Searchable Symmetric Encryption

Rachit Garg
CS14B050

Aravind Birudu
CS14B004

November 19, 2017

Based on paper
Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions.
Published by Reza Curtmola, Juan Garay, Seny Kamara, Rafail Ostrovsky
Work appeared in ACM conference on CCS in 2006 [6]

Contents

1 Problem Motivation

Nowadays searching is the primary way we access data. We google stuff for most things. We are outsourcing more and more information to third parties and we trust them less and less. This problem concerns governments, research scientists, industries and any user trying to trust other entities with their data.

Suppose the government wants to perform some filtered analysis on census data, and may wish to outsource the storage and computation on this massive dataset to some untrusted server. An example in present-day India is the Aadhaar project. The project is encountering serious difficulties due to privacy concerns and risks being shelved [5]. At least part of these concerns could be addressed by enabling controlled filtered computation over a massive encrypted database.

Research scientists who work on genomic data need sensitive data about a persons identity to continue with their research. Volunteers who would like to be a part of this research would typically want that they dont have to trust the researchers and there are privacy policies preventing the abuse of their sensitive data. In this setting too there is a need for creating protocols that can search on encrypted data and satisfy the security guarantees needed and are also efficient.

Industry also is increasingly getting involved in promoting efficient computation on encrypted data. The rise in usage of cloud computing raises a lot of question on performing secure, efficient encrypton. Also over the last few years, machine learning algorithms have solved big problems. Industry is trying to support open participation in machine learning by usage of structure preserving encryption schemes [1].

2 Problem Statement

Suppose that the client has a collection of documents that consists of a set of words. Client generates Encrypted Database (EDB), which refers to some data structure that allows some form of search and encrypts document collection and sends everything to the server. We want to achieve the following properties :-

Functionality: Server should support the queries from clients where a client asks for a keyword and the server returns the corresponding documents that contain the keyword.

Privacy: Allow server to help, but reveal as little as possible.

Goal: The leakage to the server should be minimum while the scheme being efficient.

Formally we define the problem as follows:
Let $\Delta = (w_1, ..., w_d)$ be a dictionary of d words in lexicographic order, and 2^{Δ} be the set of all possible documents with words in Δ. We assume $d = poly(k)$ and that all words $w \in \Delta$ are of length polynomial in k. Furthermore, let $D \subseteq 2^{\Delta}$ be a collection of $n = poly(k)$ documents $D = (D_1, ..., D_n)$, each containing $poly(k)$ words. Let $id(D)$ be the identifier of document D, where the identifier can be any string that uniquely identifies a document such as a memory location. We denote by $D(w)$ the lexicographically ordered list consisting of the identifiers of all documents in D that contain the word w. The client wishes to perform queries so that knowing w, it is able to get $D(w)$ from the server. Next we define the problem statement formally in the SSE scenario.

2.1 Searchable Symmetric Encryption(SSE)

Client keeps an encrypted data structure on the server such that client wants the server to send encrypted documents associated with a particular keyword. To do this, client will send a token that encapsulates the keyword without revealing information about it. The server will use the token with the help of the encrypted data structure to send encrypted documents associated with that keyword.
Note that we're not searching the entire contents of the database, only a set of keywords that have been indexed in advance.

We begin by considering the formal definition of an index-based SSE scheme. The participants in a single-user SSE scheme include a client that wants to store a private document collection $D = (D_1, ..., D_n)$ on an honest-but-curious server in such a way that

1. The server will not learn any useful information about the collection; and that
2. The server can be given the ability to search through the collection and return the appropriate (encrypted) documents to the client.

We consider searches to be over documents but any SSE scheme as described below can be trivially extended to be used with collections of arbitrary files (e.g., images or audio files) as long as the files are labeled with keywords.

Definition 2.1. *Searchable symmetric encryption* is an index-based SSE scheme over a dictionary Δ is a collection of five polynomial-time algorithms $SSE = (Gen, Enc, Trpdr, Search, Dec)$ such that,

$K \leftarrow Gen(1^k)$: is a probabilistic key generation algorithm that is run by the user to setup the scheme. It takes as input a security parameter k, and outputs a secret key K.

$(I, c) \leftarrow Enc(K, D)$: is a probabilistic algorithm run by the user to encrypt the document collection. It takes as input a secret key K and a document collection $D = (D_1, ..., D_n)$, and outputs a secure index I and a sequence of ciphertexts $c = (c_1, ..., c_n)$. We sometimes write this as $(I, c) \leftarrow Enc_K(D)$.

$t \leftarrow Trpdr(K, w)$: is a deterministic algorithm run by the user to generate a trapdoor for a given keyword. It takes as input a secret key K and a keyword w, and outputs a trapdoor t. We sometimes write this as $t \leftarrow Trpdr_K(w)$.

$X \leftarrow Search(I, t)$: is a deterministic algorithm run by the server to search for the documents in D that contain a keyword w. It takes as input an encrypted index I for a data collection D and a trapdoor t and outputs a set X of (lexicographically-ordered) document identifiers.

$D_i \leftarrow Dec(K, c_i)$: is a deterministic algorithm run by the client to recover a document. It takes as input a secret key K and a ciphertext c_i , and outputs a document D_i . We sometimes write this as $D_i \leftarrow Dec_K(c_i)$.

An index-based SSE scheme is correct if for all $k \in N$, for all K output by $Gen(1^k)$, for all $D \subseteq 2^{\Delta}$, for all (I, c) output by $Enc_K(D)$, for all $w \in \Delta$,

$$Search(I, Trpdr_K(w)) = D(w) \qquad\qquad Dec_K(c_i) = D_i, \text{ for } 1 \leq i \leq n.$$

3 Summary of known solutions

We discuss a summary of known solutions previous to the paper we studied. We know of different ways to compute on encrypted data, each based on one of the following cryptographic primitives:

- *Property-Preserving Encryption(PPE)* PPE schemes encrypt messages in a way that leaks certain properties of the underlying message. There are different types of PPE schemes that each leak different properties. The simplest form is deterministic encryption which always encrypts the same message to the same ciphertext. The property preserved by deterministic encryption is equality since, given two encryptions $c_1 = Enc_K(m_1)$ and $c_2 = Enc_K(m_2)$, one can test if the underlying messages are equal by just checking if $c_1 = c_2$. More complex PPE schemes include order-preserving encryption (OPE) and orthogonlity-preserving encryption. PPE-based encrypted search was first proposed in the Database community and later studied more formally in the Cryptography community [2]. The drawbacks of this approach were that the encrypted database was vulnerable to frequency analysis.

- *Functional Encryption(FE)* Functional encryption supports restricted secret keys that enable a key holder to learn a specific function of encrypted data, but learn nothing else about the data [4]. For example, given an encrypted program the secret key may enable the key holder to learn the output of the program on a specific input without learning anything else about the program. The FE-based approach, resulted in schemes with slow search ($O(n)$) but with better security guarantees.

- *Fully-Homomorphic Encryption(FHE)* An FHE scheme enables computations to be performed on encrypted data without needing to first decrypt the data [7]. FHE schemes are presently too slow for an efficient solution to exist using this scheme.
- *Oblivious RAM(ORAM)* Basic idea behind ORAM's is that we try to hide the access pattern(i.e. the location in memory where the client performs queries(read,write) on). Oblivious RAM solutions provide the highest security, but suffer from the drawback of being too inefficient. Goldreich and Ostrovsky showed that any ORAM solution requires a lower bound of logarithmic blow up in rounds when constructing an ORAM model [9]. Boneh-Kushilevitz-Ostrovsky-Skeith showed a scheme with sqrtDB communication with constant rounds [3].
- *Secure two-party computation(2PC)* The goal of 2PC is to create a generic protocol that allows two parties to jointly compute an arbitrary function on their inputs without sharing the value of their inputs with the opposing party. Yao introduced a garbled circuit protocol is one such protocol that showed how to achieve this property.
- *Searchable Symmetric Encryption(SSE)* SSE was first introduced by Song, Wagner and Perrig [10]. Goh gave formal definition for security in these schemes and pointed out that SSE schemes were not normal encryption schemes and, therefore, the standard notion of CPA-security was not meaningful/relevant for SSE [8]. But their definitions didn't capture the fact that those were (implicitly) restricting the adversary's power, and didn't explicitly capture the fact that the constructions were leaking information. The paper we studied was the first to provide a security definition for SSE in the adaptive setting which took into account the security of documents, trapdoors while considering leakage from the length of the documents, the search outcomes and the search pattern.

4 Security Definitions

Let Δ be a dictionary and $D \subseteq 2^{\Delta}$ be a document collection over Δ.

Definition 4.1. *History*:A q-query history over D is a tuple $H = (D, w)$ that includes the document collection D and a vector of q keywords $w = (w_1, ..., w_q)$.

Definition 4.2. *Access Pattern*: The access pattern induced by a q-query history $H = (D, w)$, is the tuple $\alpha(H) = (D(w_1), ..., D(w_q))$.

Definition 4.3. *Search Pattern*: The search pattern induced by a q-query history $H = (D, w)$, is a symmetric binary matrix $\sigma(H)$ such that for $1 \leq i, j \leq q$, the element in the i^{th} row and j^{th} column is 1 if $w_i = w_j$, and 0 otherwise.

Definition 4.4. *Trace* The trace induced by a q-query history $H = (D, w)$, is a sequence $\tau(H) = (|D_1|, ..., |D_n|, \alpha(H), \sigma(H))$ comprised of the lengths of the documents in D, and the access and search patterns induced by H.

Definition 4.5. *Non-singular history* We say that a history H is non-singular if (1) there exists at least one history $H' \neq H$ such that $\tau(H) = \tau(H')$; and if (2) such a history can be found in polynomial-time given $\tau(H)$.

We note that the existence of a second history with the same trace is a necessary assumption, otherwise the trace would immediately leak all information about the history.

Next we define the notion of adaptive semantic security in the SSE model, we note that this was one of the main contributions of the paper we studied.

Definition 4.6. *Adaptive Semantic Security*: Let $SSE = (Gen, Enc, Trpdr, Search, Dec)$ be an index based SSE scheme, $k \in N$ be the security parameter, $A = (A_0, ..., A_q)$ be an adversary such that $q \in N$ and $S = (S_0, ..., S_q)$ be a simulator and consider the following probabilistic experiments $Real^*_{SSE,A}(k)$ and $Sim^*_{SSE,A,S}(k)$:

```
$Real^*_{SSE,A}(k)$
$K \leftarrow Gen(1^k)$
$(D, st_A) \leftarrow A_0(1^k)$
$(I, c) \leftarrow Enc_K(D)$
$(w_1, st_A) \leftarrow A_1(st_A, I, c)$
$t_1 \leftarrow Trpdr_K(w_1)$
for $2 \leq i \leq q$,
    $(w_i, st_A) \leftarrow A_i(st_A, I, c, t_1, ..., t_{i-1})$
    $t_i \leftarrow Trpdr_K(w_i)$
let $t = (t_1, ..., t_q)$
output $\vee = (I, c, t)$ and $st_A$
```

```
$Sim^*_{SSE,A,S}(k)$
$(D, st_A) \leftarrow A_0(1^k)$
$(I, c, st_S) \leftarrow S_0(\tau(D))$
$(w_1, st_A) \leftarrow A_1(st_A, I, c)$
$(t_1, st_S) \leftarrow S_1(st_S, \tau(D, w_1))$
for $2 \leq i \leq q$,
    $(w_i, st_A) \leftarrow A_i(st_A, I, c, t_1, ..., t_{i-1})$
    $(t_i, st_S) \leftarrow S_1(st_S, \tau(D, w_1, w_2, ....w_i))$
let $t = (t_1, ..., t_q)$
output $\vee = (I, c, t)$ and $st_A$
```

We say that SSE is adaptively semantically secure if for all polynomial-size adversaries $A = (A_0, ..., A_q)$ such that $q = poly(k)$, there exists a non-uniform polynomial-size simulator $S = (S_0, ..., S_q)$, such that for all polynomial-size D,

$$|Pr[D(\vee, st_A) = 1 : (\vee, st_A) \leftarrow Real^*_{SSE,A}(k)] - Pr[D(\vee, st_A) = 1 : (\vee, st_A) \leftarrow Sim^*_{SSE,A,S}(k)]| \leq negl(k)$$

where the probabilities are over the coins of Gen and Enc.

5 Solution mentioned in the paper

Inverted Index Solution We introduce some more notaion. Let $\delta(D) \subseteq \Delta$ be the set of distinct keywords in the document collection D. We assume that keywords in Δ can be represented using at most l bits. Also, recall that n is the number of documents in the collection and that $D(w)$ is the set of identifiers of documents in D that contain keyword w ordered in lexicographic order.

- For each distinct keyword $w_i \in \delta(D)$, a linked list L_i is created that points to all documents containing that keyword.
- We then store all the nodes of all the lists in the array A permuted in a random order and encrypted with randomly generated keys.
- Before encrypting the j^{th} node of list L_i , it is augmented with a pointer (with respect to A) to the $(j + 1)^{th}$ node of L_i , together with the key used to encrypt it.

- Note that by storing the nodes of all lists L_i in a random order, the length of each individual L_i is hidden.
- We then build a look-up table T that allows one to locate and decrypt the first node of each list L_i .
- The client generates both A and T based on the plaintext document collection D, and stores them on the server together with the encrypted documents.
- When the user wants to retrieve the documents that contain keyword w_i , it computes the decryption key and the address for the corresponding entry in T and sends them to the server.
- The server locates and decrypts the given entry of T, and gets a pointer to and the decryption key for the first node of L_i . Since each node of L_i contains a pointer to the next node, the server can locate and decrypt all the nodes of L_i , revealing the identifiers in $D(w_i)$.

The solution also introduces padding to only reveal the leakage of the access pattern, the search pattern, the total size of the encrypted document collection, and the number of documents it contains. To achieve this, a certain amount of padding to the array and the table are necessary. This helps them in proving the inverted index solution earlier mentioned into an adaptive secure construction. The paper also mentions the full reduction to show their construction as adaptively secure.

References

[1] Encrypted data for efficient markets. https://medium.com/numerai/encrypted-data-for-efficient-markets-fffbe9743ba8.

[2] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'07.

[3] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith, III. Public key encryption that allows pir queries. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'07.

[4] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[5] C.Arun. Privacy is a fundamental right. http://www.thehindu.com/opinion/lead/lead-article-on-aadhaar-bill-by-chinmayi-arun-privacy-is-a-fundamental-right/article8366413.ece.

[6] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, 2006.

[7] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[8] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[9] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*.

[10] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, 2000.