Computer System Security - CS 628
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Abhishek Yadav , Naman Jain
Roll Number: 160040 , 160427
Date: March 1, 2019

---

1 Data Structures

1.1 struct User (AES encrypted using Argon2Key(Password))

This would have the following fields :

- username
- Argon (password)
- Private_Key of the User
- Map [ ][ ] from HMAC of (Username, Filename) → Public_Keys of all files that this User possess
- Map [ ][ ] from HMAC of (Username, Filename) → Private_Keys of all files that this User possess

1.2 struct File (RSA encrypted)

This would have the following fields :

- public_key of file
- Initialization Vector IV (Random String)
- "AES Key" for Struct File_data
- locations [ ] = list with pointers to File-data
- hash_locations [ ] = list with HMAC(address, content of stored pointer locations)

1.3 struct File-data (AES encrypted using "AES Key")

This would have the following fields :

- data
- public_key of file

1.4 username-User Look-up table

This is a Map[ ][ ] from Argon(username) to (pointer to User structure).

1.5 username_filename-File Look-up table

This is a Map[ ][ ] from HMAC(username,filename) to (pointer to File structure).

2 Function Implementation

2.1 InitUser

- It initializes the User structure for that user by generating public-private pair using *GenerateRSAKey()*.
- User structure is stored in DataStore using needed fields.
- Entry in username-User Look-up table is inserted.

2.2 GetUser

- It finds the Argon_Key corresponding to the entered Username and password using the function *Argon2Key()* with salt as "username" in both the cases.
- It finds to the location for the User using the Look-up username User (correspondingly check if the location is NULL)
- It decrypts the AES based on the Argon2Key obtained from the password and validates the login using the field "Argon(password)". The IV used is the Argon2Key(password) itself.
- Then, it returns the pointer "struct* User" if login is validated or gives an error if login fails.

2.3 StoreFile

- It generates an RSA key (Public and Private Key) for the file using *GenerateRSAKey()* and populate the corresponding entry in the Map of struct User, it also generates a random IV and AES key for encryption of subsequent appends of blocks.
- It determines the "key" = HMAC(username,filename) with which it would be put on the data store.
- It creates and entry of type "struct File" and "struct File_data" using this "key" and RSA keys.
- It encrypts the corresponding entry of "struct File" using public key of the file and "struct File_data" using the generated IV, AES key and and stores the corresponding parameters it itself.

2.4 LoadFile

- It locates the file using HMAC(username,filename) and look-up table(throws an error if no such entry is found).
- It verifies the integrity of the file using the public_Key of the file which was protected in struct User.
- Then it iterates over all the Locations to find out the corresponding appended content, decrypts them using IV and AES key stored and appends the plain text obtained and finally returns the complete file.

2.5 AppendFile

- Locates and decryptes the corresponding entry using the look-up from struct User(throws an error if unable to locate or unable to check integrity using the protected public key of the file).
- Generates a HMAC (username, filename + '_' + "SizeOf(Locations)") and uses this as the key to store the appended content(This would work because filenames can't have '_').
- Creates the corresponding entry of struct File_data using value provided.
- Encrypts the entry using IV and AES Key stored in struct File.

2.6 ShareFile

- Locates the corresponding entry using the look-up table(throws an error if unable to do so).
- Creates a message auth token of the type (Location of File, Public Key of shared User, Private_key of File encrypted by the public key of shared User, Public Key of the file). Location of file is HMAC(current User, filename).
- Uses the method *RSAsign()* to sign this usgin the private key of the current user and returns this token.

2.7 ReceiveFile

- Verifies the token using the provided username and the function *RSAVerify()* and obtains the private key for the file.
- Creates the corresponding entries in the lookup table for private key and public key for this file using "key" as HMAC(username,filename) where filename is new filename provided by the shared user.
- Creates the entry for this filename in the look-up table for location, puts "location" at map[HMAC(usrname,filename)].

2.8 RevokeFile

- If it is unable to find the file or unable to verify the integrity, it throws an error.
- Decrypts the struct and generates a new pair of RSA keys using *GenerateRSAKey()* and replaces the corresponding entries in the lookup of public and private keys of struct User.
- Re-Encrypt the struct usgin new key and puts back the content at the same location.

3 Tests

- Try to access to shared file after it has been revoked by the user.
- Append to the file and check for its correctness.
- Try to load a file that does not belong to the user.
- Login with invalid credentials.
- Try to receive a file that has not been shared.
- Check correctness for a big file size.
- Try to initialize a user with an already existing username.