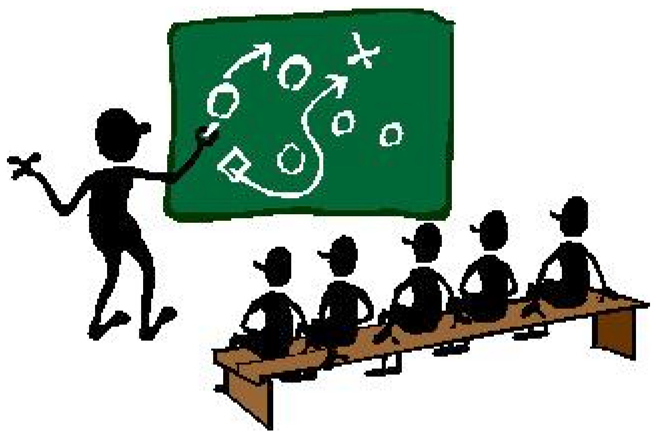
Searches Through Encrypted Data

presenter: Reza Curtmola

Introduction

- Searching usually done over plaintext

- But what if we could search encrypted data?



Bloom Filters

- Efficient method to encode set membership
- The set: n elements (n is large)
- The Bloom filter: array of m bits (m is small)
- r independent hash functions:

  $h_i:\{0,1\}^* \to [1,m]; \; i \in [1,r]$

Bloom Filters - example

h_1(‘water’)=2
h_2(‘water’)=5
h_3(‘water’)=9

h_1(‘sky’)=1
h_2(‘sky’)=5
h_3(‘sky’)=7

1	1			1		1		1	
1	2	3	4	5	6	7	8	9	10

h_1(‘air’)=2
h_2(‘air’)=5
h_3(‘air’)=7

false positive!

To minimize false positive rate, need to choose

$$r = \ln 2 * \frac{m}{n}$$

$$FP = (\frac{1}{2})^r$$

Bloom Filters

- Properties:
  - History independent
  - Once added, elements can't be removed

- Examples of usage:

  password schemes, IP traceback schemes, intrusion detection, SED

Encrypted Bloom Filter

- Restrict ability to compute the hash functions by using a secret

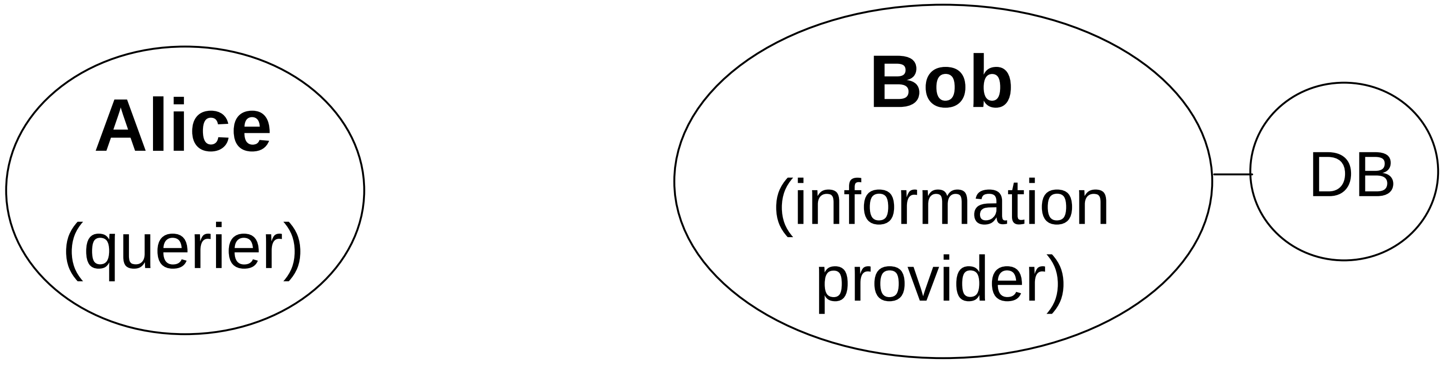
$h_1(w,k_1)$	$f(w,k_1)$
$h_2(w,k_2)$	$f(w,k_2)$
…	…
$h_r(w,k_r)$	$f(w,k_r)$

Bloom Filters used for SED

- Model 1:
  - Parties want to share data selectively
- Model 2:
  - User stores encrypted data on untrusted storage

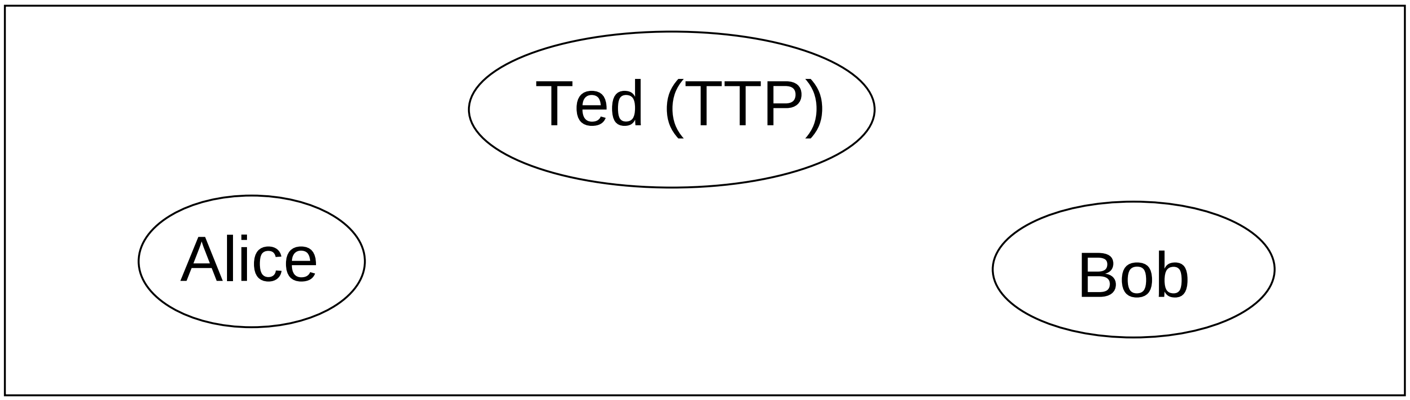
Privacy-Enhanced Searches

- Bellovin, Cheswick, "Privacy-enhanced Searches Using Encrypted Bloom Filters"
- Two parties want to share data selectively
- The parties don't trust each other



Properties

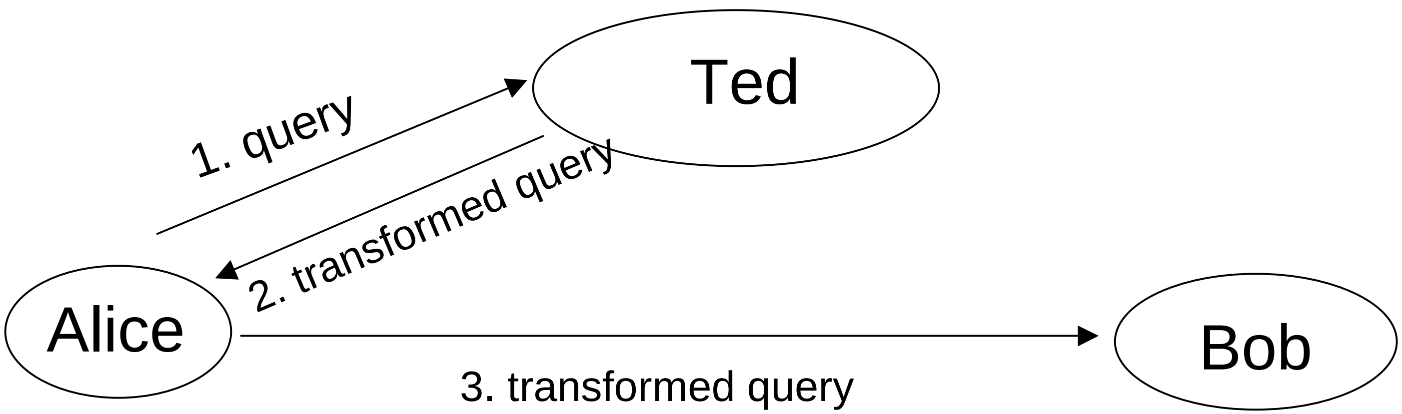
- Alice should be able to retrieve only documents matching valid queries
- Bob should not find contents of queries



- No third party should gain knowledge about queries or documents

The Basic Scheme

- Three-party negotiation between Alice, Bob and Ted to provision Ted with the transformation keys
- Bob prepares his DB as a collection of encrypted Bloom filters



Group Ciphers

- The set of all keys k forms an Abelian group under the operation composition of encryption

$$E_{k_1}(E_{k_2}(W)) = E_{k_1 \circ k_2}(W)$$

- Ted knows $r_{A,B} = k_B \circ k_A^{-1}$

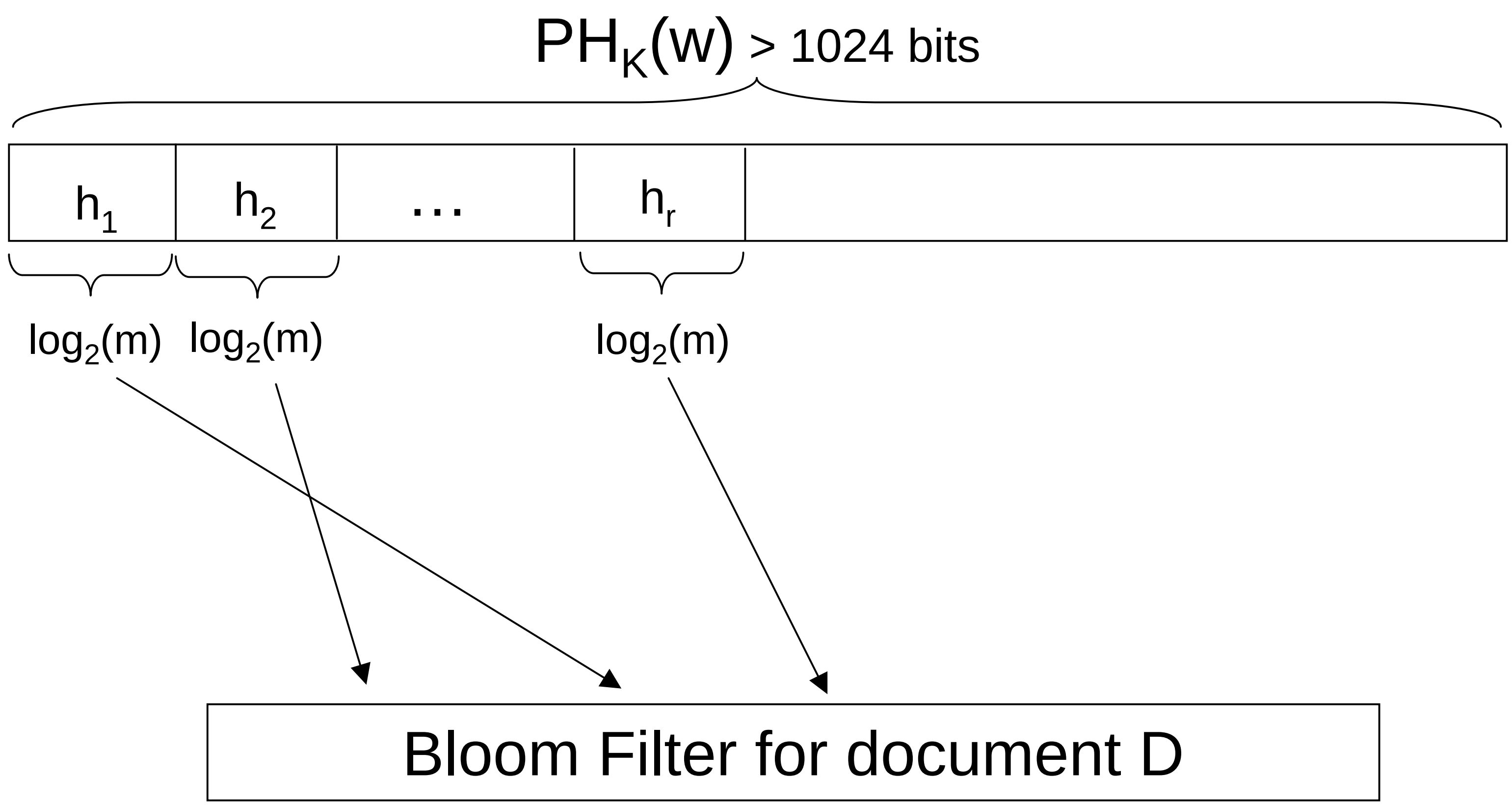
- Given $E_{k_A}(W)$, Ted can compute

$$E_{r_{A,B}}(E_{k_A}(W)) = E_{r_{A,B} \circ k_A}(W) = E_{k_B}(W)$$

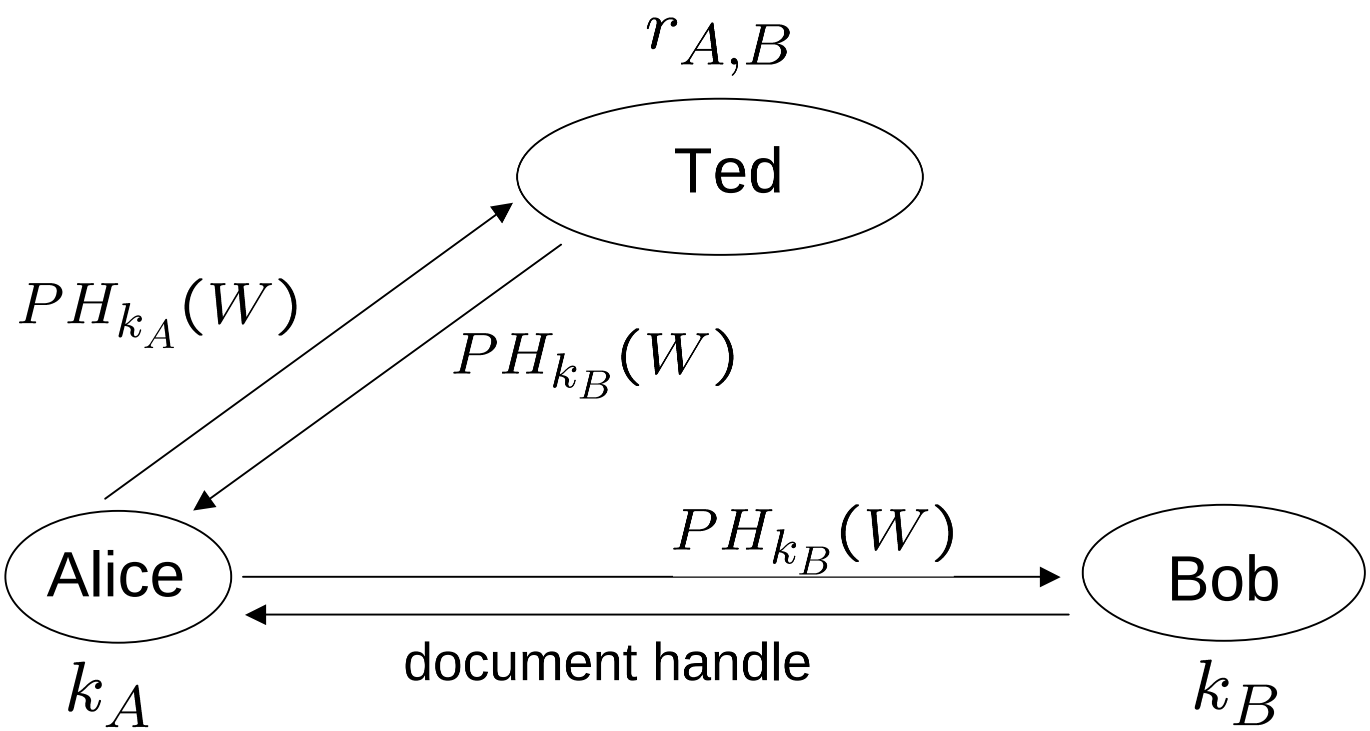
Group Ciphers as Hash Functions

- Pohlig-Hellman encryption
  $PH_k(X) = X^k \bmod p$
- Decrypt using d , such that $kd \equiv 1 \bmod (p-1)$
- Since p > 1024 bits, use output of encryption as hash function
- Bob computes encrypted Bloom filters:
  - For each document D
    - For each word W in D
      - Compute $PH_{k_B}(W)$ and use chunks of $\lceil \log_2 m \rceil$ of it as hash functions to insert into Bloom filter for document D

Group Ciphers as Hash Functions



The Basic Scheme - revisited

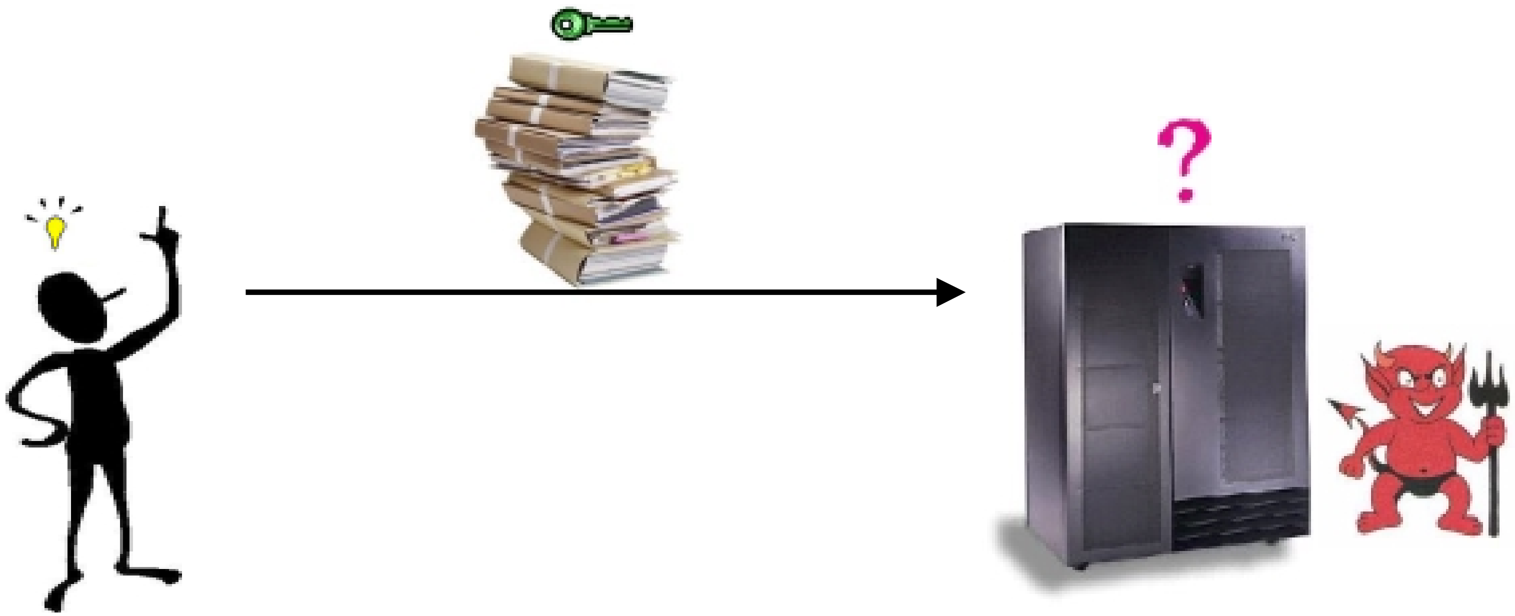


Bob uses $PH_{k_B}(W)$
to query the Bloom filter
of each document in the DB

Model #2

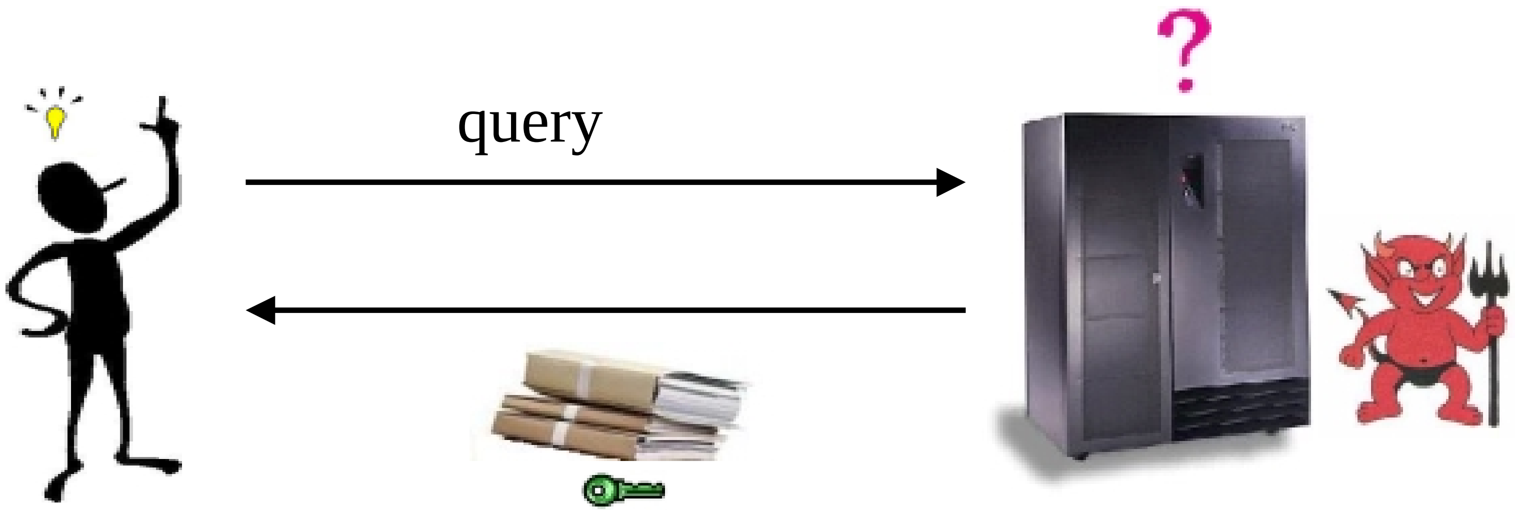
- Eu-Jin Goh, "Secure Indexes"

User submits data



User retrieves data

honest-but-curious adversary



user wants to preserve her privacy:
leak as little information as possible

Previous work

- [Song,Wagner,Perrig - 2000]
  - Query isolation
  - Controlled searching
  - Hidden queries

- Additional property:
  - Hide data access pattern

Private indexes

- Index is an additional structure that allows the remote server to perform searches efficiently
- Computed over unencrypted documents
- Private index should preserve user's privacy

Secure Indexes

- Indexes associated with each document
- Security model: IND-CKA

  (a secure index does not reveal anything about the a document’s content)
- Security game:

  given two encrypted documents of equal size, and an index, decide which document is encoded in the index

Secure Indexes

- An index is a Bloom filter, with pseudorandom functions used as hash functions
- A collection of 4 algorithms:
  - Keygen(s)
  - Trapdoor(K_{priv},w)
  - BuildIndex(D,K_{priv})
  - SearchIndex(T_w,I_D)
- Keygen generates:
  - pseudo-random function f
  - master key $K_{priv}=(k_1,\ldots,k_r)$

BuildIndex

- For each word w in document D_{id}:
  - Phase 1: compute trapdoor for w:

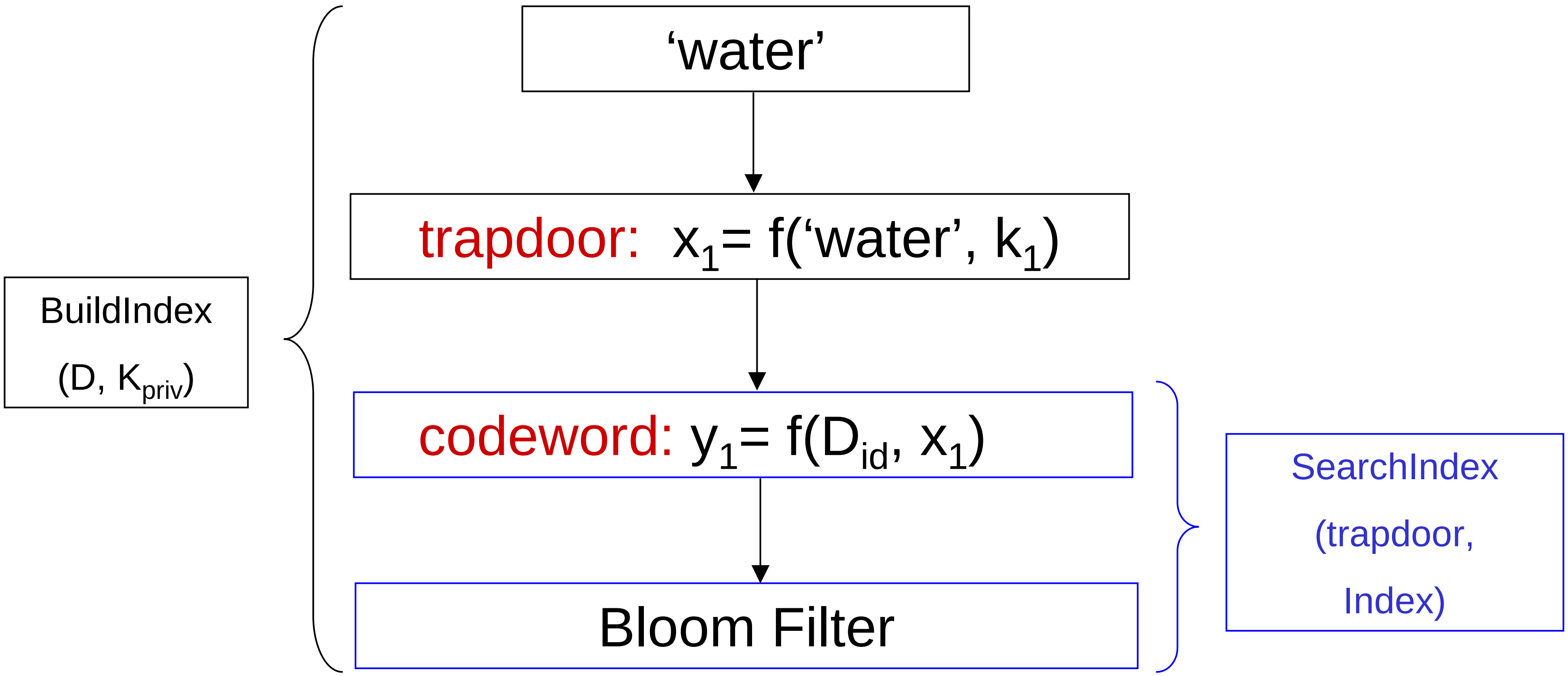
$$T_w = (x_1 = f(w, k1), ..., x_r = f(w, k_r))$$

  - Phase 2: compute codeword for w:

$$C_w = (y_1 = f(D_{id}, x1), ..., y_r = f(D_{id}, x_r))$$

  - insert codeword into document's Bloom filter

Secure Index usage



Achieving IND-CKA

- But, not enough to achieve IND-CKA:
  - Adversary can win game easily
- Solution:
  - u = upper bound on the number of words in D_{id}
  - v = number of distinct words in D_{id}
  - insert into index (u-v) random words
- But:
  - u is computed relative to the encrypted document
  - requires encryption of documents before building the index

Observations

- IND-CKA security requires “hidden queries” property, although not stated specifically

- IND-CKA2 security
  - stronger: indexes for documents with different number of keywords cannot be distinguished
  - more inefficient to obtain: need to use a global upper bound of number of words for all documents

Occurrence Search

- Allows questions like:

  "does 'word' appear at least n times?"

- Treat occurrences of same word as different words when building the index:

$$T_w = (x_1 = f(z_i || w, k1), ..., x_r = f(z_i || w, k_r))$$

where z_i is the number of times 'word' occurred so far in the document

Boolean queries

- Perform “AND” and ~~“OR”~~ queries

- Only as secure as performing individual queries for each term
- Can be done in a single pass:
  - ‘water’ AND ‘sky’
  - combine codewords for ‘water’ and ‘sky’
  - search the index

Implementation

- HMAC-SHA1 as PRFs
- FP = 2^{-10} $\rightarrow$ r = 10 (PR functions)
  (since $FP = (\frac{1}{2})^r$)
- *Claim*: search 15,151 indexes / sec on PIII 866 Mhz

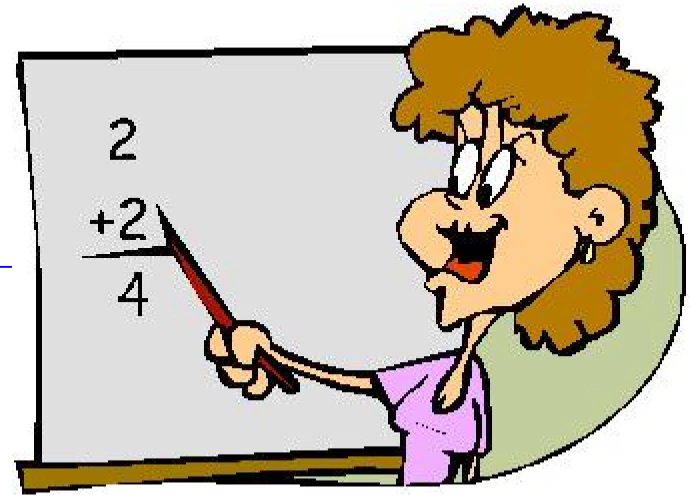
$1 + 1 \neq 2$

- Largest document
  - 876.6 Kbytes (plaintext or encrypted?)
  - contains 72,982 words (distinct or not?)
  - index is 774.3 Kbytes (difference encoded?)

- Choose BF parameters:

$$m = nr / \ln 2$$

Conclusions



- Computational complexity

  O(N)

- Communicational complexity

  1 round

- Drawbacks:
  - Bloom filters result in false positives
  - Updating procedure lacks security analysis
  - Security model not satisfactory for boolean searches
  - Unclear experimental evaluation