Secure Indexes*

Eu-Jin Goh

Stanford University

15 March 2004

* Generalizes an early version of my paper
"How to search on encrypted data"
on ePrint Cryptology Archive on 7 October 2003

Secure Indexes

Data Structures that –

- Index words $(w_1 , \ldots , w_n)$ in a doc
- Allow users with trapdoor for word *w* to search only for *w* in *O(1)* time
- Contents hidden without trapdoor
- Index preserves semantic security of encrypted documents
  - Do not hide public info about doc (e.g. encrypted file size)

Applications

1. Searching on Encrypted Data [SWP00, G03, BDOP03, CM04]
2. Encrypted Searchable Audit Logs [WBDS04]
3. Private Database Queries [BC04]
4. Accumulated Hashing
5. Private Set Membership Test

Talk Overview

1. Security model
   - IND-CKA — almost always sufficient
   - IND2-CKA — stronger (by [CM04])
2. Efficient Construction (Z-IDX)
   - Variants secure in both models

Secure Index Scheme

Consists of 4 algorithms –

1. Keygen
2. Trapdoor
3. BuildIndex
4. SearchIndex

IND-CKA Intuition

Goal – Semantic Security

$\mathcal{A}$ cannot deduce doc contents from index

IND-CKA Intuition

Goal — Semantic Security

A cannot deduce doc contents from index

Captured using standard IND Game —

1. A chooses 2 equal size docs V_0 , V_1 and is given index I for either V_0 or V_1
2. V_0 and V_1 (possibly) unequal # words
3. A guesses which doc is indexed by I

IND-CKA Intuition

Goal — Semantic Security

$\mathcal{A}$ cannot deduce doc contents from index

Captured using standard IND Game —

1. $\mathcal{A}$ chooses 2 equal size docs V_0 , V_1 and is given index I for either V_0 or V_1
2. V_0 and V_1 (possibly) unequal # words
3. $\mathcal{A}$ guesses which doc is indexed by I

Chosen Keyword Attack (CKA) — $\mathcal{A}$ given

1. plain text access to all docs + indexes
2. queries for any trapdoor of its choice (restricted after challenge)

IND2-CKA Intuition

Goal — Semantic Security

A cannot deduce doc contents from index

Captured using standard IND2 Game —

1. A chooses 2 docs V_0 , V_1 and is given index *I* for either V_0 or V_1
2. V_0 , V_1 (possibly) unequal size + # words
3. A guesses which doc is indexed by *I*

Chosen Keyword Attack (CKA) — A given

1. plain text access to all docs + indexes
2. queries for any trapdoor of its choice (restricted after challenge)

IND-CKA vs. IND2-CKA

IND-CKA

- Equal size docs have indexes that appear to contain same # of words/tokens

IND2-CKA [CM04]

- Unequal size docs have indexes that appear to contain same # of words/tokens
- But can already distinguish indexes for unequal size docs from doc size

IND-CKA vs. IND2-CKA

IND-CKA

- Equal size docs have indexes that appear to contain same # of words/tokens

IND2-CKA [CM04]

- Unequal size docs have indexes that appear to contain same # of words/tokens
- But can already distinguish indexes for unequal size docs from doc size

IND2-CKA model appears too strong

- IND-CKA probably strong enough + gives more efficient constructions

Construction Z-IDX

Z-IDX built using

1. Bloom filters (BF) –
   - Efficiently test set membership
   - *O(1)* insert/test algorithms
2. Pseudo-random functions (PRF)
   - emulate "random functions"

IND-CKA Z-IDX

Keygen (s): PRF $f: \{0,1\}^n$ ´ $\{0,1\}^s$ ® $\{0,1\}^s$

Output $K_{priv} = (k_1 , \ldots , k_r) \xleftarrow{R} \{0,1\}^{sr}$

IND-CKA Z-IDX

Keygen (s): PRF *f: {0,1}*^{n} ´ *{0,1}*^{s} ® *{0,1}*^{s}

Output $K_{priv} = (k_1 , \ldots , k_r) \xleftarrow{R} \{0,1\}^{sr}$

Trapdoor (K_{priv}, *w*):

Output $T_w = (f(w , k_1) , \ldots , f(w , k_r))$ Î $\{0,1\}^{sr}$

IND-CKA Z-IDX

Keygen (s): PRF f: $\{0,1\}^n$ ´ $\{0,1\}^s$ ® $\{0,1\}^s$

Output $K_{priv} = (k_1, \ldots, k_r) \xleftarrow{R} \{0,1\}^{sr}$

Trapdoor (K_{priv}, w):

Output $T_w = (f(w, k_1), \ldots, f(w, k_r))$ Î $\{0,1\}^{sr}$

BuildIndex (D, K_{priv}): Let $D = (D_{id}, w_0, \ldots, w_n)$,

```
$u$ = upper bound on # words for doc of size $|D|$
1) For $w_0, \ldots, w_n$, do
    a) Compute $T_{w_i} = ( x_1 = f( w_i , k_1 ), \ldots, x_r = f( w_i , k_r ) )$
    b) Compute + insert $( f( D_{id} , x_1 ), \ldots, f( D_{id} , x_r ) )$ in $BF$
2) Insert $(u - n) \cdot r$ of 1's uniformly at random in $BF$
3) Output $I_D = (D_{id}, BF)$
```

IND-CKA Z-IDX

Keygen (s): PRF f: $\{0,1\}^n$ ´ $\{0,1\}^s$ ® $\{0,1\}^s$

Output $K_{priv} = (k_1 , \ldots , k_r) \xleftarrow{R} \{0,1\}^{sr}$

Trapdoor (K_{priv}, w):

Output $T_w = (f(w , k_1) , \ldots , f(w , k_r))$ Î $\{0,1\}^{sr}$

```
BuildIndex ($D$, $K_{priv}$): Let $D = ( D_{id} , w_0 , \ldots, w_n )$,
    $u$ = upper bound on # words for doc of size $|D|$
    1) For $w_0, \ldots, w_n$, do
        a) Compute $T_{w_i} = ( x_1 = f( w_i , k_1 ) , \ldots , x_r = f( w_i , k_r ) )$
        b) Compute + insert $( f( D_{id} , x_1 ) , \ldots , f( D_{id} , x_r ) )$ in $BF$
    2) Insert $(u - n)\cdot r$ of 1's uniformly at random in $BF$
    3) Output $I_D = (D_{id} , BF)$
```

```
SearchIndex ($T_w$, $I_D$): Let $T_w = ( x_1, \ldots , x_r )$, $I_D = ( D_{id} , BF )$
    1) Compute $( y_1 = f( D_{id} , x_1 ) , \ldots , y_r = f( D_{id} , x_r ) )$
    2) Test if $BF$ contains 1's in all $y_1, \ldots , y_r$ locations
```

IND2-CKA Z-IDX

Keygen (s): PRF f: $\{0,1\}^n$ ´ $\{0,1\}^s$ ® $\{0,1\}^s$

Output $K_{priv} = (k_1 , \dots , k_r) \xleftarrow{R} \{0,1\}^{sr}$

Trapdoor (K_{priv}, w):

Output $T_w = (f(w , k_1) , \dots , f(w , k_r))$ Î $\{0,1\}^{sr}$

BuildIndex (D, K_{priv}): Let $D = (D_{id} , w_0 , \dots, w_n)$,

```
    $u$ = global upper bound on # words for single doc
    1) For $w_0, \dots, w_n$, do
        a) Compute $T_{w_i} = ( x_1 = f( w_i , k_1 ) , \dots , x_r = f( w_i , k_r ) )$
        b) Compute + insert $( f( D_{id} , x_1 ) , \dots , f( D_{id} , x_r ) )$ in $BF$
    2) Insert $(u – n)\cdot r$ of 1's uniformly at random in $BF$
    3) Output $I_D = (D_{id} , BF)$
```

SearchIndex (T_w, I_D): Let $T_w = (x_1, \dots , x_r)$, $I_D = (D_{id} , BF)$

```
    1) Compute $( y_1 = f( D_{id} , x_1 ) , \dots , y_r = f( D_{id} , x_r ) )$
    2) Test if $BF$ contains 1's in all $y_1, \dots , y_r$ locations
```

Z-IDX Properties

1. Handle arbitrary updates
2. Compressible Indexes
   - Space efficient for small and medium size docs
3. Short Trapdoors
4. Computationally very efficient
5. Occurrence Search
6. Efficient Boolean + Limited Regex Queries
7. Simple Key Management

Chang-Mitzenmacher (Feb 2004)

- Based on similar techniques as Z-IDX
- IND2-CKA secure
- Use pre-built dictionaries

Chang-Mitzenmacher (Feb 2004)

- Based on similar techniques as Z-IDX
- IND2-CKA secure
- Use pre-built dictionaries

Advantages

- More space efficient than IND2-CKA secure Z-IDX
- No false positives (negligible in Z-IDX with proper choice of BF params)

Chang-Mitzenmacher (Feb 2004)

- Based on similar techniques as Z-IDX
- IND2-CKA secure
- Use pre-built dictionaries

Advantages

- More space efficient than IND2-CKA secure Z-IDX
- No false positives (negligible in Z-IDX with proper choice of BF params)

Disadvantages

- Cannot handle arbitrary updates
- Much less comp. efficient than both Z-IDX's
- Large fixed size indexes — not compressible

  ⇒ less space efficient than IND-CKA Z-IDX for small and medium size docs