CS3031 - Project 2

Paolo Moloney - 16325409

April 9, 2019

Contents

1 Specification **3**

2 Implementation **4**
2.1 Admin Mode 4
2.2 User Mode 4
2.3 Authentication 5

3 Code **5**
3.1 admin.py 5
3.2 user.py 8

1 Specification

The objective of this project is to implement a secure cloud storage application with the following features:

1. Secures all files uploaded to the cloud, such that only users in the 'Secure Cloud Storage Group' can access them.
2. Key management system for users.
3. Add and remove users from the 'Secure Cloud Storage Group'.

2 Implementation

I chose to implement the application in Python, using the Google Drive API and the following modules:

PyDrive Wrapper library of google-api-python-client

cryptography Provides high-level and low-level interfaces to common cryptographic algorithms

2.1 Admin Mode

When the `admin.py` script is run, it first searches for a symmetric key in keys/key.txt. If none is found then it generates a new key and stores it there. This key is encrypted using RSA, providing an additional layer of security. The files in the cloud are stored in a folder on Google Drive. All these files are encryptred using this symmetric key.
The admin has a number of management commands available:

enc Pulls all the files from the Drive and encrypts using the symmetric key in keys/key.txt, then reuploads them

dec Similar to enc, but decrypts instead of encrypting

lf Lists all the files in the Drive

addu Prompts the admin to enter a username to add to the 'Secure Cloud Storage Group', then adds this username to group/ and generates a private key with RSA, storing it in group/user/privateKey.txt, used to decrypt the symmetric key

rmvu Prompts the admin to enter a username to remove from the 'Secure Cloud Storage Group', deleting the user and the corresponding private key from group/

lu Lists all the users in the 'Secure Cloud Storage Group'

q Terminates the execution of the program

2.2 User Mode

Running `user.py` is similar to admin mode, but with less privileges. The commands available to users are:

lf Lists all the files in the Drive

op Prompts the user for the filename to view, then pulls and decrypts the file and prints its contents to the console

up Prompts the user for the path to the file to upload *(Not working)*

2.3 Authentication

Authorization and authentication are handled by PyDrive. The credentials are found in client_secrets.json, which is generated when setting up the Google Drive API project.

3 Code

3.1 admin.py

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

def encrypt(f,files):
    for fi in files:
        plaintext = fi.GetContentString()
        token = f.encrypt(plaintext.encode())
        fi.SetContentString(token.decode())
        fi.Upload()

def decrypt(f,files):
    for fi in files:
        token = fi.GetContentString()
        plaintext = f.decrypt(token.encode())
        fi.SetContentString(plaintext.decode())
        fi.Upload()

def options(i):
    return {
             'enc': 1,
             'dec': 2,
             'lf': 3,
             'addu': 4,
             'rmvu': 5,
             'lu': 6,
             'q': 7
             }.get(i, 8)


def main():
    gauth = GoogleAuth()
    gauth.LocalWebserverAuth()
    drive = GoogleDrive(gauth)
```

```
40      print("** ADMIN MODE **")
41      try:
42          f = open('keys/key.txt', 'r')
43          key = f.read()
44          print("~Found existing symmetric key: '" + key + "'")
45      except:
46          key = Fernet.generate_key()
47          f = open('keys/key.txt', 'w')
48          f.write(key)
49          print("~New symmetric key generated: '" + key + "'")
50      f = Fernet(key)
51      files = drive.ListFile({'q':"'1qwdkeX0WApKsX0JA0ZctEnjkjj3MhcfK' in parents and
52      end = False
53      while not end:
54          i = raw_input("> *OPTIONS*\
55                  \n> 'enc': encrypt files\
56                  \n> 'dec': decrypt files\
57                  \n> 'lf': list files\
58                  \n> 'addu': add user\
59                  \n> 'rmvu': remove user\
60                  \n> 'lu': list users\
61                  \n> 'q': quit\n")
62          opt = options(str(i))
63          if opt is 1:
64              encrypt(f, files)
65
66          elif opt is 2:
67              decrypt(f,files)
68
69          elif opt is 3:
70              print("~All files in drive: ")
71              for fi in files:
72                  print("~" + fi['title'])
73
74          elif opt is 4:
75              username = raw_input("~Enter a username to add: ")
76              if os.path.exists("group/" + str(username)):
77                  print("~User already exists.")
78              else:
79                  print("~Adding user " + str(username))
80                  os.mkdir("group/" + str(username))
81                  print("~Generating RSA private key for " + str(username))
82                  privateKey = rsa.generate_private_key(public_exponent=65537, key_si
83                  privateSerializedKey = privateKey.private_bytes(encoding=serializat
84                  file = open("group/" + str(username) + "/privateKey.txt", "w")
85                  file.write(privateSerializedKey)
86                  file.close()
87                  print("~Added user " + str(username))
88
89          elif opt is 5:
```

```
90              username = raw_input("~Enter the username to remove: ")
91              if os.path.exists("group/" + str(username)):
92                  print("~Removing user " + str(username))
93                  shutil.rmtree("group/" + str(username))
94                  print("~Generating new symmetric key and encrypting files.")
95                  decrypt(f, files)
96                  key = Fernet.generate_key()
97                  file = open('keys/key.txt', 'w')
98                  file.write(key)
99                  file.close()
100                 f = Fernet(key)
101                 print("~New symmetric key: " + key)
102                 encrypt(f, files)
103             else:
104                 print("~User " + str(username) + " does not exist. Enter 'lu' to li
105
106         elif opt is 6:
107             users = os.listdir('group/')
108             print("~All users in drive: ")
109             for user in users:
110                 print("~" + user)
111
112         elif opt is 7:
113             print("~Exiting...")
114             end = True
115
116         elif opt is 8:
117             print("~Invalid command.")
118
119 if __name__ == "__main__":
120     main()
```

3.2 user.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
import os

def encrypt(f,files):
    for fi in files:
        unencoded = fi.GetContentString()
        encoded = f.encrypt(unencoded.encode())
        fi.SetContentString(encoded.decode())
        fi.Upload()

def decrypt(f,files):
    for fi in files:
        encoded = fi.GetContentString()
        unencoded = f.decrypt(encoded.encode())
        fi.SetContentString(unencoded.decode())
        fi.Upload()

def getKey(username):
    with open("group/" + str(username) + "/privateKey.txt", "rb") as fileWithPrivat
        privateKey = fileWithPrivateKey.read()
    privateKey = load_pem_private_key(privateKey, None, default_backend())
    publicKey = privateKey.public_key()
    key = open("keys/key.txt", "r")
    key = key.read()
    encrypted = publicKey.encrypt(key, padding.OAEP(mgf=padding.MGF1(algorithm=hash
    print("~Getting encrypted symmetric key.")
    symmetricKey = privateKey.decrypt(encrypted, padding.OAEP(mgf=padding.MGF1(algo
    print("~Decrypting symmetric key.")
    return symmetricKey

def options(i):
    return {
            'lf': 1,
            'op': 2,
            'up': 3,
            'q': 4
            }.get(i, 5)

def main():
    print("** USER MODE **")
    username = raw_input("~Username: ")
```

```
49      if os.path.exists("group/" + str(username)):
50          print("~Welcome " + str(username))
51          gauth = GoogleAuth()
52          gauth.LocalWebserverAuth()
53          drive = GoogleDrive(gauth)
54          key = getKey(username)
55          f = Fernet(key)
56          files = drive.ListFile({'q':"'1qwdkeX0WApKsX0JAOZctEnjkjj3MhcfK' in parents
57          end = False
58          while not end:
59              i = raw_input("> *OPTIONS*\
60                  \n> 'lf': list files\
61                  \n> 'op': open file\
62                  \n> 'up': upload file\
63                  \n> 'q': quit\n")
64              opt = options(str(i))
65              if opt is 1:
66                  files = drive.ListFile({'q':"'1qwdkeX0WApKsX0JAOZctEnjkjj3MhcfK' in
67                  print("~All Files in Secure Drive: ")
68                  for file in files:
69                      print("~" + file['title'])
70
71              elif opt is 2:
72                  found = 0
73                  nameOfFile = raw_input("~Enter name of file you wish to open: ")
74                  for file in files:
75                      if file["title"] == nameOfFile:
76                          found = 1
77                          token = file.GetContentString()
78                          print("~Decrypting file:")
79                          plaintext = f.decrypt(token.encode())
80
81                          print("~File Contains: ")
82                          print(plaintext.decode())
83                  if found is 0:
84                          print("~File: " + nameOfFile + " not in drive.. enter 'lf'
85
86              elif opt is 3:
87                  filePath = raw_input("~Enter the path to the file to upload: ")
88                  if os.path.exists(filePath):
89                      with open(filePath,"r") as file:
90                          parents = ["1qwdkeX0WApKsX0JAOZctEnjkjj3MhcfK"]
91                          driveFile = drive.CreateFile({ "parents": [{"kind": "drive#
92                          readFile = file.read()
93                          encoded = f.encrypt(readFile.encode())
94                          driveFile.SetContentString(encoded.decode())
95                          driveFile.Upload()
96
97              elif opt is 4:
98                  print("~Exiting...")
```

```
                end = True

            elif opt is 5:
                print("~Invalid command.")

        else:
            print("~Invalid username.")

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