Brandon Wang
UC Berkeley CS-161

0. **How is a user initialized?**
   InitUser() will first take the username and password (in addition to a predetermined salt) to deterministically create 3 new private keys: an RSA private key, an encryption key, and an HMAC key. Then, the function will create a new User struct, which contains the user's username and all of these keys. This struct is then marshaled and encrypted into a byte string, which is then stored in Datastore.
   GetUser() will recompute the 3 private keys using the given username and password. Then, it'll compare these keys with the ones stored in Datastore. If all of them match, it'll return a pointer to the user's data. Otherwise, an error will be raised.
1. **How is a file stored on the server?**
   StoreFile() will first randomly create a UUID and 2 symmetric keys (encryption and MAC). The input data will be encrypted and then a MAC will be computed, and both will be stored in Datastore.
   LoadFile() will check the user's permissions to access the file, then obtain the stored ciphertext. It'll then use the symmetric keys to verify and decrypt the ciphertext to obtain the data.
2. **How does a file get shared with another user?**
   ShareFile() will first check the current's user's permissions. Then, it'll get the targeted user's RSA public key from Keystore and use it to encrypt(file UUID + decryption key + authentication key). It'll also encrypt it with the current user's RSA private key for authenticity.
   ReceiveFile() will check the recipient's permissions. It'll use the sender's RSA public key to verify the message, then use the recipient's RSA private key to decrypt the message, which contains the file's encryption and authentication keys.
3. **What is the process of revoking a user's access to a file?**
   RevokeFile() will first fetch/download the file data. Then it'll need to go through the entire procedure again of StoreFile()+ShareFile()+RecieveFile(), with the sole difference being that the file will no longer be shared with the intended user. The original file will be deleted from Datastore.
4. **How does your design support efficient file append?**
   AppendFile() will first check if user has the right permissions. Next, it will use the same methods in StoreFile() to store the appended data in Datastore. Then, it'll fetch the encrypted data from Datastore using the given filename, decrypt it, change the file's metadata to show where the updated/appended data can be found in Datastore, encrypt it again, and store the message back into Datastore.

Security Against Attacks

1. Man-in-the-Middle - If the attacker eavesdrops while a file is being shared with another user, he won’t be able to obtain any information about the file nor be able to change any information about it because the file is encrypted using RSA signatures in functions ShareFile() and ReceiveFile(). Without the user’s RSA private key, the attacker cannot decrypt the file.
2. Password Dictionary Attacks - Even if the attacker has a precomputed hash table of all of the common passwords, a dictionary attack is thwarted because the password is salted and hashed before being stored in Datastore. This also means that even if the attacker gains full admin-level access to the contents of Datastore, he won’t be able to obtain any user’s password.
3. Modifying User Struct - If the attacker gains full admin-level access to the contents of Datastore, he won’t be able to decrypt or learn the user’s private keys because all of them were deterministically created from the user’s username and password. If the attacker changes any of these keys, when GetUser() is called, the function will return an error because the recomputed key won’t match the corresponding key in Datastore.