CS3031 Advanced Telecommunications
Project II

Séamus Woods
15317173

05/04/2019

0.1 Specification

The objective of the excercise is to develop a secure cloud storage application for Google Drive. For example my application will secure all files that are uploaded to the cloud, such that only people that are part of my 'Secure Cloud Storage Group' will be able to decrypt my uploaded files. To all other users the files will be encrypted. I was required to design and implement a suitable key management system for my application that will allow me to share files securely, and add or remove users from my group.

0.2 Implementation

The easiest way to explain my design and implementation is just by talking through the execution of the code below. I have successfully managed to implement all features listed above.
For a brief overview, there are three main files for this program. *drive.py*, *client.py* and *server.py*.

drive.py is the admin side of the application, here the admin can encrypt/decrypt all the files in the drive, add/remove users to the group and see what files are in the drive.

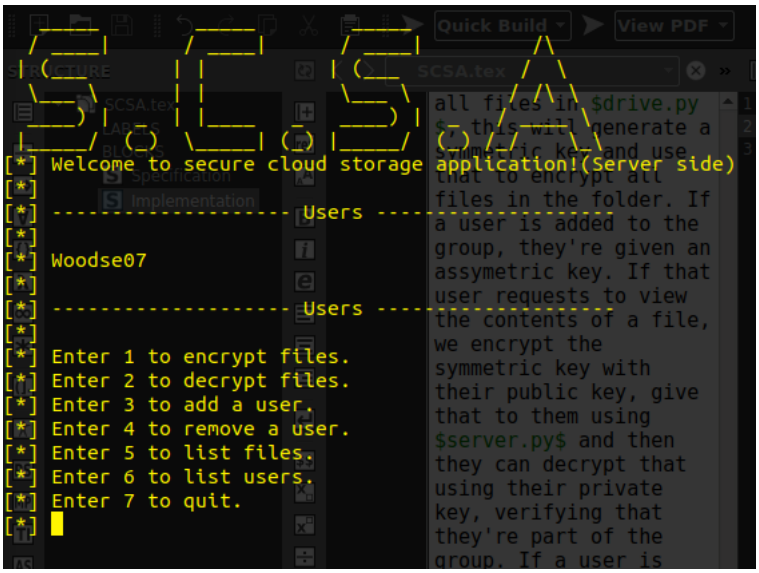
client.py is intended to be that part of the application that the client uses. Here they can login, see what files are in the drive, view the contents of files and upload/delete files from the drive.

server.py is a flask server, which listens for client.py to send it a username, and will return the encrypted version of the symmetric key for that user to the client, where it can then be decrypted.

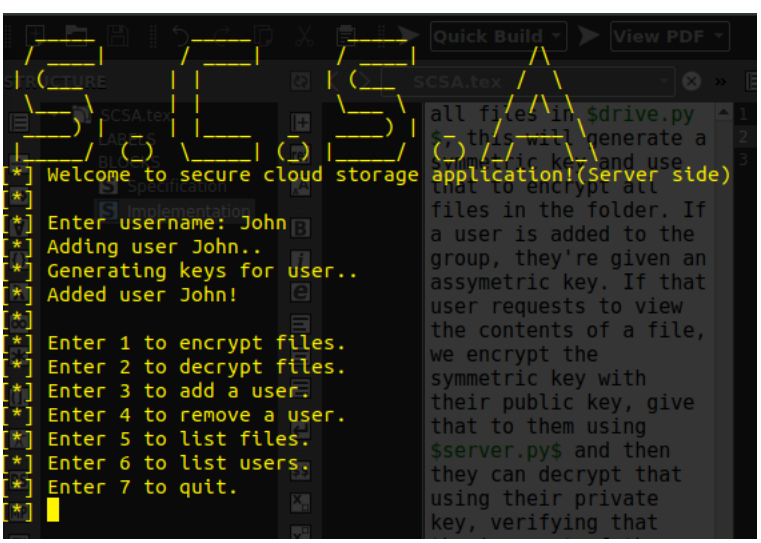
How does the encryption work? After you choose which folder of your drive you wish to be encrypted, you choose to option to encrypt all files in *drive.py*, this will generate a symmetric key and use that to encrypt all files in the folder. If a user is added to the group, they're given an assymetric key. If that user requests to view the contents of a file, we encrypt the symmetric key with their public key, give that to them using *server.py* and then they

can decrypt that using their private key, verifying that they're part of the group. If a user is removed, we delete their asymmetric key, decrypt all the files in the folder with the symmetric key, generate a new symmetric key and re-encrypt all the files in the folder of the drive.

Some screenshots and code can be seen below.
Server side after selecting to view users..

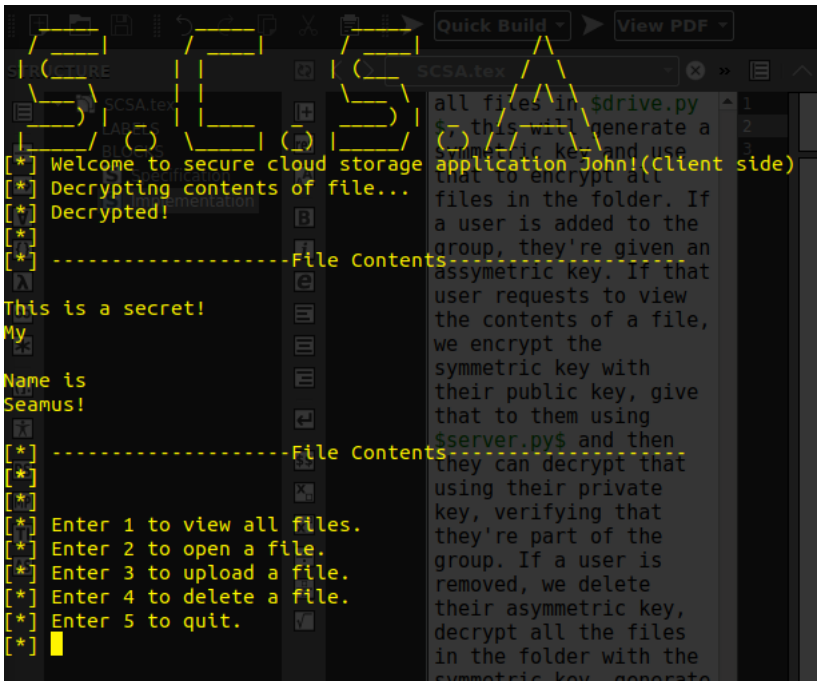


Server side after adding user 'John'



Client side after logging in as 'John'

Client side after requesting to view 'secret1.txt'



0.3 Drive.py

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from cryptography.fernet import Fernet
from encrypt import encrypt
from decrypt import decrypt
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
```

```python
import os
import shutil

# Shows all the users that are part of the admin's group
def show_users():
        subs = os.listdir('group/')
        print("[*] ---------------------- Users ----------------------")
        print("[*]")
        for sub in subs:
                print("[*] " + sub)
        print("[*]")
        print("[*] ---------------------- Users ----------------------")
        print("[*]")

# Shows all the files in the folder of the drive
def show_files(f_list):
        print("[*] ---------------------- Files ----------------------")
        print("[*]")
        for file in f_list:
                print("[*] " + file['title'])
        print("[*]")
        print("[*] ---------------------- Files ----------------------")
        print("[*]")

# Clears terminal and prints logo
def logo():
        os.system('cls' if os.name == 'nt' else 'clear')
        print("    _____          _____          _____                    ")
        print("   / ____|        / ____|        / ____|          /\       ")
        print("  | (___         | |            | (___           /  \      ")
        print("   \___ \        | |             \___ \         / /\ \     ")
        print("   ____) |  _   | |____   _   ____) |  _   / ____ \    ")
        print("  |_____/  (_)  \_____| (_) |_____/  (_) /_/    \_\  ")
        print("[*] Welcome to secure cloud storage application!(Server side)")
```

```python


def main():
        logo()

        # Google Authentication
        auth = GoogleAuth()
        auth.LocalWebserverAuth()
        drive = GoogleDrive(auth)

        # Grab the symmetric key.. if it doesn't exist, generate a new one
        try:
                f = open('keys/symmetric_key.txt', 'r')
                key = f.read()
                print("[*] Your symmetric key: '" + key + "'")
        except:
                key = Fernet.generate_key()
                f = open('keys/symmetric_key.txt', 'w')
                f.write(key)
                print("[*] Your symmetric key: '" + key + "'")
        f = Fernet(key)

        # Grabbing list of files in folder of drive.. change the id here to change folder.
        f_list = drive.ListFile({'q':"'1l53l9SNSC2qwj6wfDCMFQvXMOhX1BI0f' in parents and trashed=false"}).GetList()

        # Guts of the program..
        finished = False
        logo()
        print("[*]")
        # Present admin with list of options..
        while not finished:
                option = input("[*] Enter 1 to encrypt files.\n[*] Enter 2 to decrypt files.\n[*] Enter 3 to add a user.\n[*] Enter 4 to remove a user.\n[*] Enter 5 to list files.\n[*] Enter 6 to list users.\n[*] Enter 7 to quit.\n[*] ")
                logo()
                print("[*]")
                # Encrypts all files in the folder of the drive..
```

```
        if option is 1:
                encrypt(f, f_list)

        # Decrypts all files in the folder of the drive..
        elif option is 2:
                decrypt(f,f_list)

        # Adds a user to the group.. basically just creates a new folder whose name is the
        # name of the user, and generates an rsa key and stores it in that folder.
        elif option is 3:
                username = raw_input("[*] Enter username: ")
                if os.path.exists("group/" + str(username)):
                        print("[*] Username taken.")
                        print("[*]")
                else:
                        print("[*] Adding user " + str(username) + "..")
                        # Creating dir for user..
                        os.mkdir("group/" + str(username))

                        print("[*] Generating keys for user..")
                        # Generating private key..
                        private_key = rsa.generate_private_key(
                public_exponent=65537,
                key_size=2048,
                backend=default_backend())

                        # Serialising keys for storing..
                        private_serialized = private_key.private_bytes(
                        encoding=serialization.Encoding.PEM,
                        format=serialization.PrivateFormat.PKCS8,
                        encryption_algorithm=serialization.NoEncryption())

                        # Storing private key..
                        file = open("group/" + str(username) +
```

```
                    "/private_key.txt", "w")
                    file.write(private_serialized)
                    file.close()
                    print("[*] Added user " + str(username) + "!")
                    print("[*]")

            # Removes a user.. basically deletes the folder whose name is the same as the users,
            # decrypts all the files in the folder of the drive, generates a new symmetrics key,
            # and re-encrypts all the files in folder of the drive.
            elif option is 4:
                show_users()
                username = raw_input("[*] Enter username: ")
                logo()
                if os.path.exists("group/" + str(username)):
                    print("[*] Removing user..")
                    shutil.rmtree("group/" + str(username))
                    print("[*] Generating new symmetric key and re-encrypting all files..")
                    decrypt(f, f_list)
                    key = Fernet.generate_key()
                    file = open('keys/symmetric_key.txt', 'w')
                    file.write(key)
                    file.close()
                    f = Fernet(key)
                    print("[*] New symmetric key: " + key)
                    encrypt(f, f_list)
                    print("[*]")
                    logo()
                    print("[*]\n[*] Deleted user " + username + "!\n[*]")
                else:
                    print("[*] Username does not exist.. enter '6' to see list of users.")
```

```
                                print("[*]")

                # Shows all files in the folder of the drive..
                elif option is 5:
                        show_files(f_list)

                # Shows all users in the admin's group..
                elif option is 6:
                        show_users()

                # Exit()
                elif option is 7:
                        print("[*] Thanks for using the program!")
                        finished = True

                else:
                        print("[*] Please pick one of the options listed below..")
                        print("[*]")

if __name__ == "__main__":
        main()
```

0.4 Client.py

```
from cryptography.hazmat.primitives.serialization import load_pem_private_key
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from cryptography.fernet import Fernet
from encrypt import encrypt
from decrypt import decrypt
import base64
import requests
import os

# Takes in a username and returns the encrypted version of the symmetric key specifically
# for that user.
def getKey(username):
```

```
        # Grabbing private key..
        with open("group/" + str(username) + "/private_key.txt", "rb") as key_file:
                private_key = key_file.read()
        private_key = load_pem_private_key(private_key, None, default_backend())

        # Generating public key..
        public_key = private_key.public_key()
        public_key = public_key.public_bytes(
                        encoding=serialization.Encoding.PEM,
                        format=serialization.PublicFormat.SubjectPublicKeyInfo)

        # Sending request to flask server..
        print("[*] Requesting symmetric key from server..")
        URL = "http://127.0.0.1:5000/get_key"
        PARAMS = {'username' : username, 'pub_key': public_key}
        r = requests.get(url=URL, params=PARAMS)
        encrypted = r.text
        print("[*] Received encrypted symmetric key from server..")

        # Decrypting response from flask server..
        encrypted = base64.b64decode(encrypted)
        symmetric_key = private_key.decrypt(
    encrypted,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
        )
        )
        print("[*] Decrypted symmetric key!")

        return symmetric_key

# Shows all files in the folder of the drive..
def show_files(f_list):
        print("[*] ---------------------- Files ----------------------")
        print("[*]")
        for file in f_list:
                print("[*] " + file['title'])
        print("[*]")
        print("[*] ---------------------- Files
```

```
            ———————————————————————")
        print("[*]")

# Clears terminal and prints logo
def logo(username):
        os.system('cls' if os.name == 'nt' else 'clear')
        print("    -----          -----          -----                          ")
        print("   / ----|        / ----|        / ----|           /\            ")
        print("  | (---         | |            | (---            /  \           ")
        print("   \--- \        | |             \--- \          / /\ \          ")
        print("   ----) |  -   | |----   -    ----) |  -    / ----\  ")
        print("  |-----/  (-)  \-----| (-) |-----/  (-) /-/\-\ ")
        print("[*] Welcome to secure cloud storage application "+username+"!(Client side)")



def main():
        logo("")
        # Prompts client for username and checks if they're a valid user..
        username = raw_input("[*] Please enter your username: ")
        if os.path.exists("group/" + str(username)):
                print("[*] Welcome " + str(username) + "!")
                auth = GoogleAuth()
                auth.LocalWebserverAuth()
                drive = GoogleDrive(auth)

                # Get symmetric key..
                key = getKey(username)
                print("[*] '" + key + "'")
                f = Fernet(key)
                f_list = drive.ListFile({'q':"'1l53l9SNSC2qwj6wfDCMFQvXMOhX1BI0f' in parents and trashed=false"}).GetList()

                finished = False
                logo(username)
                while not finished:
```

```
print("[*]")
# Present user with options...
option = input("[*] Enter 1 to view all files.\n[*] Enter 2 to open a file.\n[*] Enter 3 to upload a file.\n[*] Enter 4 to delete a file.\n[*] Enter 5 to quit.\n[*] ")
logo(username)
print("[*]")

# Shows all files in the folder of the drive..
if option is 1:
        show_files(f_list)

# Opens the contents of a file.. basically grabs contents of the file in the
# drive and drcrypts it using the symmetric key.
elif option is 2:
        found = 0
        show_files(f_list)
        file_name = raw_input("[*] Enter name of file: ")
        logo(username)
        for file in f_list:
                if file["title"] == file_name:
                        found = 1
                        encoded = file.GetContentString()
                        print("[*] Decrypting contents of file...")
                        decoded = f.decrypt(encoded.encode())
                        print("[*] Decrypted!")
                        print("[*]")
                        print("[*] ----------File Contents----------")
                        print("")
```

```
                                        print(decoded.decode())
                                        print("[*] ----------File Contents----------")
                                        print("[*]")
                                if found is 0:
                                        print("[*] No such file name in drive.. enter '1' to see list of files.")
                                        print("[*]")

                        # Uploads a file.. just takes in the name of a file form secret_files/ and
                        # encrypts it before uploading..
                        elif option is 3:
                                file_name = raw_input("[*] Enter name of file to upload: ")
                                file = drive.CreateFile({"parents": [{"kind": "drive#fileLink", "id": "1l53l9SNSC2qwj6wfDCMFQvXMOhX1BI0f"}], 'title':file
                                source_file = open("secret_files/" + file_name, "r")
                                print("[*] Encrypting file and uploading...")
                                data = source_file.read()
                                encrypted_data = f.encrypt(data)
                                file.SetContentString(encrypted_data.decode())
                                file.Upload()
                                print("[*] Uploaded!")
                                f_list = drive.ListFile({'q':"'1l53l9SNSC2qwj6wfDCMFQvXMO in parents and trashed=false"}).GetList()
                                print("[*]")

                        # Removes a file.. self explanatory.
                        elif option is 4:
                                found = 0
                                show_files(f_list)
                                file_name = raw_input("[*]
```

```
                                        Enter name of file: ")
                                logo(username)
                                for file in f_list:
                                        if file["title"] == file_name:
                                                found = 1
                                                file.Delete()
                                                print("[*] Deleted " + file_name + "!")
                                                print("[*]")
                                if found is 0:
                                        print("[*] No such file name in drive.. enter '1' to see list of files.")
                                        print("[*]")

                        # Exit.
                        elif option is 5:
                                print("[*] Thanks for using the program!")
                                print("[*]")
                                finished = True

                        else:
                                print("[*] Please pick one of the options listed below..")
                                print("[*]")




        # If user is not a valid user, print this.
        else:
                print("[*] You are not part of the admin's group.. please contact admin for an invite.")

if __name__ == "__main__":
        main()
```

0.5 Server.py

```python
from cryptography.hazmat.primitives.serialization import load_pem_private_key
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from cryptography.fernet import Fernet
from encrypt import encrypt
from decrypt import decrypt
import requests
import base64
import os

from flask import Flask
from flask import request
app = Flask(__name__)

# Create route for get_key function
@app.route('/get_key', methods=['GET'])
def index():
        # Grab username and public_key for that user.
        username = request.args.get('username')
        pub_key = request.args.get('pub_key').encode('ascii')
        public_key = serialization.load_pem_public_key(pub_key,backend=default_backend())

        # Grab the unencrypted symmetric key..
        key = open("keys/symmetric_key.txt", "r")
        key = key.read()

        # Encrypt the symmetric key..
        encrypted = public_key.encrypt(
    key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
        )
        )

        # Return the symmetric key specifically encrypted for that user...
        encrypted = base64.b64encode(encrypted)
```

```python

        print("encrypted symmetric key.. sending to client..")
        return encrypted

if __name__ == '__main__':
    app.run(debug = True)
```