Advanced Telecommunications
Securing the cloud

Abhay Singh Khanka (1830999)

1. Project Description

The aim of this project was to build a secure cloud storage application for Google drive. With this application, the user can upload and download files to their Google drive folder, as well as be able to encrypt and decrypt certain files that need more security. The user is also able to share files with another user by just entering his email ID and can also choose to remove this user or delete the file entirely.

High level Description

To begin with, the project is made in Python 3 and requires a few extra libraries which are under the requirements section.

On each execution of the main file, the method establishAuthFlow is called. This function establishes the connection to google drive and verifies the users' credentials.json file.

The user is showed a menu and from there they have several choices. The user would initially have to generate a new key, which is done using the Fernet library's inbuilt function which is then stored in a creds.key file.

To upload a file, the user is asked the name of the file and its extensions which is then converted to the standardized MIME type and uploaded.

For more security, the user can encrypt this file at the menu. The encrypt function uses fernets inbuilt encrypt function and the key stored in the creds.key file to secure the file. Similarly, the user can decrypt a file with the in-built decrypt method of fernet, however the key used to encrypt the file must be the same as the one used to decrypt it otherwise the file would be lost.

There is even functionality to share a file on the users google drive with other users. The required input is for the name of the file and the email of the person the file is to be shared with and thus by calling the google drive API, this person can access the shared file. Furthermore, the user can choose to list the people that have access to a particular file and if he wishes revoke this access with the unshare user command.

Finally, there is functionality to search a file on the users google drive and the functionality to delete a file as well. If the user wishes to exit the program, he can type either exit or quit to end the program.

Requirements

- https://developers.google.com/workspace/guides/create-credentials, The user would need to have his credentials.json file from the above link.
- The following libraries:
  - Fernet
  - Cryptography
  - Pycrypto
  - Apiclient
- pip3 install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
  The following command would enable the user to establish auth flow to the google drive account

2. References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types
https://developers.google.com/drive/api/v3/quickstart/python
https://pypi.org/project/cryptography/
https://cryptography.io/en/latest/
https://cryptography.io/en/latest/fernet/

3. Code Listing

```
#imports
from __future__ import print_function
from apiclient.http import MediaFileUpload, MediaIoBaseDownload
from cryptography.fernet import Fernet
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
import pickle
import os
import io
import base64

#https://developers.google.com/drive/api/v3/about-sdk - GoogleDriveAPI

# to convert to MIME type standard
mediaTypes={
    "xls":'application/vnd.ms-excel',
    "xlsx":'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',
    "xml":'text/xml',
    "ods":'application/vnd.oasis.opendocument.spreadsheet',
    "csv":'text/plain',
    "tmpl":'text/plain',
    "pdf": 'application/pdf',
    "php":'application/x-httpd-php',
    "jpg":'image/jpeg',
    "png":'image/png',
```

```python
    "gif":'image/gif',
    "bmp":'image/bmp',
    "txt":'text/plain',
    "doc":'application/msword',
    "js":'text/js',
    "swf":'application/x-shockwave-flash',
    "mp3":'audio/mpeg',
    "zip":'application/zip',
    "rar":'application/rar',
    "tar":'application/tar',
    "arj":'application/arj',
    "cab":'application/cab',
    "html":'text/html',
    "htm":'text/html',
    "default":'application/octet-stream',
    "folder":'application/vnd.google-apps.folder'
}

SCOPES = ['https://www.googleapis.com/auth/drive']

def main():
    os.system('clear')
    Key = importKey()

    programEnd = False
    establishAuthFlow()

    # Menu
    print("#############################")
    print("!Menu!")
    print("Enter the commands for the following actions")
    print("Search file: search")
    print("Upload file: upload")
    print("Download file: download")
    print("Encrypt file: enc")
    print("Decrypt file: dec")
    print("Share with user: share")
    print("List users that have access to a file: list")
    print("Delete user: unshare")
    print("Delete  file: del")
    print("Regen key : gen")
    print("WARNING : When regening key any file that is currently encrypted will not be able to be decrypted")
    print("Exit : exit")
    print("############################")
    print('\n')
    while programEnd == False:
        nextAction = (str(input("Please enter a command or type quit to exit. ")))
        if(nextAction == 'search'):
            fileName = (str(input("Enter a file name to search for: ")))
            search(fileName)
        elif(nextAction == 'upload'):
            fileName = (str(input("Enter full file name for upload: ")))
            fileType = (str(input("Enter the file exstension (ex : txt or rar)")))
            fileType = mediaTypes[fileType]
            upload(fileName, fileType)
        elif(nextAction == 'download'):
            fileName = (str(input("Enter a file name for download: ")))
            download(getID(fileName), fileName)
        elif(nextAction =='enc'):
            fileName = (str(input("Enter a file name with exstension to encrypt: ")))
            encryptFile(fileName, Key)
        elif(nextAction == 'dec'):
            fileName = (str(input("Enter a file name with exstension to decrypt: ")))
            decryptFile(fileName, Key)
        elif (nextAction == 'exit' or nextAction == 'quit'):
            programEnd = True
            print('Exiting program.')
            exit()
        elif(nextAction == 'gen'):
            print('Regening key, Program will quit. Please restart to use new key')
            keyGen()
            print("Program terminated")
            exit()
        elif(nextAction == 'share'):
            fileName = (str(input("Enter a file name to share:  ")))
            userEmail = (str(input("Enter users email address to share with: ")))
            shareFile(getID(fileName), userEmail)
        elif(nextAction == 'list'):
            fileName = (str(input("Enter a file name to list user access:  ")))
            listPerm(getID(fileName))
        elif(nextAction == 'unshare'):
            fileName = (str(input("Enter a file name to unshare:  ")))
            permissionID = (str(input("Enter the id of the permission for the file name to unshare:  ")))
            removeUser(getID(fileName),permissionID)
        elif(nextAction =='del'):
            fileName = (str(input("Enter a file name to delete:  ")))
            deleteFile(getID(fileName))
        else:
            print("Please try again: ")

# Function to generate keys for user
def keyGen():
        key = Fernet.generate_key()
        file = open('creds.key', 'w+')
        file.write(key.decode())
        return key

# Function to import key from creds.key file
def importKey():
    try:
        file = open("creds.key", 'rb')
        key = file.read()
        return key
    except:
        print("No creds.key file found")

# Function to encrypt a file using Fernets inbuilt encrypt method
def encryptFile(fileName, Key):
    try:
        data = open(fileName, "rb")
        data = data.read()
        f = Fernet(Key)
        encrypted = f.encrypt(data)
        file = open(fileName, "wb")
        file.write(encrypted)
        print("Success!")
        print("File Encrypted")
    except:
        print("Encryption failed")

# Function to decrypt a file using key from key.key and fernets inbuilt decrypt method
def decryptFile(fileName, Key):
    try:
        data = open(fileName, "rb")
        data = data.read()
        f = Fernet(Key)
        decrypted = f.decrypt(data)
        file = open(fileName, "wb")
        file.write(decrypted)
        print("Success!")
        print("File Decrypted")
    except:
        print("Decryption failed")

# Function to establish connection to google drive
# saves the credentials for the next run by writing them to a token.pickle file
def establishAuthFlow():
    try:
        global service
        creds = None
        if os.path.exists('token.pickle'):
            with open('token.pickle', 'rb') as token:
                creds = pickle.load(token)
                #creds = Credentials.from_authorized_user_file('token.json', SCOPES)
        if not creds or not creds.valid:
            if creds and creds.expired and creds.refresh_token:
                creds.refresh(Request())
            else:
                flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
                creds = flow.run_local_server(port=0)
            with open('token.pickle', 'wb') as token:
                pickle.dump(creds, token)
                #token.write(creds.to_json())
```

```python
        service = build('drive', 'v3', credentials=creds)
        #document = service.documents().get(documentId=DOCUMENT_ID).execute()
        print("Auth Flow Established")
    except:
        print("Establish Auth Flow Failed")

#Function to search for a file s
#Change
def search(fileName):
  #  try:
        global service
        results = service.files().list(pageSize=15, q="name contains '" + fileName +"'", fields="files(id, name)").execute()
        items = results.get('files', [])
        if not items:
            print('No files found.')
        else:
            print('Files:')
            for item in items:
                print(u'{0} ({1})'.format(item['name'], item['id']))
   # except:
    #    print("Search failed")

# used to get ID for downloading file
def getID(fileName):
    try:
        global service
        results = service.files().list(pageSize=1, q="name='" + fileName +"'", fields='*').execute()
        items = results.get('files', [])
        if not items:
            print('No files found.')
        else:
            for item in items:
                return item['id']
    except:
        print("Get ID failed")

#Function to upload a file
def upload(fileName, fileType):
    try:
        global service
        file_metadata = {'name': fileName}
        media = MediaFileUpload(fileName,mimetype=fileType)
        file = service.files().create(body=file_metadata,media_body=media,fields='id').execute()
        print ("File  "+fileName+" uploaded")
    except KeyboardInterrupt:
        print("Interrupted")

#Function to download a file
def download(file_id, fileName):
    try:
        global service
        request = service.files().get_media(fileId=file_id)
        fileRequest = io.BytesIO()
        downloader = MediaIoBaseDownload(fileRequest, request)
        done = False
        while done is False:
            status, done = downloader.next_chunk()
            print( "Download status %d%%." % int(status.progress() * 100))
        with io.open(fileName, 'w+') as file:
            fileRequest.seek(0)
            file.write(fileRequest.read().decode())
    except:
        print("Downlaod Failed")

#
def shareFile(fileID, userEmail):
    try:
        global service
        batch = service.new_batch_http_request(callback=callback)
        user_permission = {
            'type': 'user',
            'role': 'writer',
            'emailAddress': userEmail,
        }
        batch.add(service.permissions().create(fileId=fileID,body=user_permission,fields='id',))
        batch.execute()
        print("File shared with "+userEmail)
    except:
        print("File share failed")

#Function to list permissions of a file in your google drive
def listPerm(fileID):
    try:
        global service
        permissions = service.permissions().list(fileId=fileID,fields='permissions(id, emailAddress, displayName)').execute()
        result = permissions.get('permissions', [])
        print(result)
    except:
        print("Permission list failed")

#Function to delete permissions of a file in your google drive
def removeUser(fileID, permissionID):
    try:
        global service
        service.permissions().delete(fileId=fileID,permissionId=permissionID).execute()
        print("User removed")
    except:
        print("Failed to remove user")

def callback(request_id, response, exception):
    if exception:
        print (exception)
    else:
        print ("Permission Id: %s" % response.get('id'))

#Function to delete a google drive file
def deleteFile(fileid):
    try:
        service.files().delete(fileId=fileid).execute()
        print("file deleted")
    except:
        print("Unable to delete file")

if __name__ == '__main__':
    main()

```