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

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

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 = 89e99f568c088b6a022074a1c7a691a275f7253ac

Create a codeword by calling the HMAC-SHA1 function again using the trapdoor and document name as the parameters
- HMAC-SHA1(secret-key, document1)
- Codeword = 09b8066f84b3a9f213436bc1c20ada2c50e7d

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

Searching the Secure Index
Continue this process for each unique word in the document

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 its own bloom filter

References