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 aes.key using password
- Use aes.key and iv to encrypt data block using CFBEncrypt
- Create HMAC of {VALUE} using argon2 key from step3 and append block on {VALUE}
- Store {key}{VALUE} on server

Property 2 : Get User
Input : Username, Password

- Generate the argon2key using password as pass and username as salt ->Convert it to string ->use first 10 bits for key {KEY}
- Use {KEY} to get {VALUE} from DataStore
- Verify HMAC using argon2 key from step1
- Create aes.key using password

- 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

var User struct { salt, 'list of allowed users' }

- Create random key{m} and store (fname, m) in file_key map
- Create salt and add owner to the allowed user list.
- Use first 10 Bytes of {m} and store (m, SharingRecord) in DataStore.
- Use Sharing Record to create hash and use its first 10 bits as {KEY}
- Use {m} to generate aes.key and encrypt data in a similar way. (as used in Property 1) (Using CFB)
- Create HMAC using {m} as key and append to data block.
- Store (KEY, data) in DataStore.

Property 4 : Append Data
Input : fname,data

- Use fname to get secretkey {m} from file_key map.
- Use first 10 Bytes of {m} to retrieve SharingRecord.
- Create hash of SharingRecord and first 10 Bytes to get file block.
- Verify HMAC using {m}.
- Create aes.key using {m}.
- Use last block of cipher text as iv and {m} as key to encrypt further blocks using CFB method.

Property 5 : Load File
Input : fname, offset

- Use fname to get secretkey {m} from file_key map
- Create hash of SharingRecord and first 10 Bytes to get file block
- Verify HMAC using {m}
- Create aes.key using {m}
- Use offset-1 block ciphertext is iv and aes.key to decrypt blocks from offset and futher

Property 6 : Share File
Input : fname, receiver

- Use fname to get secretkey {m} from file_key map
- Add reciever to Sharing Record
- Create hash of new sharing record and use its first 10 Bytes as {KEY}
- Relocate data to new {KEY}
- Retrieve recievers public key from keystore
- Use it to encrypt {m}
- Return encrypted {m} as msg_id

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

- Decrypt msg_id as {m} using rsa.privateKey
- Update (fname', m) in file_key map
- Use first 10 Bytes of {m} to get Sharing Record
- Use first 10 Bytes of hash of Sharing Record to get data block
- verify HMAC using {m}
- Create aes.key using {m} and first data block as iv to decrypt data

Property 8 : Revoke Access
Input : fname

- Use fname to get secretkey {m} from file_key map.
- Use first 10 Bytes of {m} to get Sharing Record.
- Use first 10 Bytes of hash of Sharing Record to get data.
- Create a new key {m'}.
- Now Remove all other users from Sharing Record and store it using {m'}.
- Verify HMAC and Decrypt data using {m}
- Encrypt again using {m'} and calculte HMAC using {m'}.
- update (fname, m') in file_key map.