End-To-End Encrypted Distributed Cloud Storage

Daniel Wagner

June 2020

Abstract

Current cloud storage solutions store the user data unencrypted on centralized server infrastructures where it often gets analyzed to build profiles of their customers. Even when data is encrypted at rest the private keys are kept on the same infrastructure as the data to decrypt it whenever a user sends a request. This opens a door for hackers since all data is in one place with their respective decryption keys as they gain access to the network.

This paper takes a look at how a distributed architecture can solve the problems mentioned above and how to make it usable for the average user. Blockchain technology enables a decentralized identity system for permission management of stored files in the peer to peer cloud storage. Preparing files on the users device before encrypting them mitigates some issues of distributed systems.

1 Introduction

Cloud Storage is expected to grow at a rapid pace over the next decade while the amount of data generated increases. With more digital infrastructure we store more sensitive information in the cloud. A distributed end-to-end encrypted approach provides users more security and the ability to own their data. Such an ecosystem consists of multiple important parts with their own problems to solve. An identity system for authentication, a peer to peer file storage and a end user client for convenient access to the data.

Decentralized identifiers stored on a public blockchain to ensure data consistency with their public encryption keys allow every participant to verify the integrity of the system. This also enables access control for the files.

Data is stored on a peer to peer system with content addressable hashes of files. To locate a file a peer has to query the distributed hash table stored by network participants.

An end user client combines those systems for an easy to use experience. It handles management of cryptographic keys, caching of data and preperation of files before encryption to save bandwidth and optimize performance.

2 State of the art

Current cloud storage providers dont offer end to end encryption for their users and store data on centralized servers. With this approach data can be used for artificial intelligence to learn from big datasets it comprises on privacy and security.

The W3C has developed a standard for decentralized identifiers (DID). It can handle authentication with asymetric encryption and provides public key discovery of users for sharing data. [1]

Ethereum is a distributed smart contract platform based on blockchain technology that allows anyone to develop simple decentralized programs. Those programs can be accessed by everyone. [2]

A working distributed storage network is IPFS. It lets users store files that a referenced by a content hash. Everyone can request files from the network by this hash. [3]

3 Problem statement

While the key components of an end to end encrypted distributed cloud storage exists, fitting them together and make the solution work afor end users with a good user experience is no trivial task.

One aspect of cryptographic systems is key management. It is different from current authentication, since the users are responsible for protecting their keys. There is no method of account recovery when the keys are lost.

The DID standard only describes how the documents are structured, but the implementation of a registry is not defined. The registry must be on a distributed network aswell, to guarantee access to the file permissions for every user.

To fulfill the needs of users to be as fast as possible, decentralized networks fall short in comparison to centralized solutions.

The last disadvantage of an end to end encrypted cloud is that data cannot be used to drive development in artificial intelligence forward.

4 Approach

The approach takes a look at what techniques can be used to solve the problems mentioned above and how to combine them in a easy to use application.

With smart contract wallets [4] handling multiple keys, for instance with one key per device, and implementing custom recovery solution less difficult. Those wallets are on the Ethereum blockchain with keys mapped to the DID Document. A recovery process similar to existing account systems can be implemented using hardware security modules. [5]

Those wallets also store the reference hash of the DID document uploaded to the IPFS network. It enables resolution of public keys by knowing the wallet address or a human readable domain.

To prepare data for performance before encryption unique methods must be used for different data types. Images as an example can be transformed to different sizes. A blury low resolution placeholder with a size of 1-2 Kb cached on the device makes scrubbing through all images quickly feel instant while the thumbnails for the next page are loaded in the background before they appear on the users screen. Only when a user clicks on an image a bigger size that fits the screen resolution gets requested from the network. [6] This approach increases bandwidth when uploading the file, but saves a lot of it the more often the file gets requested.

Federated learning enables artificial intelligence to train its models on a clients device with the data store locally. [8]

To work on all platforms and devices the client can be developed as a progressive web application. Utilizing the webs standards makes it accessible and platform agnostic. [7]

References

[1] Decentralized Identifiers (DIDs) v1.0, 2020, Drummond Reed, Manu Sporny, Dave Longley, Christopher Allen, Ryan Grant, Markus Sabadello, Jonathan Holt
https://w3c.github.io/did-core/

[2] Ethereum Whitepaper, 2013, Vitalik Buterin
https://ethereum.org/en/whitepaper/

[3] IPFS Docs: How IPFS works, 2019, Alan Shaw, Mikeal Rogers, Steven Allen
https://docs.ipfs.io/concepts/how-ipfs-works/

[4] Smart Wallets are Here, 2019, Eric Conner
https://blog.gnosis.pm/smart-wallets-are-here-121d44519cae

[5] Security Infrastructure at Fortmatic, 2019, Sean Li
https://medium.com/fortmatic/security-infrastructure-at-fortmatic-4a95c3688997

[6] Building the Google Photos Web UI, 2018, Antin Harasymiv
https://medium.com/google-design/google-photos-45b714dfbed1

[7] Progressive web apps (PWAs), 2020, MDN Contributers
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

[8] Introducing TensorFlow Federated, 2019, Alex Ingerman, Krzys Ostrowski
https://blog.tensorflow.org/2019/03/introducing-tensorflow-federated.html?hl=pt-BR