Searchable symmetric encryption (SSE)

Tom Ristenpart
CS 6431

Outsourced storage settings

Dropbox

Client wants to store data up on Dropbox

- High availability, synch across devices
- Server includes much value-add functionality

  Keyword search (find all files with “Tom” in text)

  Deduplication (if two files same, store only one copy)

  Thumbnail generation for images

Plaintext keyword search

Email client

Keyword stemming

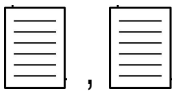
The attached contract is ready for signature. Please print 2 documents and have Atmos …

Keyword	Documents
contract	1, 7
signatur	8, 9, 1, 15, 200

Email storage provider

Upload documents

Search: "contract"



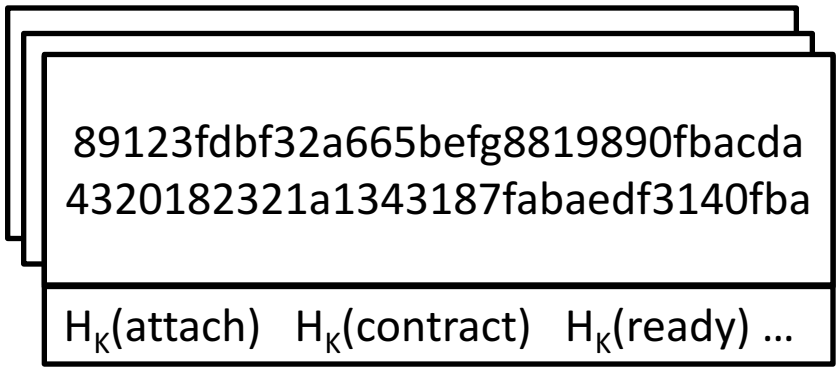
Standard encryption

Encryption prevents keyword search on server side.
Must store an index on client

Appended-PRF Searchable Encryption

Email client

Encrypt plaintext & keyed hash of keywords



Upload encrypted documents

Keyword	Documents
H_K(contract)	1, 7
H_K(signatur)	8, 9, 1, 15, 200

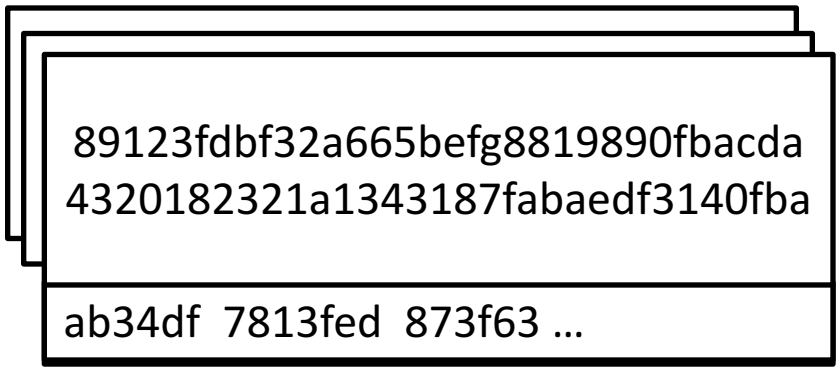
Email storage provider

Appended-PRF Searchable Encryption

Email client

Encrypt plaintext & keyed hash of keywords

7813fed = H_K(contract)



Keyword	Documents
7813fed	1, 7
456abc3	8, 9, 1, 15, 200

Email storage provider

Upload encrypted documents →

Search: "7813fed" →

← ,

Legacy compatible:
Works with existing plaintext storage interfaces

Two more schemes to consider

(2) Unordered appended-PRFs

Randomize order of PRF values

The attached contract is ready for signature. Please print 2 documents and have Atmos …
H_K(contract) H_K(ready) H_K(attach) …

(3) Encrypted index

Keyword	Documents
H_K(contract)	
H_K(signatur)	

Encrypt each document list under keyword-specific key

Encrypted index schemes

Email client

89123fdbf32a665befg8819890fbacda
4320182321a1343187fabaedf3140fba

Encrypt plaintext & keyed hash of keywords

7813fed = H_K(1||contract1)
K1' = H_K(0||contract)

Keyword	Documents
H_K(1||contract)	$E_{K1'}$(1,7)
H_K(1||signatur)	$E_{K2'}$(8,9,1,15,200)

Email storage provider

Upload encrypted documents →

Search: "7813fed", K1' →

← ,

What still leaks to the server?

Searchable symmetric encryption

Email client

Keyword	Documents
89123fdb	f32a665befg881
5befg8819	1a1343188gf781

Email storage provider

Upload encrypted documents →

Search: "7813fed", K_1 →

← ,

Informally, want to leak as little information ***as possible***:

- Obfuscated co-occurrence matrix
- Total number of documents
- Document lengths

Cash et al. basic construction

Problem with the simple encryption of document ID lists:

- Leaks number of documents associated to (some) keyword

Keyword	Documents
contract	1, 7
signatur	8, 9, 1, 15, 200

Keyword	Documents
H_K(contract)	
H_K(signatur)	

<u>Setup(DB):</u>

```
K <- {0,1}                    // pick a key K
For each w in DB:             // all keywords in DB
    Ki || Ki'  <- H_K(w)
    c <- 0
    For id in DB(w):          // all docs contain w
        L <- H_Ki(c)
        d <- E_Ki'(id)
        Put(L,d) in dictionary
        c++
Return dictionary
```

Key	Value
H_{K1}(1 || contract)	$E_{K1'}$(1)
H_{K2}(5 || signatur)	$E_{K2'}$(200)
H_{K1}(2 || contract)	$E_{K1'}$(7)
H_{K2}(2 || signatur)	$E_{K2'}$(9)

Cash et al. basic construction

Problem with the simple encryption of document ID lists:

- Leaks number of documents associated to (some) keyword

Keyword	Documents
contract	1, 7
signatur	8, 9, 1, 15, 200

Keyword	Documents
H_K(contract)	
H_K(signatur)	

<u>Client Search(K,w):</u>

Ki || Ki' <- H_K(w)

Send Ki, Ki' to server

<u>Server Search(Ki,Ki'):</u>

c <- 0

While d ≠ ⊥ :

    d <- Get(H_{ki}(c))

    Add $Dec_{Ki'}$(d) to IDs

Return IDs

Key	Value
H_{K1}(1 || contract)	$E_{K1'}$(1)
H_{K2}(5 || signatur)	$E_{K2'}$(200)
H_{K1}(2 || contract)	$E_{K1'}$(7)
H_{K2}(2 || signatur)	$E_{K2'}$(9)
…	…

[Curtmola et al. 2006]
basic scheme

$\mathsf{Gen}(1^k)$: sample $K_1, K_2, K_3 \xleftarrow{\$} \{0,1\}^k$, generate $K_4 \leftarrow \mathsf{SKE2.Gen}(1^k)$ and output $K = (K_1, K_2, K_3, K_4)$.

$\mathsf{Enc}_K(\mathbf{D})$:

Initialization:

1. scan $\mathbf{D}$ and generate the set of distinct keywords $\delta(\mathbf{D})$
2. for all $w \in \delta(\mathbf{D})$, generate $\mathbf{D}(w)$
3. initialize a global counter $\mathsf{ctr} = 1$

Building the array $\mathtt{A}$:

4. for $1 \le i \le |\delta(\mathbf{D})|$, build a list $\mathtt{L}_i$ with nodes $\mathtt{N}_{i,j}$ and store it in array $\mathtt{A}$ as follows:
   (a) sample a key $K_{i,0} \xleftarrow{\$} \{0,1\}^k$
   (b) for $1 \le j \le |\mathbf{D}(w_i)| - 1$:
   - let $\mathsf{id}(D_{i,j})$ be the jth identifier in $\mathbf{D}(w_i)$
   - generate a key $K_{i,j} \leftarrow \mathsf{SKE1.Gen}(1^k)$
   - create a node $\mathtt{N}_{i,j} = \langle \mathsf{id}(D_{i,j}) \| K_{i,j} \| \psi_{K_1}(\mathsf{ctr}+1) \rangle$
   - encrypt node $\mathtt{N}_{i,j}$ under key $K_{i,j-1}$ and store it in $\mathtt{A}$:

$$\mathtt{A}[\psi_{K_1}(\mathsf{ctr})] \leftarrow \mathsf{SKE1.Enc}_{K_{i,j-1}}(\mathtt{N}_{i,j})$$

   - set $\mathsf{ctr} = \mathsf{ctr} + 1$

   (c) for the last node of $\mathtt{L}_i$,
   - set the address of the next node to NULL: $\mathtt{N}_{i,|\mathbf{D}(w_i)|} = \langle \mathsf{id}(D_{i,|\mathbf{D}(w_i)|}) \| 0^k \| \mathsf{NULL} \rangle$
   - encrypt the node $\mathtt{N}_{i,|\mathbf{D}(w_i)|}$ under key $K_{i,|\mathbf{D}(w_i)|-1}$ and store it in $\mathtt{A}$:

$$\mathtt{A}[\psi_{K_1}(\mathsf{ctr})] \leftarrow \mathsf{SKE1.Enc}_{K_{i,|\mathbf{D}(w_i)|-1}}\left(\mathtt{N}_{i,|\mathbf{D}(w_i)|}\right)$$

   - set $\mathsf{ctr} = \mathsf{ctr} + 1$
5. let $s' = \sum_{w_i \in \delta(\mathbf{D})} |\mathbf{D}(w_i)|$. If $s' < s$, then set the remaining $s - s'$ entries of $\mathtt{A}$ to random values of the same size as the existing s' entries of $\mathtt{A}$

Building the look-up table $\mathtt{T}$:

6. for all $w_i \in \delta(\mathbf{D})$, set $\mathtt{T}[\pi_{K_3}(w_i)] = \langle \mathsf{addr}_{\mathtt{A}}(\mathtt{N}_{i,1}) || K_{i,0} \rangle \oplus f_{K_2}(w_i)$
7. if $|\delta(\mathbf{D})| < |\Delta|$, then set the remaining $|\Delta| - |\delta(\mathbf{D})|$ entries of $\mathtt{T}$ to random values of the same size as the existing $|\delta(\mathbf{D})|$ entries of $\mathtt{T}$

Preparing the output:

8. for $1 \le i \le n$, let $c_i \leftarrow \mathsf{SKE2.Enc}_{K_4}(D_i)$
9. output $(I, \mathbf{c})$, where $I = (\mathtt{A}, \mathtt{T})$ and $\mathbf{c} = (c_1, \ldots, c_n)$

$\mathsf{Trpdr}_K(w)$: output $t = (\pi_{K_3}(w), f_{K_2}(w))$

$\mathsf{Search}(I, t)$:

1. parse t as (γ, η), and set $\theta \leftarrow \mathtt{T}[\gamma]$
2. if $\theta \neq \perp$, then parse $\theta \oplus \eta$ as $\langle \alpha || K' \rangle$ and continue, otherwise return $\perp$
3. use the key K' to decrypt the list $\mathtt{L}$ starting with the node stored at address α in $\mathtt{A}$
4. output the list of document identifiers contained in $\mathtt{L}$

$\mathsf{Dec}_K(c_i)$: output $D_i \leftarrow \mathsf{SKE2.Dec}_{K_4}(c_i)$

Figure 1: A non-adaptively secure SSE scheme (SSE-1)

Evaluating security

Prove upper bound on information leaked:

- Assuming cryptographic components good, then nothing revealed beyond specified leakage model $\mathcal{L}$

Real

<u>Init(DB):</u>
(K,EDB) <- Setup(DB)
Return EDB

<u>Search(w):</u>
(Ki,Ki') <- F(K,w)
Return (Ki,Ki')

Ideal

<u>Init(DB):</u>
$(\alpha, st_L) \leftarrow \mathcal{L}(DB)$
$(EDB, st_S) \leftarrow \mathcal{S}(\alpha)$
Return EDB

<u>Search(w):</u>
$(\alpha, st_L) \leftarrow \mathcal{L}(st_L, w)$
$(Ki, Ki', st_S) \leftarrow \mathcal{S}(st_s, \alpha)$
Return (Ki,Ki')

Specify leakage $\mathcal{L}$:
$\mathcal{L}(DB)$ outputs total # of keywords in DB
$\mathcal{L}(st_L, w)$ outputs DB(w), which other queries were on w

Simulator. Goal is to trick adversary into thinking this is real world

Scheme secure if exists efficient $\mathcal{S}$ s.t. no efficient adversary can distinguish Real vs Ideal

Cash et al. other results

- Improved HDD efficiency by batching docs
- Ability to add / delete documents
- Reported on implementation results for large document corpuses

Qualitative comparison of schemes

Appended-PRF scheme used in industry

Unordered appended-PRF used in research literature

Mimesis Aegis [Lau et al. 2014]
ShadowCrypt [He et al. 2014]

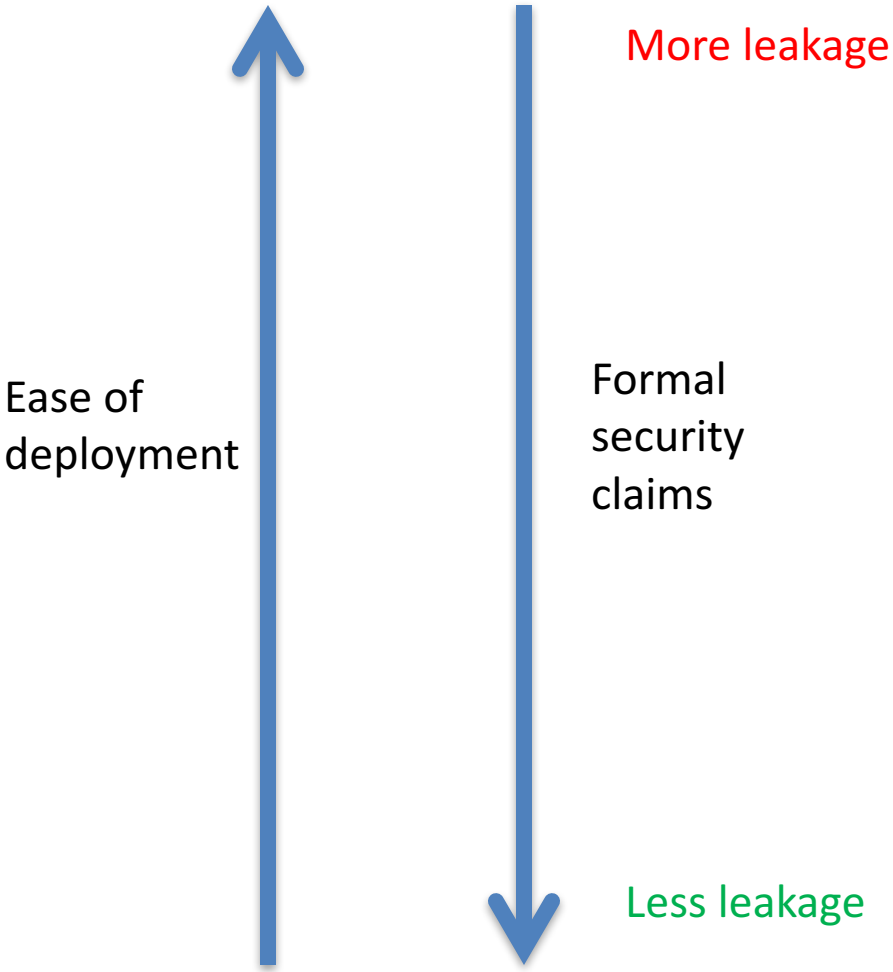
Encrypted index in literature & starting to appear in industry

Qualitative comparison of schemes

Appended-PRF scheme used in industry

Unordered appended-PRF used in research literature

Encrypted index in literature & starting to appear in industry



Leakage-abuse attacks

[Islam, Kuzu, Kantarcioglu – 2013]
[Cash, Grubbs, Perry, R. – 2015]

All searchable encryption leaks information about plaintexts and queries. Appended-PRF case:

Keyword	Documents
H_K(contract)	1, 7
H_K(signatur)	8, 9, 1, 15, 200

H_K(attach) H_K(contract) H_K(ready) …

Upload encrypted documents →

Search: "H_K(contract)" →

Adversarial storage provider

Leakage-abuse attacks

[Islam, Kuzu, Kantarcioglu – 2013]
[Cash, Grubbs, Perry, R. – 2015]

All searchable encryption leaks information about plaintexts and queries. Appended-PRF case:

"Keyword 7813fed came second in Document 1"
(Keyword location)

ab34df 7813fed 873f63 …

Upload encrypted documents

Search: "7813fed"

"Keyword 7813fed searched often"
(Search frequency)

"Document 1 and 7 both contain 7813fed" (Co-occurrence relationships)

Keyword	Documents
7813fed	1, 7
456abc3	8, 9, 1, 15, 200

Adversarial storage provider

Unordered appended-PRF: order of keywords not leaked

Encrypted index: order of keywords not leaked & leakage only after queries made

A lot of basic security questions:

- Does leakage damage confidentiality?
- How much more security does one achieve via more complex schemes?
- What adversarial capabilities are likely to arise in practice?

Leakage-abuse attack taxonomy

Attacker goal	Query recovery	
	Plaintext recovery	
Attacker capabilities	Passive	Observe queries and stored ciphertexts
	Active	Force insertion of documents and/or queries
Document knowledge	Full	Know all plaintexts exactly
	Partial	Know some plaintexts
	Distributional	Know similar plaintexts
	Support	Know support of distribution

IKK 2013 against encrypted index: Query recovery | Passive | Full

Simulations with Enron email corpus: 80% of queries recoverable

Case studies of three attacks

1. Simple attack against *appended-PRF*

Plaintext recovery	
Passive	Partial

2. Query recovery against *encrypted index schemes*

Query recovery	
Passive	Distributional

3. Chosen-email attack against *unordered appended-PRF*

Plaintext recovery	
Active	Support

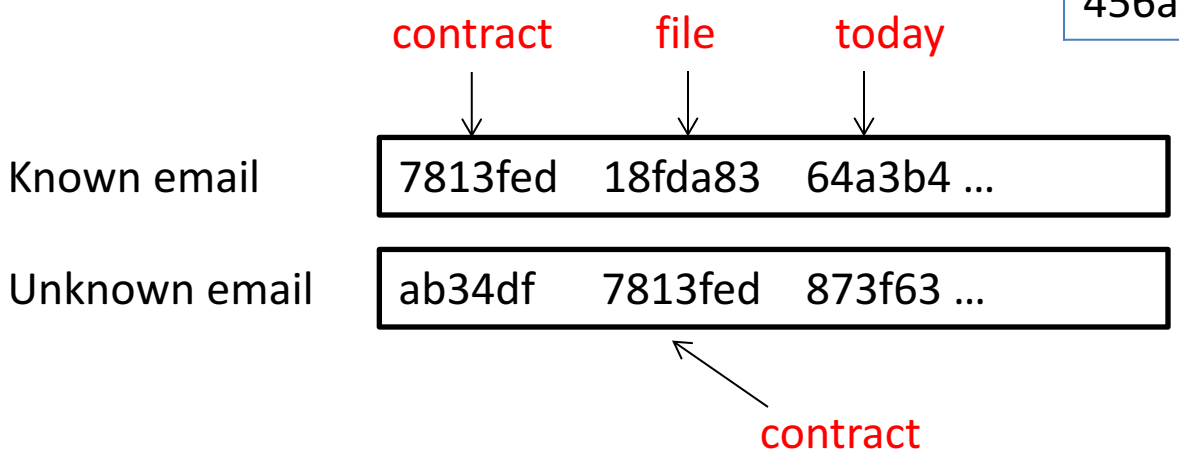
Partial plaintext recovery against appended-PRF

[Cash, Grubbs, Perry, R. – 2015]

Plaintext recovery

Passive | Partial

Keyword	Documents
7813fed	1, 7
456abc3	8, 9, 1, 15, 200



Adversarial storage provider

Partial plaintext recovery against appended-PRF

[Cash, Grubbs, Perry, R. – 2015]

Plaintext recovery

Passive | Partial

Simulations with Enron email corpus

- 30,109 emails from employee sent_mail folders
- Adversary knows 20 random emails (0.06%)
- Simply match keywords in known emails to unknown

Unknown email plaintext

```
 The attached contract is ready for signature.
Please print 2 documents and have Atmos execute
both and return same to my attention.  I will re-
turn an original for their records after ENA has
signed.  Or if you prefer, please provide me with
the name / phone # / address of your customer and
I will Fed X the Agreement.
```

Recovered information

```
attach contract signatur pleas print 2 document
have execut both same will origin ena sign prefer
provid name agreement
```

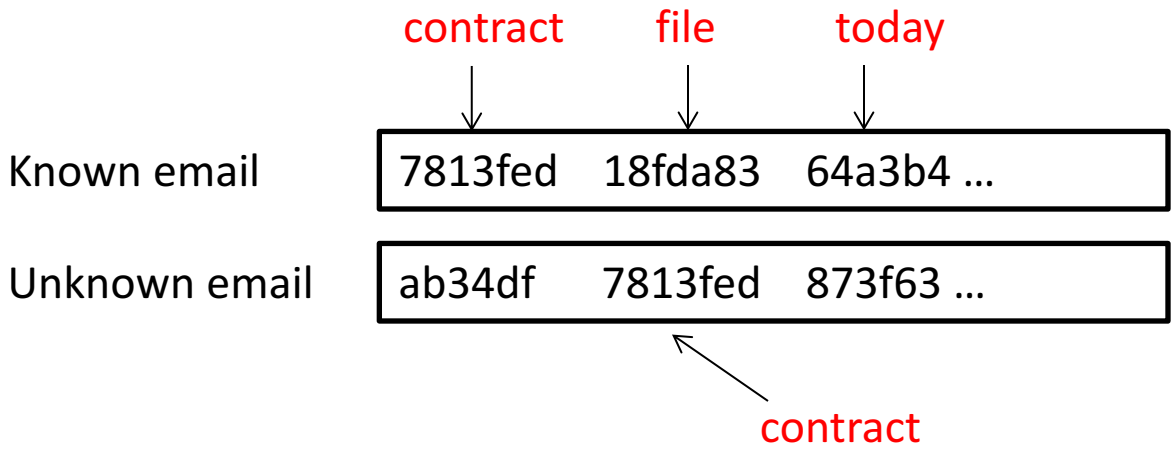
Randomizing hash order

Plaintext recovery

Passive | Partial

Leaving hashes in document order makes attack easy

Simple change: randomize order of hashes to leak less information (sort by hash value)



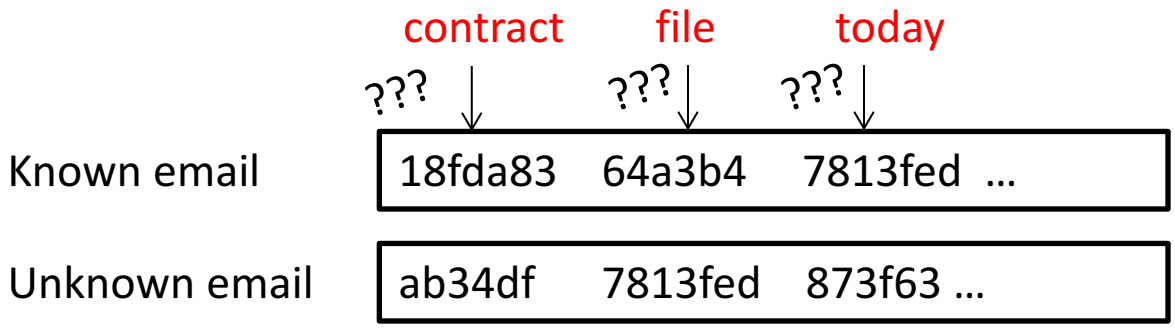
Randomizing hash order

Plaintext recovery

Passive | Partial

Leaving hashes in document order makes attack easy

Simple change: randomize order of hashes to leak less information
(sort by hash value)



Order issue left implicit in prior work

Mimesis Aegis: randomizes order due to Bloom filter

ShadowCrypt: implementation randomizes order, paper does not discuss

IKK query recovery attack

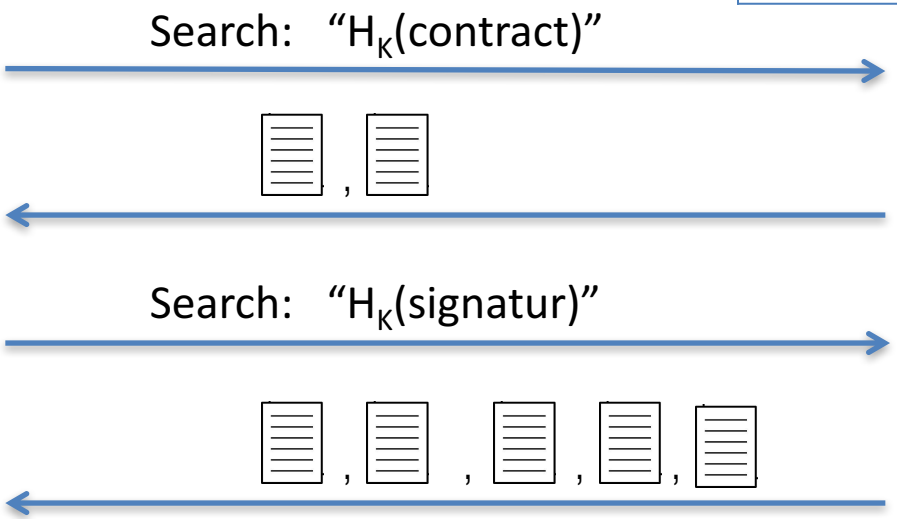
Query recovery	
Passive	Full

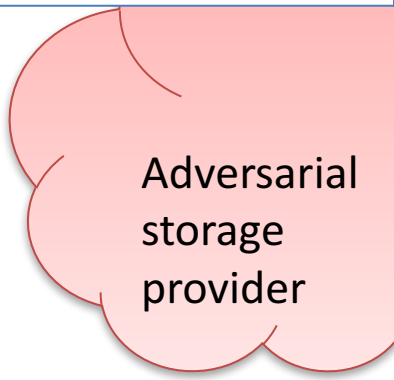
Adversary knows ***full plaintext corpus***
Goal is to uncover search query keywords used by client

Keyword	Documents
H_K(contract)	1, 7
H_K(signatur)	8, 9, 1, 15, 200

Email client

Uniformly selects keywords to search





IKK detail expensive attack using simulated annealing to solve NP-complete problem sufficient to reveal queries

Cash et al. 2016 count attack

Query recovery	
Passive	Full

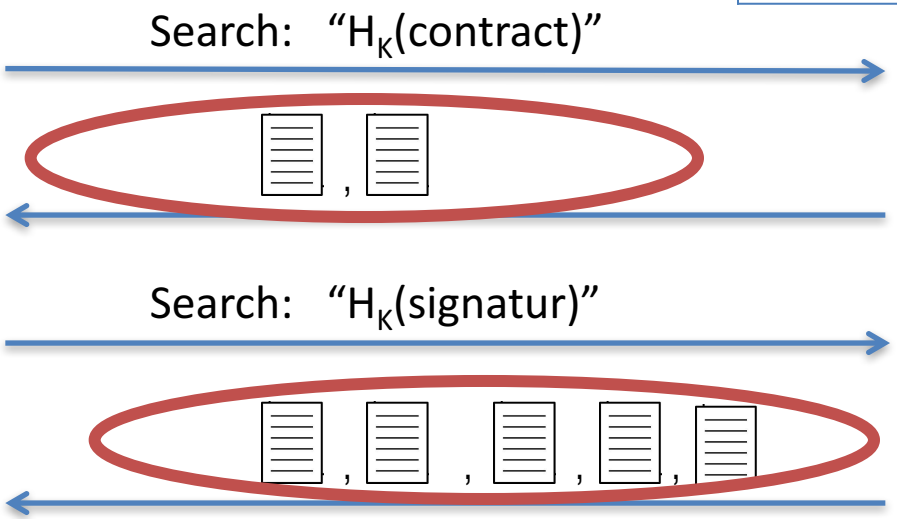
Adversary knows ***full plaintext corpus***

Goal is to uncover search query keywords used by client

Keyword	Documents
H_K(contract)	1, 7
H_K(signatur)	8, 9, 1, 15, 200

Email client

Uniformly selects keywords to search



Adversarial storage provider

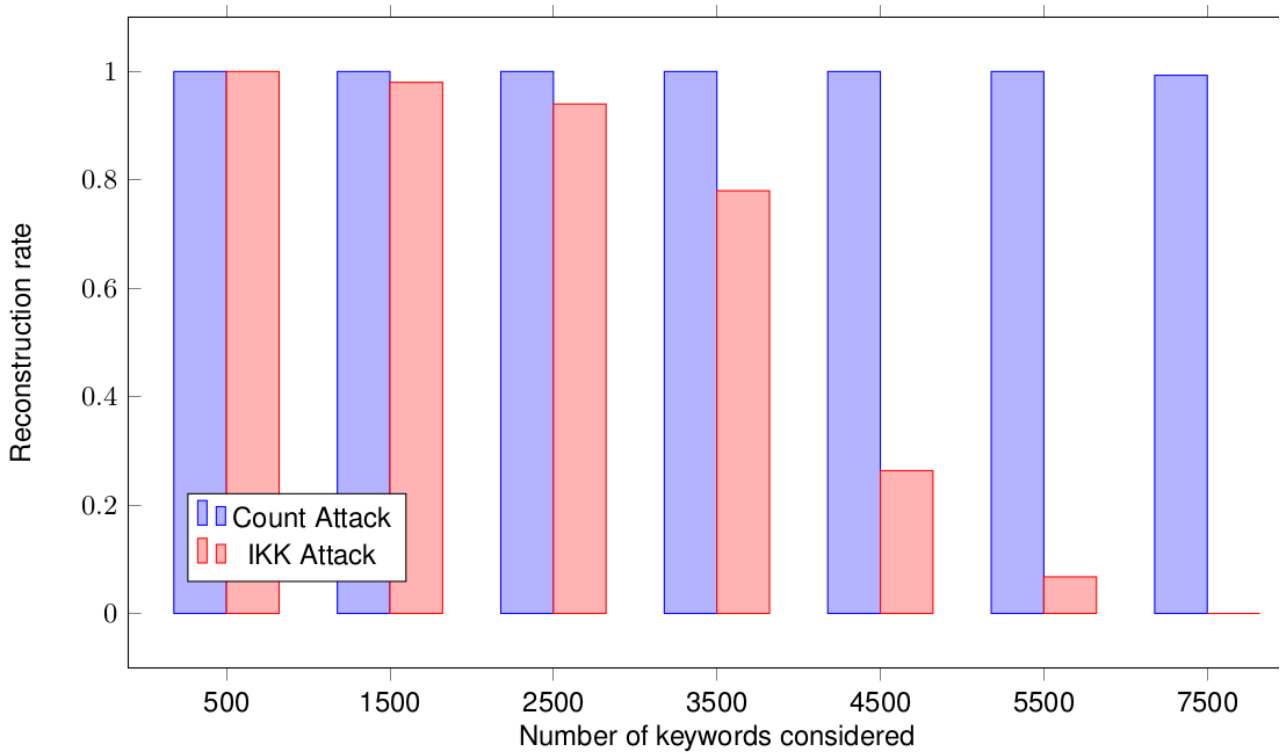
Attacker sees number of documents returned

Many keywords appear in a unique number of documents

Disambiguate with co-occurrence relationships

IKK vs “count” attack

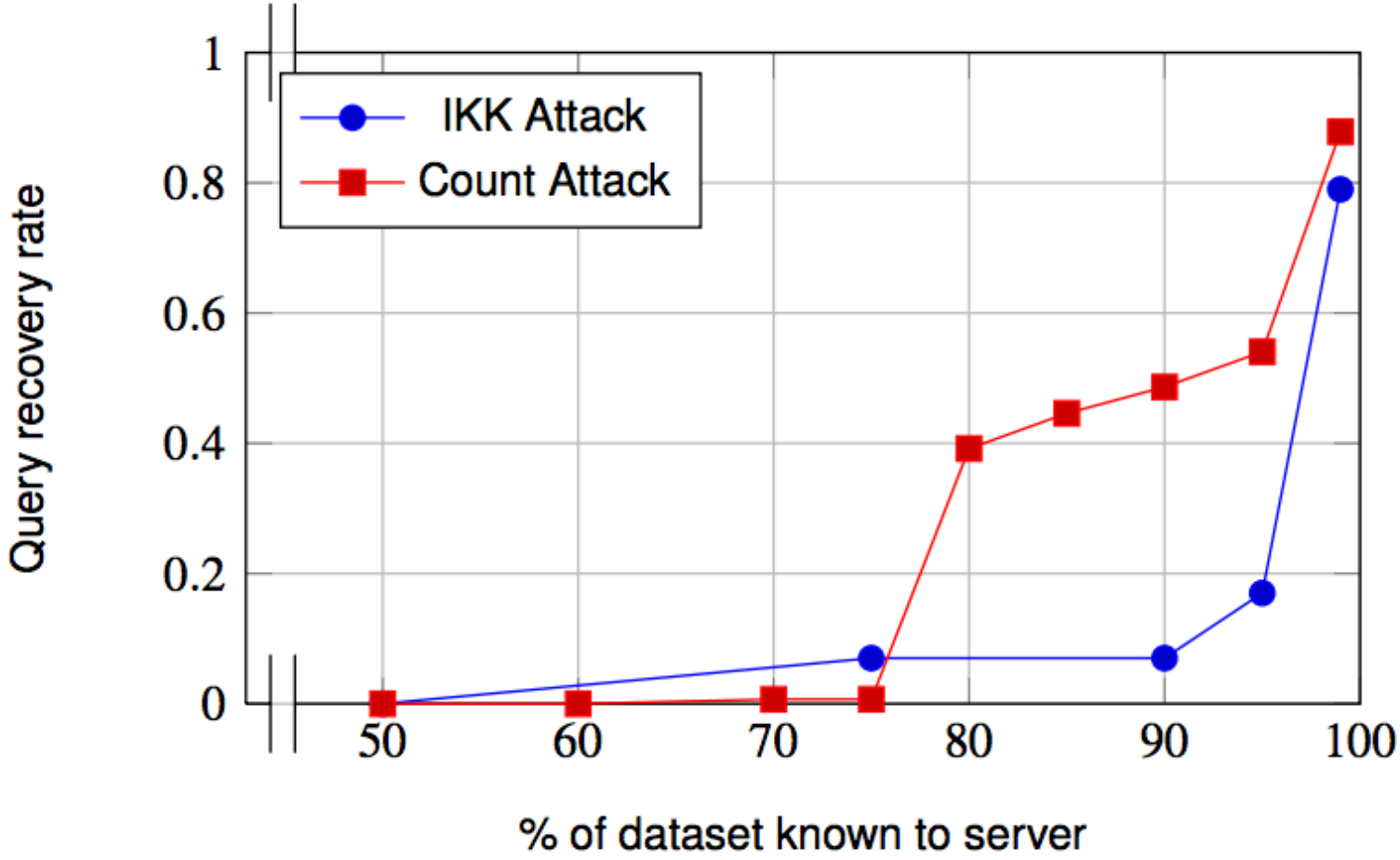
Query recovery	
Passive	Full



Subset of Enron emails (known to attacker)
Most popular x keywords considered
10% of keywords uniformly sampled and queried

Count attack with partial knowledge

Query recovery	
Passive	Partial



Chosen-email attacks (aka file-injection attacks)

Query recovery	
Active	Support

Attacker can plant chosen emails into client's inbox

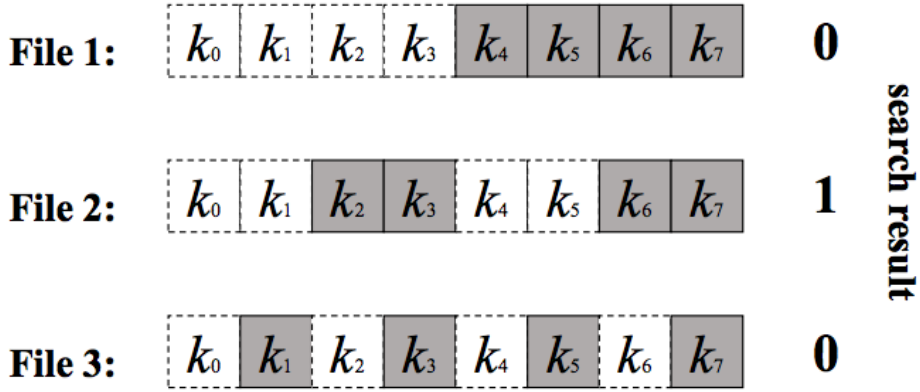
Email client

Insert new emails →

Search: "$H_K(k_5)$" →

Keyword	Documents
$H_K(k_0)$	
$H_K(k_5)$	
$H_K(k_2)$	
…	…

Adversarial storage provider



[Zhang et al. 2016]

Say all keywords = $\{k_0,k_1,\ldots,k_7\}$
Send three emails that include just shaded keywords

Case studies of three attacks

1. Simple attack against *appended-PRF*

Plaintext recovery	
Passive	Partial

2. Query recovery against *encrypted index schemes*

Query recovery	
Passive	Distributional

3. Chosen-email attack against *unordered appended-PRF*

Plaintext recovery	
Active	Support

Summary of leakage-abuse attacks

Provable security must be (at least) paired with empirical security analyses

Lots of open questions:

- Leakage of richer queries
- Role of updates
- Effect of re-encryption
- Exploitability of active attacks in practice

And challenges:

- Better data sets for simulations
- Query traces
- ***Countermeasures***