POTSHARDS
Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, and Kaladhar Voruganti

Presented by: Jason Gustafson
Overview

- Secure long-term storage
- Background technology review
- POTSHARDS overview
- Implementation details
- System evaluation
The Challenge

- In the long term:
  - People pass away.
  - Companies merge or go bankrupt.
  - Governments are overthrown.
  - Natural disasters occur.

- More technically:
  - Hardware fails.
  - Crypto-systems are broken!

- The challenge is to design a secure long-term storage system which maintains security even in the occurrence of these events.
POTSHARDS from 10,000 Feet

- The basic idea is to split the data across multiple institutions in such a way that significant collusion is required to recreate the data.

- No cryptography is used!

- Builds on many prior technologies
  - Secret splitting
  - Distributed RAID
  - Algebraic signatures
Background: Secret Splitting

• A \((K, N)\) secret splitting scheme divides a single piece of data into \(N\) separate pieces such that no fewer than \(K\) are needed for reconstruction.

• The Shamir method is to use polynomial interpolation
  • Let \(D\) be the data. Chose, at random, a \(K - 1\) polynomial \(q(x)\) such that \(q(0) = D\).
  • Evaluate \(q(x)\) at \(N\) separate points to get the separate pieces.
Background: Secret Splitting

• An XOR-based method:
  • Let $D$ be the block of data. Choose $N - 1$ random blocks of equal size.
  • Choose one more block of data such that the XOR of the $N - 1$ blocks and it is equal to $D$. 
Background: RAID 5

- Data is XOR'd together to create a parity bit.
- Can recover from the loss of one disk.
- Parity is spread across disks to avoid write bottleneck.

Source: https://secure.wikimedia.org/wikipedia/en/wiki/Standard_RAID_levels
Background: Algebraic Signatures

- Parity of the signature is equal to the signature of the parity.
- Allows verification of the possession of data without actually possessing that data (or any signatures).

Source: Schwarz T., Miller E.; “Store, Forget, and Check: Using Algebraic...”
POTSHARDS Security Overview

- **Confidentiality**
  - Original data is divided using secret-splitting techniques into “shards,” which are distributed to separate organizations (ideally with competing agendas).
  - The user holds an index which shows how to recombine the shards to get the original data.

- **Integrity**
  - Distributed RAID technologies.
  - Algebraic signature verification.

- **Availability**
  - Redundancy and distribution.
  - Approximate pointers in case the user's index is unavailable.
POTSHARDS Implementation

- Data is processed in four steps before being placed at storage locations.
- Each step provides a different protection.
Preprocessing

- Converts a chunk of data into a set of fixed-sized objects.
- Each object gets:
  - An identifier.
  - A hash to help with data reconstitution
Secret Splitting

Uses an XOR-based secret splitting algorithm (tuned for secrecy) to produce $N$ fragments. Add another identifier and hash and some additional metadata.
Availability Splitting

- Uses Shamir \((M, N)\) secret splitting on the fragments from previous step to produce “shards.”
- Throw on another identifier and an approximate pointer to the next shard (discussed later).
- Individuals shards no longer contain the fragment metadata.
Placement

- Select archives for each shard so that no single archive has enough shards to recover the data.
- Ideally, each archive is in a separate “security domain.”
- Can take into account geographic location of archives.
Reliability Enhancements

- Archives are required to participate in a distributed RAID over the data they store.
  - Each archive is divided into fixed-sized blocks.
  - When a shard is inserted, a random block is chosen, the shard is placed in the last slot, and a parity update is sent along.
  - The update includes the data in the block.
  - The parity is computed across redundancy groups.
Secure Reconstruction

- If an archive is lost, recovery proceeds as follows:
  - Choose an archive which has space and is not a member of the redundancy group being recovered.
  - Elect one archive to manage the rebuilding through a sequence of chained requests which proceed according to the diagram below.
Secure Integrity Checking

- Each archive must check the internal consistency of each block.
- Algebraic signatures are used for archives to verify that the other members of each redundancy group are actually storing the data.
How to Retrieve Data

- After the transformation process completes, the user gets an index mapping the original data to its shards.

- If the index is lost...
  - Each shard has an approximate pointer.
  - “Following” the pointer involves retrieving many shards (some of which might not exist).
  - Since you cannot confirm the data until you have enough shards, this potentially involves retrieving many shards.
Evaluation

• Implementation:
  • 15,000 lines of Java
  • Standard TCP/IP communication
  • BerkeleyDB for storage

• Experiments:
  • 750 KB objects
  • (2, 2) first-stage secret splitting
  • (2, 3) second-stage secret splitting
  • Local experiments on 1 Gbps network
  • Global experiments on PlanetLab
## Read Performance

<table>
<thead>
<tr>
<th>Ingestion Profile</th>
<th>Cluster</th>
<th>PlanetLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secret Splitting</td>
<td>time (ms)</td>
<td>1509</td>
</tr>
<tr>
<td>Layers</td>
<td>msgs in</td>
<td>1</td>
</tr>
<tr>
<td>Request</td>
<td>msgs out</td>
<td>1</td>
</tr>
<tr>
<td>Placement</td>
<td>time (ms)</td>
<td>37</td>
</tr>
<tr>
<td>Layer</td>
<td>msgs in</td>
<td>1</td>
</tr>
<tr>
<td>Request</td>
<td>msgs out</td>
<td>6</td>
</tr>
<tr>
<td>Archive</td>
<td>time (ms)</td>
<td>67</td>
</tr>
<tr>
<td>Layer</td>
<td>msgs in</td>
<td>6</td>
</tr>
<tr>
<td>Request</td>
<td>msgs out</td>
<td>6</td>
</tr>
<tr>
<td>Response Trip</td>
<td>time (ms)</td>
<td>88</td>
</tr>
<tr>
<td>Total Round Trip</td>
<td>time (ms)</td>
<td>1731</td>
</tr>
</tbody>
</table>
### Write Performance

<table>
<thead>
<tr>
<th>Extraction Profile</th>
<th>Cluster</th>
<th>PlanetLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Trip</td>
<td>time (ms) 28</td>
<td>6493</td>
</tr>
<tr>
<td>Shard Acquisition</td>
<td>time (ms) 832</td>
<td>29666</td>
</tr>
<tr>
<td>Transformation Layer</td>
<td>msgs 34</td>
<td>34</td>
</tr>
<tr>
<td>Transformation Response</td>
<td>time (ms) 1009</td>
<td>1698</td>
</tr>
<tr>
<td>Total Round Trip</td>
<td>time (ms) 1843</td>
<td>31410</td>
</tr>
</tbody>
</table>
Performance with Multiple Clients

![Graph showing throughput (MB/s) vs. number of clients, with lines for extraction and ingestion.]
Data Recovery

(a) 2 of 3 split.

(b) 3 of 5 split.