CS628A - Assignment 1

Apoorva Jain
19111016

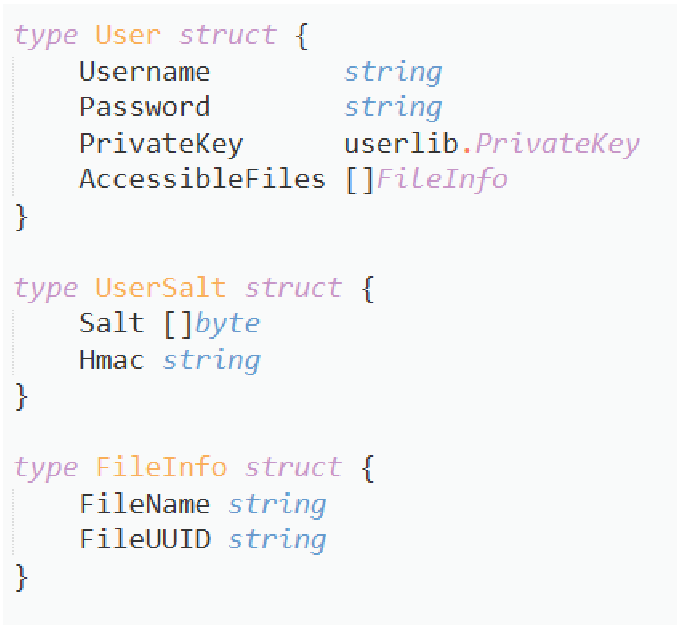
Aakrati Jain
19111001

<u>Property 1 InitUser(UserName string, Password string)</u>

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 returns a hashedPassword . Use this hashedPassword as a key to store the user information in the DataStore. Generate the HMAC { to maintain integrity } of the user information. Use the username_passWord as a key to store the salt and the HMAC in the DataStore.

<u>Property 2 GetUser(UserName string, Password string)</u>

If the UserName and PassWord are correct then the HMAC verifies 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 .



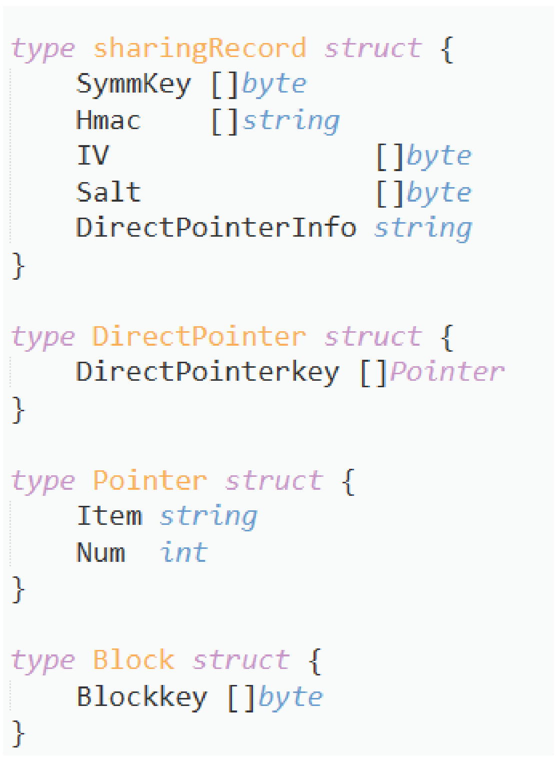
```
type User struct {
    Username       string
    Password       string
    PrivateKey     userlib.PrivateKey
    AccessibleFiles []FileInfo
}

type UserSalt struct {
    Salt []byte
    Hmac string
}

type FileInfo struct {
    FileName string
    FileUUID string
}
```

<u>Property 3 StoreFile(filename string, data []byte)</u>

Generate a uuid. Use SHA256 to generate a hashedUUID. Use this hashedUUID as a key to store the sharingRecord in the DataStore. An array of pointers is maintained which is stored corresponding to the DirectPointerkey. The Pointer further consist of the block UUID as Item and offset as Num. The block UUID points to the block which consist of the file data. Encrypt each block of the 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 FileUUID in the user information.



```go
type sharingRecord struct {
    SymmKey []byte
    Hmac    []string
    IV                  []byte
    Salt                []byte
    DirectPointerInfo string
}

type DirectPointer struct {
    DirectPointerkey []Pointer
}

type Pointer struct {
    Item string
    Num  int
}

type Block struct {
    Blockkey []byte
}
```

Property 4 AppendFile(filename string, data []byte)
HMAC verifies whether the file data is corrupted or not. A new block UUID is created corresponding to which the data is stored. Also a Pointer is created which consist of the block UUID as Item and offset as Num. Append this pointer in the DirectPointerkey .

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 Num in the Pointer acts as the block offset. HMAC verifies if the file data is corrupted or not .

Property 6 ShareFile(filename string, recipient string) && 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 FileUUID using the receiver's public key { to maintain confidentiality } which is decrypted at the receiver using its own private key. But before sending the encrypted version, the UUID is signed using sender's private key { to maintain authenticity } which is verified at receiver's side using sender's public key. Receiver adds the file in his AccessibleFiles. All the changes are made directly in the DataStore.

Property 7 RevokeFile(filename string)
Generate a new uuid. 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 file uuid with new file uuid in AccessibleFiles in user data who is revoking. Use UUID as a key to store the symmKey and HMAC in the DataStore.