Percival: A Searchable Secret Split Datastore

Joel C. Frank
Shayna M. Frank, Lincoln A. Thurlow, Thomas M. Kroeger¹, Ethan L. Miller and Darrell D. E. Long

Center for Research in Storage Systems
University of California, Santa Cruz
Sandia National Laboratories, Livermore¹
Untrusted Environment

- Distributed Datastore
- Main Concern: Information Privacy
- Trust the whole, not the individual
- Secret split the data
  - N pieces, each the size of the original
  - T pieces required for reassembly \((1 < T \leq N)\)
  - Minimizes insider threat
  - No single point of failure
- Information-theoretically secure, but…
  - Either can’t search it, or
  - Need to reassemble prior to searching
- Pre-index
  - Current methods rely on fixed-key encryption
  - Not well suited for long-term storage
Percival

- **Goal:** To enable searching without the need for reassembly
- **Solution:** Store secret split pre-generated queries (reverse indexes)
  - Query Servers: key, value store (hash :: secret-split reverse index)
  - Clients retrieve reverse index shares using a custom hash
  - Reassembled Query: maps search term to data share(s)
- **Result**
  - Secure and searchable data store
  - Aids in information sharing
  - Assumes insider threat
    - Single repository
    - No collusion between attackers
Goal: To enable searching without the need for reassembly

Solution: Store secret split pre-generated queries (reverse indexes)
  - Query Servers: key, value store (hash :: secret-split reverse index)
  - Clients retrieve reverse index shares using a custom hash
  - Reassembled Query: maps search term to data share(s)

Result
  - Secure and searchable data store
  - Aids in information sharing
  - Assumes insider threat
    - Single repository
    - No collusion between attackers

POTSHARDS, Cleversafe, etc…
Percival

- Goal: To enable searching without the need for reassembly
- Solution: Store secret split pre-generated queries (reverse indexes)
  - Query Servers: key, value store (hash :: secret-split reverse index)
  - Clients retrieve reverse index shares using a custom hash
  - Reassembled Query: maps search term to data share(s)
- Result
  - Secure and searchable data store
  - Aids in information sharing
  - Assumes insider threat
    - Single repository
    - No collusion between attackers

# of Query Servers ≠ # of Data Servers
Splitting Scheme Flexibility

### High Threshold
- Example: \((n - 2)\) shares
- More shares needed for reconstruction
- Higher barrier to compromise
- Denial of Service attack
- Ransomware

### Low Threshold
- Example: 2 shares
- Fewer shares needed for reconstruction
- Lower barrier to compromise
- Improves data availability
Ingestion: Reverse Indexes

- Identify the keywords for each piece of data
  - Choose top 10, 20, etc… keywords
  - Security is based on there being millions of files stored (more on that later)
- Generate reverse indexes: each reverse index is a query result
- Secret split each query
Ingestion: Query Server

Keywords are never visible to the custodian

Assumes secure comms (more on this later)
Hardware Security Module

- Physical device that:
  - Safeguards and manages keys
  - Provides crypto-processing
  - Has its own NIC
  - Can be a plug-in card or external device
- Provides tamper evidence and resistance
  - Logs suspected tamper attempts
  - Deletes its internal memory upon tampering
- Cost: Low bandwidth (~1.4MB/s)
- Easily parallelizable
- Secure channel between HSMs
Hardware Security Module

- Physical device that:
  - Safeguards and manages keys
  - Provides crypto-processing
  - Can be a plug-in card or external device

- Provides tamper evidence and resistance
  - Logs suspected tamper attempts
  - Deletes its internal memory upon tampering

- Cost: Low bandwidth (~1.4MB/s)

- Easily parallelizable

- Secure channel between HSMs
Secret Salt

- Each query server has a unique salt, that is:
  - Each salt is:
    - Generated by its Hardware Security Module (HSM)
    - Never leaves the HSM
    - Write only. No interface to read the salt from the HSM
    - Ensures sibling query shares are stored with different hashes

- Targeted vs wholesale theft

- Theoretically possible to be brute forced, but:
  - Landauer limit: \( L = kT \ln(2) \)
  - 256 bit salt requires \( 8.9 \times 10^{39} \) TWh
  - More on threat analysis later
Performing a Query

- Client reconstructs the reverse index
- Conjunctive search: intersection of queries

```
Client

h(search_term + ACL_role) = h_{acl}

qS_{Ax} + qS_{Bx} + ... + qS_{nx}

Query Reconstruction

[ DocID_{i}, DocID_{j}, ... ]

Document IDs for all documents containing search_term

Query Server

HSM

h(h_{acl} + salt_{B}) = (h_{qs})

<table>
<thead>
<tr>
<th>Hash</th>
<th>Query Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>h_{0}</td>
<td>qs_{0}</td>
</tr>
<tr>
<td>h_{1}</td>
<td>qs_{1}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h_{x}</td>
<td>qs_{x}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h_{m}</td>
<td>qs_{m}</td>
</tr>
</tbody>
</table>

1. QS_{A} QS_{B} ... QS_{n}
2. HSM
3. Query Reconstruction
4. h_{acl}=h(search_term + ACL_role)
```
Adding new content

- Similar to ingestion process
- Identify keywords for new document
- Query for each of the existing reverse indexes
- Add the DOC ID to each reverse index
- Secret split them and push to each Query Server
- QS updates its key:value store with the new shares
• Goal: maintain information privacy in a distributed, untrusted environment

  • At most \((\text{threshold} - 1)\) query servers are compromised
  • Able to read data sent to server (not the HSM)
  • Potentially unlimited time to carry out attack
  • Can run arbitrary code on a compromised server
  • Targeted vs wholesale theft

• Numerous side channel attacks
  • Range from cold boot to social engineering
  • Not trying to solve

• Assume one or more clients \textit{will} be compromised
• Access to client’s RBAC credential
• Does not reveal salts or information not related to that role
**Threat Analysis**

**HSM**
- High barrier for attackers to overcome
- Salt prevents targeted theft since all query servers differ
- If compromised, stored keywords are revealed
- Compromise does not lead to data release
- Recovery = rebuilding the server

**SSL**
- Assume secure connections between client and servers
- SSL is not inviolate, just outside of Percival’s scope
- Unable to read contents of encrypted data stream
- Can get quantity of search requests, but not result

**K:V Store**
Performance: Digital Corpora

- 1 million files of various types (e.g. pdf, txt, html, etc...)
- Keywords found by a Stemmed TF-IDF
- 5 : 3 splitting scheme
- Ingest into a BerkeleyDB key:value store
- Very corpus dependent

64 bit Linux
4 core
24 GB
Intel 4764 HSM

- 80% contained < 3 Doc IDs
- All shares are of equal size
- Avg query completion time: < 1s
- Precision and recall: based on number and accuracy of keywords
- Salt rotation: < 2 min
- Query Server rebuild: 53 min
- 32B DocID with 100 keywords: 9.6GB
Future Work

- Support additional ACL methods (currently limited to RBAC)
- Hierarchical ACL support
- Keyword locality: a.k.a exact phrase matching
- Improve query server recovery time
- Evaluate performance using real search workload
- Disaster recovery
Thank You

Joel C. Frank

jcfrank@soe.ucsc.edu

This work was supported in part by Sandia National Laboratories, the National Science Foundation under award IIP-1266400 and industrial members of the Center for Research in Storage Systems. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
Backup Slides
Access Control

- Leverages a secure, external access control service
- Segregates the query servers to localize data loss
- Unique set of reverse indexes for each credential
  - Potentially large space overhead
  - Role-based access control (RBAC)
  - Organizations typically have ~20 defined roles
Concurrency Control

- Can potentially corrupt a set of reverse index shares
- Strong Strict Two-Phase Locking (SS2PL)
- Distributed Lock Manager (DLM)
- Operations that rely on the DLM:
  - Salt rotation
  - Performing a query
  - Adding new content
Salt Rotation

- New salt generated by the HSM
- Sent to other HSMs via secure channel
- HSM iterates over its stored keywords
- Does not help if already compromised