Searchable Encryption: Part 1

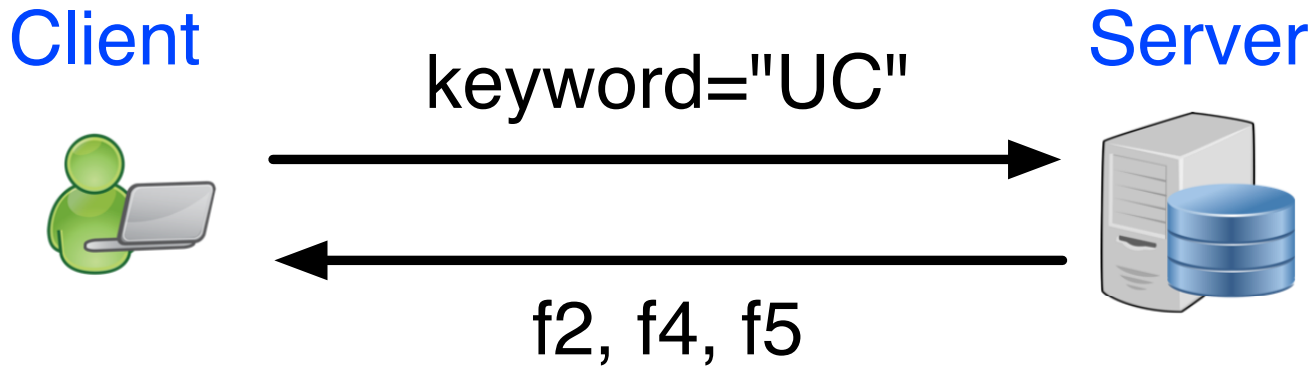
CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

Keyword Search

- Server maintains many files for client
- Client submits a keyword to server
- Server returns files containing this keyword



  - E.g., search a keyword in your UCmail/Gmail
  - Client searches “UC”, server returns f2, f4, f5

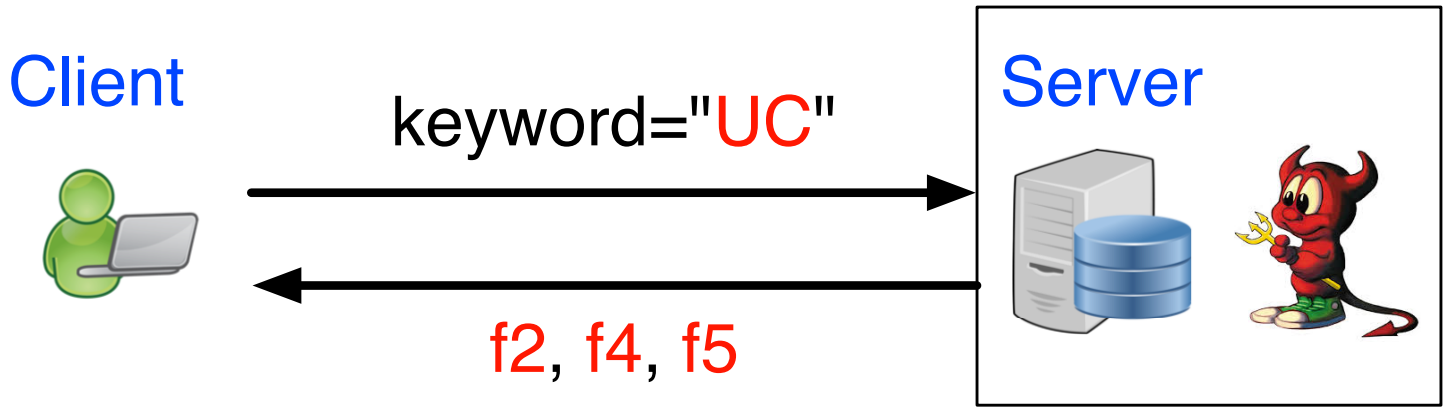
Keyword Search

- Example: files f1, f2, f3, f4
    - f1 includes k1, k2
    - f2 includes k3
    - f3 includes k1, k3, k4
    - f4 includes k2,
- Client searches k3, Server returns f2, f3

- Client saves storage and querying costs
    - Client does not need to maintain all the files
    - Client does not need to perform search

Untrusted Server

- Client does not (fully) trust server
  - A hacker on the server
    - Equifax leaked 143 million SSNs (2017)
    - Yahoo leaked 3 billion account info (2013)
  - A curious manager sells data



Untrusted Server

- Client does not (fully) trust server
  - Server/Attacker learns the keyword query
  - Server/Attacker learns data in each file

- Client wants to keep data/queries private
  - Does not want server know what is the keyword
  - Does not want server know content in its files
  - E.g., medical records, locations, financial data, credit card information, home address, travel plans, etc.

Secure Keyword Search

- Option 1: Client keeps all the files, searches locally
  - Pros:
    - Data and queries are private
    - Server knows nothing
  - Cons:
    - Client spends huge costs, storage & querying
    - Server is idle

- Client still wants to save costs by using server

Secure Keyword Search

- Option 2: Client encrypts all the files (AES), keeps secret key, then uploads encrypted files to server
  - Pros:
    - Data is private
    - Server knows nothing (does not know the key)
    - Client saves storage
  - Cons:
    - Server cannot search (AES leaks nothing)
    - Client has to retrieve all the (encrypted) files decrypt all, and search locally for each query

Secure Keyword Search

- Option 3: Client sends data to server, server encrypts data with AES, server has AES key. (encrytion-at-rest, e.g., Dropbox, iCloud)
  - Pros:
    - Data is relatively private
    - Client saves storage
  - Cons:
    - Server decrypts all the data for search
    - Untrusted server still knows data

Secure Keyword Search

- Searchable Encryption: Client encrypts all the files, keeps secret key, uploads encrypted files to server
  - Pros:
    - Data and queries are private
    - Server knows nothing (does not know the key)
    - Server can search
    - Client retrieves only matched files for a keyword
    - Client saves storage & querying
  - Cons:
    - Search time is slower but still practical

Information Retrieval

- Search in plaintext
  - Option 1: Given a keyword, server scans data (string by string) in each file and checks whether this keyword is inside a file

  - Not efficient
    - need to scan all the files string by string
    - E.g., each file has $\mathbf{s}$ strings, and $\mathbf{n}$ files in total
    - Search time: O($\mathbf{sn}$) for each keyword query

Keyword-File Matrix

- Build an index in advance
  - Extra pre-processing time, but boost each query
- Keyword-File Matrix (a binary matrix)
  - m keywords, n files, —> m*n matrix
  - M(i,j) = 1 if file f_j has keyword k_i

f1: k1, k3
f2: k2
f3: k4
f4: k2
f5: k1, k2

	f1	f2	f3	f4	f5
k1	1	0	0	0	1
k2	0	1	0	1	1
k3	1	0	0	0	0
k4	0	0	1	0	0

Keyword-File Matrix

- Practice: 4 files and 4 keywords
  - f1 includes k1, k2; f2 includes k3;
  - f3 includes k1, k3, k4; f4 includes k2
  - Keyword-File Matrix??

	f1	f2	f3	f4
k1	?	?	?	?
k2	?	?	?	?
k3	?	?	?	?
k4	?	?	?	?

Keyword-File Matrix

- Practice: 4 files and 4 keywords
  - f1 includes k1, k2; f2 includes k3;
  - f3 includes k1, k3, k4; f4 includes k2

	f1	f2	f3	f4
k1	1	0	1	0
k2	1	0	0	1
k3	0	1	1	0
k4	0	0	1	0

Keyword-File Matrix

- Keyword-File Matrix (a binary matrix)
  - Search keyword k_i, only need to check the i-th row in the matrix, if M(i, j) = 1, return f_j
  - No need to scan all the files (v.s. straightforward)

Search k3, check 3-rd row in matrix

	f1	f2	f3	f4	f5
k1	1	0	0	0	1
k2	0	1	0	1	1
k3	1	0	0	0	0
k4	0	0	1	0	0

Keyword-File Matrix

- If implement matrix as a <u>two-dimensional array</u>
  - Search is a two-layer for loop
  - E.g., given keyword k', search in matrix M

```
for (i=1; i<=4; i++)
   if (k'==k[i])
      for (j=1; j<=5; j++)
         M[i][j]?=1
```

	f1	f2	f3	f4	f5
k1	1	0	0	0	1
k2	0	1	0	1	1
k3	1	0	0	0	0
k4	0	0	1	0	0

Keyword-File Matrix

- If implement matrix as a two-dimensional array
  - Search is a two-layer for loop
  - Finding a matched keyword item takes O(m)
  - Searching the row takes O(n)
  - Total search time O(m+n)

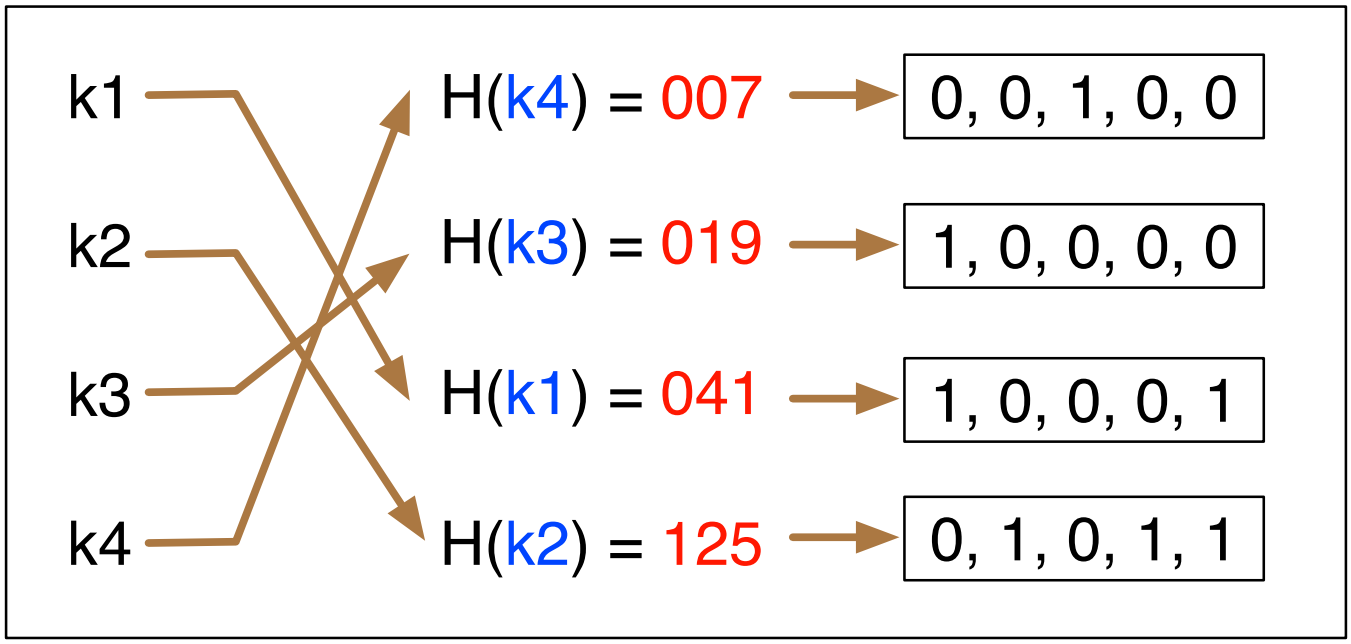
Search k3, check 3-rd row in matrix

	f1	f2	f3	f4	f5
k1	1	0	0	0	1
k2	0	1	0	1	1
k3	1	0	0	0	0
k4	0	0	1	0	0

Keyword-File Matrix

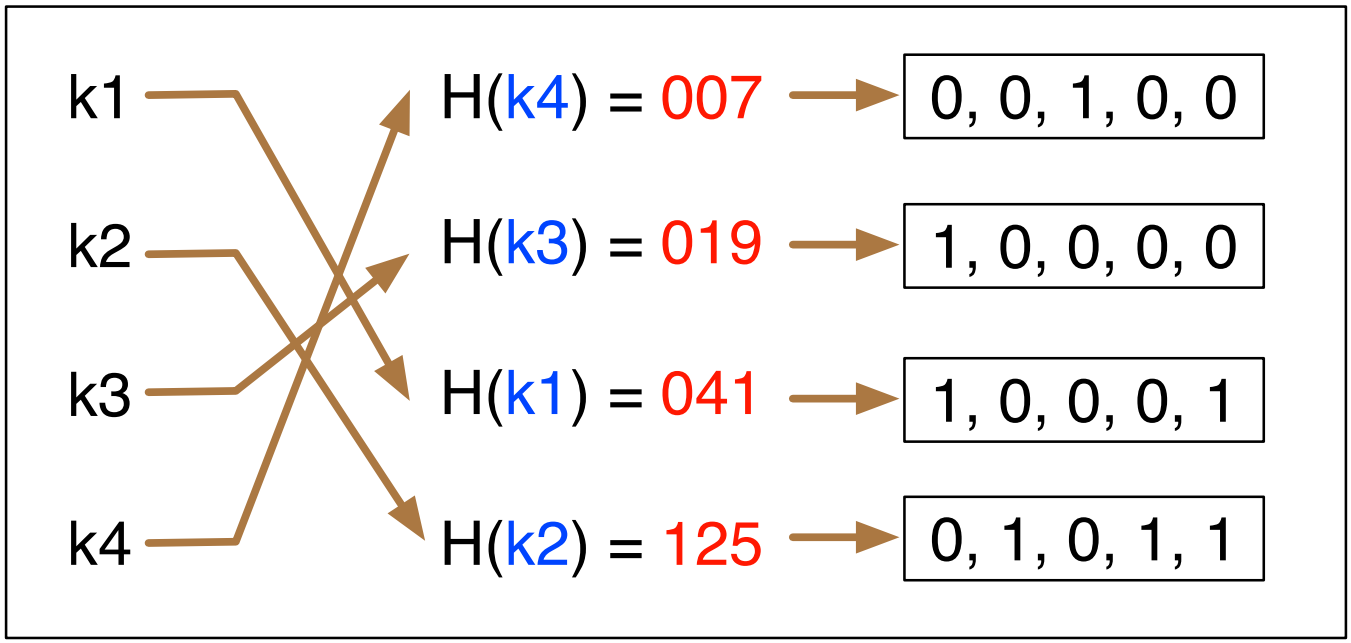
- All the keywords are stored in a hash table

```
h = H(k') // (H is non-crypto hash fun.)
    for (j=1; j<=5; j++)
        M[h][j]?=1
```



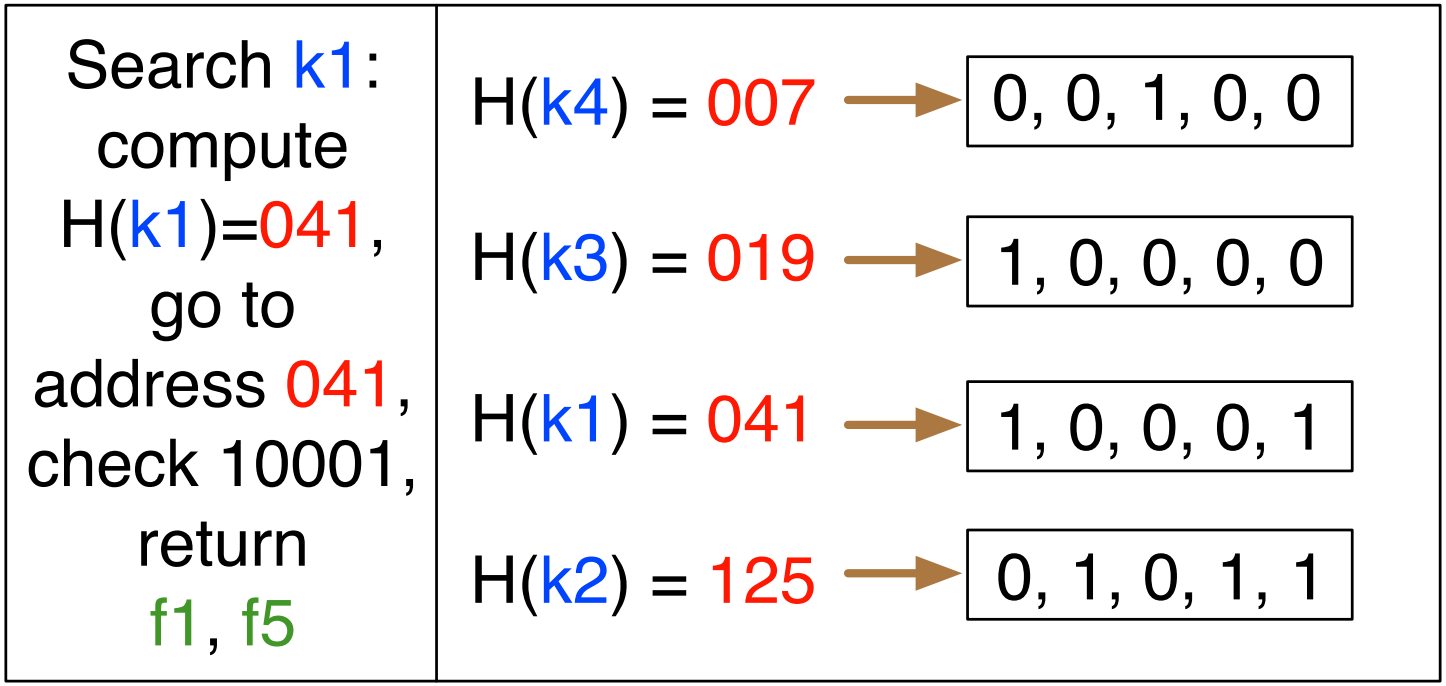
Keyword-File Matrix

- All the keywords are stored in a hash table
  - Find address of matched keyword with O(1)
  - Search its corresponding row with O(n)
  - Total search time: O(n)



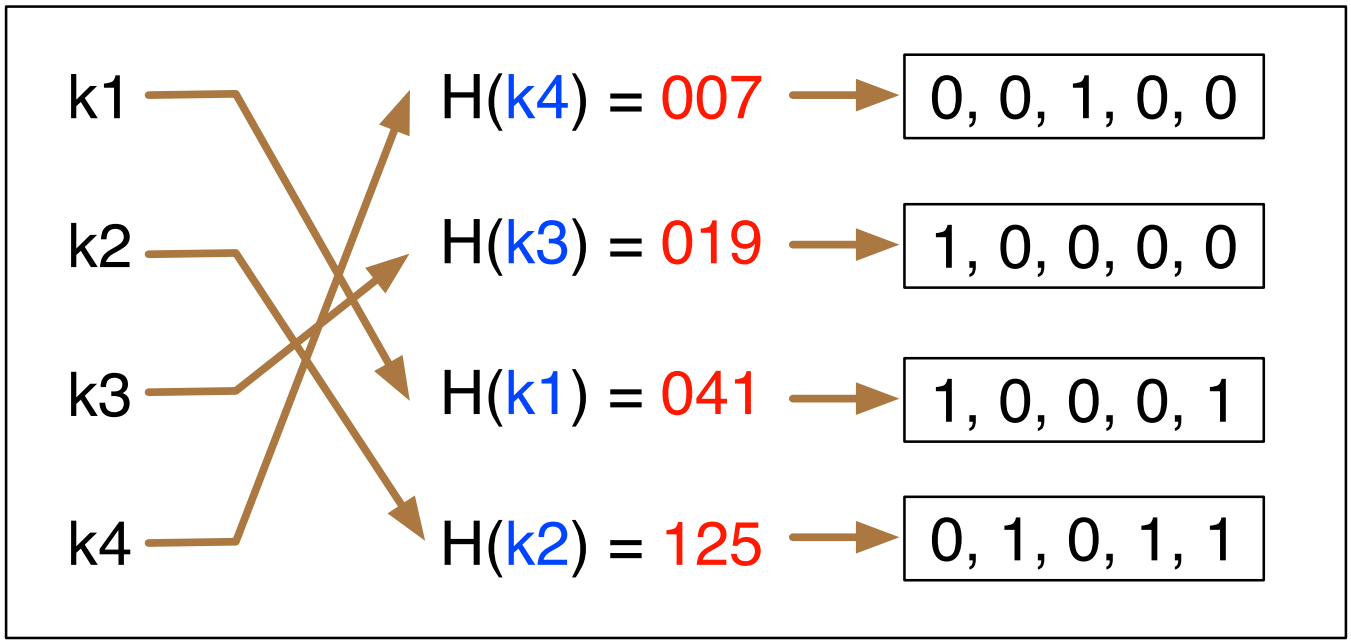
Keyword-File Matrix

- Return associated files based 1s
  - Client searches k1, server returns f1, f5 to client



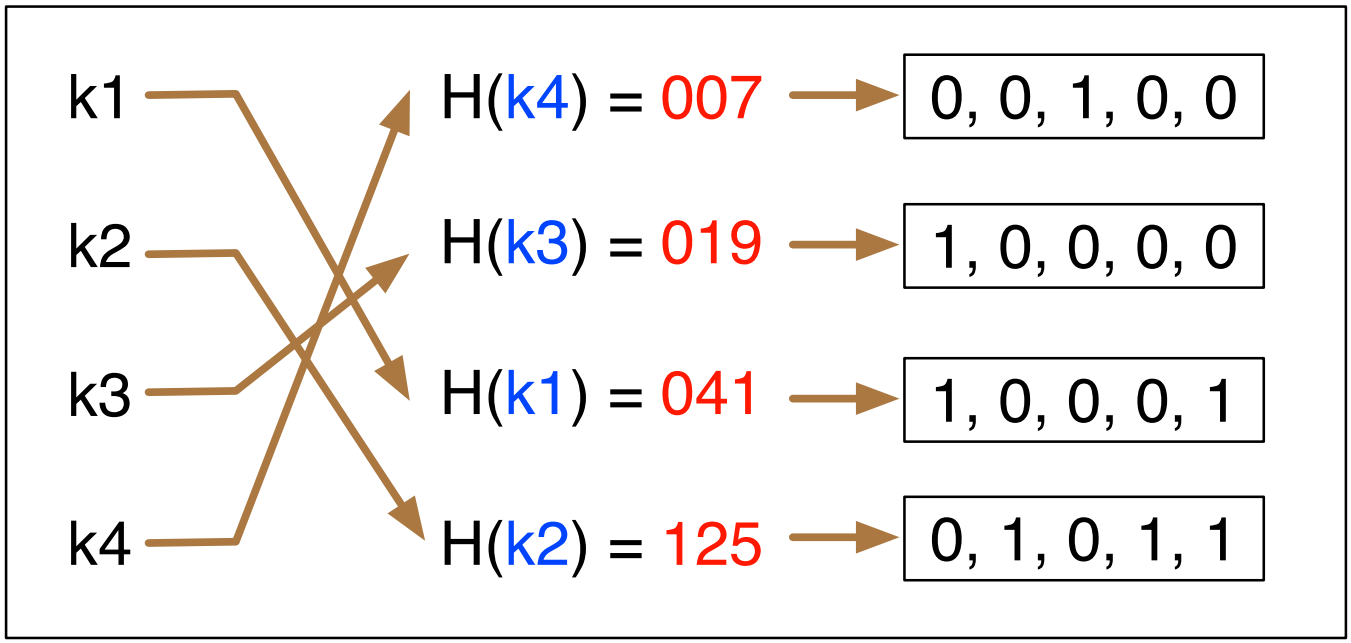
Keyword-File Matrix

- Practice: Search process for k' = k2
  - Compute ????
  - Go to address ???, check ????
  - Return ????



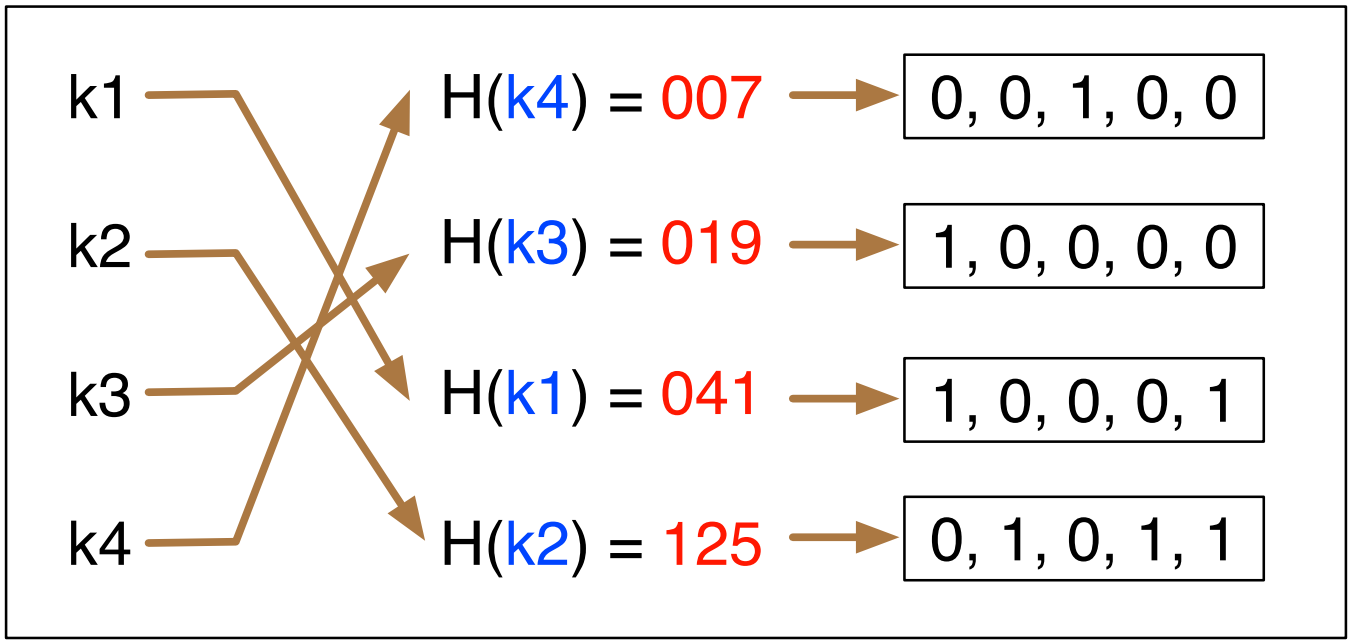
Keyword-File Matrix

- Practice: Search process for k' = k2
  - Compute H(k2) = 125
  - Go to address 125, check 0, 1, 0, 1, 1
  - Return f2, f4, f5



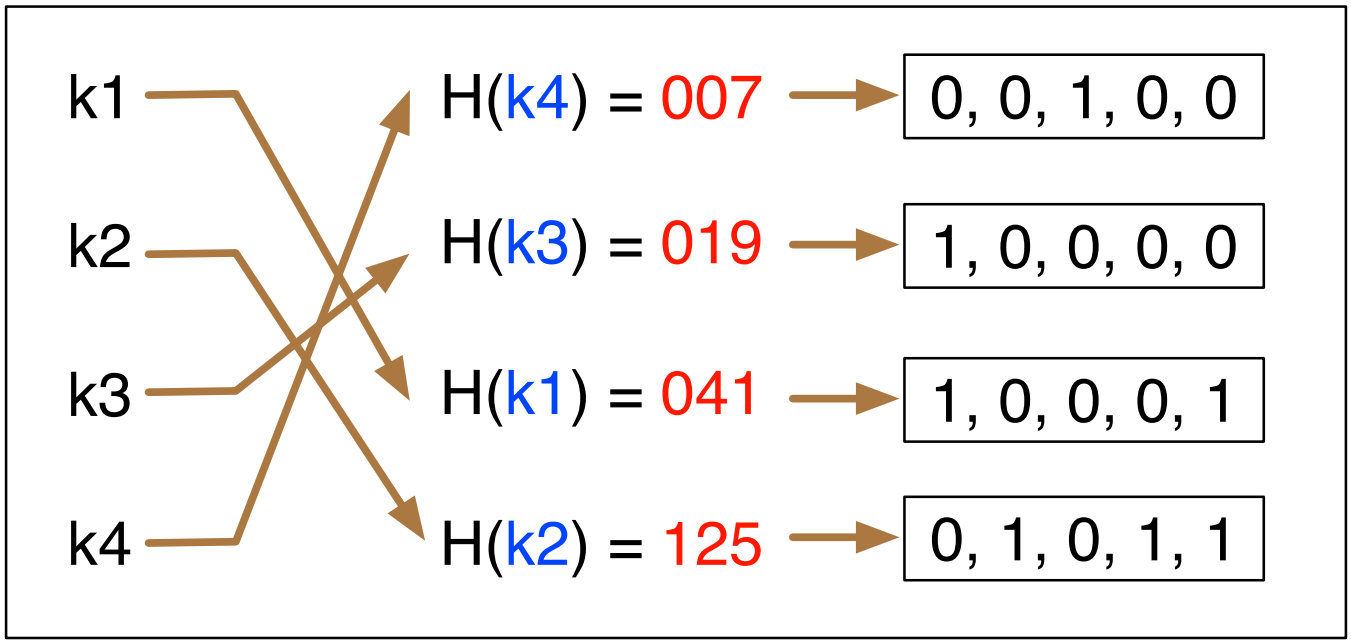
Keyword-File Matrix

- Practice: Search process for k' = k6
  - Compute H(k6) = 098
  - Go to address ????, check ????
  - Return ????



Keyword-File Matrix

- Practice: Search process for k' = k6
  - Compute H(k6) = 098
  - Go to address 098, empty
  - Return null



Keyword-File Matrix

- Keyword-File Matrix is sparse
  - No. of files is much greater than no. of keywords
    - E.g., 200 keywords, 10,000 files
  - Each file only has a small subset of keywords
  - Many 0s, which slow down search time

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12
k1	1	0	1	0	1	0	0	0	1	0	1	0
k2	0	1	0	1	0	0	0	0	0	1	0	0
k3	1	1	0	0	1	1	0	0	0	0	1	0
k4	0	0	1	0	0	0	1	1	0	0	0	1

Inverted Index

- Inverted Index (built from keyword-file matrix)
  - Only keeps associated file identifiers (pointers or memory addresses) for each keyword

	f1	f2	f3	f4	f5
k1	1	0	0	0	1
k2	0	1	0	1	1
k3	1	0	0	0	0
k4	0	0	1	0	0

Inverted Index

k1: id1, id5
k2: id2, id4, id5
k3: id1
k4: id3

Inverted Index

- Practice: 4 files and 4 keywords
  - f1 includes k1, k2; f2 includes k3;
  - f3 includes k1, k3, k4; f4 includes k2
  - What is the inverted index of this matrix?

	f1	f2	f3	f4
k1	1	0	1	0
k2	1	0	0	1
k3	0	1	1	0
k4	0	0	1	0

Inverted Index

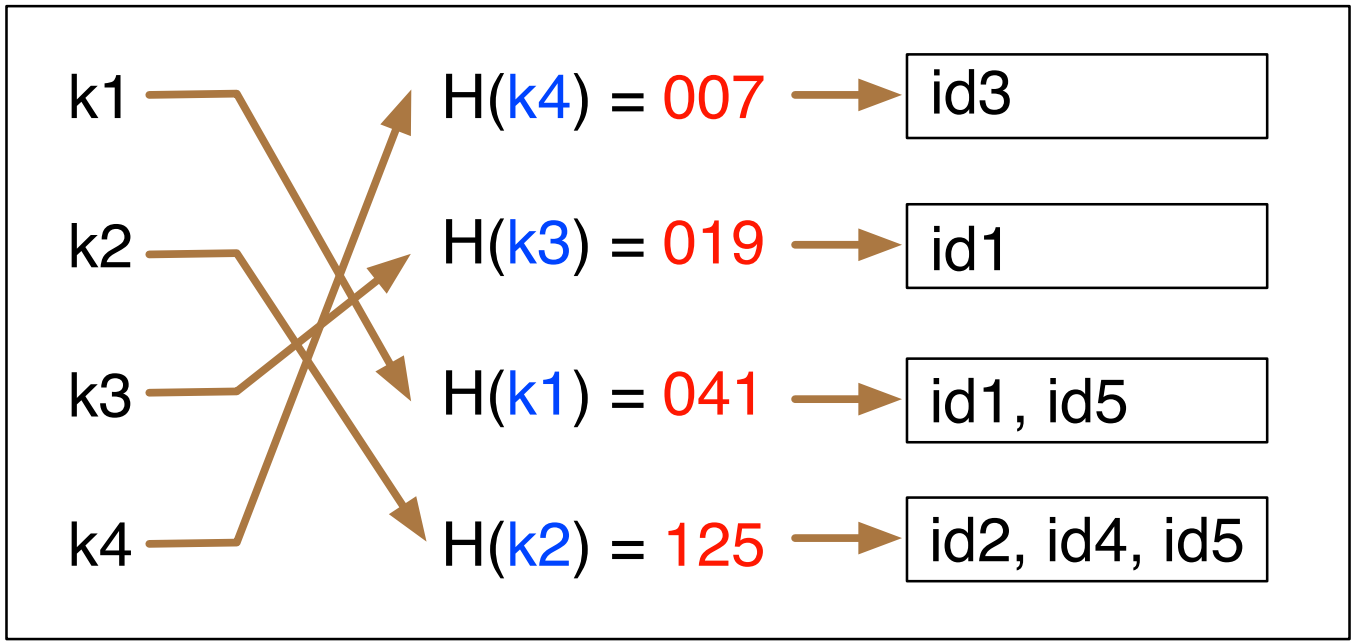
- Practice: 4 files and 4 keywords
  - f1 includes k1, k2; f2 includes k3;
  - f3 includes k1, k3, k4; f4 includes k2
  - What is the inverted index of this matrix?

	f1	f2	f3	f4
k1	1	0	1	0
k2	1	0	0	1
k3	0	1	1	0
k4	0	0	1	0

k1	id1, id3
k2	id1, id4
k3	id2, id3
k4	id3

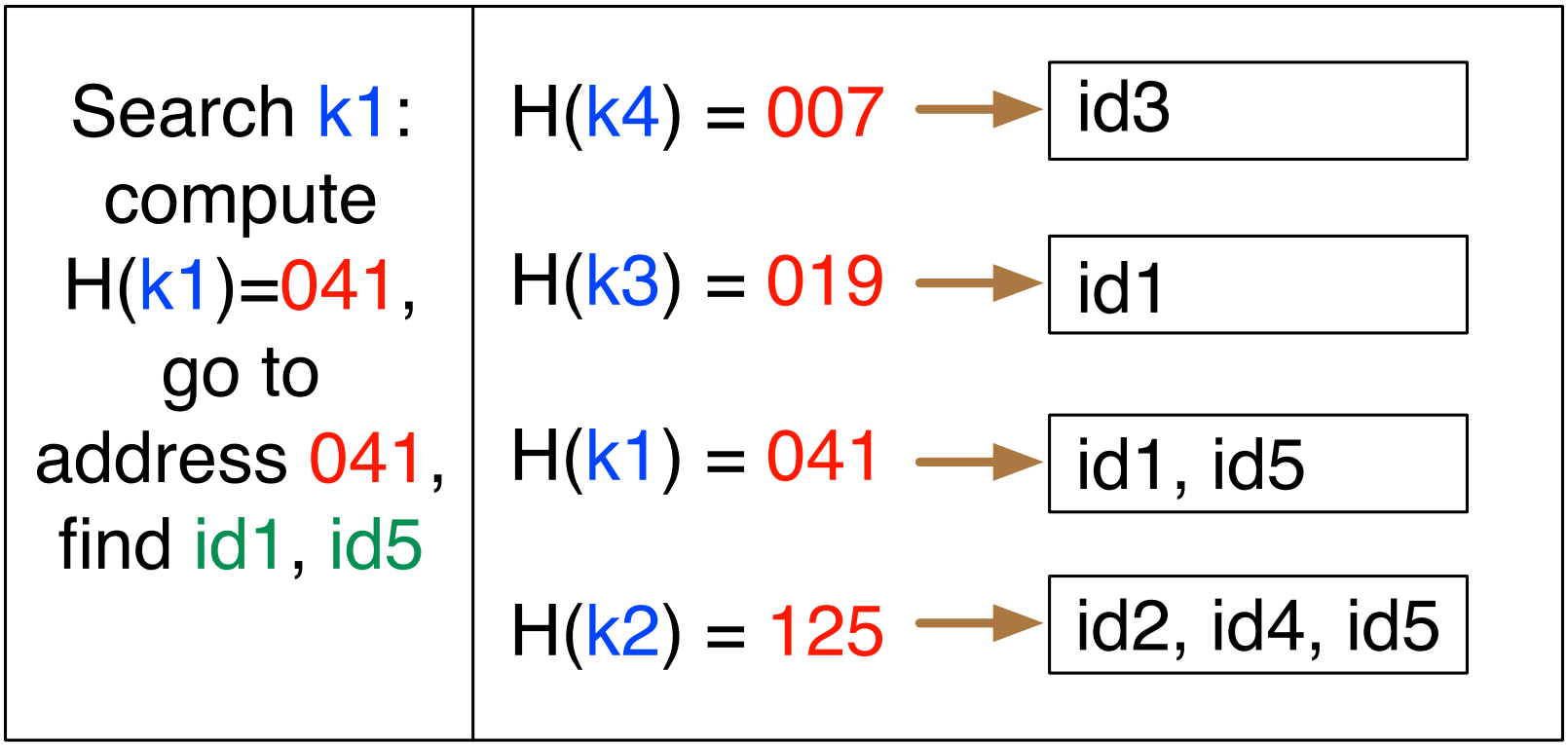
Inverted Index

- All the keywords are stored in a hash table
- Find address of matched keyword with O(1)
- Then search its corresponding row with O(r)
- Sublinear search time: O(r), r << n



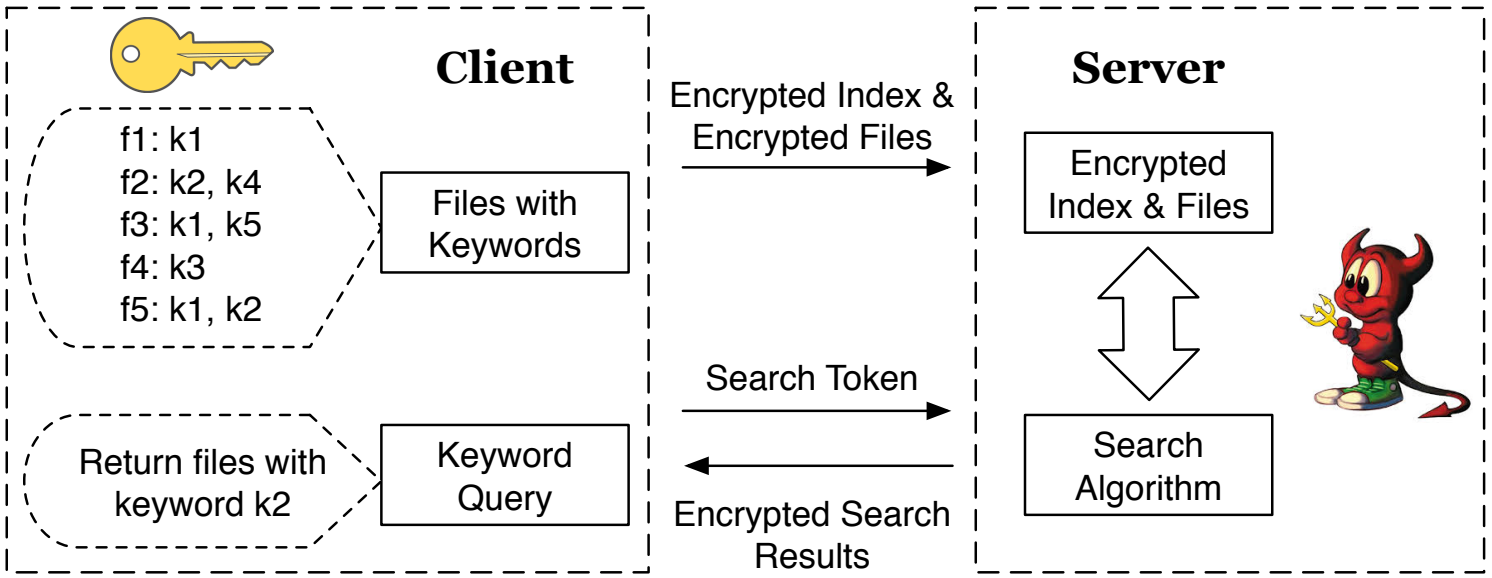
Inverted Index

- Return associated files based associated identifiers
  - Client searches k1, server returns f1, f5 to client



Searchable Encryption

- Client builds an encrypted index, and encrypts files with AES
- Server can still search, but does not learn keyword query or returned files



- KeyGen: given a security parameter λ, client outputs a secret key sk.
- Enc: given a set of files $f_1, ..., f_n$ and a secret key sk, client outputs an encrypted index Γ and a set of encrypted files $c_1, ..., c_n$.
- Token: given a keyword k and a secret key sk, client outputs a token tk.
- Query: given an encrypted index Γ, a set of encrypted files $c_1, ..., c_n$ and a token tk, server returns all the encrypted files that associated with tk.

Building Blocks

- Pseudo Random Function (PRF)
  - Deterministic encryption
    - E.g., $PRF_k(m_1) = w_1$, $PRF_k(m_2) = w_2$
    - If $m_1 = m_2$, then $w_1 = w_2$
- Advanced Encryption Standard (AES)
  - AES-CBC: probabilistic encryption
- Inverted Index with non-crypto hash functions

Additional Reading

R. Curmola, J. Garay, S. Kamara, R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," in the Proceedings of the 13th ACM Conference on Computer and Communication Security (CCS) 2006