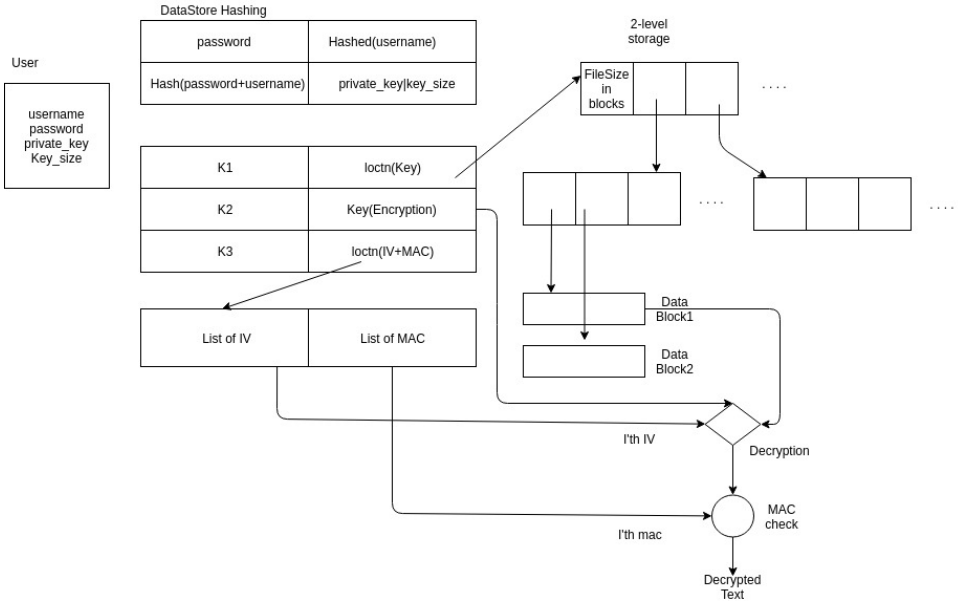
Assignment 1

Debdan Bhandari 19111026

Raunak Kumar 19111069



File structure is stored using a 2 level inode storage. Every file can be accessed using three keys, k1, k2 and k3. k1 stores the location of the file. k2 stores the symmetric encryption key and k3 holds an adress to the IVs and MACs of the file blocks. After every decryption, the IV corresponding to the block is changed to incorporate randomized encryption.

Important Notations:
1. password referred here are all argon2 passwords unless mentioned as pwd.
2. [key] refers to the value stored corresponding to key string in the database.
3. a|b refers to appending a and b.
4. IV denotes Initialization vector. IVMAC is a list of IV's followed by a list of MACs

The functions are given as

Struct user

The user structure must contain 4 values, those are
1. Username
2. Argon2 Password
3. Private key
4. Key size

InitUser(username string, password string)

create a new user structure. put the username, argon2 password and the privatekey in the structure. Keep an hashed version of username at [password] location of datastore. Also keep the (privatekey|Key size) in the location obtained as [hash(username+password)] in the datastore

Getuser(username, password)
Generate argon2 password from user password. Hash the username and check it with [password]. If hashed usernames don't match, return error. else go to location [hash(username|password)] and fetch the (privatekey|Key size) pair. Store the privatekey and key size into the user structure

StoreFile(filename, data)
Create three keys as follows
k1 = hash(filename|password)
k2 = hash(filename|privatekey)
k3 = hash(filename|password|privatekey)
total_num_blocks = 800+(800/(blocksize/keysize))+1 ///number of data blocks + 2nd level inode blocks + 1st level inode block
IVMAC list = [random IVs for all the data blocks] + [MAC for all the data blocks of file]
store the values at locations as
DatastoreSet(K1, random1)
DatastoreSet(K2, random2)
DatastoreSet(K3, random3)
DatastoreSet(random3, IVMAC)
num_keys_per_block = (blocksize/keysize) //number of addresses per block
num_blocks = (size(data)/blocksize) //number of blocks of given file
root = num_keys_per_block number of random numbers // the root inode
root[0]=num_blocks // put size of current file at first location of root
The inode structure is created as follows
for i in 1 to num_keys_per_block:
{
value = num_keys_per_block number of random numbers // second level of inode
[root[i]]=value //put the ith second level inode in ith location of first level inode
}
for i in 0 to num_blocks:
{
$first\_level\_index = i/num\_keys\_per\_block + 1$
$second\_level\_index = i\%num\_keys\_per\_block$
go to root[first_level_index] fetch value, then go to value[second_level_index] to fetch block_address.
encrypt block using i'th IV and put in the block address location. Also calculate the MAC value and store at i'th location of MAC list
}

AppendFile(filename, data)
Generate k1,k2,k3 and Get file key, encryption key, IVMAC list using K1, K2, K3
$file\_size = root[0]$
$numblocks = size(data)/blocksize$
for i in 0 to numblocks:
{
$first\_level\_index = file\_size/num\_keys\_per\_block + 1$
$second\_level\_index = file\_size\%num\_keys\_per\_block$
encrypt the block using IV[file_size] and put it in the location obtained from inodes. Put the mac at MAC[file_size]
file_size+ = 1
}
Put root[0] =file_size

Loadfile(filename, offset)
Generate k1,k2,k3 and Get file key, encryption key, IVMAC list using K1, K2, K3
$first\_level\_index = offset/num\_keys\_per\_block + 1$
$second\_level\_index = offset\%num\_keys\_per\_block$
get value at root[first_level_index] and then get block from value at [second_level_index].
Decrypt block using IV[block offset]
change the IV value corresponding to the block and re-encrypt
check MAC of obtained block with MAC[block offset], if match then return block, Otherwise return error

Sharefile(filename, username)
get public key of receiver and get k1, k2, k3 of sender.
Get file key, key for encryption and IVMAC location using k1, k2, k3 sign and encrypt (key, key for encryption, IV location) using RSAOAEP send to username

Receivefile(filename, username)
Get (key, key for encryption, IV location) from received data
Generate k1,k2,k3 for receiving user using new filename and receiver's argon2 password and privatekey
put the values as
DatastoreSet(K1, file key)
DatastoreSet(K2, key for encryption)
DatastoreSet(K3, IVMAC location)

Revoke(filename)
Load the file from current location and store at a new location with new key and IV values
data = loadfile(filename)
storefile(filename,data)