'} + 2^{s'} = 2^{s'+1} < 2^s, \quad (7)$$

$$\left(1 - \left(1 - \frac{1}{2^{s'}}\right)^{rl}\right)^{2r} < \left(1 - \left(1 - \frac{1}{2^s}\right)^{rl}\right)^r. \quad (8)$$

Equations (7) and (8) demonstrate the reductions in space and false positive rate, respectively. If such s' exists, that means we can improve the efficiency of the original Bloom filter B by two smaller Bloom filters B_1 and B_2. If there exists a range of values for s', then it is up to the users to choose the one that balances both the space and false positive rate.

Table 1: Original Bloom filter B false positive rate R and space m.

s	R	m	$\log_{10}(m)$
30	8.67362E-15	1073741824	9.03089987

Table 2: New smaller Bloom filter false positive rate $R^{'}$ and space $2m^{'}$

s'	$R^{'}$	$2m^{'}$	$\log_{10}(2m^{'})$
15	8.62138E-11	65536	4.816479931
16	5.40466E-12	131072	5.117509926
17	3.38302E-13	262144	5.418539922
18	2.11598E-14	524288	5.719569918
19	1.32299E-15	1048576	6.020599913
20	8.27024E-17	2097152	6.321629909
21	5.16939E-18	4194304	6.622659905
22	3.23102E-19	8388608	6.9236899
23	2.01944E-20	16777216	7.224719896
24	1.26216E-21	33554432	7.525749892
25	7.88856E-23	67108864	7.826779887
26	4.93037E-24	134217728	8.127809883
27	3.08148E-25	268435456	8.428839879
28	1.92593E-26	536870912	8.729869874
29	1.20371E-27	1073741824	9.03089987

3.2 An example

Given an original Bloom filter B with $r = 2$ hash functions h_1 and h_2 of hash length $s = 30$. Let $l = 50$ be the number of keywords in B. Table 1 calculates the false positive rate $R = \left(1 - \left(1 - 1/2^{30}\right)^{100}\right)^2$ and the space $m = 2^s = 2^{30}$. Suppose we want to replace B with two smaller Bloom filters B_1 and B_2 with 2 hash functions h_1' and h_2' of hash length $2 \times s'$. We want to find out a range of s' such that there will be a reduction in both false positive rate and space. Table 2 calculates the false positive rates $R^{'} = \left(1 - \left(1 - 1/2^{s'}\right)^{100}\right)^4$ and the space $2m^{'} = 2^{s'+1}$.

Based on Equations (7) and (8), we are developing a method to effectively estimate the feasible range of s'. From Tables 1 and 2, hash lengths s' from 20 to 28 fulfill the requirements of the reductions in both false positive rate and space. The calculation of false positive rate is based on Equation (5). However, we can follow the idea mentioned in [4]. For the purpose of this paper, the exact calculation of the rate is not so important. Again, we assume there is no collision among l words in the Bloom filters B, B_1, and B_2. It means that rl bits are set to 1 in each Bloom filter.

4. CONCLUSIONS

We propose a technique to improve the efficiency of Bloom filters used for searchable encryption schemes.

A) Practical considerations: As we know the false positive rate will be reduced if we use a larger Bloom filter, or more hash functions. However, using a larger filter may not be efficient and difficult to implement. In the same way, using more hash functions may not be necessary, as they may degrade the performance.

We suggest using a cryptographic hash function for the following reasons:

1. By repeating the underlying compress function, it can take any message of arbitrary length.
2. Good performance due to collision, pre-image, and second pre-image resistance properties of a cryptographic hash function.
3. We can simply extract certain portions (for example, the first 30 bits), or combine portions of the hash value, as a new hash function.

B) Implementation of the automated system: After estimating a reasonable upper bound for the number of keywords and the acceptable false positive rate, we design a Bloom filter. Then we apply the above technique to subdivide the original Bloom filter B into smaller ones B_1 and B_2 for improvement. Based on the goals of the user, space vs. false positive rate, the optimum size of the smaller Bloom filters can thus be adjusted.

5. REFERENCES

[1] S. Bellovin and W. Cheswick. Privacy-enhanced searches using encrypted bloom filters. Technical Report CUCS-034-07, Dept. of Computer Science, Columbia Univ., Sept. 2007, 1–16.

[2] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Comm. of the ACM*, 13(7):422–426, 1970.

[3] C. Bosch, P. Hartel, W. Jonker, and A. Peter. A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2), 2014. Article 18.

[4] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, and P. Morin. On the false-positive rate of bloom filters. Technical Report TR-07-07, School of Computer Science, Carleton University.

[5] E.-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003.

[6] S. Pal, P. Sardana, and A. Sardana. Efficient search on encrypted data using bloom filter. In *Proc. of 2014 Int. Conf. on Computing for Sustainable Global Development (INDIACom)*, pages 412–416, 2014.

[7] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. Choi, W. George, A. Keromytis, and S. Bellovin. Blind seer: A scalable private dbms. In *Proc. of 2014 IEEE Symposium on Security and Privacy*, pages 359–374, 2014.

[8] M. Raykova, B. Vo, S. Bellovin, and T. Malkin. Secure anonymous database search. In *Proc. of 2009 CCSW, ACM*, pages 115–126, 2009.

[9] Q. Tang. Search in encrypted data: theoretical models and practical applications. Technical Report 2012/648, IACR ePrint Cryptography Archive, 2012.