DYNAMIC SEARCHABLE SYMMETRIC ENCRYPTION

Arpan Kapoor

October 20, 2015

National Institute of Technology, Calicut

OUTLINE

1. Introduction
2. Definitions
3. The construction
4. Example

INTRODUCTION

THE SCENARIO

- Rise of cloud storage.
- Outsource data storage.
- Security concerns regarding data privacy.
- Naïve solution: Encrypt data beforing uploading.

Searchable Symmetric Encryption

- Encrypt data such that it can still be searched.
- Generate search tokens to send as queries to server.
- Return appropriate encrypted files.
- Application: Cloud storage.

DEFINITIONS

DEFINITIONS

Symmetric Key Encryption

- Same key for encryption and decryption.

  $c = E_K(m)$ $\qquad$ $m = D_K(c)$

Homomorphic Encryption

- Permit computations on encrypted data.
- Obtaining $E_K(f(x))$ from $E_K(x)$.
- Server learns nothing about data it computed on.
- 2 types: Partially HE & Fully HE.

DEFINITIONS

Psuedorandom Function

- Polynomial time function whose output is indistinguishable from a random function.

$$F\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$$

- Given F, K, $x_1, \ldots, x_a$ and $F_K(x_1), \ldots, F_K(x_a)$, $F_K(x_{a+1})$ can't be predicted for any x_{a+1}.

THE CONSTRUCTION

REQUIREMENTS

- A private-key encryption scheme SKE.
- 2 pseudorandom functions F and G.
- A_s - search array.
- T_s - search table.

INPUT

- Collection of files $\mathbf{f} = (f_1, \ldots, f_n)$
- Each file has unique identifier $\mathrm{id}(f_i)$
- W = keyword space.
- Map each file to a list of keywords from W.
- $\mathbf{f}_w$ = set of files in $\mathbf{f}$ that contain w.

WORKING I

- $\forall w \in W$, construct $\mathsf{L}_w = (N_1, \ldots, N_{|f_w|})$
- Each node stored at random locations in A_s
- $\mathsf{N}_i = \langle \mathrm{id}, \mathrm{addr}_s(\mathsf{N}_{i+1}) \rangle$
- K_1 and K_2 are the keys to the PRF F and G.
- $\mathsf{T}_s[F_{K_1}(w)] =$ head of L_w
- Each list encrypted using SKE under key $G_{K_2}(w)$

WORKING II

- Send *search array* A_s, *search table* T_s and the collection of encrypted files $\mathsf{c} = (c_1, \ldots, c_n)$ to the server.
- To search for w, send $F_{K_1}(w)$ and $G_{K_2}(w)$.
- Use $F_{K_1}(w)$ to recover the pointer to head of L_w.
- Use $G_{K_2}(w)$ to decrypt the list.
- Running time - $O(|\mathsf{f_w}|)$
- Leakage of statistical information.

MAKING SSE DYNAMIC I

- Allow addition, deletion or modification of a file.
- Difficulties:
    1. Nodes corresponding to a file *f* are unknown.
    2. Can't modify pointer of the previous node as it is encrypted.
    3. Free locations in search array are unknown.

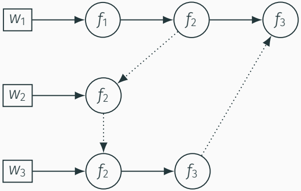
making sse dynamic II

1. Store list of pointers to nodes in A_s corresponding to a file f in the data structures A_d and T_d called the *deletion array* and *deletion table*.
2. Encrypt pointers with a homomorphic encryption scheme.
3. Keep track of free locations in A_s in a *free list*.

EXAMPLE

EXAMPLE

Index



Search Table T_s

$F_{K_1}(w_1) \rightarrow 4$

$F_{K_1}(w_2) \rightarrow 0$

$F_{K_1}(w_3) \rightarrow 5$

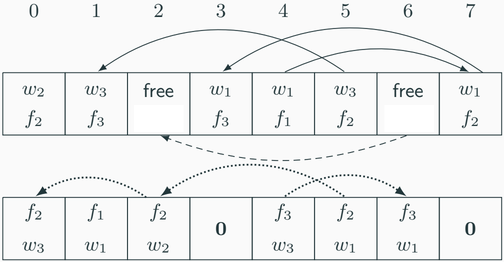
free → 6

Deletion Table T_d

$F_{K_1}(f_1) \rightarrow 1$

$F_{K_1}(f_2) \rightarrow 5$

$F_{K_1}(f_3) \rightarrow 4$



0	1	2	3	4	5	6	7
w_2 f_2	w_3 f_3	free	w_1 f_3	w_1 f_1	w_3 f_2	free	w_1 f_2

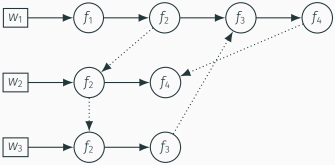
Search Array $\mathtt{A}_s$

f_2 w_3	f_1 w_1	f_2 w_2	**0**	f_3 w_3	f_2 w_1	f_3 w_1	**0**

Deletion Array $\mathtt{A}_d$

EXAMPLE – ADDING A FILE

Index



Search Table T_s

$F_{K_1}(w_1) \rightarrow 4$

$F_{K_1}(w_2) \rightarrow 0$

$F_{K_1}(w_3) \rightarrow 5$

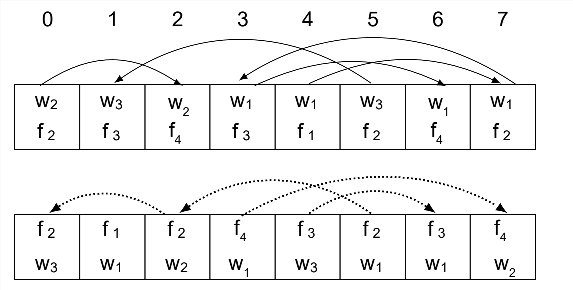
Deletion Table T_d

$F_{K_1}(f_1) \rightarrow 1$

$F_{K_1}(f_2) \rightarrow 5$

$F_{K_1}(f_3) \rightarrow 4$

$F_{K_1}(f_4) \rightarrow 3$

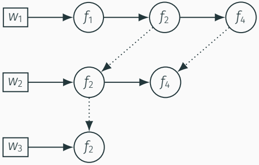


0	1	2	3	4	5	6	7
w_2 f_2	w_3 f_3	w_2 f_4	w_1 f_3	w_1 f_1	w_3 f_2	w_1 f_4	w_1 f_2

f_2 w_3	f_1 w_1	f_2 w_2	f_4 w_1	f_3 w_3	f_2 w_1	f_3 w_1	f_4 w_2

EXAMPLE - DELETING A FILE

Index



Search Table T_s

$F_{K_1}(w_1) \rightarrow 4$

$F_{K_1}(w_2) \rightarrow 0$

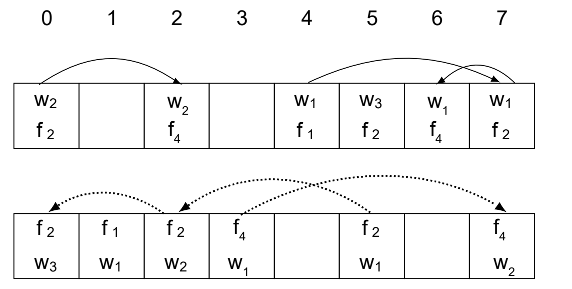
$F_{K_1}(w_3) \rightarrow 5$

Deletion Table T_d

$F_{K_1}(f_1) \rightarrow 1$

$F_{K_1}(f_2) \rightarrow 5$

$F_{K_1}(f_4) \rightarrow 3$



0	1	2	3	4	5	6	7
w_2 f_2		w_2 f_4		w_1 f_1	w_3 f_2	w_1 f_4	w_1 f_2
f_2 w_3	f_1 w_1	f_2 w_2	f_4 w_1		f_2 w_1		f_4 w_2

Questions?

REFERENCES I

[1] C. Bösch, P. Hartel, W. Jonker, and A. Peter.
A survey of provably secure searchable encryption.
ACM Computing Surveys (CSUR), 47(2):18, 2014.

[2] S. Kamara, C. Papamanthou, and T. Roeder.
Dynamic searchable symmetric encryption.
In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM, 2012.