Main Idea Behind The Design:
Whenever data is to be stored or appended under the filename f, in the datastore, we will hash the data, then encrypt the data and hash together using the file key $\{K_f\}$ corresponding to the file f which is generated when the file was created for the first time. Whenever the file is accessed the confidentiality is held because you would need K_f to access it and integrity is held by computing the hash and comparing it with the stored value. All these files would be stored at random locations with no relation to each other.The locations of all files under one user would be stored in a data structure say, File_info which would be accessible only by the user.

We are using three types of **Data structures**:

1.User_info(contains private key and File_info) :
Everytime a new user registers, User_info will be created containing all the info of the user like the private key and the File_info data structure, and his password would be used to create a new key K_u. Then we will hash the User_info and encrypt the hash and the original User_info using the key K_u. Thus User_info would be confidential and we can check if some data was changed.

2.File_info(List of all files) :
This would basically contain the list of all the files a user has created with their names, location for the Data_info, and the corresponding K_f for the file.

3.Data_info(contains locations of parts of a file) :
This is required in order to make efficient appends, so that every time a file is appended we only need to store the append data at a new location and store this location in the Data_info. So, Data_info would basically contain the locations of all the parts of a file.

1. InitUser(username string, password string): This would create the User_info data structure, generate the key K_u using the password, and encrypt the User_info along with the hash of the User_info. The User_info would be

stored at the location given by the HMAC of username using K_u as key. Thus both username and password are required to get the location of User_info.

2. GetUser(username string, password string): This would return the User_info data structure if the username and password are correct.
It will be decrypted using the Key K_u which is generated by password and then the integrity is checked by hashing the data and comparing it with the stored hash.

3. LoadFile(filename string): This would get the file by accessing the data_info of the corresponding file from file_info and decrypt it using K_f of that file from file_info.
If K_f is incorrect or the location is incorrect or no such file exists this would throw an error.

4. StoreFile(filename string, data []byte): This would create a new entry in file_info, create a new K_f, and assign a random address and store it in data_info.

5. AppendFile(filename string, data []byte): This would add a new random location to data_info and encrypt the given data together with it's hash by K_f. The encrypted data would now be stored at this random location.

6. Sharing: In order to share a file the user needs to send the data_info, and the K_f of the file. We will encrypt this using the public key of the user we need to send to, and append the hash of this message, then send it to the user. He can decrypt it using his public key and check the integrity by comparing the hash value.

7. Revocation: In order to revoke access to the file we need to change the location of the file (data_info) and generate a new key {not needed but as a extra security measure} for this file so that other users with whom the data_info and K_f was shared now can neither access the file nor make any changes to it.