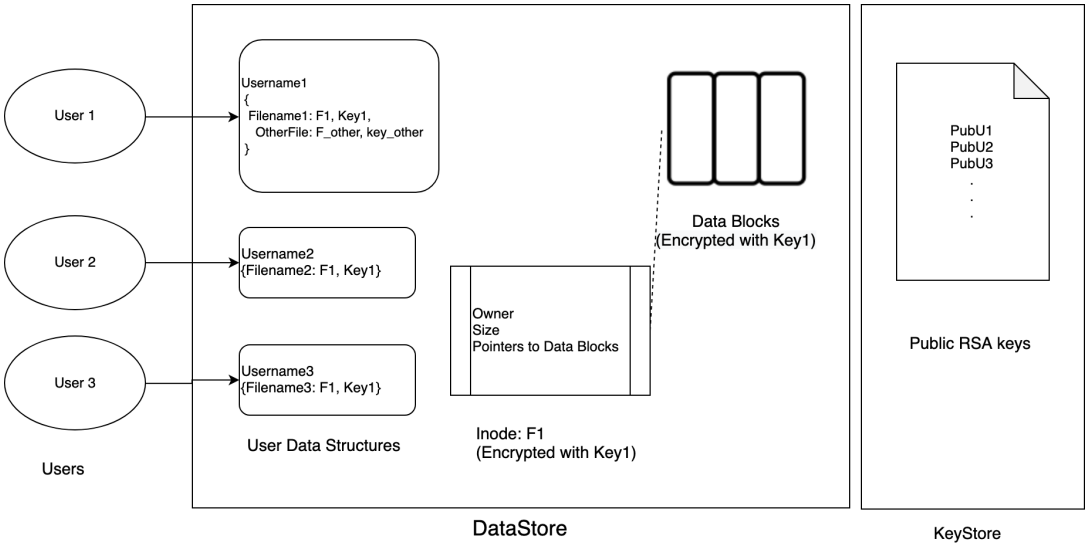
CS628 Homework 1 - Design Document

Anant Prakash (160110) Sriram Varun Vobilisetty (160706)

1 Overview



2 Data Store

2.1 Storage Abstraction layer

Our storage server is malicious, hence we always require that the data we store/retrieve using DatastoreSet(), DatastoreGet() and DatastoreDelete() has confidentiality and integrity maintained. Hence, let us define secure versions of the above which does exactly that.

SecureDatastoreSet(SecretKey, DataKey, DataValue)

```
unknownLocation = Argon2Key(SecretKey+DataKey, nil, KeyLen)
hmac = HMAC(SecretKey, unknownLocation+DataValue)
encryptedData = CFBEncrypter(SecretKey, fixedIV, hmac+DataValue)
DatastoreSet(unknownLocation, encryptedData)
```

SecureDatastoreGet(SecretKey, DataKey)

```
unknownLocation = Argon2Key(SecretKey+DataKey, nil, KeyLen)
encryptedData = DatastoreGet(unknownLocation)
storedHmac, DataValue = CFBDecrypter(SecretKey, encryptedData)
calculatedHmac = HMAC(SecretKey, unknownLocation+DataValue)
if Equal(storedHmac, calculatedHmac){
return DataValue
} else {return nil}
```

SecureDatastoreDelete(SecretKey, DataKey)

```
unknownLocation = Argon2Key(SecretKey+DataKey, nil, KeyLen)
DatastoreDelete(unknownLocation)
```

2.2 User Data Structure

UserData

```
SecretKey // KDF (Argon2Key) generated string
PrivateKey // RSA private key
OwnedFiles // map of filenames to {InodeAddress, FileSecretKey}
```

2.3 File Storage Scheme

2.3.1 Inode Structure

Inode
FileSize
DirectPointers * 12
SingleIndirect * 1
DoubleIndirect * 1

In the Inode of a file, we store file size(in blocks), 12 direct pointers(DataStore keys corresponding to file data blocks), one indirect pointer (DataStore key corresponding to a block containing direct pointers), and one double indirect pointer(DataStore key corresponding to a block containing indirect pointers)

Anyone with access to the File can access the Inode as they will have in their OwnedFiles map the corresponding InodeAddress and FileSecretKey used to encrypt the Inode as well as the actual FileData.

2.4 `InitUser(username string, password string)`

- We generate a SecretKey using Argon2(password, salt, keyLen). Where the *salt* could be derived from password. Save the SecretKey in the UserData struct.
- Generate a RSA key pair, register the publicKey on the Keystore. Save the privateKey in the UserData struct.
- Initialize empty values for the other fields of UserData struct. Then call SecureDatastoreSet(SecretKey, username, UserData).

2.5 `GetUser(username string, password string)`

- We compute the SecretKey using Argon2(password, salt, keyLen). Where the *salt* could be derived from password. Save the SecretKey in the UserData
- Use SecureDatastoreGet(SecretKey, username) to get the UserData struct.

2.6 `StoreFile(filename string, data []byte)`

2.6.1 New FileName

- Create following random keys with helper function given:
  - (Key_1) to be used as InodeAddress, (Key_2) to be used as FileSecretKey
- If entry for FileName is not found in OwnedFiles map of user, create an entry {FileName: Key_1 , Key_2}.
- Create InodeBlock. Determine file size in blocks and store in field 'FileSize'. Sequentially, keep generating as many random keys(DataStoreKeys) as there are blocks in file, and calling SecureDatastoreSet(Key_2,DataStorekey_i,DataStoreBlock_i) and storing the DatastoreKeys in the corresponding pointers (first direct, then SingleIndirect, then DoubleIndirect) until the file is stored entirely.
- Call SecureDatastoreSet(Key_2,Key_1,InodeBlock) to store the Inode block in Datastore and then call SecureDatastoreSet(SecretKey, username, UserData) to store the updated UserData block.

2.6.2 Existing FileName

- Do everything similar to LoadFile, except the last step where we call SecureDatastoreSet() instead of SecureDatastoreGet() to rewrite/write the file data.

2.7 `AppendFile(filename string, data []byte)`

- Get and decrypt the Inode of the file by using the keys corresponding to FileName (Key_1 and Key_2) from the User's OwnedFiles map.
- Check filesize, go to pointer corresponding to the correct block. Generate random DatastoreKey(Key_3), call SecureDatastoreSet(Key_2, Key_3, DataBlock). Populate the pointer with Key_3. Call SecureDatastoreSet(Key_1, Key_2, InodeBlock) to update the InodeBlock.

2.8 `LoadFile(filename string, blockOffset integer)`

- Get and decrypt the Inode of the file by using the keys corresponding to FileName (Key_1 and Key_2) from the User's OwnedFiles map.
- Get the required block of file by accessing the correct pointer (getting the block's DatastoreKey, say Key_3) from the Inode and calling SecureDatastoreGet(Key_1,Key_3).

2.9 `ShareFile(filename string, userToShare string)`

- First, we RSAsign the fileEntry i.e {InodeAddress, FileSecretKey} corresponding to the filename with the sender's private key, and append the sign to the fileEntry. Call this the payload.
- Then we generate two random keys $Key1$ and $Key2$. We use $Key1$ as a symetric key to encrypt the payload and use $Key2$ as the key(address) to securely store it in the dataStore.
- Then we send to userToShare $E(Key1||Key2)(\textit{sharing message})$ RSAencrypted with userToShare's public key.

2.10 `ReceiveFile(filename string, sharedFromUser string, sharing string)`

- RSADecrypt *sharing* using self private key and load the payload from dataStore using SecureDatastoreGet ($Key1$, $Key2$).
- Obtain sharedFromUser's public key, break up the payload into fileEntry and sign, and RSAverify the sign to confirm if the data was sent from sharedFromUser.
- If it is, do UserData.OwnedFiles[filename]=fileEntry and store the state of UserData using SecureDatastoreSet().

2.11 `RevokeFile(filename string)`

- We will copy over the old file data with a new InodeAddress and FileSecretKey.
- Generate two random string with the helper functions to use as InodeAddress and FileSecretKey.
- Create a new inode as described in StoreFile, Use the LoadFile iteratively to copy over old data to new inode. Delete old inode and data blocks.
- Store the new InodeAddress and FileSecretKey in the UserData in corresponding maps. Store UserData back with SecureDatastoreSet.