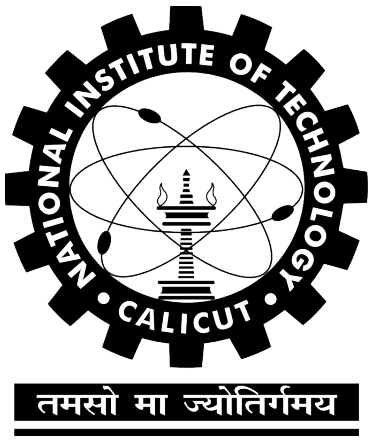
Dynamic Searchable Symmetric Encryption

A
Seminar Report

by

Arpan Kapoor
B120555CS



Department of Computer Science and Engineering
National Institute of Technology, Calicut
Monsoon-2015

National Institute of Technology, Calicut
Department of Computer Science and Engineering

Certified that this Seminar Report entitled

Dynamic Searchable Symmetric Encryption

is a bonafide record of the Seminar presented by

Arpan Kapoor
B120555CS

in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Vinith R.
(Group Seminar Coordinator)
Department of Computer Science and Engineering

Acknowledgement

I would like to acknowledge the help and guidance received from several people in preparation of this report.

First and foremost, I would like to thank the course coordinators Lijiya A, Sameer Mohamed Thahir, Vinith R and Ibrahim A for helping us choose an appropriate topic for the seminar.

I would like to thank Dr. Vinod Pathari for his Computer Security course, which spurred my interest in that domain.

Finally, I would like to thank my family and friends for their continual support in all my endeavours.

Abstract

Searchable symmetric encryption (SSE) allows a client to encrypt its data in such a way that this data can still be searched. The most immediate application of SSE is to cloud storage, where it enables a client to securely outsource its data to an untrusted cloud provider without sacrificing the ability to search over it. A practical SSE scheme should (at a minimum) satisfy the following properties: sub-linear search time, security against adaptive chosen-keyword attacks, compact indexes and the ability to add and delete files efficiently. A SSE scheme to satisfy all the properties outlined above is discussed.

Contents

1 Introduction

Cloud storage is ubiquitous and widely used for services such as backups or outsourcing data to reduce operational costs. However, these remote servers cannot be trusted, because administrators, or intruders with root rights, have full access to the server and consequently to the plaintext data. Thus, to store sensitive data in a secure way on an untrusted server, the data has to be encrypted. Simple encryption is however a hindrance to search capabilities for the data owner. A trivial but *impractical* solution to re-enable searching functionality is to download the whole database, decrypt it locally, and then search for the desired results in the plaintext data.

Searchable symmetric encryption (SSE) is a cryptographic primitive addressing encrypted search. It allows a client to encrypt data in such a way that it can later generate search tokens to send as queries to a storage server. Given a token, the server can search over the encrypted data and return the appropriate encrypted files.

The most immediate application of SSE is to the design of searchable cryptographic cloud storage systems which can provide end-to-end security for cloud storage systems without sacrificing utility. Other applications include the design of graph encryption schemes and controlled disclosure mechanisms [1].

A perfect solution to searchable encryption is theoretically offered by Fully Homomorphic Encryption. It is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. If the operation is that of searching, we have a perfect solution to our problem of searching encrypted data. However, Fully Homomorphic Encryption was an open problem till 2009, when Craig Gentry described the first plausible construction for a fully homomorphic encryption scheme using lattice-based cryptography.

Even though it is now known that we can have Fully Homomorphic Encryption, all current solutions are extremely inefficient and impractical for most real-world situations.

2 Preliminaries

Definition 1 (Symmetric Key Encryption)**.** [1] A private-key encryption scheme is a set of three polynomial-time algorithms $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. The *key-generation algorithm* Gen is a probabilistic algorithm that outputs a secret key K.
2. The *encryption algorithm* Enc takes as input a key K and a message m and returns a ciphertext c.
3. The *decryption algorithm* Dec takes as input a key K and a ciphertext c and returns m if K was the key under which c was produced.

Definition 2 (Pseudorandom Function)**.** [2] A polynomial-time computable function that cannot be distinguished from random functions by any probabilistic polynomial-time adversary.
A function $F\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is a PRF if

- Given a key $K \in \{0,1\}^s$ and an input $X \in \{0,1\}^n$ there is an *efficient* algorithm to compute $F_K(X) = F(X, K)$.
- $F_K(i)$ is indistinguishable from a random function, that is, given any $x_1, \ldots, x_m$, $F_K(x_1), \ldots, F_K(x_m)$, no adversary can predict $F_K(x_{m+1})$ for any x_{m+1}.

Definition 3 (Homomorphic Encryption)**.** [2] An operation performed on a set of ciphertexts such that decrypting the result of the operation is the same as the result of some operation performed on the plaintexts.

3 Definitions

Searchable encryption allows a client to encrypt data in such a way that it can later generate search tokens to send as queries to a storage server. Given a search token, the server can search over the encrypted data and return the appropriate encrypted files.

The data can be viewed as a sequence of n files $\mathbf{f} = (f_1, \ldots, f_n)$, where file f_i is a sequence of words $(w_1, \ldots, w_m)$ from a universe W. Each file has a unique identifier $\mathsf{id}(f_i)$. The data is dynamic, so at any time a file may be added or removed. The files can be any type of data as long as there exists an efficient algorithm that maps each document to a file of keywords from W. Given a keyword w, $\mathbf{f}_w$ denotes the set of files in $\mathbf{f}$ that contain w. If $\mathbf{c} = (c_1, \ldots, c_n)$ is a set of encryptions of the files in $\mathbf{f}$, then $\mathbf{c}_w$ refers to the ciphertexts that are encryptions of the files in $\mathbf{f}_w$.

Definition 4 (Dynamic SSE)**.** [1] A dynamic index-based SSE scheme is a set of nine polynomial-time algorithms $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SrchToken}, \mathsf{AddToken}, \mathsf{DelToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Del}, \mathsf{Dec})$ such that:

1. $K \leftarrow \mathsf{Gen}(1^k)$: is a probabilistic algorithm that takes as input a security parameter k and outputs a secret key K.

2. $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \mathbf{f})$: is a probabilistic algorithm that takes as input a secret key K and a sequence of files $\mathbf{f}$. It outputs an encrypted index γ, and a sequence of ciphertexts $\mathbf{c}$.

3. $\tau_s \leftarrow \mathsf{SrchToken}(K, w)$: is a (possibly probabilistic) algorithm that takes as input a secret key K and a keyword w. It outputs a search token τ_s.

4. $(\tau_a, c_f) \leftarrow \mathsf{AddToken}(K, f)$: is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f. It outputs an add token τ_a and a ciphertext c_f.

5. $\tau_d \leftarrow \mathsf{DelToken}(K, f)$: is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f. It outputs a delete token τ_d.

6. $\mathbf{I}_w := \mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$: is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$ and a search token τ_s. It outputs a sequence of identifiers $\mathbf{I}_w \subseteq \mathbf{c}$.

7. $(\gamma', \mathbf{c}') := \mathsf{Add}(\gamma, \mathbf{c}, \tau_a, c)$: is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$, an add token

τ_a and a ciphertext c. It outputs a new encrypted index γ' and sequence of ciphertexts $\mathbf{c}'$.

8. $(\gamma', \mathbf{c}') := \mathsf{Del}(\gamma, \mathbf{c}, \tau_d)$: is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$, and a delete token τ_d. It outputs a new encrypted index γ' and sequence of ciphertexts $\mathbf{c}'$.

9. $f := \mathsf{Dec}(K, c)$: is a deterministic algorithm that takes as input a secret key K and a ciphertext c and outputs a file f.

A dynamic SSE scheme is correct if for all $k \in \mathbb{N}$, for all keys K generated by $\mathsf{Gen}(1^k)$, for all $\mathbf{f}$, for all $(\gamma, \mathbf{c})$ output by $\mathsf{Enc}(K, \mathbf{f})$, and for all sequences of add, delete or search operations on γ, search always returns the correct set of indices.

4 Construction

The dynamic SSE scheme is an extension of the non-dynamic SSE-1 construction, which is based on the inverted index data structure.

4.1 SSE-1 construction

The scheme makes use of a symmetric-key encryption scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, two pseudorandom functions F and G, an array $\mathtt{A}_s$ referred to as the *search array* and a dictionary $\mathtt{T}_s$ referred to as the *search table*.

To encrypt a collection of files $\mathbf{f}$, the scheme constructs for each keyword $w \in W$ a list $\mathtt{L}_w$ composed of $\#\mathbf{f}_w$ nodes $(\mathtt{N}_1, \ldots, \mathtt{N}_{\#\mathbf{f}_w})$ that are stored at random locations in the search array $\mathtt{A}_s$. The node $\mathtt{N}_i$ is defined as $\mathtt{N}_i = \langle \mathsf{id}, \mathsf{addr}_s(\mathtt{N}_{i+1}) \rangle$, where id is the unique file identifier of a file that contains w and $\mathsf{addr}_s(\mathtt{N})$ denotes the location of node $\mathtt{N}$ in $\mathtt{A}_s$. The unused cells in $\mathtt{A}_s$ can be padded with random strings of appropriate length.

For each keyword w, a pointer to the head of $\mathtt{L}_w$ is then inserted into the search table $\mathtt{T}_s$ under search key $F_{K_1}(w)$, where K_1 is the key to the PRF F. Each list is then encrypted using SKE under a key generated as $G_{K_2}(w)$, where K_2 is the key to the PRF G.

To search for a keyword w, the client sends the values $F_{K_1}(w)$ and $G_{K_2}(w)$. The server then uses $F_{K_1}(w)$ with $\mathtt{T}_s$ to recover the pointer to the head of $\mathtt{L}_w$. $G_{K_2}(w)$ is then used to decrypt the list and recover the identifiers of the files that contain w. If $\mathtt{T}$ supports $O(1)$ lookups (for example, using a hash table), the total search time for the server is linear in $\#\mathbf{f}_w$, which is optimal.

4.2 Dynamic SSE

The limitations of SSE-1 is that it is not explicitly dynamic.

The difficulty is that the addition, deletion or modification of a file requires the server to add, delete or modify nodes in the encrypted lists stored in $\mathtt{A}_s$. This is difficult for the server to do since:

1. upon deletion of a file f, it does not know where (in $\mathtt{A}$) the nodes corresponding to f are stored.

2. upon insertion or deletion of a node from a list, it cannot modify the pointer of the previous node since it is encrypted.

3. upon addition of a node, it does not know which locations in $\mathtt{A}_s$ are free.

At a high level, these limitations are addressed as follows [1]:

1. Add an extra (encrypted) data structure $\mathtt{A}_d$ called the *deletion array* that stores for each file f a list $\mathtt{L}_f$ of nodes that point to the nodes in $\mathtt{A}_s$ that should be deleted if file f is ever removed. So every node in the search array has a corresponding node in the deletion array and every node in the deletion array points to a node in the search array.

2. Encrypt the pointers stored in a node with a homomorphic encryption scheme. By providing the server with an encryption of an appropriate value, it can then modify the pointer without ever having to decrypt the node.

3. To keep track of which locations in $\mathtt{A}_s$ are free, a free list is maintained to add and manage extra space the server uses to add new nodes.

5 Conclusion

Searchable encryption has become an important problem in security and cryptography primarily due to the wide proliferation of sensitive data in open information and communication infrastructures like Amazon S3 and Microsoft Azure Storage. For example, legislation around the world stipulates that electronic health records (EHRs) should be encrypted, which immediately raises the question of how to search EHRs efficiently and securely [3].

A lot of developments have taken place in this line of study over the past decade and research continues in the uncharted areas that are not covered by the techniques currently in existence.

References

[1] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption", in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12, Raleigh, North Carolina, USA: ACM, 2012, pp. 965–976, ISBN: 9781450316514. DOI: 10.1145/2382196.2382298. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382298.

[2] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*, ser. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2014, ISBN: 9781466570276.

[3] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption", *ACM Comput. Surv.*, vol. 47, no. 2, 18:1–18:51, Aug. 2014, ISSN: 0360-0300. DOI: 10.1145/2636328. [Online]. Available: http://doi.acm.org/10.1145/2636328.

[4] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries", English, in *Advances in Cryptology – CRYPTO 2013*, ser. Lecture Notes in Computer Science, R. Canetti and J. Garay, Eds., vol. 8042, Springer Berlin Heidelberg, 2013, pp. 353–373, ISBN: 9783642400407. DOI: 10.1007/978-3-642-40041-4_20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40041-4_20.