Design Document

Bingsong Tong & Gordan Lee

Data Structures

User structure

We divide the user structure into two layers: inner and outer. The outer structure contains three elements. One is used to store the inner user secret, including the user's uuid, two RSA keys, one encryption and one signature; the second is the HMAC value of the user's secret. , used to detect changes; the last one is salt, used to generate a unique key. After the generation of inner user structure, we json.Marshal(), encrypt and sign it. This is how it is stored in the outer user structure.

File structure

Like user, file is also composed of inner and outer layers. The inner structure contains the uuid of the file (stored in blocks, with HMAC placed at the beginning to detect the latest update of the file) and the symmetric key of the file. The outer structure contains four elements: the first is the inner file structure, the second is the name of the source file owner, the third is the digital signature of the source file owner, and the last one is the file sharing record (large array). (Shared person, pointer to the shared structure) An array composed of these multiple key-value pairs. The source file owner encrypts the fileSecret(the inner file structure) with his own private key, and we save filename-fileuuid into datastore.

Share structure

Like user, invitation is also composed of inner and outer layers. The inner invitation consists of fileScrete, the name of the source file owner, and the electronic signature of the source file owner. This inner structure is encrypted using the sender's public key and the receiver's private key and is transmitted using RSA. The outer structure consists of an inner structure, the sender's signature, the receiver's signature and a pointer to the shared file structure.

Help function

encryptThenMAC(): use SymEnc() first then use HMACEval() to MAC the ciphertext.
blockStore(): Store files in chunks
autofill(): Serves for generating keys
computation(): Compute password and file name

User Authentication

There is hmac in user_struct. Once an incorrect password is entered, the symmetric key generated through salt, etc. will change and cannot pass hmac verification, so the verification fails.

Multiple Devices

Calculate the hmac of the file block file before each operation on the file. If another device is already open when the file is changed, the old hmac will exist.
The old device will be inconsistent with the latest hmac of the datastore, so you will be asked to update again.

File Storage and Retrieval

Storage: Each time you use filename to calculate pwd to get uuid, find the file's unique struct file struct to find the file name.

Retrieval: File storage is composed of blocks. Every time the file is modified, the blocks in the file are modified. So add whichever block you want to change.

Efficient Append

Total Bandwidth = Size of the append operation (data volume) + Constant factor.

The pieces of data that we need to upload (DatastoreSet) or download (DatastoreGet) from Datastore in a call to append are as follows: 1. The content parameter of the AppendToFile function: This is the data we want to append to the file, the size of which is determined by the number of bytes in the data itself. The amount of data in this section will vary depending on the amount of data to be appended. 2. HMAC (for message authentication code): When calculating HMAC-SHA-512, we need to upload or download the HMAC, which is 64 bytes in size. 3. Metadata related to data storage: We may also need to upload or download metadata related to storing and managing files, such as file names, UUIDs, keys, etc. This portion of data is of constant size.
So, the total number changes as the amount of data to be appended increases.

File Sharing

Whenever user A shares a file with user B, A will add a record to his file sharing record, which includes a pointer to a struct share structure. Since the struct share secret structure in the struct share structure is sent in an asymmetric encryption manner, the struct user secret will be encrypted with the recipient's public key in the keystore and then sent to the recipient. Once B accepts the share, it decrypts the struct share information and then creates a new struct file struct, which contains the file's UUID and key. This UUID is generated by calculating B's login password, but can be disclosed. This enables B to access and decrypt the shared file because it now has the file's UUID and key.

File revocation: If the owner of the source file wants to revoke the file, he should change the uuid of his file and the symmetric key of the file, and then send the new key to the current authorized person. The method to find is to share the file from his own struct file. The record starts to be searched. The pointer in this record points to B's struct share, and B's file struct can be found in the struct share, so that we can continue to find who B shared it with, so that we can find the entire share tree. thereby sharing keys based on this new tree.

UUID	Encrypted	Key Derivation	Value at UUID	Description/Relationship
User.uuid= Hash(username)[0:16]	Yes	Symmetric; deterministic; hash(username)	Hash(username)	Help logged-in users find their corresponding structures
File.uuid=computation (filename, password)	No	N/A	computation(filename, password)	Generate file uuid based on the file name and password contained by the user to determine the pointed file
Invitation.uuid=uuid.New()	No	N/A	uuid.New()	The uuid of the invitation generated by sharing the file

Notes:

• UUID: How is this UUID generated? If it is a deterministic process, which function is used and with which parameters?

• Encrypted: Is the value stored at this UUID encrypted? Yes/No.

• Key Derivation: Is the key symmetric/asymmetric? Is the key derived through a random or deterministic process? If it is deterministic, which function is used and with which parameters?

• Value at UUID: What is stored at this UUID? If it is a struct, what are the members of this struct?

• Description/Relationship: Describe how the value stored at this UUID is used through your program. Does it interact with other structs?