CS628A Assignment - Design Document

Aakrati Jain (19111001) Apoorva Jain (19111016)

August 7, 2020

Property 1 InitUser(UserName string, Password string)

Fetch userName and passWord. Generate public and private key for the user using RSA. Use the userName as key and store the public key in KeyStore as value. Generate a random salt . Use password and salt as parameters to argon2key to prevent brute force attack function which will return a hashedPassword . Use this hashedPassword as a key to store the user information in the DataStore. Encrypt this user information using symmetric key encryption to achieve confidentiality with key as symmKey. Generate the HMAC to maintain integrity of the user information. Use the passWord as a key to store the userName, symmKey, salt and the HMAC in the DataStore.

```
userPassword {
    "salt":
    "HMAC":
}

hashedPassword {
    "username":
    "privateKey":
    "accessibleFiles":{
        "FileName1":uuid1
        "FileName2":uuid2
                   :
        "FileNameN":uuidN
    }
}
```

Property 2 GetUser(UserName string, Password string)

If the UserName and PassWord are correct then the HMAC will verify if the UserData in the DataStore is corrupted or not. If UserName and PassWord are correct and UserData is not corrupted, UserStructure will be populated from the DataStore else error will be returned .

Property 3 StoreFile(filename string, data []byte)

Generate a uuid. Use SHA256 to generate a hashedUUID. Use this hashedUUID as a key to store the file data in the DataStore as mentioned in structure . Encrypt this file data using symmetric key encryption to achieve confidentiality with key as symmKey. Generate the HMAC to maintain integrity of the file data. Use UUID as key to store the symmKey and HMAC in the DataStore. Add fileName and uuid in the user information.

```
uuid {
    "symmKey":
    "HMAC":
}

hashedUUID {
    "1":uuidDoubleIndirect1
    "2":uuidDoubleIndirect2
       :
    "n":uuidDoubleIndirectn
}

uuidDoubleIndirect {
    "1":uuidBlock1
    "2":uuidBlock2
       :
    "n":uuidBlockn
}
```

Property 4 AppendFile(filename string, data []byte)

HMAC verifies whether the file data is corrupted or not. A new uuidBlock is allocated and added to uuidDoubleIndirect.

Property 5 LoadFile(filename string, blockOffset integer)

The user obtains the uuid corresponding to the fileName from UserData. Existence of the file is verified by uuid as key in DataStore. The calculation of the location of blockOffest which DoubleIndirect and Block to access will be done further. HMAC verifies if the file data is corrupted or not.

Property 6 ShareFile(filename string, recipient string) and ReceiveFile(filename string, sender string, msgid string)

Sender can only send the files whose entry is present under accessibleFiles in its own user data. Sender would encrypt the file data using its own privateKey which will be decrypted at the receiver using the publicKey of the sender. Receiver adds the file by creating a new fileName and storing the uuid as its value in its own user data. All the changes are made directly in the DataStore.

Property 7 RevokeFile(filename string)

Generate a new uuid and a new SymmKey. Decrypt the file using old SymmKey and encrypt it using new SymmKey to prevent previous users from accessing the old data . Use SHA256 to generate a new hashedUUID. Use this new hashedUUID as a key and copy the value of old hashedUUID. Delete the old uuid and old hashedUUID from the DataStore. Update the old uuid with new uuid corresponding to the fileName in user data who is revoking. Generate the HMAC to maintain integrity of the file data. Use UUID as a key to store the symmKey and HMAC in the DataStore.