POTSHARDS: Secure Long-Term Storage Without Encryption

Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, Kaladhar Voruganti
Storage System Research Center, University of California, Santa Cruz; *IBM Almaden Research Center

The Need for Secure, Long-Term Storage
Recent legislation has placed strict requirements on preservation and retrieval properties of business data. Additionally, as users increasingly create and store images, video, family documents, medical records and legal records digitally, the need to securely preserve this data for future generations grows correspondingly. This information often needs to be stored securely: such as medical records and legal documents that could be important to future generations must be kept indefinitely but not be publicly accessible.

To address this need we have created POTSHARDS:

Protection Over Time, Securely Harboring And Reliably Distributing Stuff

The Problem with Long-Term Encryption
Encryption is only computationally secure and its security often relies upon the difficulty of a related problem such as factoring a large number. While useful for short term security, in the long-term it is vulnerable to attacks. For digital “time capsules” that must last for decades or even centuries, the writer is assumed to be gone soon after the data has been written.

Solution: Secret Splitting
Secret splitting algorithms produce a set of number of shares from a secret. These shares can be used to reconstruct the original secret. The long-term advantage of secret splitting is that it can be proven that an insufficient number of shares reveals no information about the secret. A variety of secret splitting algorithms exist differing in performance and features. Some require no information about the original secret. In such threshold schemes, both m of n can be set independently.

1 POTSHARDS Mechanism: Secret Splitting
A simple n of m secret splitting algorithm using XOR

1. Generate n - random pieces of data the same size as the secret in size
2. XOR the n - random pieces and store the result in S'
3. Take away the secret S and distribute the n - random pieces and S'

With less than all n - random pieces and S' no information about S is revealed. However, with all of the pieces R, S = R ⊕ S'

More advanced n of m threshold algorithms about the secret to securely both m and n shares can rebuild the secret while the others have no information.

The Digital Time-Capsule Scenario
One possible use of POTSHARDS is as a secure repository for data being saved for a future generation. This adds the unique property that the reader may have little knowledge of the system's contents and no contact with the original writer, while file bitmasks may be indecipherable, the human readable format of the file is preserved. Hence, POTSHARDS can be used to store virtually any kind of data, from old letters and diaries to future reports.

POTSHARDS Archives and Distributed RAID
The security in POTSHARDS depends on each archive keeping its contents secure. Thus the system uses a parity-like model where each participant is independent and has an incentive to provide availability. All archives are not necessarily connected and can maintain a private index that maps their data to shards. This can be shared within POTSHARDS itself.

This approach has several advantages:
• A user can use the approximate pointers to find a given object. With the shard index, the user can find any shard.
• The approximate pointers do not provide an adversary with enough information to launch a targeted attack. They would need to access every shard for every object.
• Since there is no central index, the threat of a malicious insider is mitigated.

Why two-levels of secret splitting?
Each level can be tuned for a specific property and performance. For example, the first level can be tuned specifically for secrecy and the second for I/O performance. The second level is tuned to provide availability and utilize Shamir's linear interpolation based threshold scheme.

POTSHARDS Mechanism: Approximate Pointers

What if a user loses the index over their shards or never had one?
All of the data content of POTSHARDS can be recovered without an index (albeit with a performance penalty). Through the use of approximate pointers, standard linear pointers indicate a range in the POTSHARDS namespace. The shards are not sequential and the user can maintain a private index that maps the data to shards. This index can be shared within POTSHARDS itself.

Solution:
1) Choose a new destination for the shard
2) Split each shard into two random halves, and send each half on a separate path to the destination archive

Now do you rebuild the contents of a lost archive without rewriting existing shards to any other archives other than the shard itself to the archive?

POTSHARDS Mechanism: Secure Archive Rebuilding