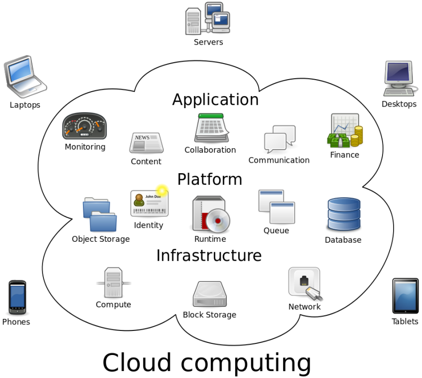
Searchable Encryption

Laltu Sardar
Indian Statistical Institute, Kolkata

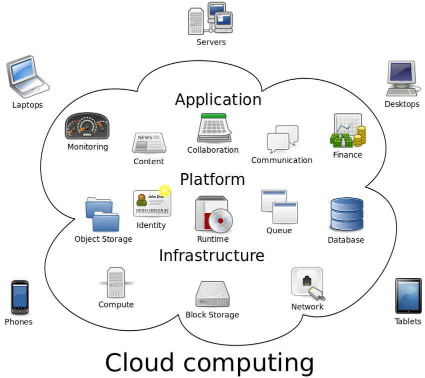
Summer Internship in Cryptology
R. C. Bose Centre for Cryptology and Security

May 22-23, 2018

Cloud Services



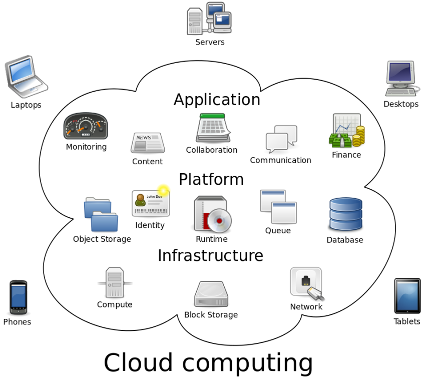
Cloud Services



Cloud Computing Services

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform
- IBM Cloud

Cloud Services



Cloud Computing Services

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform
- IBM Cloud

Cloud Storage Services

- Google Drive
- Dropbox
- Microsoft Onedrive

Can we trust Remote Storage Service Providers?

Cloud Computing and Storage

- Email service providers- Gmail, outlook.com, Yahoo! Mail etc.
- Stoarge service providers- Google Drive, Dropbox etc.
- Institutional Server

Can we trust Remote Storage Service Providers?

Cloud Computing and Storage

- Email service providers- Gmail, outlook.com, Yahoo! Mail etc.
- Stoarge service providers- Google Drive, Dropbox etc.
- Institutional Server

Survey Report [CER12]

- 53% attacks are insider

Can we trust Remote Storage Service Providers?

Cloud Computing and Storage

- Email service providers- Gmail, outlook.com, Yahoo! Mail etc.
- Stoarge service providers- Google Drive, Dropbox etc.
- Institutional Server

Survey Report [CER12]

- 53% attacks are insider
- 67% are sensitive or personal data.

Can we trust Remote Storage Service Providers?

Cloud Computing and Storage

- Email service providers- Gmail, outlook.com, Yahoo! Mail etc.
- Stoarge service providers- Google Drive, Dropbox etc.
- Institutional Server

Survey Report [CER12]

- 53% attacks are insider
- 67% are sensitive or personal data.

Can we trust Remote Storage Service Providers?

Cloud Computing and Storage

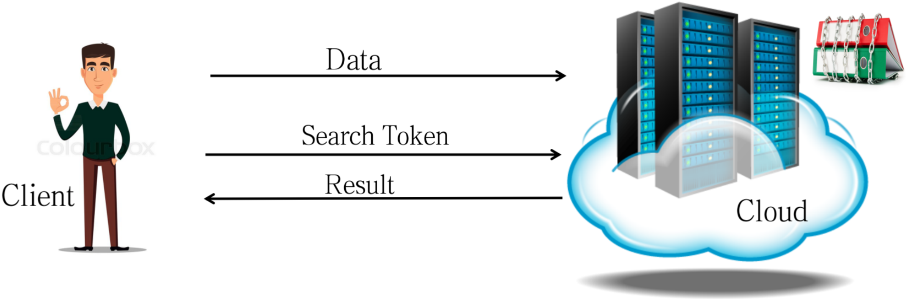
- Email service providers- Gmail, outlook.com, Yahoo! Mail etc.
- Stoarge service providers- Google Drive, Dropbox etc.
- Institutional Server

Survey Report [CER12]

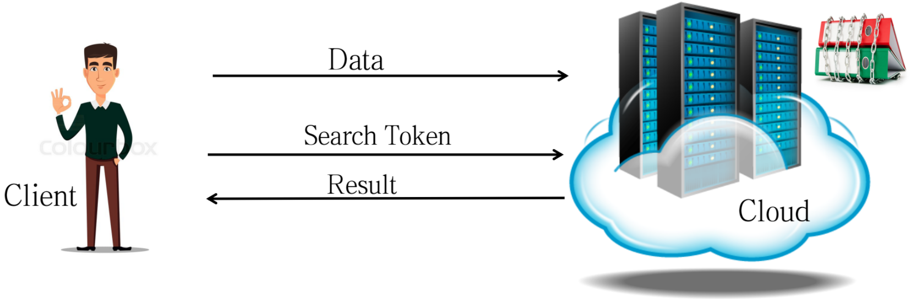
- 53% attacks are insider
- 67% are sensitive or personal data.

No!

Privacy-Preserving Computation

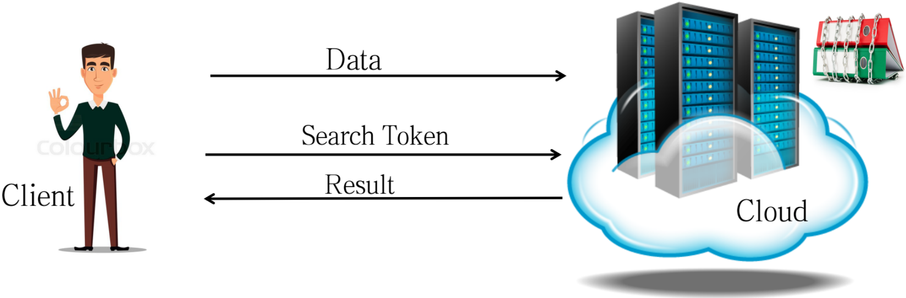


Privacy-Preserving Computation



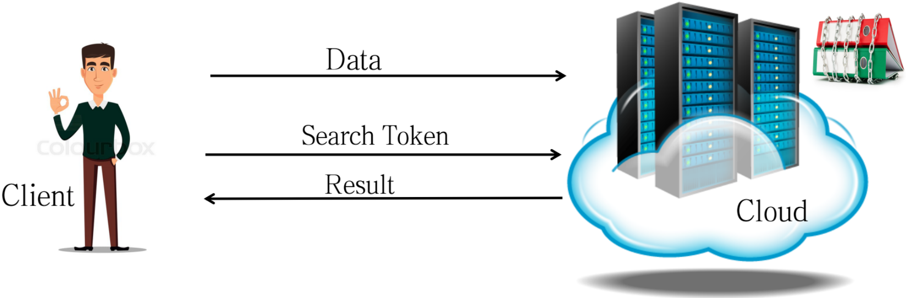
- Preserve search privacy $\rightarrow$ Private Information Retrieval

Privacy-Preserving Computation



- Preserve search privacy → Private Information Retrieval
- Data repository is huge → Privacy-preserving data mining

Privacy-Preserving Computation



- Preserve search privacy → Private Information Retrieval
- Data repository is huge → Privacy-preserving data mining
- Data are encrypted → Searchable Encryption

Trivial Solution

Trivial Solution

- Encrypt data

Trivial Solution

- Encrypt data
- Upload data to the cloud server

Trivial Solution

- Encrypt data
- Upload data to the cloud server

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search
- Re-encrypt and upload

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search
- Re-encrypt and upload

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search
- Re-encrypt and upload

Problems

- Huge Communication overhead for the client

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search
- Re-encrypt and upload

Problems

- Huge Communication overhead for the client
- Huge Computation at client side

Trivial Solution

- Encrypt data
- Upload data to the cloud server

To perform SEARCH

- Download all data
- Decrypt data
- Perform Search
- Re-encrypt and upload

Problems

- Huge Communication overhead for the client
- Huge Computation at client side
- Does NOT solve the purpose of using cloud

Searchable Encryption Goals

Searchable Encryption Goals

- Data should be

Searchable Encryption Goals

- Data should be
  - Outsourced

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data
  - Compute much

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data
  - Compute much
- Protection Needed

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data
  - Compute much
- Protection Needed
  - Source Data

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data
  - Compute much
- Protection Needed
  - Source Data
  - Search keywords

Searchable Encryption Goals

- Data should be
  - Outsourced
  - Encrypted
- Set Search Goals
  - What to search
  - Who can search
- Client should not
  - Wait long for search $\rightarrow$ Inefficiency
  - Download all data
  - Compute much
- Protection Needed
  - Source Data
  - Search keywords
  - Search Results

Adversarial Model

Adversarial Model

- Who is the adversary?

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider
- What is the power of adversary?

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider
- What is the power of adversary?
  - Infinite Power can not be assumed

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider
- What is the power of adversary?
  - Infinite Power can not be assumed
- What about channel?

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider
- What is the power of adversary?
  - Infinite Power can not be assumed
- What about channel?
  - Secure?

Adversarial Model

- Who is the adversary?
  - Cloud Service Provider
- What is the power of adversary?
  - Infinite Power can not be assumed
- What about channel?
  - Secure?
  - Can it be aborted?

Cryptographic Tools

Pseudo Random Function (PRF)

Definition

$F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$

- $\forall$ key $K \in \{0,1\}^k$, and $\forall x \in \{0,1\}^n$, $F(K, x)$ or $(F_K(x))$ is Efficiently computable
- F is Indistinguishable from a random Function

Pseudo Random Permutation (PRP)

Definition

$F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

- $\forall$ key $K \in \{0,1\}^k$, and $\forall x \in \{0,1\}^n$, $F(K,x)$ or $(F_K(x))$ is Efficiently computable
- Indistinguishable from a random Permutation

Pseudo Random Permutation (PRP)

Definition

$F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

- $\forall$ key $K \in \{0,1\}^k$, and $\forall x \in \{0,1\}^n$, $F(K,x)$ or $(F_K(x))$ is Efficiently computable
- Indistinguishable from a random Permutation

Examples

- AES
- DES
- 3DES

Pseudo Random Permutation (PRP)

Definition

$F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

- $\forall$ key $K \in \{0,1\}^k$, and $\forall x \in \{0,1\}^n$, $F(K,x)$ or $(F_K(x))$ is Efficiently computable
- Indistinguishable from a random Permutation

Examples

- AES
- DES
- 3DES

Note

- a PRP is a PRF

Pseudo Random Generator (PRG)

Definition

Deterministic random bit generator

Pseudo Random Generator (PRG)

Definition

Deterministic random bit generator

Properties

- Given a seed (start state), produces a sequences of random numbers/bits
- Efficient: Can produce many numbers/bits in a short time
- Deterministic: Same seeds generate same sequences of numbers/bits
- Periodic: Sequence will eventually repeat itself

Pseudo Random Generator (PRG)

Definition

Deterministic random bit generator

Properties

- Given a seed (start state), produces a sequences of random numbers/bits
- Efficient: Can produce many numbers/bits in a short time
- Deterministic: Same seeds generate same sequences of numbers/bits
- Periodic: Sequence will eventually repeat itself

Examples

- Stream cipher
- linear congruential generator
- Multiple-recursive generators

Hash Function

Definition

An algorithm/function that produces sequences of random numbers/bits.

Hash Function

Definition

An algorithm/function that produces sequences of random numbers/bits.

Features

- Publicly known key or no key
- Maps arbitrary-size bit-string to a fixed-size bit-string
- Deterministic: Same bit-string always results in the same hash
- Efficient: Quick to compute the hash value

Hash Function

Definition

An algorithm/function that produces sequences of random numbers/bits.

Features

- Publicly known key or no key
- Maps arbitrary-size bit-string to a fixed-size bit-string
- Deterministic: Same bit-string always results in the same hash
- Efficient: Quick to compute the hash value

Properties

- Pre-image Resistance, Second pre-image Resistance, Collision Resistance

Hash Function

Definition

An algorithm/function that produces sequences of random numbers/bits.

Features

- Publicly known key or no key
- Maps arbitrary-size bit-string to a fixed-size bit-string
- Deterministic: Same bit-string always results in the same hash
- Efficient: Quick to compute the hash value

Properties

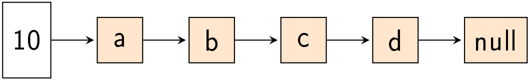
- Pre-image Resistance, Second pre-image Resistance, Collision Resistance

Examples

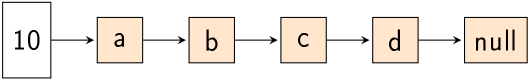
- SHA-0, SHA-1, SHA-2, SHA-3, MD5, SHA256 etc.

Data Type and Structures

Linked List



Linked List



Operations

- Create (Link List)
- Insert (a Node)
- Delete (a Node)

Dictionary

Definition

A collection of (key-value) pairs, such that each possible key appears at most once in the collection.

Dictionary

Definition

A collection of (key-value) pairs, such that each possible key appears at most once in the collection.

Operations

- Create (a Dictionary)
- Insert a (key-value) pair (a Node)
- Search whether a key exists

Dictionary

Definition

A collection of (key-value) pairs, such that each possible key appears at most once in the collection.

Operations

- Create (a Dictionary)
- Insert a (key-value) pair (a Node)
- Search whether a key exists

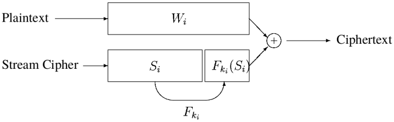
Properties

- Creation: In constant time
- Insertion: In logarithmic time
- Search: In constant time

Song et al. [SWP00] Scheme

Scheme I

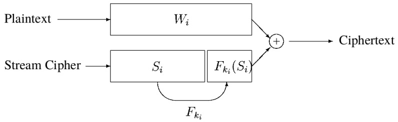
Encrypt a document $D = (W_1, W_2, \ldots, W_l)$ as follows



- s_i are generated using stream cipher
- k_i are fixed

Scheme I

Encrypt a document $D = (W_1, W_2, \ldots, W_l)$ as follows



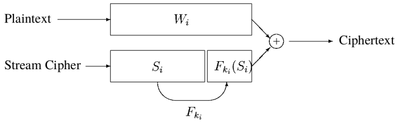
- s_i are generated using stream cipher
- k_i are fixed

For each i

- $T_i = S_i || F_{k_i}(S_i)$
- $C_i = W_i + T_i$

Scheme I

Encrypt a document $D = (W_1, W_2, \ldots, W_l)$ as follows



- s_i are generated using stream cipher
- k_i are fixed

For each i

- $T_i = S_i || F_{k_i}(S_i)$
- $C_i = W_i + T_i$

Finally uploads $Enc(D) = (C_1, C_2, \ldots, C_l)$

Scheme I

Search

To search for a word W

- Must reveal all the k_i

Scheme I

Search

To search for a word W

- Must reveal all the k_i

Problems

- Potentially revealing the entire document

Scheme I

Search

To search for a word W

- Must reveal all the k_i

Problems

- Potentially revealing the entire document

Solution

- Alice must know in advance which locations W may appear

Scheme II

- $k_i = f_{k'}(W_i)$, solves problem with keys
- $T_i = S_i || f_{k_i}(S_i)$, f is a PRF
- $C_i = W_i + T_i$

Scheme II

- $k_i = f_{k'}(W_i)$, solves problem with keys
- $T_i = S_i || f_{k_i}(S_i)$, f is a PRF
- $C_i = W_i + T_i$

Search

To search for a word W

- Only reveals all the $f_{k'}(W)$, Controlled searching.
- Check all positions
- If any decryption matches, returns the Doc

Scheme II

- $k_i = f_{k'}(W_i)$, solves problem with keys
- $T_i = S_i || f_{k_i}(S_i)$, f is a PRF
- $C_i = W_i + T_i$

Search

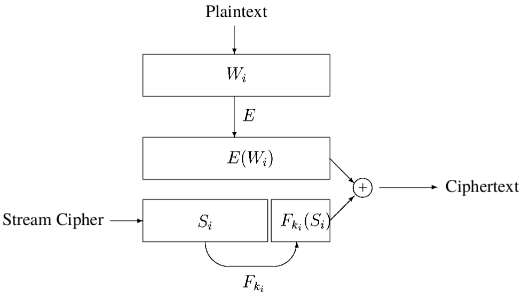
To search for a word W

- Only reveals all the $f_{k'}(W)$, Controlled searching.
- Check all positions
- If any decryption matches, returns the Doc

Problems

- W is revealed during search

Scheme III



- $Xi = E_{k''}(W_i)$, $E_{k''}$ is a deterministic encryption algorithm
- $k_i = f_{k'}(X_i)$
- $T_i = S_i || F_{k_i}(X_i)$,
- $C_i = X_i + T_i$

Scheme III

Search

To search for a word W

- Compute $X = E_{k''}(W)$
- Compute $k = f_{k'}(X)$
- Sends (X, k)

Scheme III

Search

To search for a word W

- Compute $X = E_{k''}(W)$
- Compute $k = f_{k'}(X)$
- Sends (X, k)

Advantages

- Searched keyword W is not revealed

Scheme III

Search

To search for a word W

- Compute $X = E_{k''}(W)$
- Compute $k = f_{k'}(X)$
- Sends (X, k)

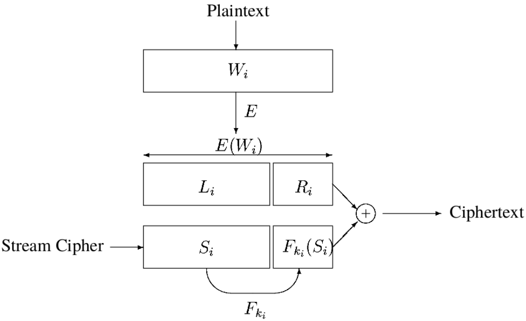
Advantages

- Searched keyword W is not revealed

Problems

- Owner can't recover the plaintext as $E_{k''}(W_i)$ is needed for decryption
- Applicable for Scheme II

Scheme IV- Final Scheme



- $X_i = E_{k''}(W_i)$, $E_{k''}$ is a deterministic encryption algorithm
- $X_i =< L_i||R_i >$
- $k_i = f_{k'}(L_i)$,
- $T_i = S_i||F_{k_i}(X_i)$,
- $C_i = X_i + T_i$

Scheme IV

Search

To search for a word W

- Sends (X, k) computed similarly

Scheme IV

Search

To search for a word W

- Sends (X, k) computed similarly

Decryption

To search for a word W

- Generate S_i
- Recover L_i XORing S_i with C_i
- Recover $k_i = f_{k'}(L_i)$,
- Recover X_i
- Get W_i Decrypting X_i

Major Disadvantage

- Every keywords of every files have to be decrypted

Eu-Jin Goh [Goh03] Scheme

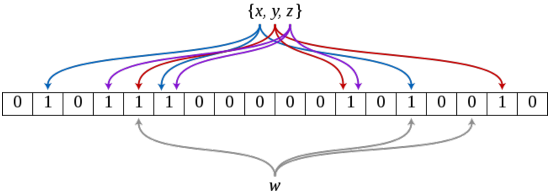
Main Contribution

- Defined Secure index
- Formulated Security Model for indexes

Background

Bloom Filter

- A set $S = s_1, \ldots, s_n$, represented by an array of m bits.
- All array bits are initially set to 0
- The filter uses r independent hash functions $h_1, \ldots, h_r$,
- To determine if an element a belongs to the set S, checks whether all $h_i(a)$ are 1 or not



Scheme Overview

Key Generation

Given Security parameter s

- $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^s$, pseudo-random function
- $k_1, \ldots, k_r \leftarrow \{0,1\}^s$, keys for hash functions
- $K_{priv} \leftarrow (k_1, \ldots, k_r)$

Scheme Overview

Key Generation

Given Security parameter s

- $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^s$, pseudo-random function
- $k_1, \ldots, k_r \leftarrow \{0,1\}^s$, keys for hash functions
- $K_{priv} \leftarrow (k_1, \ldots, k_r)$

Build Index

Given K_{priv} and a document $D = (w_0, \ldots, w_t)$ with identifier D_{id}

- For each unique word wi for $i \in [0, t]$, -
  - Compute trapdoor: $(x_1 = f(w_i, k_1), \ldots, x_r = f(w_i, k_r)) \in \{0,1\}^{sr}$,
  - Compute codeword for w_i in $D_{id} : (y_1 = f(D_{id}, x_1), \ldots, y_r = f(D_{id}, x_r)) \in \{0,1\}^{sr}$
  - Insert $y_1, \ldots, y_r$ into D_{id} 's Bloom filter BF.
- Output $\mathcal{I}_{D_{id}} = (D_{id}, BF)$ as the index for D_{id} .

Scheme Overview

Trapdoor Generation

Given a keyword w

- $T_w = (f(w, k_1), \ldots, f(w, k_r))$

Search

- $(x_1, \ldots, x_r) \leftarrow Tw$
- The index $\mathcal{I}_{D_{id}} = (D_{id}, BF)$ for document D_{id}
- For w Compute
  $D_{id} : (y1 = f(D_{id}, x1), \ldots, yr = f(D_{id}, xr)) \in \{0, 1\}^{sr}$.
- Test if BF contains 1's in all r locations denoted by $y1, \ldots, yr$

Chang and Mitzenmacher [CM05] Scheme

Scheme Description

Privacy Preserving Keyword Searches on Remote Encrypted Data [CM05]

Scheme overview

- Skip Now

Major Issues of Earlier Schemes

- Greater Search Complexity: Linear in number of documents

Major Issues of Earlier Schemes

- Greater Search Complexity: Linear in number of documents
- Leaks Access Pattern: Memory addresses of documents that contain the searched keywords

Major Issues of Earlier Schemes

- Greater Search Complexity: Linear in number of documents
- Leaks Access Pattern: Memory addresses of documents that contain the searched keywords
- Leaks Search Pattern: Whether two queries were for the same keyword or not

Major Issues of Earlier Schemes

- Greater Search Complexity: Linear in number of documents
- Leaks Access Pattern: Memory addresses of documents that contain the searched keywords
- Leaks Search Pattern: Whether two queries were for the same keyword or not
- Leakages were not defined

Is there any Solution?

Is there any Solution?

Yes, there is.

Is there any Solution?

Yes, there is.

SSE can be achieved using oblivious RAMs (O-RAM)

- Functionality: can simulate any data structure in a hidden way, and can support conjunctive queries, B-trees etc...
- Privacy: hides everything, even the access pattern
- Efficiency: logarithmic number of rounds per each read/write

Is there any Solution?

Yes, there is.

SSE can be achieved using oblivious RAMs (O-RAM)

- Functionality: can simulate any data structure in a hidden way, and can support conjunctive queries, B-trees etc...
- Privacy: hides everything, even the access pattern
- Efficiency: logarithmic number of rounds per each read/write

Question?

Can we search over encrypted data in single/constant rounds?

- with privacy,
- with efficiency

Curtmola et al. [CGKO06] Scheme

Background

Inverted Index

- Index data structure
- Maps content to its locations in a database

Background

Inverted Index

- Index data structure
- Maps content to its locations in a database

$\mathcal{D} \leftarrow \{D_1, D_2, D_3, D_4\}$

Background

Inverted Index

- Index data structure
- Maps content to its locations in a database

$\mathcal{D} \leftarrow \{D_1, D_2, D_3, D_4\}$

D_1 ← {cryptography, search, symmetric, encryption}

D_2 ← {public, encryption, add}

D_3 ← { add, public, cryptography}

D_4 ← {search, symmetric, encryption, decryption, add}

Background

Inverted Index

- Index data structure
- Maps content to its locations in a database

$\mathcal{D} \leftarrow \{D_1, D_2, D_3, D_4\}$

$D_1 \leftarrow$ {cryptography, search, symmetric, encryption}

$D_2 \leftarrow$ {public, encryption, add}

$D_3 \leftarrow$ { add, public, cryptography}

$D_4 \leftarrow$ {search, symmetric, encryption, decryption, add}

Content	Locations
encryption	$[D_1, D_2, D_4]$
symmetric	$[D_1, D_4]$
decryption	$[D_4]$
cryptography	$[D_1, D_3]$
add	$[D_3, D_4]$
search	$[D_1, D_4]$
public	$[D_2, D_3]$

Table: Inverted Index corr. to $\mathcal{D}$

Notations

- $D = (D_1, \ldots, D_n)$- Document Collection
- D_i- Document
- T- A Table
- A- An Array
- L_i- The Link list corr. to D_i
- F- A PRP
- G- A PRG
- H- A keyed Hash function

Definition

A tuple of PPT algorithms as follows

Definition

A tuple of PPT algorithms as follows

- **Key Generation:**
  - Input: A security parameter k
  - Output: A secret key K

Definition

A tuple of PPT algorithms as follows

- **Key Generation:**
  - Input: A security parameter k
  - Output: A secret key K
- **Encryption:** a document collection D
  - Input: A secret key K and a document collection $D = (D_1, \ldots, D_n)$
  - Output: A secure index I and a sequence of ciphertexts $c = (c_1, \ldots, c_n)$

Definition

A tuple of PPT algorithms as follows

- **Key Generation:**
  - Input: A security parameter k
  - Output: A secret key K
- **Encryption:** a document collection D
  - Input: A secret key K and a document collection $D = (D_1, \ldots, D_n)$
  - Output: A secure index I and a sequence of ciphertexts $c = (c_1, \ldots, c_n)$
- **Trapdoor Gen:** for a keyword w
  - Input: A secret key K and a keyword w
  - Output: A trapdoor $t \leftarrow Trpdr_K(w)$

Definition

A tuple of PPT algorithms as follows

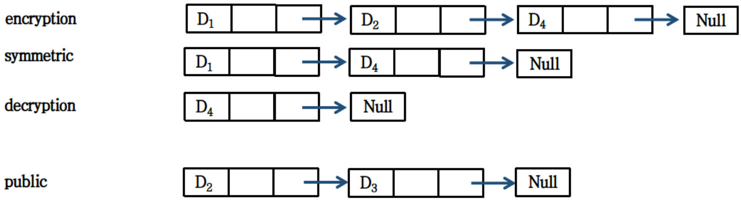
- **Key Generation:**
  - Input: A security parameter k
  - Output: A secret key K
- **Encryption:** a document collection D
  - Input: A secret key K and a document collection $D = (D_1, \ldots, D_n)$
  - Output: A secure index I and a sequence of ciphertexts $c = (c_1, \ldots, c_n)$
- **Trapdoor Gen:** for a keyword w
  - Input: A secret key K and a keyword w
  - Output: A trapdoor $t \leftarrow Trpdr_K(w)$
- **Search:** for the documents in D that contain a keyword w
  - Input: An encrypted index I for a data collection D and a trapdoor t
  - Output: a set X of document identifiers

Definition

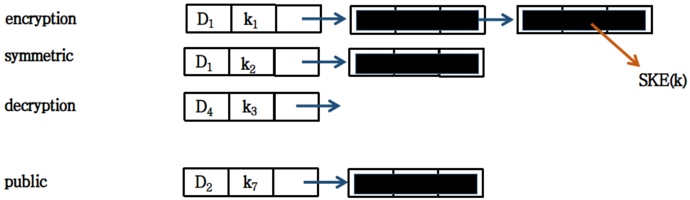
A tuple of PPT algorithms as follows

- **Key Generation:**
  - Input: A security parameter k
  - Output: A secret key K
- **Encryption:** a document collection D
  - Input: A secret key K and a document collection $D = (D_1, \ldots, D_n)$
  - Output: A secure index I and a sequence of ciphertexts $c = (c_1, \ldots, c_n)$
- **Trapdoor Gen:** for a keyword w
  - Input: A secret key K and a keyword w
  - Output: A trapdoor $t \leftarrow Trpdr_K(w)$
- **Search:** for the documents in D that contain a keyword w
  - Input: An encrypted index I for a data collection D and a trapdoor t
  - Output: a set X of document identifiers
- **Decryption:** for an encrypted document D_i
  - Input: A secret key K and a ciphertext ci
  - Output: A document D_i

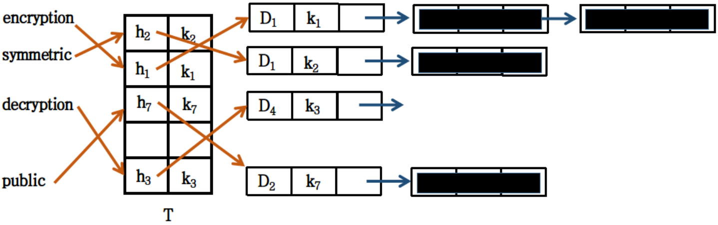
Build Inverted Index



Encrypt List Entries



Make Search Table



Encrypt 1st Node

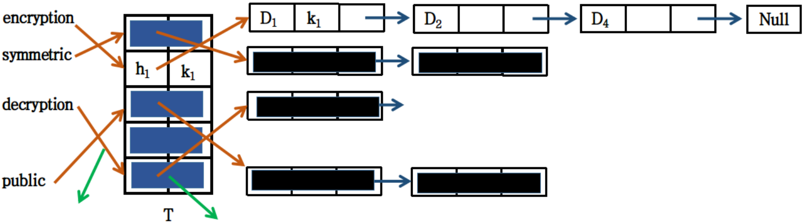
Encrypt Table

Search: Generate Trapdoor

Trapdoor = $<\mathrm{PRP}(k'', w_2),\ \mathrm{PRG}(k', w_2)>$

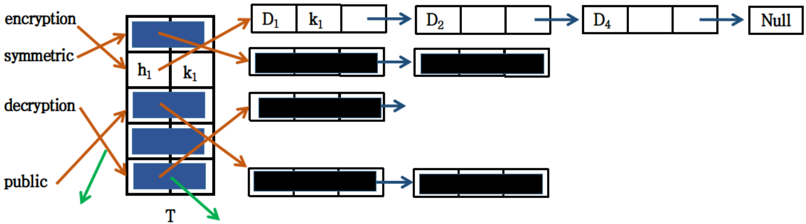
Search: Decrypt List

Trapdoor = < PRP (k'',w_2), PRG (k',w_2) >



Search: Return Result

Trapdoor = < PRP (k'', w_2), PRG (k', w_2) >



Returns = { D_1, D_2, D_4 }

Advantages

Advantages

Search Complexities

- # decryption $\leftarrow$ O(Search Result)

Advantages

Search Complexities

- # decryption $\leftarrow$ O(Search Result)

Communication Complexities

- # rounds $\leftarrow$ constant

Advantages

Search Complexities

- # decryption $\leftarrow$ O(Search Result)

Communication Complexities

- # rounds $\leftarrow$ constant

Privacy

- Yes

Issues

Previous Schemes are STATIC

- One encrypted index is generated, Can't be changed
- Does not support Addition of document
- Does not support Deletion of document
- Does not support Addition of word in a document
- Does not support Deletion of word from a document

Issues

Previous Schemes are STATIC

- One encrypted index is generated, Can't be changed
- Does not support Addition of document
- Does not support Deletion of document
- Does not support Addition of word in a document
- Does not support Deletion of word from a document

In Practical

- Database should support word of file updates

Issues

Previous Schemes are STATIC

- One encrypted index is generated, Can't be changed
- Does not support Addition of document
- Does not support Deletion of document
- Does not support Addition of word in a document
- Does not support Deletion of word from a document

In Practical

- Database should support word of file updates

Dynamic SSE

- SSE that Supports updates

Definition of Dynamic SSE

Definition of Dynamic SSE

Skip Now :)

Few Remarkable works on Dynamic SSE

Kamara et al. [KPR12] Scheme

Scheme Overview

- 1st ever work on Dynamic SSE

Kamara et al. [KPR12] Scheme

Scheme Overview

- 1st ever work on Dynamic SSE
- Improvement over Curtmola et al. [CGKO06].

Kamara et al. [KPR12] Scheme

Scheme Overview

- 1st ever work on Dynamic SSE
- Improvement over Curtmola et al. [CGKO06].
- Inverted Index Based

Kamara et al. [KPR12] Scheme

Scheme Overview

- 1st ever work on Dynamic SSE
- Improvement over Curtmola et al. [CGKO06].
- Inverted Index Based
- Instead of one, used TWO indexes-

Kamara et al. [KPR12] Scheme

Scheme Overview

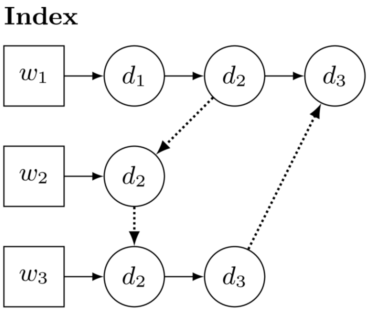
- 1st ever work on Dynamic SSE
- Improvement over Curtmola et al. [CGKO06].
- Inverted Index Based
- Instead of one, used TWO indexes-
  - Search index - Inverted index

Kamara et al. [KPR12] Scheme

Scheme Overview

- 1st ever work on Dynamic SSE
- Improvement over Curtmola et al. [CGKO06].
- Inverted Index Based
- Instead of one, used TWO indexes-
  - Search index - Inverted index
  - Deletion Index - General index

Example



Main Index M

$f_{k_c}(w_1) \longrightarrow (4 \,||\, 1) \oplus f_{k_b}(w_1)$

$f_{k_c}(w_2) \longrightarrow (0 \,||\, 2) \oplus f_{k_b}(w_2)$

$f_{k_c}(w_3) \longrightarrow (5 \,||\, 0) \oplus f_{k_b}(w_3)$

$f_{k_c}(\text{free}) \longrightarrow 6 \oplus f_{k_b}(\text{free})$

Deletion Index I

$f_{k_c}(d_1) \longrightarrow 1 \oplus f_{k_b}(d_1)$

$f_{k_c}(d_2) \longrightarrow 5 \oplus f_{k_b}(d_2)$

$f_{k_c}(d_3) \longrightarrow 4 \oplus f_{k_b}(d_3)$

Example

Main Index M

$f_{k_c}(w_1) \longrightarrow (4 \mid\mid 1) \oplus f_{k_b}(w_1)$

$f_{k_c}(w_2) \longrightarrow (0 \mid\mid 2) \oplus f_{k_b}(w_2)$

$f_{k_c}(w_3) \longrightarrow (5 \mid\mid 0) \oplus f_{k_b}(w_3)$

$f_{k_c}(\text{free}) \longrightarrow 6 \oplus f_{k_b}(\text{free})$

Deletion Index I

$f_{k_c}(d_1) \longrightarrow 1 \oplus f_{k_b}(d_1)$

$f_{k_c}(d_2) \longrightarrow 5 \oplus f_{k_b}(d_2)$

$f_{k_c}(d_3) \longrightarrow 4 \oplus f_{k_b}(d_3)$

Main List L

0	1	2	3	4	5	6	7
w_2	w_3	free	w_1	w_1	w_3	free	w_1
d_2	d_3	D_7	d_3	d_1	d_1	D_3	d_2

Deletion List D

d_2	d_1	d_2	r	d_3	d_2	d_3	r'
w_3	w_1	w_2		w_3	w_1	w_1	

Issues

- Complex Scheme- Difficult To Implement
- Nodes were at Random location- Sequential operation

Kamara et al. [KP13] Scheme

Scheme Overview

- Search or update can be done in Parallel

Kamara et al. [KP13] Scheme

Scheme Overview

- Search or update can be done in Parallel
- Extra: do not leak information about the keywords contained in a newly added or deleted document

Kamara et al. [KP13] Scheme

Scheme Overview

- Search or update can be done in Parallel
- Extra: do not leak information about the keywords contained in a newly added or deleted document
- Used *tree-based multi-map data structure*- keyword red-black (KRB) tree

Background

(k, m) Hash Table

- A table of $(key, value)$ pairs
- $key \in \{0,1\}^k$
- at most m entries

Background

(k, m) Hash Table

- A table of $(key, value)$ pairs
- $key \in \{0, 1\}^k$
- at most m entries

KRB Tree

- A table of $(key, value)$ pairs

Background

(k, m) Hash Table

- A table of $(key, value)$ pairs
- $key \in \{0, 1\}^k$
- at most m entries

KRB Tree

- A table of $(key, value)$ pairs
- $key \in \{0, 1\}^k$

Background

(k, m) Hash Table

- A table of $(key, value)$ pairs
- $key \in \{0,1\}^k$
- at most m entries

KRB Tree

- A table of $(key, value)$ pairs
- $key \in \{0,1\}^k$
- at most m entries

KRB-Based Dynamic SSE

Scheme Overview

- On White-Board

Remarkable Works Till Today

Stefanov et al. [SPS14] Scheme

- Practical Dynamic Searchable Encryption with Small Leakage

Remarkable Works Till Today

Stefanov et al. [SPS14] Scheme

- Practical Dynamic Searchable Encryption with Small Leakage

Naveed et al. [NPG14] Scheme

- Dynamic Searchable Encryption via Blind Storage

Remarkable Works Till Today

Stefanov et al. [SPS14] Scheme

- Practical Dynamic Searchable Encryption with Small Leakage

Naveed et al. [NPG14] Scheme

- Dynamic Searchable Encryption via Blind Storage

Cash et al. [CJJ+14] Scheme

- Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation

Remarkable Works Till Today

Stefanov et al. [SPS14] Scheme

- Practical Dynamic Searchable Encryption with Small Leakage

Naveed et al. [NPG14] Scheme

- Dynamic Searchable Encryption via Blind Storage

Cash et al. [CJJ+14] Scheme

- Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation

Hahn and Kerschbaum. [HK14] Scheme

- Searchable Encryption with Secure and Efficient Updates

Remarkable Works Till Today

Stefanov et al. [SPS14] Scheme

- Practical Dynamic Searchable Encryption with Small Leakage

Naveed et al. [NPG14] Scheme

- Dynamic Searchable Encryption via Blind Storage

Cash et al. [CJJ+14] Scheme

- Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation

Hahn and Kerschbaum. [HK14] Scheme

- Searchable Encryption with Secure and Efficient Updates

Kamara et al. [KM17] Scheme

- Boolean SSE with Worst-Case Sub-linear Complexity

Attacks on Searchable Encryption Scheme

Islam et al. [IKK12] Attack

Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation

- 1st to investigate- Access Pattern disclosure on Searchable Encryption
- Attack the existing Schemes with few assumptions
- Provide solution to the problem

Attack Overview

Assumptions

- Attacker observes $Q =< Q_1, \ldots, Q_l >$ and their responses $< R_{Q_1}, \ldots, R_{Q_l} >$
- Attacker knows the underlying keywords for a set of k queries: K_Q
- Attacker has access to a $(m \times m)$ matrix M s.t. $M_{i,j} = Pr[(k_i \in d) \wedge (k_j \in d)]$, here d is sampled uniformly from D.

Attack Overview

Assumptions

- Attacker observes $Q =< Q_1, \ldots, Q_l >$ and their responses $< R_{Q_1}, \ldots, R_{Q_l} >$
- Attacker knows the underlying keywords for a set of k queries: K_Q
- Attacker has access to a $(m \times m)$ matrix M s.t. $M_{i,j} = Pr[(k_i \in d) \wedge (k_j \in d)]$, here d is sampled uniformly from D.

Attack Process

- From knowledge of d (sampled uniformly from D)
  - From publicly known large dataset, ex. WikiLeaks
  - Inside Attacker may have access to the sizable subset of the dataset
- From publicly known large dataset
  - Attacker can calculate frequency of keywords i.e., $Pr[(k_i \in d)]$
  - Attacker can calculate $M_{i,j} = Pr[(k_i \in d) \wedge (k_j \in d)]$
- They later considered $Pr[(k_{i_1} \in d) \wedge \ldots \wedge (k_{i_r} \in d)]$

Attack Overview

Attack Result

- Knowing only subset of D significant # queries can be guessed

Attack Overview

Attack Result

- Knowing only subset of D significant # queries can be guessed

$(\alpha, 0)$- secure index

- For each keyword, there are at least $\alpha - 1$ keywords which appear exactly in the same set of documents.
- It's hard for an attacker to distinguish a keyword given the query response of that particular keyword.

Attack Overview

Attack Result

- Knowing only subset of D significant # queries can be guessed

$(\alpha, 0)$- secure index

- For each keyword, there are at least $\alpha - 1$ keywords which appear exactly in the same set of documents.
- It's hard for an attacker to distinguish a keyword given the query response of that particular keyword.

Proposed a noise addition technique

- Inject false positive docs so that index remains $(\alpha, 0)$- secure
- User can later decrypt the document and reject if the keywords is not present

Inference Attacks on Property-Preserving EDB (Naveed et al. [NKW15])

- SKIP NOW

Inference Attacks on Property-Preserving EDB (Naveed et al. [NKW15])

- SKIP NOW

Property Preserving Encryption (PPE)

Leaks a certain property of the plaintext

- Order Preserving Encryption (PPE): Reveals the order of the messages (i.e., the order property).
- Deterministic Encryption (PPE): Reveals whether they are equal or not (i.e., the equality property).

Inference Attacks on Property-Preserving EDB (Naveed et al. [NKW15])

- SKIP NOW

Property Preserving Encryption (PPE)

Leaks a certain property of the plaintext

- Order Preserving Encryption (PPE): Reveals the order of the messages (i.e., the order property).
- Deterministic Encryption (PPE): Reveals whether they are equal or not (i.e., the equality property).

Where is it Applicable?

- Searchable encryption that supports Range queries
- PPE Based database CryptDB and its variants

Inference Attacks on Property-Preserving EDB (Naveed et al. [NKW15])

Attack Techniques

- Frequency analysis: DTE
- l_p-optimization: DTE
- Sorting attack: OPE
- Cumulative attack: OPE

Leakage-Abuse Attacks (Cash et al. [CGPR15])

Leakage-Abuse Attacks Against Searchable Encryption

- Query recovery attack: Determining the plaintext of queries that have been issued by the client
- Partial plaintext recovery attack: Reconstruct indexed documents as much as possible

Query recovery attack

Attack Model

- Count Attack:
- Server knows $count(w) \forall w \in W$
- Fully document knowledge

Query recovery attack

Attack Model

- Count Attack:
- Server knows $count(w) \forall w \in W$
- Fully document knowledge

Solution?

- Padding
- Adding Garbage doc id in the index

Query recovery attack from Partially known Docs

- See Islam et al. [IKK12]

Partial plaintext recovery attack

- Known-Document Attack
- Active Attacks

Partial plaintext recovery attack

- Known-Document Attack
- Active Attacks

Known-Document Attack

- Order of Hashes Known
- Order of Hashes Unknown

Partial plaintext recovery attack

- Known-Document Attack
- Active Attacks

Known-Document Attack

- Order of Hashes Known
- Order of Hashes Unknown

Active Attacks

- Hash order known for chosen document
- Hash order unknown for chosen documents

File Injection Attack (Zhang et al. [ZKP16])

File Injection Attack

All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption

Attack Overview

- **Focused on Query Recover Attack**

File Injection Attack

All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption

Attack Overview

- **Focused on Query Recover Attack**
- Applicable for Dynamic SSEs

File Injection Attack

All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption

Attack Overview

- **Focused on Query Recover Attack**
- Applicable for Dynamic SSEs
- Attack does not require the server to have any knowledge about the client's files

File Injection Attack

All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption

Attack Overview

- **Focused on Query Recover Attack**
- Applicable for Dynamic SSEs
- Attack does not require the server to have any knowledge about the client's files
- Recovers all the keywords being searched by the client with 100% accuracy

Binary search Attack

Process

- Insert $\#\log K$ files.
- ith file contains exactly those keywords whose ith most-significant bit is 1
- If a keyword w is searched and returns then it matches returned files with its injected ones.

Binary search Attack

Process

- Insert #$\log K$ files.
- ith file contains exactly those keywords whose ith most-significant bit is 1
- If a keyword w is searched and returns then it matches returned files with its injected ones.

Reduction in # files

- If targeted keyword set $K' \subset K$

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil . (\lceil \log 2T \rceil + 1)$

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil . (\lceil \log 2T \rceil + 1)$

Solution

- The Rest of the paper **Assume partial knowledge** of documents

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil.(\lceil \log 2T \rceil + 1)$

Solution

- The Rest of the paper **Assume partial knowledge** of documents
- Attack adaptive and statistical

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil . (\lceil \log 2T \rceil + 1)$

Solution

- The Rest of the paper **Assume partial knowledge** of documents
- Attack adaptive and statistical
- Applicable for the scheme which are not **Forward Private**.

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil . (\lceil \log 2T \rceil + 1)$

Solution

- The Rest of the paper **Assume partial knowledge** of documents
- Attack adaptive and statistical
- Applicable for the scheme which are not **Forward Private**.
- Forward Privacy: The server cannot tell if a newly inserted file matches previous search queries

Binary search Attack

Hierarchical File Injection

- Considers K' instead of K
- Apply Binary search on K
- # files to be injected $\approx \lceil |K|/2T \rceil . (\lceil \log 2T \rceil + 1)$

Solution

- The Rest of the paper **Assume partial knowledge** of documents
- Attack adaptive and statistical
- Applicable for the scheme which are not **Forward Private**.
- Forward Privacy: The server cannot tell if a newly inserted file matches previous search queries
- Examples: Stefanov et al. [SPS14], Raphael Bost [Bos16]

Forward Private DSSE

Few Examples of Forward-Secure DSSE

- Stefanov et al. [SPS14]
- $\Sigma o\phi o\varsigma$ (Sophos) Bost [Bos16] in 2016
- Bost et al. [BMO17] in 2017
- Rizomiliotis and Gritzalis [RG15], ORAM Based
- Lai and Chow [LC17] based on Bipartite Graph

We have focused on $\Sigma o\phi o\varsigma$

$\Sigma o\phi o\varsigma$

- Devided into two parts
  - $\Sigma o\phi o\varsigma$-B
  - $\Sigma o\phi o\varsigma$

$\Sigma o\phi o\varsigma$-B -> Idea

$$\begin{tikzcd}[column sep=large, row sep=large]
ST_0(w) \arrow[d, "H_{K_w}"'] \arrow[r, bend left, "\pi_{\mathsf{SK}}^{-1}"] & ST_1(w) \arrow[l, "\pi_{\mathsf{PK}}"] \arrow[d, "H_{K_w}"'] & \cdots & ST_c(w) \arrow[ll, dashed, "\pi_{\mathsf{PK}}^{(c-1)}"] \arrow[d, "H_{K_w}"'] \arrow[r, bend left, "\pi_{\mathsf{SK}}^{-1}"] & ST_{c+1}(w) \arrow[l, "\pi_{\mathsf{PK}}"] \arrow[d, "H_{K_w}"'] \\
UT_0(w) & UT_1(w) & \cdots & UT_c(w) & UT_{c+1}(w)
\end{tikzcd}$$

$\Sigma o\phi o\varsigma$-B -> Setup

<u>Setup()</u>

1: $K_S \xleftarrow{\$} \{0,1\}^\lambda$
2: $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3: $\mathbf{W}, \mathbf{T} \leftarrow \text{empty map}$
4: $\textbf{return } ((\mathbf{T}, \mathsf{PK}), (\mathbf{K_S}, \mathsf{SK}), \mathbf{W})$

$\Sigma o \phi o \varsigma$-$B$ -> Search

$$Search(w, \sigma, EDB)$$

Client:

$K_w \leftarrow F_{K_S}(w)$
$(ST_c, c) \leftarrow \mathbf{W}[w]$
if $(ST_c, c) = \perp$
  return $\emptyset$
Send (K_w, ST_c, c) to the server.

Server:

for $i = c$ **to** 0 **do**
  $UT_i \leftarrow H_1(K_w, ST_i)$
  $e \leftarrow \mathbf{T}[UT_i]$
  $\mathsf{ind} \leftarrow e \oplus H_2(K_w, ST_i)$
  Output each ind
  $ST_{i-1} \leftarrow \pi_{\mathsf{PK}}(ST_i)$
end for

Σοφος-*B* -> Update

Update(add, w, ind, σ; EDB)

Client:

$K_w \leftarrow F(K_S, w)$
$(ST_c, c) \leftarrow \mathbf{W}[w]$
if $(ST_c, c) = \perp$ **then**
  $ST_0 \xleftarrow{\$} \mathcal{M}$, $c \leftarrow -1$
else
  $ST_{c+1} \leftarrow \pi^{-1}_{\mathsf{SK}}(ST_c)$
end if
$\mathbf{W}[w] \leftarrow (ST_{c+1}, c+1)$
$UT_{c+1} \leftarrow H_1(K_w, ST_{c+1})$
$e \leftarrow \mathsf{ind} \oplus H_2(K_w, ST_{c+1})$
Send (UT_{c+1}, e) to the server.

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs

Problem with $\Sigma o \phi o \varsigma$-$B$

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Client-Storage Reduction

- ST_0 can be generated using PRF

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Client-Storage Reduction

- ST_0 can be generated using PRF
- From ST_0 compute ST_c

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Client-Storage Reduction

- ST_0 can be generated using PRF
- From ST_0 compute ST_c
- Still, keyword id and counter have to be stored

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Client-Storage Reduction

- ST_0 can be generated using PRF
- From ST_0 compute ST_c
- Still, keyword id and counter have to be stored
- Results large Computation

Problem with $\Sigma o\phi o\varsigma$-B

- No Deletion and Huge Client Storage

Enabling Deletion

- Adding Extra Database for Deletion
- Searching eliminate the deleted docs
- Problem: Database size grows with time even after deletion
- Any Solution?

Client-Storage Reduction

- ST_0 can be generated using PRF
- From ST_0 compute ST_c
- Still, keyword id and counter have to be stored
- Results large Computation
- Any solution?

Security of Searchable Encryption Schemes

Security Definition of SSE

First Formal Definition by Curtmola et al. [CGKO06].

Approaches

- Indistinguishability
- Semantic Security

Security Definition of SSE

First Formal Definition by Curtmola et al. [CGKO06].

Approaches

- Indistinguishability
- Semantic Security

Adversary Types

- Non-Adaptive: Queries don't depend on previous results
- Adaptive: Queries depend on previous results

Security Definition of SSE

Notations

- $\mathbf{D} \leftarrow$ Collection of documents
- $\mathbf{w} \leftarrow \{w_1, \ldots, w_q\}$, set of keywords for queries
- History $H \leftarrow (\mathbf{D}, \mathbf{w})$

Security Definition of SSE

Notations

- $\mathbf{D} \leftarrow$ Collection of documents
- $\mathbf{w} \leftarrow \{w_1, \ldots, w_q\}$, set of keywords for queries
- History $H \leftarrow (\mathbf{D}, \mathbf{w})$
- Access pattern $\alpha(H) \leftarrow (\mathbf{D}(w_1), \ldots, \mathbf{D}(w_q))$,
- Search pattern $\sigma(H) \leftarrow M(= (m_{ij})_{q \times q}$ where $m_{ij} = 1$ if $w_i = w_j$ else $0)$
- Trace $\tau(H) \leftarrow (|D_1|, \ldots, |D_n|, \alpha(H), \sigma(H))$

.

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$ Adversary $\mathcal{A}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

Adversary $\mathcal{A}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$	Adversary $\mathcal{A}$
• $K \leftarrow Gen(1^k)$	

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0, 1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$
- $t_b = (t_{b,1}, \ldots, t_{b,q})$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$
- $t_b = (t_{b,1}, \ldots, t_{b,q})$
- Sends (I_b, c_b, t_b) to $\mathcal{A}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$
- $t_b = (t_{b,1}, \ldots, t_{b,q})$
- Sends (I_b, c_b, t_b) to $\mathcal{A}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$
- $t_b = (t_{b,1}, \ldots, t_{b,q})$
- Sends (I_b, c_b, t_b) to $\mathcal{A}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Outputs 1 if $b = b'$, else output 0

Non-Adaptive Indistinguishability Game $Ind_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0, 1\}$
- parse H_b as $(\mathbf{D}_b, w_b)$
- $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- for $1 \leq i \leq q$ do $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$
- $t_b = (t_{b,1}, \ldots, t_{b,q})$
- Sends (I_b, c_b, t_b) to $\mathcal{A}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, H_0, H_1) \leftarrow \mathcal{A}_1(1^k)$
- Sends (H_0, H_1) to $\mathcal{C}$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Outputs 1 if $b = b'$, else output 0

SSE is secure if $Pr[Ind_{SSE,\mathcal{A}}(k) = 1] \leq \frac{1}{2} + negl(k)$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$ | Adversary $\mathcal{A}$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

Adversary $\mathcal{A}$

Adaptive Indistinguishability Game $Ind^*_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$ | Adversary $\mathcal{A}$

- $K \leftarrow Gen(1^k)$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$	Adversary $\mathcal{A}$
• $K \leftarrow Gen(1^k)$	• $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$

Adaptive Indistinguishability Game $Ind^*_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$

Adaptive Indistinguishability Game $Ind^*_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$
- $(st_{\mathcal{A}}, w_{0,i}, w_{1,i}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I_b, c_b, t_{b,1}, \ldots, t_{b,q-1})$ and Send $(w_{0,i}, w_{1,i})$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$
- Generate and Send $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$
- $(st_{\mathcal{A}}, w_{0,i}, w_{1,i}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I_b, c_b, t_{b,1}, \ldots, t_{b,q-1})$ and Send $(w_{0,i}, w_{1,i})$

Adaptive Indistinguishability Game $Ind^{*}_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$
- Generate and Send $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$
- $(st_{\mathcal{A}}, w_{0,i}, w_{1,i}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I_b, c_b, t_{b,1}, \ldots, t_{b,q-1})$ and Send $(w_{0,i}, w_{1,i})$
- Let $t_b = (t_{b,1}, \ldots, t_{b,q})$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Adaptive Indistinguishability Game $Ind^*_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$
- Generate and Send $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$
- $(st_{\mathcal{A}}, w_{0,i}, w_{1,i}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I_b, c_b, t_{b,1}, \ldots, t_{b,q-1})$ and Send $(w_{0,i}, w_{1,i})$
- Let $t_b = (t_{b,1}, \ldots, t_{b,q})$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Outputs 1 if $b = b'$, else output 0

Adaptive Indistinguishability Game $Ind^*_{SSE,\mathcal{A}}(k)$

Challenger $\mathcal{C}$

- $K \leftarrow Gen(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- Generate and Send $(I_b, c_b) \leftarrow Enc_K(\mathbf{D}_b)$
- Generate and Send $t_{b,1} \leftarrow Trpdr_K(w_{b,1})$
- Generate and Send $\{t_{b,i} \leftarrow Trpdr_K(w_{b,i})\}$

Adversary $\mathcal{A}$

- $(st_{\mathcal{A}}, D_0, D_1) \leftarrow \mathcal{A}_0(1^k)$
- Sends (D_0, D_1) to $\mathcal{C}$
- $(st_{\mathcal{A}}, w_{0,1}, w_{1,1}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I_b, c_b)$
- Sends $w_{0,1}, w_{1,1}$
- $(st_{\mathcal{A}}, w_{0,i}, w_{1,i}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I_b, c_b, t_{b,1}, \ldots, t_{b,q-1})$ and Send $(w_{0,i}, w_{1,i})$
- Let $t_b = (t_{b,1}, \ldots, t_{b,q})$
- $b' \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, I_b, c_b, t_b)$

Outputs 1 if $b = b'$, else output 0

SSE is secure if $Pr[Ind^*_{SSE,\mathcal{A}}(k) = 1] \leq \frac{1}{2} + negl(k)$

Non-Adaptive Semantic Security

$\mathbf{Real}_{\mathsf{SSE},\mathcal{A}}(k)$

$K \leftarrow \mathsf{Gen}(1^k)$

$(st_{\mathcal{A}}, H) \leftarrow \mathcal{A}(1^k)$

$parse\ H\ as\ (\mathbf{D}, \boldsymbol{w})$

$(I, \boldsymbol{c}) \leftarrow \mathsf{Enc}_K(\mathbf{D})$

$for\ 1 \le i \le q,$

$\quad t_i \leftarrow \mathsf{Trpdr}_K(w_i)$

$let\ \boldsymbol{t} = (t_1, \ldots, t_q)$

$output\ \mathrm{v} = (I, \boldsymbol{c}, \boldsymbol{t})\ and\ st_{\mathcal{A}}$

$\mathbf{Sim}_{\mathsf{SSE},\mathcal{A},\mathcal{S}}(k)$

$(H, st_{\mathcal{A}}) \leftarrow \mathcal{A}(1^k)$

$\mathrm{v} \leftarrow \mathcal{S}(\tau(H))$

$output\ \mathrm{v}\ and\ st_{\mathcal{A}}$

81

Adaptive Semantic Security

$\mathbf{Real}^{\star}{}_{\mathsf{SSE},\mathcal{A}}(k)$

$K \leftarrow \mathsf{Gen}(1^k)$
$(\mathbf{D}, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^k)$
$(I, \boldsymbol{c}) \leftarrow \mathsf{Enc}_K(\mathbf{D})$
$(w_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I, \boldsymbol{c})$
$t_1 \leftarrow \mathsf{Trpdr}_K(w_1)$
for $2 \leq i \leq q,$
  $(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I, \boldsymbol{c}, t_1, \ldots, t_{i-1})$
  $t_i \leftarrow \mathsf{Trpdr}_K(w_i)$
let $\boldsymbol{t} = (t_1, \ldots, t_q)$
output $\mathrm{v} = (I, \boldsymbol{c}, \boldsymbol{t})$ *and* $st_{\mathcal{A}}$

$\mathbf{Sim}^{\star}{}_{\mathsf{SSE},\mathcal{A},\mathcal{S}}(k)$

$(\mathbf{D}, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^k)$
$(I, \boldsymbol{c}, st_{\mathcal{S}}) \leftarrow \mathcal{S}_0(\tau(\mathbf{D}))$
$(w_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, I, \boldsymbol{c})$
$(t_1, st_{\mathcal{S}}) \leftarrow \mathcal{S}_1(st_{\mathcal{S}}, \tau(\mathbf{D}, w_1))$
for $2 \leq i \leq q,$
  $(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, I, \boldsymbol{c}, t_1, \ldots, t_{i-1})$
  $(t_i, st_{\mathcal{S}}) \leftarrow \mathcal{S}_i(st_{\mathcal{S}}, \tau(\mathbf{D}, w_1, \ldots, w_i))$
let $\boldsymbol{t} = (t_1, \ldots, t_q)$
output $\mathrm{v} = (I, \boldsymbol{c}, \boldsymbol{t})$ *and* $st_{\mathcal{A}}$

81

Adaptive Semantic Security for DSSE

Real$_{\mathcal{A}}(\lambda)$:

1. The challenger $\mathcal{C}$ generates a key K by running $Gen(1^{\lambda})$.
2. $\mathcal{A}$ generates a set of files $\mathbf{f}$ and sends it to $\mathcal{C}$.
3. $\mathcal{C}$ computes $(\gamma, \mathbf{c}) \leftarrow Build(K, \mathbf{f})$ and sends $(\gamma, \mathbf{c})$ to $\mathcal{A}$
4. $\mathcal{A}$ makes polynomial number of adaptive queries. In each query $\mathcal{A}$ sends either a search query for a keyword w or an add query for a file f_1 or a delete query for a file f_2 to $\mathcal{C}$.
5. Depending on the query, $\mathcal{C}$ returns either the search token $t_s \leftarrow SearchToken(K, w)$ or the add token $t_a \leftarrow AddToken(K, f_1)$ or the delete token $t_d \leftarrow DelToken(K, f_2)$ to $\mathcal{A}$.
6. Finally $\mathcal{A}$ returns a bit b that is output by the experiment.

Adaptive Semantic Security for DSSE

Ideal$_{\mathcal{A},\mathcal{S}}(\lambda)$:

1. $\mathcal{A}$ generates a set of files $\mathbf{f}$. It gives $\mathbf{f}$ and $\mathcal{L}_{bld}(\mathbf{f})$ to $\mathcal{S}$.
2. On receiving $\mathcal{L}_{bld}(\mathbf{f})$, $\mathcal{S}$ generates $(\gamma, \mathbf{c})$ and sends it to $\mathcal{A}$
3. $\mathcal{A}$ makes polynomial number of adaptive queries $q \in \{w, f_1, f_2\}$. For each query, $\mathcal{S}$ is given either $\mathcal{L}_{srch}(w, \mathbf{f})$ or $\mathcal{L}_{add}(f_1, \mathbf{f})$ or $\mathcal{L}_{del}(f_2, \mathbf{f})$.
4. Depending on the query q, $\mathcal{S}$ returns to $\mathcal{A}$ either search token t_s or add token t_a or delete token t_d.
5. Finally $\mathcal{A}$ returns a bit b that is output by the experiment.

Security

$$|Pr[\mathbf{Real}_{\mathcal{A}}(\lambda) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mu(\lambda)$$

Searchable Encryption with Complex Queries

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

- Conjunctive Queries:

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

- Conjunctive Queries:
- Disjunctive Queries

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

- Conjunctive Queries:
- Disjunctive Queries
- Boolean Queries:

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

- Conjunctive Queries:
- Disjunctive Queries
- Boolean Queries:
- Substring Queries:

Different type of Queries

Range Queries

- Given two keywords w_1 and w_2, find all keywords between w_1 and w_2.
- Order should be defined
- Existing schemes: Ishai et al. [IKLO16], Fisch et al. [FVK$^+$15]

- Conjunctive Queries:
- Disjunctive Queries
- Boolean Queries:
- Substring Queries:
- Phrase Queries:

Generalization of Searchable Encryption

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
    - Set of documents

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
  - Set of documents
  - set of keywords

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
  - Set of documents
  - set of keywords
  - Each document is connected with multiple keywords

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
  - Set of documents
  - set of keywords
  - Each document is connected with multiple keywords
  - Each keyword is connected with multiple documents

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
  - Set of documents
  - set of keywords
  - Each document is connected with multiple keywords
  - Each keyword is connected with multiple documents
- Lai and Chow [LC17] proposed a forward-secure Searchable Encryption considering it as a Bipartite Graph

Graph Encryption

- Graph Encryption is a generalization of Searchable Encryption
- It can be considered as Bipartite Graph
  - Set of documents
  - set of keywords
  - Each document is connected with multiple keywords
  - Each keyword is connected with multiple documents
- Lai and Chow [LC17] proposed a forward-secure Searchable Encryption considering it as a Bipartite Graph
- More complex queries can be solved if Graph encryption become efficient

Scope of Research

Future Research Directions

- Efficient Forward Secure Scheme Design

Future Research Directions

- Efficient Forward Secure Scheme Design
- New techniques can be applied to propose new SSE/DSSE scheme

Future Research Directions

- Efficient Forward Secure Scheme Design
- New techniques can be applied to propose new SSE/DSSE scheme
- Efficient Attacks on existing schemes

Future Research Directions

- Efficient Forward Secure Scheme Design
- New techniques can be applied to propose new SSE/DSSE scheme
- Efficient Attacks on existing schemes
- Provide Solutions of the attacks

Future Research Directions

- Efficient Forward Secure Scheme Design
- New techniques can be applied to propose new SSE/DSSE scheme
- Efficient Attacks on existing schemes
- Provide Solutions of the attacks
- Complex queries on Encrypted Data/DSSE

Future Research Directions

- Efficient Forward Secure Scheme Design
- New techniques can be applied to propose new SSE/DSSE scheme
- Efficient Attacks on existing schemes
- Provide Solutions of the attacks
- Complex queries on Encrypted Data/DSSE
- Complex queries on Encrypted Graph

Raphaël Bost, Brice Minaud, and Olga Ohrimenko.
Forward and backward private searchable encryption from constrained cryptographic primitives.
In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1465–1482, New York, NY, USA, 2017. ACM.

Raphael Bost.
Sophos - forward secure searchable encryption.
IACR Cryptology ePrint Archive, 2016:728, 2016.

CERT.
2012 cybersecurity watch survey: How bad is the insider threat?
2012.

Reza Curtmola, Juan A. 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 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 79–88, 2006.

David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart.
Leakage-abuse attacks against searchable encryption.
In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 668–679, 2015.

David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner.
Dynamic searchable encryption in very-large databases: Data structures and implementation.
In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

Yan-Cheng Chang and Michael Mitzenmacher.
Privacy preserving keyword searches on remote encrypted data.
In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 442–455, 2005.

Ben A. Fisch, Binh Vo, Fernando Krell, Abishek Kumarasubramanian, Vladimir Kolesnikov, Tal Malkin, and Steven M. Bellovin.
Malicious-client security in blind seer: A scalable private DBMS.
In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 395–410, 2015.

Eu-Jin Goh.
Secure indexes.
IACR Cryptology ePrint Archive, 2003:216, 2003.

Florian Hahn and Florian Kerschbaum.
Searchable encryption with secure and efficient updates.
In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 310–320, 2014.

Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu.
Access pattern disclosure on searchable encryption: Ramification, attack and mitigation.
In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky.
Private large-scale databases with distributed searchable symmetric encryption.
In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 90–107, 2016.

Seny Kamara and Tarik Moataz.
Boolean searchable symmetric encryption with worst-case sub-linear complexity.
In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 94–124, 2017.

Seny Kamara and Charalampos Papamanthou.
Parallel and dynamic searchable symmetric encryption.
In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 258–274, 2013.

Seny Kamara, Charalampos Papamanthou, and Tom Roeder.
Dynamic searchable symmetric encryption.
In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976, 2012.

Russell W. F. Lai and Sherman S. M. Chow.
Forward-secure searchable encryption on labeled bipartite graphs.
In *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, pages 478–497, 2017.

Muhammad Naveed, Seny Kamara, and Charles V. Wright.
Inference attacks on property-preserving encrypted databases.
In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 644–655, 2015.

Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter.
Dynamic searchable encryption via blind storage.
In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 639–654, 2014.

Panagiotis Rizomiliotis and Stefanos Gritzalis.
ORAM based forward privacy preserving dynamic searchable symmetric encryption schemes.
In *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop, CCSW 2015, Denver, Colorado, USA, October 16, 2015*, pages 65–76, 2015.

Emil Stefanov, Charalampos Papamanthou, and Elaine Shi.
Practical dynamic searchable encryption with small leakage.
In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig.
Practical techniques for searches on encrypted data.
In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55, 2000.

Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou.
All your queries are belong to us: The power of file-injection attacks on searchable encryption.
In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 707–720, 2016.

Questions?

Thank You!