DESIGN DOCUMENT CS628A:COMPUTERSYSTEM SECURITY ASSIGNMENT 1

Harshit Gupta
18111020
guptah@iitk.ac.in

Somesh Kumar Jaishwal
18111071
someshk@iitk.ac.in

March 6, 2019

Components Involved

1. For each file, we have data structures:
   (a) FileBlock(Stores the content of the file and it's HMAC.)
   (b) FileData(Stores a list of keys used for encypting each block of the file and a list of addresses of all the FileBlock in the file. There is a new block for each append. Also stores HMAC.)
2. For each user, we have data structures:
   (a) User(Stores the name of user, password of user, RSA private key and HMAC).
3. For each User-File pair:
   (a) UserFile(Stores the location of file(FileData), password to decrypt the file and HMAC).

Part 1

InitUser(username string,password string)

1. Generate RSA key pair.
2. Generate E=argon2(password,salt="username")
3. Populate User data structure with username, password, RSA private key and HMAC=0. Calculate HMAC(User data structure, E) and store it in HMAC field of User data structure.
4. Store the RSA public key in KeyStore with key-value pair as (username, RSA public key).
5. Encrypt User data structure using E as key and store it at location hash(password+username).

GetUser(username string, password string)

1. Obtain hash of (username+password).
2. Retrieve value from DataStore with key as hash of password, if not found generate error.
3. Decrypt obtained value from above step using E=argon2(password,salt="username"). If decryption doesn't give the object in correct format, return error.
4. h=User.HMAC.
5. User.HMAC=0. Generate h1=HMAC(User,E).
6. if h==h1, return User data structure pointer otherwise return error.

StoreFile(filename string, data byte[])

1. Addr=hash(username+ password + filename) and E=argon2(password+username,salt="filename").
2. If entry for this Addr exists in DataStore, then get Encryption key(E1) and location(Addr1) of FileData from that location(using E, create new file as described below if unable to) otherwise, generate Addr1=hash(random bytes) and E1=argon2(password+filename,salt=random bytes) and store at Addr(populated UserFile structure) using E for encryption and HMAC.
3. Addr2=hash(random bytes) and E2=argon2(random bytes,salt=random bytes). Create a list and store Addr2 in this list. Store this list, E2 and generated HMAC(as before) at Addr 1 using E1.
4. Store data at Addr2 using E2 for encryption and HMAC

LoadFile(filename string, data byte[])

1. Addr=hash(username+password+filename) and E=argon2(password+username,salt="filename").
2. Fetch, decrypt and verify UserFile data structure at Addr using E. Get location of FileData(Addr1) and Encryption key(E1) from UserFile.
3. Load FileData from Addr1, decrypt and verify using E1.
4. Load FileBlock from each location in the list in FileData, decrypt and verify(using key stored in FileData) and keep adding data in it to the "final string". Return this final string.

AppendFile(filename string, data byte[])

1. Load FileData as done in LoadFiles function.
2. Addr2=hash(random bytes), E2=argon2(random bytes,salt=random bytes).
3. FileBlock=(data, it's HMAC). Encrypt this FileBlock using key E2 and store at Addr2.
4. Append Addr2 and E2 to their respective lists in FileData generate HMAC again, encrypt and store back.

Part 2

ShareFile(filename string, recipient string)

1. Get UserFile data structure as in LoadFile function.Create a copy of this, say "Share".
2. Calculate and store Share.HMAC.
3. Generate three random unique ids(UUIDs).(say seed1, seed2 and seed3).
4. Encrypt share with E1=argon2(seed2,salt=seed3) and store at hash(seed1).
5. Encrypt these 3 UUIDs with Receiver's public key and sign with sender's public key. Store the sign in Datastore at hash(encrypted message) and return the encrypted string.

ReceiveFile(filename string, sender string, msgid string)

1. Decrypt msgid using Recipient's Private key and then verify using Sender's public Key(sign stored at hash(msgid)) to get the three UUIDs. Use this UUIDs to get the UserFile data structure. Return error if verification fails.
2. Addr=hash(username+password+filename) and E=argon2(password+username,salt="filename").
3. Generate HMAC of UserFile data structure using E and store it in HMAC field.
4. Encrypt the final UserFile structure using E and store at Addr.

Part 3

RevokeFile(filename string)

1. Load the file and delete all the FileData structures and FileBlock structures stored in Datastore.
2. Store the file by calling StoreFile on this data.

Testing

1. Loaded and Retrieved the user. Swapped the records of two different users.
2. Checking if Storing, retrieving, appending functionalities are working correctly or not.
3. Sharing the file and checking if the load file for Receiver gives the same file.
4. Checking that after Revoke, other user must not be able to load the file.
5. Mutating the file stored in memory and then checking if our implementation returns error or not.