Eugene Brodsky
Fall 2020

# End to End Encryption on Untrusted Server Design Doc

The analogy for this program is that of an estate (metafile) that contains a house (file), and real-estate agent keyboxes which are organized in a hierarchical way (share tree).

1. How is a file stored on the server?

The house is the data structure that, given data, can perform store, load, overwrite, and append operations. The house is composed of rooms, each of which contains enough information to retrieve a chunk of data. Storing a new file or overwriting a file gives you a house with one room, and every append operation creates an additional room, this makes for efficient append operations because there is no need to decrypt/encrypt previous data. To load the file, we load the data from each room and concatenate the data in the order it was added. The house is encrypted, MAC'd, signed with keys available to every user with access to the file (stored in a share tree node, discussed below) and the house is then stored in the cloud.

The hierarchical real-estate agent keyboxes are a cryptographically enforced access tree. The basis for the cryptographic enforcement is the notion of a cryptographic link, A -> B. Suppose A is the root and should have the most power over others' access, and A wants to add a child node B. A can generate a key K0 and use it to encrypt the child node B. A stores K0 within his own node and also gives the key to user B. In this way, A is able to access B's node, but B cannot access A's node. These links are repeated to create a cryptographically enforced tree. The house and the share tree are both accessible from the metafile.

2. How does a file get shared with another user?

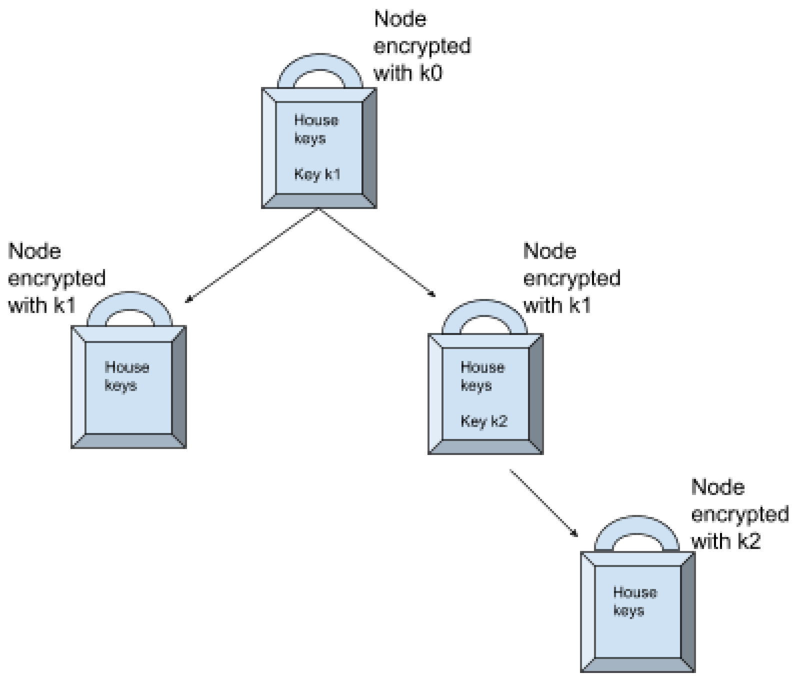
Each node of the share tree corresponds to one user that has access to the file, and the node contains all the information necessary to access the house (file). When a user A with access to the file shares to another user B, A creates a new node on the tree for user B, using a cryptographic link as discussed above. Then user A can make an access token for user B, which contains all the information necessary to access the metafile. The access token is encrypted and mac'd with keys that are sent to B in a secure manner, and the access token is signed by A.

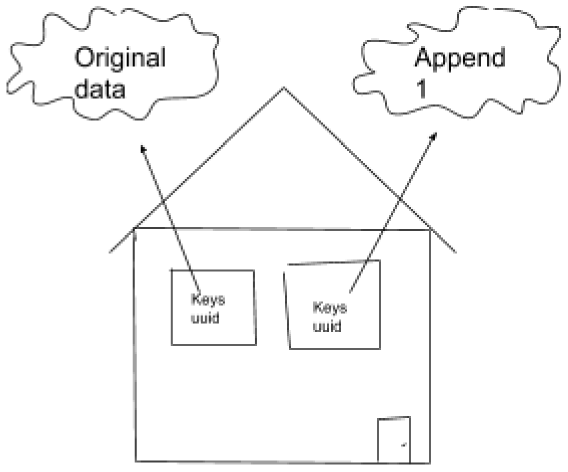
3. What is the process of revoking a user's access to a file?

The cryptographically enforced share tree makes revoking a user's access very easy. The structure of it ensures that a malicious user can only alter nodes in their subtree and this is not considered a security threat because the users in the subtree may as well have not been shared access to the file by such a malicious user. Revoking is done by removing a child node (and all of the child's subtree also), and then going through each remaining node in the tree and changing the keys to the house (file), this is done recursively. Then those same keys are used to secure the house. This is like finding out that a real-estate agent has been sleeping in the property, so you change the lock on the house and go through each agent's (except the bad one) lockbox and change the keys inside the new key.

4. How does your design support efficient file append?

The system supports efficient appends by storing each append to the file in a separate location from the original file. The rooms of the house tell a user where to look for each append. Creating an append involves adding a room, generating a set of keys for the data, encrypting the data and storing it in the cloud with a runtime proportional to the size of the append, not the number of appends made.





## Security Analysis

Attack 1: Assume a malicious user is granted access to a file, the main security considerations relate to the issue of revoking access for other users, specifically where the malicious user sets themselves as the owner of the file. This is not possible because of the cryptographic enforcement imbued on the share tree.

Attack 2: A adversary could try to alter the data stored in a particular file by altering the data stored in the cloud by a particular room in the house, but when a person loads the files contents, the room will not be able to establish the integrity of the data because it will not match the hash of the data stored in the room, so a flag will be raised and an error returned.

Attack 3: After user A shares a file with user B and later revokes access, user B may try to call ReceiveFile with the original access token to regain access to the file. This will not work because to have access to the file, a user must have a node set for them in the sharing tree, B's node is removed when A revokes access to the file, so when B attempts to access the file they will receive an error message that they do not have a node in the share tree, which implies that they do not have access to the file.