CS 628: PROGRAMMING ASSIGNMENT-1(Design Document)

```
struct file{
        rk=empty array which will be initialized later
        key=location of the index file}
struct user
{
    username
    RSA public key
    RSA private key
    password
    An array of the the structure struct file
    struct file
}
```

Inituser(): key=Argon2(username,password)
key1 is a part of the key
key2 is the remaining part of the key
This above user structure is used to create an user record in the datastore and key store with some cryptographic modification. In datastore following information are stored as an user information:
1. key1 is used as an key location of the map.
2. e_user: Encrypt(key2,user)
3. hmac: HMAC(e_user,key2)(for integrity check)
4. list1=List is created with [value,hmac]
5. value:list1

Therefore, key:key1,value:list1
Key is the location of the user record and the corresponding record is the value.
When the user invokes the inituser with username and password the above informations are stored in datastore.

The username and public key is stored in the keystore.

Get_user(): The user provides username and password.
The following are the steps that will be performed when this function is invoked:
1. key=Argon2(username,password)
2. key1 and key2 is derived from the key.
3. Go to the key1 in datastore.
4. Fetch the value field and hmac field content.
5. if HMAC(value,key2)!=hmac then an error is returned else v=Decrypt(value,key2) and v is used to populate the user.

Storefile():
keyf=Argon2(username,password,filename)
A new random key is generated.
A slice here is an index file of the file.
This slice will be encrypted by the random key.
A encrypted slice content is created which will be stored in the location which is keyf.
Initially, index value is 0 when the file is first created.
keyfile=Argon2(keyf+index of the array which will point to this content)
slice[0]=keyfile(location at which the file is stored)
Hmac:(encrypted slice,randomkey)
value:list[encrypted slice,hmac]
The keyf is the location where the value is stored.

e_file=Encrypt(filecontent,random key+index of the array which will point to this content)

hmac: HMAC(e_file,randomkey+index of the array which will point to this content)(for integrity check)
value:list[e_file,hmac]
Therefore, key:keyfile,value:list[e_file,hmac]
Keyfile is the location of the file record and the corresponding value is the filecontent.

Then, user data structure is already populated. The rk is added in the array rk of user struct and updated in the datastore accordingly.

LoadFile():
keyf=Argon2(username,password,filename).
Keyf is the location in datastore where the index file is placed. Random key is derived from the user data by decrypting the rk field in the user data.
Then HMAC(encrypted slice,randomkey) is calculated. If it matches with the content of the hmac field then no corruption of data took place. Otherwise we will return error. Then the value is decrypted using the random key. Decrypting this gives us the array with elements having locations where each file chunk is located.
For i=0 to sizeof(array/slice)-1:

keyd=Argon2(randomkey+i)

Fetch e_file from the location stored in the array[i].

Calculate HMAC(e_file,keyd).If it matches with the content of the hmac field then no corruption of data took place. Otherwise we will return error. Then the value is decrypted using the random key and the file is read.

Appendfile():

If the file is appended by the file creator:
location of the index file=Argon2(username,password,filename)

If the file is appended by one of collaborator:
location of the file = is retrieved by first reading the array of the structure file in the user structure where the location of the file is stored corresponding to the filename. Random key is derived from the user data by decrypting the rk field in the user data.

Then HMAC(encrypted slice,randomkey) is calculated. If it matches with the content of the hmac field then no corruption of data took place. Otherwise we will return error. Then the value is decrypted using the random key. Decrypting this gives us the array with elements having locations where each file chunk is located.

keyfile=Argon2(keyf+sizeofarray)
slice[sizeofarray]=keyfile(location at which the file is stored)
Hmac:(encrypted slice,randomkey)
value:list[encrypted slice,hmac]
The keyf is the location where the value is stored.
e_file=Encrypt(filecontent,random key+index of the array which will point to this content)
hmac: HMAC(e_file,randomkey+index of the array which will point to this content)(for integrity check)
value:list[e_file,hmac]
Therefore, key:keyfile,value:list[e_file,hmac]
Keyfile is the location of the file record and the corresponding value is the filecontent.

<u>Part2:</u>

sharefile():

The sender needs to send the location of the index file and the key to decrypt the file on an untrusted channel such that the receiver will be able to check the integrity of the content sent and also verify the identity of the sender.
Struct sharing{
Key(location of the index file)
key used to encrypt the file
}
The sender will encrypt this structure with the public key of the receiver. Then the sender will digitally sign this cipher. The cipher and the signature together forms the msgid. This msgid is send to receiver.

Receivefile():
The receiver receives the string and verifies the signature of the sender which verifies the integrity of the sent content. The cipher is then decrypted with the private key of the receiver. This enables the receiver to read the content which is the index file location in the datastore and the key used to encrypt that index file. This content is the new entry in the array of struct file in the user structure corresponding to the receiver.

Revoke():
In the revoke mechanism the creator of the file will go to the datastore location where the file is saved. Decrypt the file using the key and the index file and the file content will be encrypted with a different random key which will unable the other collaborators to further access the file.