CS-628 Assignment 1
Design

Ayush Nagal, Ashish Kumar

August 16, 2019

A User data structure and a file directory is created for every user.

1. **User Struct**
   - **Username**
   - **Private RSA key :** It will be used later for sending encrypted text.
   - **Symmetric Key(K0) :** It will be used for encrypting file directory.
   - **SHA256(password)**
2. **File Directory Struct :** It is encrypted using CFBEncrypter with K0 as the key and stored at location SHA256(username). HMAC(Encrypted File directory + SHA256(username)) is also stored along with the encrypted file for integrity check.
   - **SHA256(filename)**
   - **SHA256(username + filename) :** Location of meta-data
   - **Symmetric Key(K1) :** Key for encrypting meta-data
3. **Sharing Struct**
   - **SHA256(sender name + filename) :** Location of meta-data of the file to be shared
   - **Symmetric Key(K1) :** Key for decrypting meta-data of the shared file

Question 1: Simple Upload/Download

InitUser :

- Location of user data struct = SHA256(username + password).
- Generate a key **KK** = Argon2Key(SHA256(password) + username, username, 16). Encrypt user data using CFBEncrypter with **KK** as the key and generate the HMAC(Encrypted user data + location of user data) using the same key and store it along with user data at the same location.
- Generate an RSA key pair, random symmetric key(K0) and SHA256(password). Push the RSA public key to the keystore and store the remaining information in the user data struct.

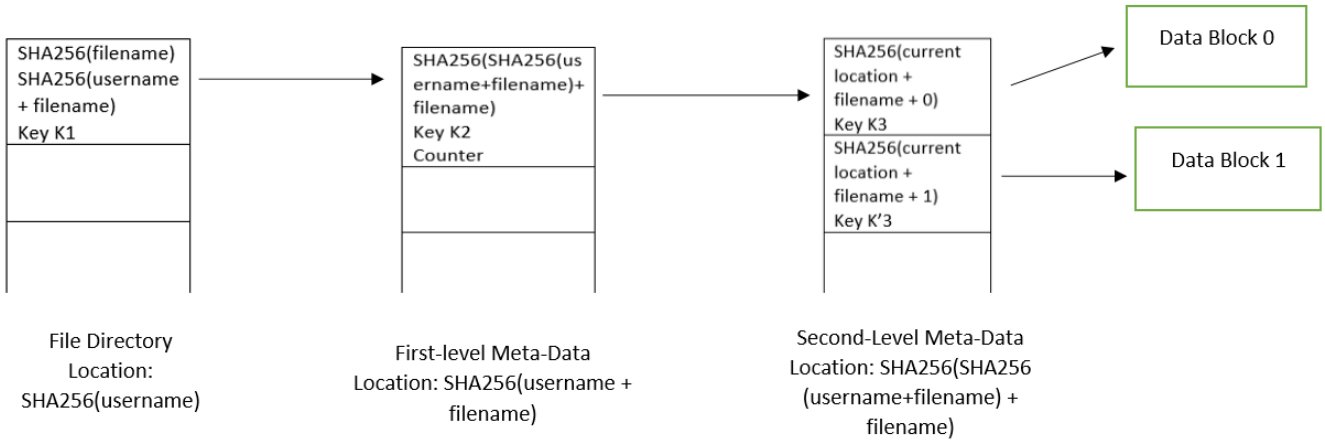
GetUser :

- Get the location of user data struct using SHA256 and key(KK) using the above invocation of Argon2. If password or username is incorrect, return an error.
- Generate HMAC as explained in InitUser and compare it with the stored HMAC. If all checks are satisfied, decrypt the user data using CFBDecrypter with key KK and return the user data.

StoreFile :

- Fetch the file directory from SHA256(username). Generate HMAC(Encrypted File directory + SHA256(username)) and compare. If no error then, decrypt the file directory using key K0 which is stored in user data. If the size of data is not a multiple of blocksize then simply return error.
- Generate a random symmetric key(K1) and store it in file directory along with other details(filename and meta-data location). Encrypt the file directory using key K0 again and update HMAC as well.
- Meta-data will initially contain a single indirect pointer. If the need arises, further single and double indirect pointers will be used. We will encrypt it like we did for file directory but the key will be K1 and generate HMAC(encrypted meta-data + SHA256(username + filename)) as well. Store these two at location SHA256(username + filename).

- At each level a key is stored, which is used to encrypt data stored at the next level. HMAC is also stored using the same key. A detailed figure is given below:



LoadFile :

- Use the method given in StoreFile() to reach, decrypt and verify each level of meta-data for the given filename. If no error then decrypt and return the file.

AppendFile :

- Use the method given in LoadFile() to reach, decrypt and verify each level of meta-data for the given filename. If the size of data is not a multiple of blocksize then simply return error.
- Generate a new key to encrypt the data block and for generating HMAC as well. Increment the counter and store the encrypted data block at SHA256(current location + filename + counter) along with the HMAC. Current location is the location of second-level meta-data.

Question 2: Sharing

ShareFile :

- Fetch the file directory and decrypt it. Create a sharing data structure to store the location of file and its key.
- First encrypt the sharing data structure using receiver's public key, then by sender's private key. Then generate a key ks = Argon2Key(SHA256(SHA256(password) + username)), username, 16). Use ks to generate a HMAC(encrypted sharing struct + SHA256(username + filename)).
- These two(encrypted sharing structure and its HMAC) are stored at location SHA256(sender + receiver + filename). Encrypt this location using the receiver's private key and sender's public key and return it.

ReceiveFile :

- Decrypt the message "sharing" using the public key of sender followed by the private key of the receiver. Generate the HMAC as explained in ShareFile and compare. If no error occurs, decrypt the sharing struct using sender's public key and receiver's private key.
- Copy the file location and its key from the sharing struct to the receiver's file directory.

Question 3: Revocation

RevokeFile :

- Fetch the file directory and decrypt it as explained in StoreFile().
- Fetch the first level meta-data and decrypt it using the key(K1) stored in file directory. Now generate a new key to encrypt it and store this key at the previous level(i.e., file directory), as explained in StoreFile(). Update the HMAC as well.
- Likewise do the same for next level meta-data and after that for the data blocks as well. So that all the old encryption keys get replaced by new ones.