Searchable Symmetric Encryption Scheme Implementation of a Secure Index

Student: Jacob Chen
Faculty Advisor: Dr. Xiaowen Zhang

Introduction

Cloud computing has experienced exponential growth over the last decade. Public cloud computing is when a third-party cloud service provider manages all hardware, software, and other supporting infrastructure for a company. Public cloud computing provides lower costs, no maintenance, better scalability, and high reliability.

The purpose of this project is to develop a secure index which will allow a user to search encrypted data. This is achieved by creating a trapdoor when building the secure index. A trapdoor can only be generated with a private key. This trapdoor is then used as a private key to generate a codeword. The codeword is added to a bloom filter. To search the encrypted data, the user must have the private key. Without the private key the user will not be able to generate the correct trapdoor which will also cause the codeword to be incorrect.

Hypothesis

How can a user store encrypted documents on an untrusted third-party server and search their files without first decrypting?

Terminology

Bloom filter – a space-efficient probabilistic data structure that has two main functions: to add an item to a set and to determine membership of a set.

HMAC-SHA1 – a type of message authentication code (MAC) with two parameters a secret key and a message of any size. The output (MAC) is a string with a fixed length of 160 bits

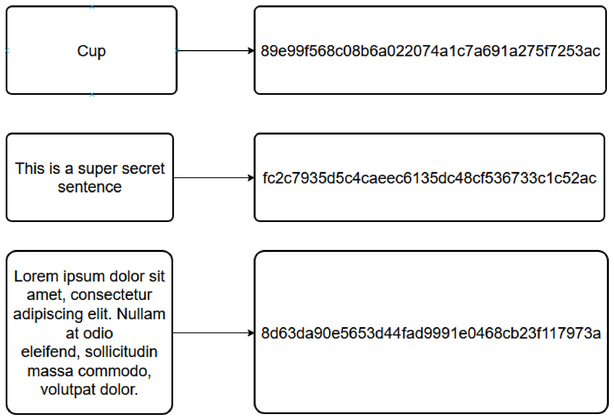

Figure 0. Passing a secret-key and message into a HMAC-SHA1 function

Trapdoor – the output from a HMAC-SHA1 function whose parameters are the private key and plaintext word

Codeword – the output from a HMAC-SHA1 function whose parameters are the trapdoor and document name

Screenshots

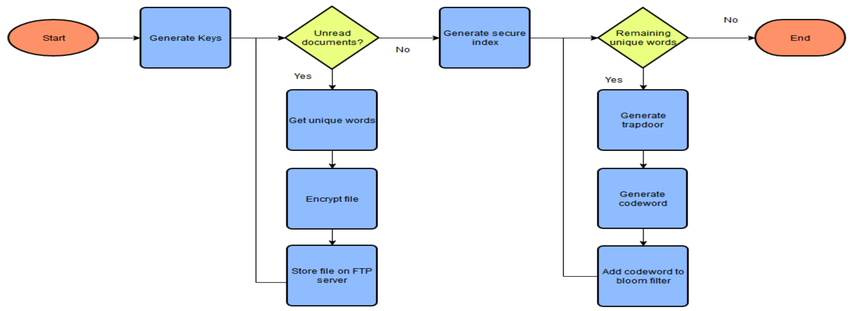

Figure 1. Flowchart for creating a secure index

Figure 2. Console output of the secure index

Figure 3. Encrypted documents located on a server

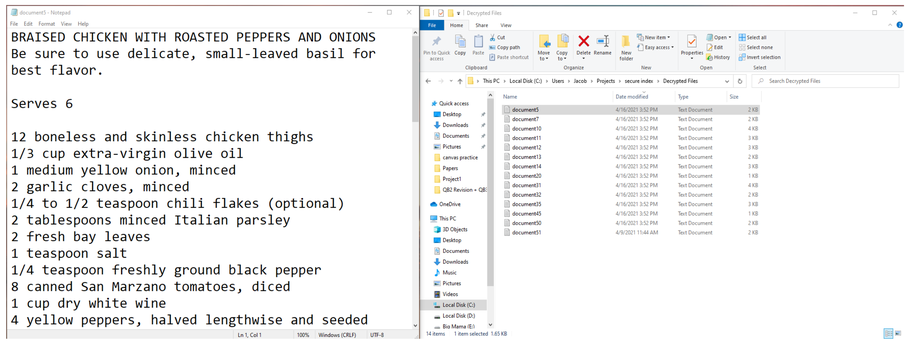

Figure 4. Decrypted documents on a local machine

Insertion into a Bloom Filter

For each unique word in a document, create a trapdoor by calling the HMAC-SHA1 function

- HMAC-SHA1(secret-key, cup)
- Trapdoor = 89e99f568c08b6a022074a1c7a691a275f7253ac

Create a codeword by calling the HMAC-SHA1 function again using the trapdoor and document name as the parameters

- HMAC-SHA1(89e9…53ac, document1)
- Codeword = 0f9b0606f84f83aaf821346cb1c420da25c0**0e7d**

Take the last two bytes of the codeword and convert it into an integer

- 0e7x (hex) = 3709 (integer)

In the bloom filter set the bit in index[3709] to 1.
This effectively adds the word cup to document1

Value	0	0	0	…	1	0	0	…	0
Index	0	1	2	…	3709	3710	3711	…	65535

Figure 5. Sample bloom filter for insertion

Continue this process for each unique word in the document

Searching the Secure Index

Value	0	0	0	…	1	…	1	…	1	…	0
Index	0	1	2	…	23654	…	37016	…	54017	0	65535

Figure 6. A secure index is an array of bloom filters. This figure represents a sample bloom filter for document3. It has three unique words whose index locations are set to 1.

Generate the trapdoor by using the private key and search word

- HMAC-SHA1(secret-key, beef)
- Trapdoor = f9c97f054ac8cfb31989a7de4c41f701c01300a8

Generate a codeword using the trapdoor and each document name then convert the last two bytes of the codeword to an integer

- HMAC-SHA1(f9c9…00a8, document1) = 731b = 29467
- HMAC-SHA1(f9c9…00a8, document2) = 5bd6 = 23510
- HMAC-SHA1(f9c9…00a8, document3) = 5c66 = **23654**
- HMAC-SHA1(f9c9…00a8, document4) = c52b = 50475

In document3 index[23654] is set to 1 indicating that the word beef is in document3

Space and Time Complexity

Time complexity = O(n)

- It takes O(1) time to check an individual bloom filter but there are n bloom filters, therefore the time complexity to search is O(n)

Space complexity = O(n)

- O(n) = each document has their own bloom filter

References

- Bloom, Burton H. "Space/time trade-offs in hash coding with allowable errors." *Communications of the ACM* 13.7 (1970): 422-426.
- Chum, Chi Sing, Xinzhou Wei, and Xiaowen Zhang. "A Split Bloom Filter for Better Performance." *Journal of Applied Security Research* 15.2 (2020): 147-160.
- Goh, Eu-Jin. "Secure indexes." *IACR Cryptol. ePrint Arch.* 2003 (2003): 216.