CS628a Assignment 1
Design Document

Suraj Gour
19111094

Amiya Tripathi
19111010

The document discribes how all the 8 properties are satisfied for the storage server using the provided library and methods

Property 1 : User Creation :
Input : Username, Password

var User struct { uname, password, rsa.key (public/private key pair), file_key as map (key: "filename", value:"secretkey") }

- Store Username, Password in *User
- Generate rsa key pair and store it in *User
- create argon2key using password as pass and username as salt ->Convert it to string ->use first 10 bits for key {KEY} storage key of datastore
- Serialise *User and store it in []byte and prepend random iv {VALUE}
- Create argon2key using password and empty string and use it as encryption key for User struct
- Use aes.key and iv to encrypt data block using CFBEncrypt
- Create HMAC of encrypted User sturct using encryption key and append HMAC block to Encrypted User block

Property 2 : Get User
Input : Username, Password

- Generate the argon2key using password as pass and username as salt ->Convert it to string ->use it to get encrypted User from datastore.
- Create argon2key using password and empty string for decryption.
- Verify HMAC using decryption key.
- Use this key and first block as iv to decrypt data blocks {possible only if key is generated using correct password}

- Deserialise data bocks to get *User

Property 3 : Storing Data
Input : Username, Password

- Create UUID and store (fname, uuid) in file_key map
- remove '' from uuid to get file encyption key
- Check data length to be divisible by blocksize
- Create indirectInode of array of uuid
- Create inode of array of uuid and a inodeID (uuid)
- Append inode ID to indirectInode then store inode using inodeID
- Store indorectInode using first 8Bytes of encryption key
- Call Appendfile and pass whole data as argument

Property 4 : Append Data
Input : fname,data

- Use fname to get encryption key from file_key map
- Use first 10 Bytes of encryption key to retrieve indirectInode
- Traverse to the last uuid of indirectInode ->get the last inode
- Traverse to the last uuid of the inode and append new uuid
- Slice first blocksize from data ->create 16 Byte ramdom iv ->encrypt data block using iv then prepend iv on encrypted data
- Create hmac using encryption key and append hmac to encrypted data
- Store this whole block (iv + encrData + hmac) using new uuid recently stored to the inode
- Repeat this process till all data is stored

Property 5 : Load File
Input : fname, offset

- Use fname to get encryption key from file_key map
- Retrieve indirectInode using first 8 Bytes for encryption key
- Traverse to the indirectInode offset (offset/blockspernode) and get inodeID
- Get inode from inode ID and traverse to inode offset(offset%blockspernode) to get blockID
- Get data block from blockID

- Slice hmac from last 32 Bytes of encrypted block
- Verify hmac using encryption key
- Slice iv from first 16 Bytes of encryted block
- Decrypt data using iv and encryption key
- Decrypted block

Property 6 : Share File
Input : fname, receiver

- Use fname to get encryption key from file_key map
- Sign key using own private key
- Enrypt encryption key using reciepent public key
- Append sign to encypted key
- Share this as message 'm'

Property 7 : Receive File
Input : fname', sender, message

- Slice sign (last 256 Bytes) from message
- Decrypt key (first 256 Bytes) using own private key
- Verify sign using sender public key
- On succesful verification parse uuid from decrypted key and store it on file_key map

Property 8 : Revoke Access
Input : fname

- Use fname to get encryption key {m} from file_key map
- Retrieve whole file from data store using loadfile till the last uuid of last inode
- Again call store file using oldData and same filename (this process will restore the data using new key)