CS2: A Searchable Cryptographic Cloud Storage System

Seny Kamara (MSR)
Charalampos Papamanthou (UC Berkeley)
Tom Roeder (MSR)
Cloud Computing
Cloud Computing

- **Main concern**
  - will my data be safe?
  - will anyone see it?
  - can anyone modify it?

- **Security solutions**
  - VM isolation
  - Single-tenant servers
  - Access control
  - ...

- Cloud provides *stronger* security than self-hosting \[\text{[Molnar-Schecter-10]}\]

- **Q**: but what if I don’t trust the *cloud operator*?
Cloud Storage
Traditional Approach
Search-based Access

- File-based access is hard (esp. for large data)
- Search-based access is preferred
  - Web search
  - Desktop search
    - Apple Spotlight, Google Desktop, Windows Desktop
  - Enterprise search
Two Simple Solutions to Search

Q: can we achieve the best of both?
Outline

- Motivation
- CS2 building blocks
  - Symmetric searchable encryption
  - Search authenticators
  - Proofs of storage
- CS2 Protocols
  - for standard search
  - for assisted search
- Experiments
CS2 Building Blocks
Searchable Symmetric Encryption [SWP01]

$\text{Enc}_K$
Searchable Symmetric Encryption

- [Goldreich-Ostrovsky-96]
  - 😊: hides everything
  - 😞: interactive

- [Song-Wagner-Perrig-01]
  - 😊: non-interactive
  - 😞: static, linear search time, leaks information

- [Goh03, Chang-Mitzenmacher-05]
  - 😊: non-interactive, dynamic
  - 😞: linear search time, non-adaptive security (CKA1-security)

- [Curtmola-Garay-K-Ostrovsky-06]
  - 😊: non-interactive, sub-linear search (optimal), adaptive security
  - 😞: static
Proofs of Storage \[\text{[ABC+07, JK07]}\]
Proofs of Storage

- [ABC+07, JK07, SW08, DVW09, AKK09]
  - ☺: efficient
  - ☹: static

- [APMT08]
  - ☺: efficient and dynamic
  - ☹: bounded verifications

- [EKPT09]
  - ☺: efficient, dynamic, unlimited verification
  - ☹: patented

We need new PoS!
Search Authenticicator

\[ W \quad \pi \]
Search Authenticators

- [GGP10,CVK10,CVK11]
  - ☻: general-purpose
  - ☹: inefficient (due to FHE) & static
- [CRR11]
  - ☻: general-purpose, efficient
  - ☹: requires two non-colluding clouds
- [BGV11]
  - ☻: proof generation is linear & static

We need new VC/SA!
Outline

- Motivation
- CS2 building blocks
  - Symmetric searchable encryption
  - Search authenticators
  - Proofs of storage
- CS2 Protocols
  - for standard search
  - for assisted search
- Experiments
SSE-1 [CGKO06]

1. Build inverted/reverse index

Posting list

2. Randomly permute array & nodes
SSE-1 [CGKO06]

2. Randomly permute array & nodes

3. Encrypt nodes
3. Encrypt nodes

4. “Hash” keyword & encrypt pointer

\[
\begin{array}{c|c|c|c|c|}
F_K(GOOG) & Enc(\bullet) \\
F_K(IBM) & Enc(\bullet) \\
F_K(AAPL) & Enc(\bullet) \\
F_K(MSFT) & Enc(\bullet) \\
\end{array}
\]
Limitations of SSE-1

- Non-adaptively secure $\Rightarrow$ adaptive security
  - Idea #1 [Chase-K-10]
    - replace encryption scheme with symmetric non-committing encryption
    - only requires a PRF + XOR
    - $\oplus$: doesn’t work for dynamic data
  - Idea #2
    - Use RO + XOR
Limitations of SSE-1

- Static data $\Rightarrow$ dynamic data
  - Problem #1:
    - given new file $F_N = (AAPL, \ldots, MSFT)$
    - append node for $F$ to list of every $w_i$ in $F$

1. Over unencrypted index

2. Over encrypted index ???
Limitations of SSE-1

- Static data $\Rightarrow$ dynamic data
  - Problem #2:
    - When deleting a file $F_2 = (AAPL, \ldots, MSFT)$
    - delete all nodes for $F_2$ in every list

1. Over unencrypted index

2. Over encrypted index ???

<table>
<thead>
<tr>
<th>Static data</th>
<th>Dynamic data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem #2</td>
<td></td>
</tr>
<tr>
<td>When deleting a file $F_2 = (AAPL, \ldots, MSFT)$</td>
<td>delete all nodes for $F_2$ in every list</td>
</tr>
</tbody>
</table>
Limitations of SSE-1

- Static data $\Rightarrow$ dynamic data
  - Idea #1
    - Memory management over encrypted data
    - Encrypted free list
  - Idea #2
    - List manipulation over encrypted data
    - Use homomorphic encryption (here just XOR) so that pointers can be updated obliviously
  - Idea #3
    - Deletion is handled using an “dual” SSE scheme
      - Given deletion/search token for $F_2$, returns pointers to $F_2$'s nodes
      - Then add them to the free list homomorphically
Outline

- Motivation
- Related work & our approach
- CS2 building blocks
  - Symmetric searchable encryption
  - Search authenticators
  - Proofs of storage
- CS2 Protocols
  - for standard search
  - for assisted search
- Experiments
Limitations of Verifiable Computation

- Inefficient $\Rightarrow$ practical
  - Idea #1
    - Design special-purpose scheme (i.e., just for verifying search)
  - Idea #2
    - Use Merkle Tree “on top” of inverted index
    - For keyword $w$: we efficiently verify its posting list and associated files
    - Generating proof is $O(w^*)$ instead of $O(n)$

- Static $\Rightarrow$ dynamic
  - Idea #1
    - Replace bottom hash with *incremental* hash
    - [Bellare-Goldreich-Goldwasser94, Bellare-Micciandio97]
Search Authenticators

1. Build inverted/reverse index

2. Build Merkle tree w/ IH at leaves

Problem: hash functions are not hiding!
Search Authenticators

2’. Build Merkle tree w/ IH at leaves over encrypted files

Problem: server has file encryptions so he can
1. IH a set of files
2. check result against a leaf hash
3. determine if files contain common keyword
Search Authenticators

2''. Build Merkle tree w/ IH at leaves over keyed hash of encrypted files

Problem: server has file encryptions so he can
1. IH a set of files
2. check result against a leaf hash
3. determine if files contain common keyword
Proofs of Storage
CS2 Protocols
CS2 Protocols

- **Standard search**
  - User searches for \( w \)
  - Server returns documents with \( w \)
  - Relatively straightforward combination of (dynamic) SSE, PoS & SA

- **Assisted search**
  - User searches for \( w \)
  - Server returns summaries of files with \( w \)
  - User chooses a subset to retrieve
  - Server returns subset of files with \( w \)
  - More complex combination of (dynamic) SSE, PoS, SA + CRHF
  - *Search can be more efficient* (since less data is returned)
CS2 Protocols

- Definitions in ideal/real-world model
  - Cloud storage w/ standard search
  - Cloud storage w/ assisted search
  - 😊
    - easier to use within larger protocols (i.e., hybrid security models)
    - Single definition for all desired properties
    - guarantees composition of underlying primitives is OK
  - 😞: definitions & proofs are complicated

- Protocols make black-box use of primitives
  - 😊: modularity -- replace underlying primitives
Experiments
Implementation

- C++
- Microsoft Cryptography API: Next Generation
  - RO: SHA256
  - PRFs: HMAC-SHA256
  - SKE: 128-bit AES/CBC
- Bignum library
  - Prime fields
- We test only the cryptobyte overhead
  - No file transfers over network
  - No reading from disk
  - No indexing costs
Experiments

- Intel Xeon CPU 2.26 GHz
  - Windows Server 2008
- 4 datasets
  - Email (enron): 4MB, 11MB, 16MB
    - ≈ every byte is a word
  - Office docs: 8MB, 100MB, 250MB, 500MB
    - Relatively few keywords
  - Media (MP3, WMA, JPG, ...): 8MB, 100MB, 250MB, 500MB
    - Barely any keywords
- Average over 10 executions
STORE

- Total
  - Email (16MB): 2 mins
  - Office (500MB): 1.5 mins
  - Media (500MB): 30 s
  - Email (16GB): 40/15 hours

- Distribution
  - Verifiability: 2/3 of cost
  - SSE: 1/3 cost
  - PoS: negl
SEARCH

- **Total**
  - Email (16MB): 0.5 secs
  - Office (500MB): 0.1 secs
  - Media (500MB): 0.025 secs

- **Distribution**
  - Client verification: 80%
  - Client decryption: 10%
  - Server search + proof: 10%
CHECK

- Total
  - Email (16MB): 12 secs
  - Office (500MB): 12 secs
  - Media (500MB): 12 secs

- Distribution
  - Server Proof: 95%
  - Client verify: 5%
- **Total**
  - Email (16MB): 1.5 secs
  - Office (500MB): 1.5 secs
  - Media (500MB): 1.5 secs

- **Distribution**
  - Email (16MB)
    - 40% client auth state update
    - 40% server auth update
    - 20% add token
o Total
  o Email (16MB): 1.5 secs
  o Office (500MB): 0.7 secs
  o Media (500MB): neglig

o Distribution
  o 40% server auth update
  o 40% client auth update
  o 20% server index update
Summary

- **New Crypto**
  - Dynamic and CKA2-secure SSE with sub-linear search
  - Sub-linear verifiable computation for search
  - Unbounded dynamic PDP

- **New Protocols**
  - Ideal/real-world definitions for secure cloud storage
  - Protocol for standard search
  - Protocol for assisted search

- **Implementation & experiments**
  - First experimental results for sub-linear SSE
  - Identified verification as bottleneck
  - Office docs seem to be the best workload
Questions?