Jigsaw – Survivable distributed data Storage (SD^2S)

***Abstract**:*

*SD^2S is a secure data storage method called **Jigsaw**, based on Byzantine fault tolerant data storage in a distributed and decentralized network. Jigsaw enforces the 'double-blind' principle: The fragmentation process knows how to reconstruct a data object but has no knowledge or where fragments are stored; the fragment storage nodes have no knowledge in which data object a fragment belongs.*

"Cybersecurity of data at rest"

Written By: Andre Szykier - UbiVault

UbiVault © 2019

Background

Decentralized storage systems partition data among **nodes** similar to what RAID does (Redundant Array of Independent Disks). A storage system stores **fragments** of a data object on distributed storage nodes, with copies of each data block on at least ***k*** of ***n*** nodes. Decentralized storage systems, designed for scalability and tolerance of infrequent faults, do not have a single failure point.

What is **survivable storage**? At a minimum, it provides a way to store data not in a single instance such as a database or file. Rather, it distributes a data object as fragments across a federation of data storage mediums. One can store fragments across devices such as smartphones, computers, cloud storage devices or any hybrid mixture. The key is that storage is no longer in one place but across many resources. Survivability is defined as the 'ability to recover data when one or more storage resources is unavailable during a request'.

Redundancy is the key to Byzantine fault tolerant solutions. It supports the ability to retrieve the original data object from a distributed storage system. The objective of **Byzantine fault tolerance**[1] is to defend against failures of system components that prevent other components from reaching an agreement (consensus) among themselves, where such an agreement is needed for the correct operation of the system. Redundancy in storage requires:

a. At least ***k*** of ***n*** storage nodes, available at any time

b. The fragmentation method allocates redundant fragments based on **k**

There are many examples of distributed, fault tolerant storage. Early work at Carnegie Mellon produced PASIS[2]. University of California Berkley developed OceanStore[3], OASIS[4] from NEC, Tahoe-LAFS[5], Storj[6] and, more recently, IPFS.[7] One company, Lacero[8] has adopted the Jigsaw principle in their product using smartphones as a distributed storage medium. Further extensions include the use of blockchains and smart ledger support.

Jigsaw – Data Store

The fundamental principle for ensuring security is to protect the data target from MiM (man in the middle) attacks and reduce the information value of any data stored on a node. The latter is important in that a trusted party, like a systems administrator, cannot access information using elevated privileges and compromise the storage system. Even if they were to compromise all storage nodes (extracting content), the data has no intuitive data structure that can be deduced from the fragments across the storage nodes.

The Jigsaw process is relatively simple.

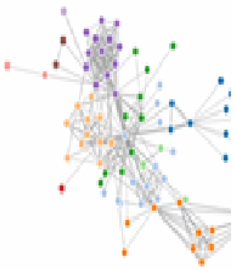
1. A data owner has a data object in their possession. Jigsaw treats all information as a '**blob' –** a data type that stores binary data for access by a computer enabled process. It may be text, video(s), image(s), file(s), file directories, even another data store such as a database extract. Based on their intent, the user can issue a request to encrypt the object using various methods before handing the object to the shredder process. The shredding process can be local to the user or performed on a dedicated server.

2. The shredder process is designed to break up the data objects into multiple fragments that are to be stored in the distributed storage system. Each fragment has a unique id (hashing plays a seminal role) which can be queried across the nodes of the storage network. During this request, the Jigsaw shredder knows how many of the ***n*** are active and online ***n'*** to calculate a ***k*** node redundancy threshold.

   There are two ways to support redundancy: (a) duplicate a fragment with separate unique ids or, (b) notify the storage system to replicate the fragment across different nodes using the same id. Both methods have different strategies that are however not germane to this topic.

3. The shredder creates an object key containing ***metadata*** on how to reconstruct the fragments. Metadata is data that describes other data. For Jigsaw, it is a companion file that the user protects and stores outside of the distributed storage network. It is feasible to fragment the metadata file and store it in the same network creating a special metadata key file to recover the metadata itself.

4. The date storage node network consists of distributed nodes whose sole purpose is to negotiate whether they accept a fragment for storage. Upon acceptance, they create a token that is signed to indicate that the fragment was accepted.



   If they reject a fragment, they remain silent. The shredder waits until all fragments were accepted to finalize the metadata key file returned to the user. The shredder receives the token from a proxy service that certifies that a fragment was recorded.

UbiVault © 2019

Note that metadata file has no information on which storage node a fragment was stored. Storage nodes can securely message (broadcast) that they refused a connection to allow other storage nodes to decide about storing the available fragment.

To minimize visibility to anyone observing the system all communications between storage nodes use secure sessions based on *InterVault's* ***Perfect Forward Secrecy.***

> **Perfect forward secrecy** (PFS), establishes key agreement protocols that gives assurances that session keys will not be compromised even if the private key of the server is compromised. Forward secrecy protects past sessions against future compromises of secret keys[9]

Jigsaw – Data Retrieval

Once a data object is fragmented and stored in the distributed network of storage nodes it prevents the ability of an unauthorized party to reconstruct any object from the universe of stored fragments; there is no information on which object a fragment is from. To recover the object requires the metadata key in control of the user. The shredder receives metadata and queries the distributed storage network with fragment id.

The process of passing fragment ids to the network is further obfuscated such that the storage system has no knowledge of the requestor. Further every user has a unique session within which fragments are passed, either for writing or recovering.

Jigsaw Applications

The principle of ***security of data at rest*** in Jigsaw is independent of the cybersecurity practices applied to a data storage environment. It assumes that hosting services have perimeter defenses, audit mechanisms and permission rules that apply to any application on their system. Jigsaw is designed to increase the ***entropy***[10] of information in the distributed network of nodes. In a simple analog, the storage network is a giant box of compartments containing many pieces to a puzzle, but compartments also contain pieces to other puzzles. Without the metadata key there is no way to retrieve only the pieces that belong to a specific puzzle.

Here are some examples of Jigsaw in practice.

✓ Health Records	✓ Track & Trace data
✓ Legal Documents	✓ Logistics
✓ Media	✓ Financial Reporting
✓ Time sensitive data	✓ Voting Records
✓ Insurance	✓ Alt-currency offchain

Summary

Secure distributed data storage networks are a strong alternative against attacks on centralized data systems. The concept of data shredding (fragmentation) allows parts of data objects to be stored in different nodes.

Redundancy of fragments ensures that the storage system is survivable to allow data retrieval when one or more nodes (k < n) is offline. Hashing mechanisms along with encryption allow for data integrity to prevent alteration of stored data fragments.

Both the Jigsaw process that creates the fragments (shredder) and the network nodes that store them, follow the double-blind principle: shredders don't know where fragments are stored, and storage nodes don't know what data object fragments belong to.

The metadata key to reconstruct a data object is kept outside of the storage network to reduce potential targets for unauthorized retrieval. In short, stored data is useless without the key.

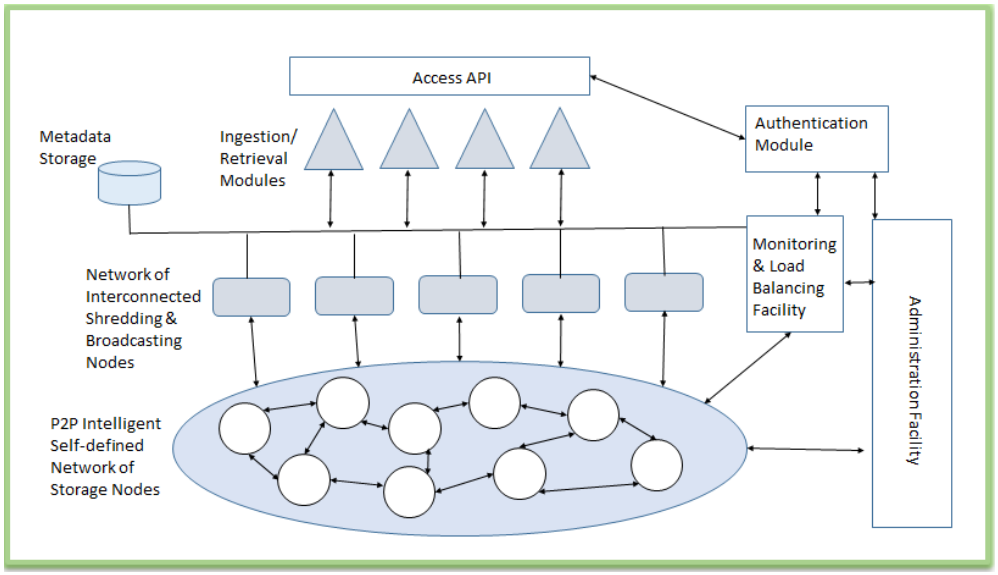
The primary advantage of Jigsaw is in securing data with low volatility (data half-life). When properly applied, it provides an audit record of creation, access, change and removal to any stored data object (data blob.)

Jigsaw is integrated with InterVault™, the data in motion security module in the UbiVault Fabric, along with connectivity to blockchain and smart ledger deployments (UbiChain™.)

UbiVault © 2019

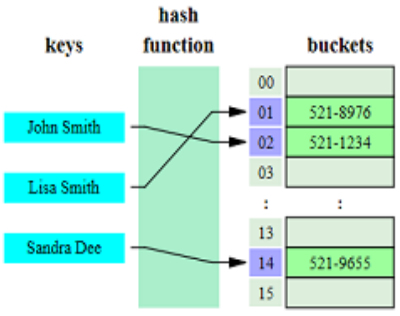
Appendix

Jigsaw Architecture



The diagram above describes an architecture where the shredder modules are instances (virtual machines) in a distributed secure perimeter on the **cloud**. Each storage node can communicate with other nodes. A storage node follows the Kubernetes model.

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications that automatically mount a storage system, whether from local storage, a public cloud provider such as GCP or AWS, or a network storage system. K8s is self-healing – it restarts containers that fail, replaces and reschedules containers when nodes die, and doesn't advertise them to a system until they are ready.



Interrogating a storage node for storing or retrieving a fragment by its id can be done by any algorithm that treats hashes as an indexing method. A hash table is a data structure that is used to store keys/value pairs.

The hash function computes an index into an array in which an element will be inserted or searched.[11] In a well-dimensioned hash table, the average cost for each lookup is independent of the number of elements stored in the table.

UbiVault © 2019

Jigsaw supports *Homomorphic Encryption*[12], allowing computations to be carried out on encrypted data and obtain an encrypted result which when decrypted matches the result of operations as if performed on plaintext. Homomorphic Encryption is a breakthrough which enables cloud storage and computation. HE is considered a necessary component of federated data networks where source data is kept in separate sources and a merged into a virtual encrypted instance for analytics, either centrally or at a source. It does however require a creating a standard, most likely by multiple standardization bodies and government agencies. An important part of standardization is broad agreement on security levels for varying parameter sets.[13]

Jigsaw Application-level Protocol

Receive *Data Object (DO) Store Request from Ingestion Module*

- *Confirm ready state with Ingestion Module*
- *Shred the DO*
  - *Apply hashing algorithm*
  - *Shred DO into fragments*
  - *Insert checksum segments*
- *Check authorization from Monitoring engine (heuristic pattern matching)*
- *Establish session with Storage P2P network*
  - *Issue one-time certificate*
  - *Apply Session Key*
- *Send Fragments to the Storage P2P network*
- *Receive confirmation of storage*
- *Send Confirmation (and optionally Metadata) to Ingestion Module*

Retrieve *DO from Ingestion Module*

- *Confirm ready state with Ingestion Module*
- *Retrieve the Metadata*
  - *Option 1 – from the request*
  - *Option 2 – from the Metadata storage associated with the Broadcasting Network*
    - *Establish Session /Authenticate*
- *Retrieve Metadata*
  - *Retrieve Fragments from the P2P storage network*
  - *Validate Fragment hasn't Been Tampered with (comparing to other copies returned)*
- *Apply Metadata to retrieved fragments to reconstitute DO*
- *Send reconstituted DO back to the Ingestion/Retrieval Module*

Jigsaw Storage Node Functions

*Cyclical Self-Verification (**heartbeat**)*

- *Check own Availability*
- *Check own Capacity*
- *Verify own State with the Monitoring Facility (Unavailable, Rebuilding, Ready for Storage, Retrieval, or Both)*

***Write** Fragment Request*

- *Authenticate The message*
- *Validate Applicable State*
- *Store Fragment or pass to another storage node*
- *Confirm Write Request to Requester*

***Read** Fragment Request*

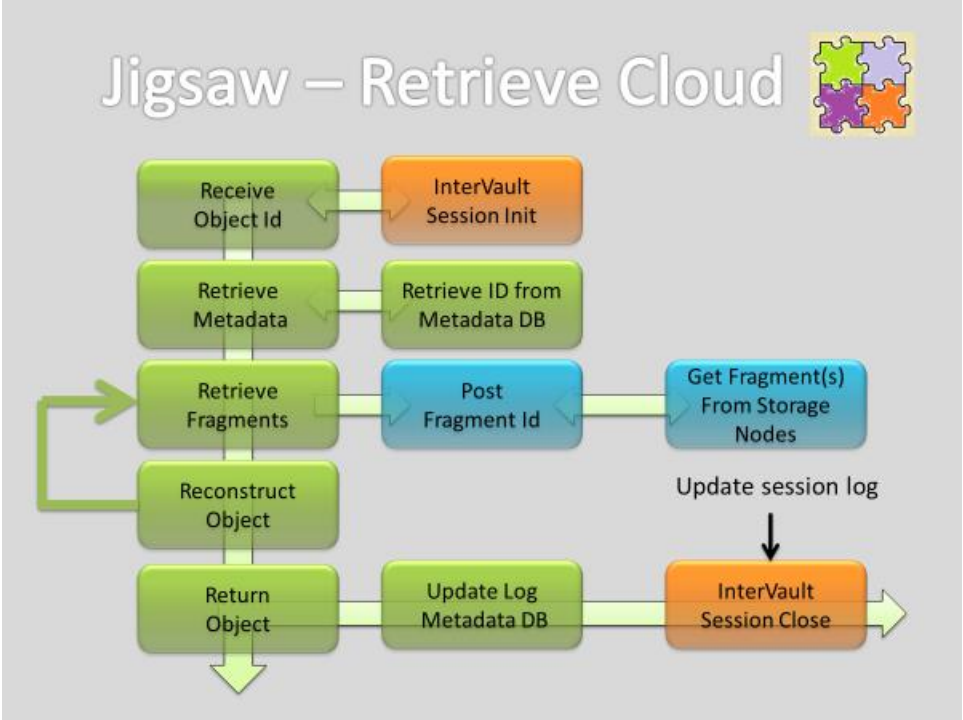
- *Authenticate Message*
- *Validate Applicable State*
- *Retrieve Fragment from Storage node*
- *Return Retrieved Fragment to Requester*

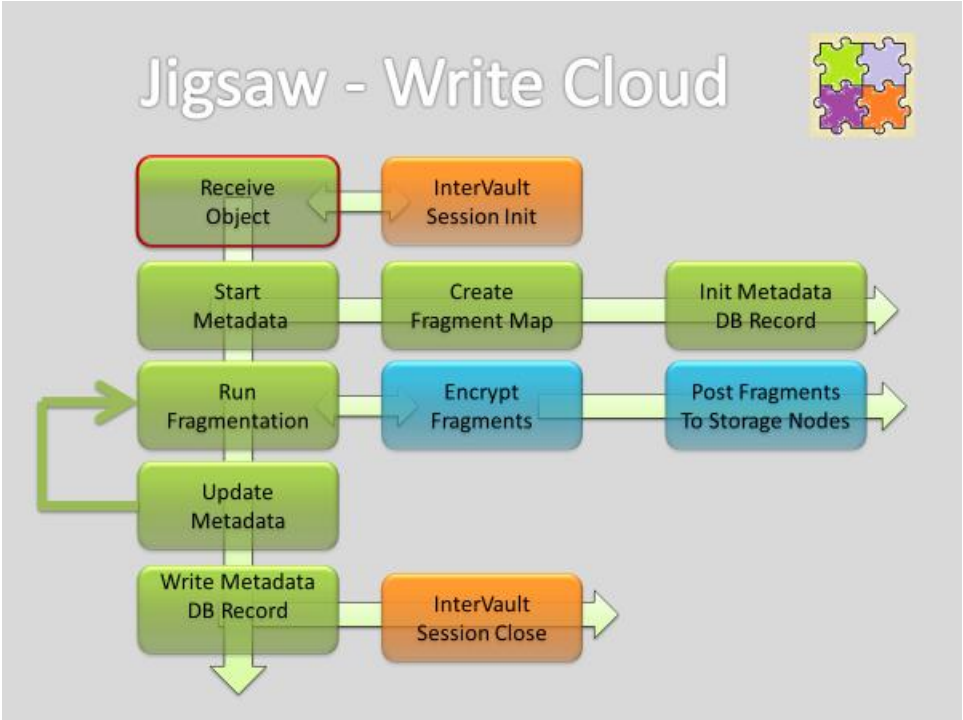
***Rebuild** Node Content*

- *Authenticate The message*
- *Validate Applicable State*
- *Validate received fragment (lack of its presence)*
- *Store Fragment*
- *Confirm request to Requester*

UbiVault © 2019

Jigsaw Protocol Flow





UbiVault © 2019

Author

Andre Szykier lives in Bermuda Dunes California, is a mathematician and founder of ***UbiVault***, a leader in security solutions for a decentralized Web. He is also the CTO of a cryptocurrency ATM network – ***BlockchainBTM***, and contributing scientist to ***Aegis Health Analytics*** in Washington DC.

andre@ubivault.com

References

[1] One example is *Tendermint*, a general purpose software for BFT state machine replication. Using a socket protocol, it enables state machines to be written in any language, that allow a state machine to influence elements of the consensus, such as the list of active processes. Tendermint is implemented in the style of a blockchain, which amortizes the overhead of BFT and allows for faster recovery from failure.

[2] *PASIS*, *apps.dtic.mil/dtic/tr/fulltext/u2/a436245.pdf*

[3] *OceanStore*, *oceanstore.cs.berkeley.edu/info/overview.html*

[4] *OASIS*, *www.oasis.com.hk/data/NEC_White%20Paper_on_Fault_Tolerant.pdf*

[5] *Tahoe-LAFS*, *tahoe-lafs.org/trac/tahoe-lafs*

[6] *Storj*, https://storj.io/

[7] *IPFS*, *ipfs.io/*

[8] *Lacero*, *lacero.io/*

[8] *InterVault*

[9] *PFS* *en.wikipedia.org/wiki/Forward_secrecy*

[10] Information entropy is the average rate at which information is produced by a stochastic source of data where possible data values are the negative logarithm of the probability mass function **S.** The amount of information conveyed by each event defined in this way becomes a random variable whose expected value is the information entropy which refers to disorder or uncertainty. The definition of entropy used in information theory is analogous to the definition used in statistical thermodynamics:

$S = -\sum_i P_i \log P_i$

[11] *Hashing*, *www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/*

[12] *Homomorphic encryption*, *homomorphicencryption.org/*

[13] *HE Standards*, http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf