

CS3031 Advanced Telecommunications Project II

Séamus Woods
15317173

05/04/2019

0.1 Specification

The objective of the exercise 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.

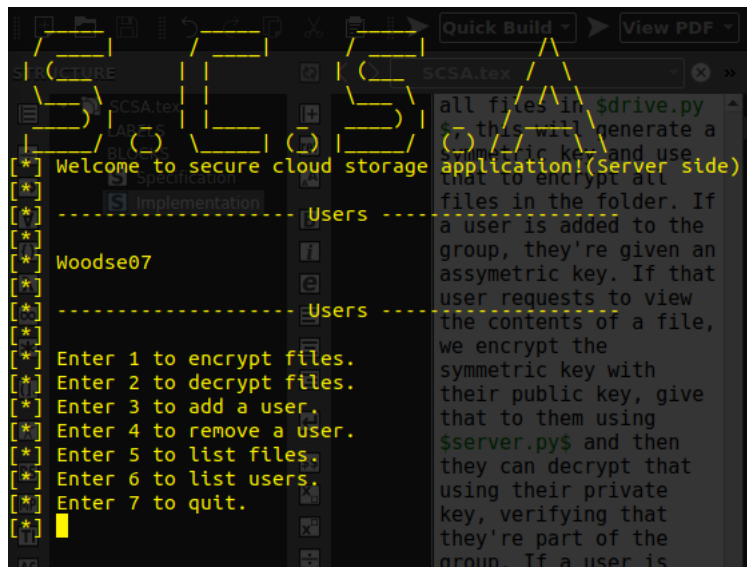
server.py is a flask server, which listens for *client.py* to send it a username, and will return the encrypted version of the symmetric key for that user to the client, where it can then be decrypted.

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 asymmetric 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..



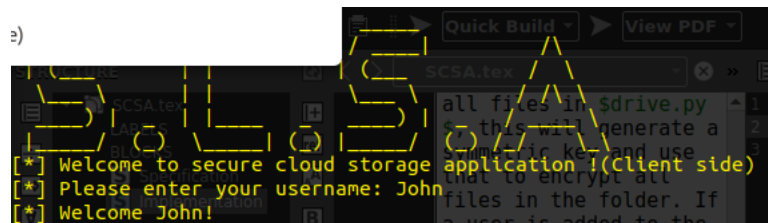
```
[*] Welcome to secure cloud storage application!(Server side)
[*] -----Users-----
[*] Woodse07
[*] -----Users-----
[*] Enter 1 to encrypt files.
[*] Enter 2 to decrypt files.
[*] Enter 3 to add a user.
[*] Enter 4 to remove a user.
[*] Enter 5 to list files.
[*] Enter 6 to list users.
[*] Enter 7 to quit.
[*] 
```

Server side after adding user 'John'



```
[*] Welcome to secure cloud storage application!(Server side)
[*] Enter username: John
[*] Adding user John..
[*] Generating keys for user..
[*] Added user John!
[*] Enter 1 to encrypt files.
[*] Enter 2 to decrypt files.
[*] Enter 3 to add a user.
[*] Enter 4 to remove a user.
[*] Enter 5 to list files.
[*] Enter 6 to list users.
[*] Enter 7 to quit.
[*] 
```

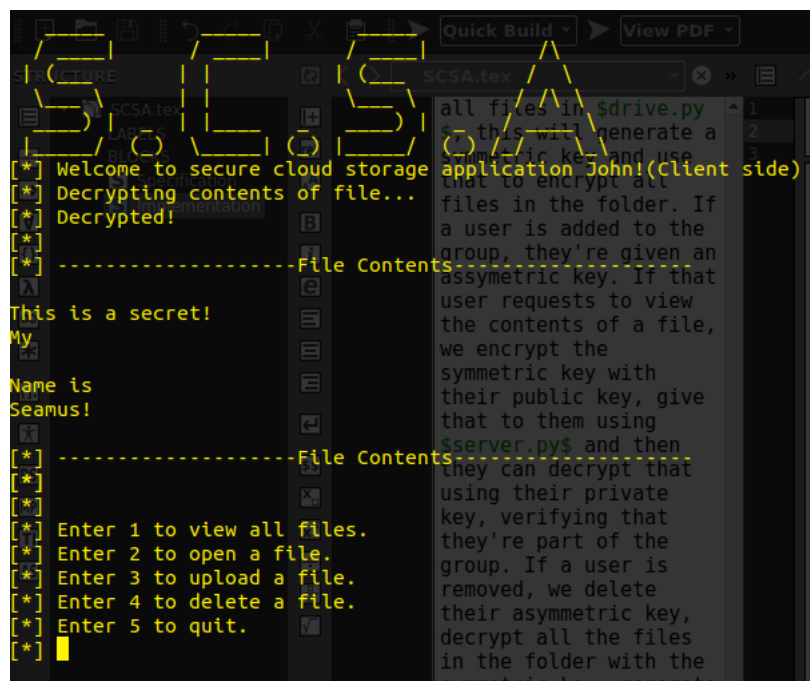
Client side after logging in as 'John'



a)

```
[*] Welcome to secure cloud storage application!(Client side)
[*] Please enter your username: John
[*] Welcome John!
```

Client side after requesting to view 'secret1.txt'



```
[*] Welcome to secure cloud storage application John!(Client side)
[*] Decrypting contents of file...
[*] Decrypted!
[*] -----File Contents-----
This is a secret!
My
Name is
Seamus!
[*] -----File Contents-----
[*] Enter 1 to view all files.
[*] Enter 2 to open a file.
[*] Enter 3 to upload a file.
[*] Enter 4 to delete a file.
[*] Enter 5 to quit.
[*] █
```

0.3 Drive.py

```
1 from pydrive.auth import GoogleAuth
2 from pydrive.drive import GoogleDrive
3 from cryptography.fernet import Fernet
4 from encrypt import encrypt
5 from decrypt import decrypt
6 from cryptography.hazmat.backends import default_backend
7 from cryptography.hazmat.primitives.asymmetric import rsa
8 from cryptography.hazmat.primitives import serialization
```

```

9  import os
10 import shutil
11
12 # Shows all the users that are part of the admin's group
13 def show_users():
14     subs = os.listdir('group/')
15     print("[*] _____ Users
16         _____")
17     print("[*]")
18     for sub in subs:
19         print("[*] " + sub)
20     print("[*] _____ Users
21         _____")
22     print("[*]")
23 # Shows all the files in the folder of the drive
24 def show_files(f_list):
25     print("[*] _____ Files
26         _____")
27     print("[*]")
28     for file in f_list:
29         print("[*] " + file['title'])
30     print("[*] _____ Files
31         _____")
32     print("[*]")
33 # Clears terminal and prints logo
34 def logo():
35     os.system('cls' if os.name == 'nt' else 'clear')
36     print("      -----      ")
37     print(" /  ---|      /  ---|      /  ---|      /\
38     ")
39     print(" | (---      |      | (---      /  \
40     ")
41     print(" \--- \      |      \--- \      / /\ \
42     ")
43     print(" ----) | - | |---- - ----) | - / ----
44     \
45     ")
46     print(" |-----/ (-) \-----| (-) |-----/ (-) /- /
47     \-\
48     ")
49     print("[*] Welcome to secure cloud storage
50         application!(Server side)")

```

```

43
44
45 def main():
46     logo()
47
48     # Google Authentication
49     auth = GoogleAuth()
50     auth.LocalWebserverAuth()
51     drive = GoogleDrive(auth)
52
53     # Grab the symmetric key.. if it doesn't exist,
       generate a new one
54     try:
55         f = open('keys/symmetric_key.txt', 'r')
56         key = f.read()
57         print("[*] Your symmetric key: '" + key + "'")
58     except:
59         key = Fernet.generate_key()
60         f = open('keys/symmetric_key.txt', 'w')
61         f.write(key)
62         print("[*] Your symmetric key: '" + key + "'")
63     f = Fernet(key)
64
65     # Grabbing list of files in folder of drive.. change
       the id here to change folder.
66     f_list =
        drive.ListFile({'q':"1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f'
        in parents and trashed=false"}).GetList()
67
68     # Guts of the program..
69     finished = False
70     logo()
71     print("[*]")
72     # Present admin with list of options..
73     while not finished:
74         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[*] ")
75         logo()
76         print("[*]")
77         # Encrypts all files in the folder of the
           drive..

```

```

78         if option is 1:
79             encrypt(f, f_list)
80
81         # Decrypts all files in the folder of the
82         drive..
83         elif option is 2:
84             decrypt(f, f_list)
85
86         # Adds a user to the group.. basically just
87         creates a new folder whose name is the
88         # name of the user, and generates an rsa key
89         and stores it in that folder.
90         elif option is 3:
91             username = raw_input("[*] Enter
92             username: ")
93             if os.path.exists("group/" +
94             str(username)):
95                 print("[*] Username taken.")
96                 print("[*]")
97             else:
98                 print("[*] Adding user " +
99                 str(username) + "..")
100                # Creating dir for user..
101                os.mkdir("group/" +
102                str(username))
103
104                print("[*] Generating keys for
105                user..")
106                # Generating private key..
107                private_key =
108                rsa.generate_private_key(
109                public_exponent=65537,
110                key_size=2048,
111                backend=default_backend())
112
113                # Serialising keys for storing..
114                private_serialized =
115                private_key.private_bytes(
116                encoding=serialization.Encoding.PEM,
117                format=serialization.PrivateFormat.PKCS8,
118                encryption_algorithm=serialization.NoEncryption())
119
120                # Storing private key..
121                file = open("group/" +
122                str(username) +

```

```

112         "/private_key.txt", "w")
113     file.write(private_serialized)
114     file.close()
115     print("[*] Added user " +
116           str(username) + "!")
117     print("[*]")
118
119     # Removes a user.. basically deletes the folder
120     whose name is the same as the users,
121     # decrypts all the files in the folder of the
122     drive, generates a new symmetric key,
123     # and re-encrypts all the files in folder of
124     the drive.
125     elif option is 4:
126         show_users()
127         username = raw_input("[*] Enter
128             username: ")
129         logo()
130         if os.path.exists("group/" +
131             str(username)):
132             print("[*] Removing user..")
133             shutil.rmtree("group/" +
134                 str(username))
135             print("[*] Generating new
136                 symmetric key and
137                 re-encrypting all files..")
138             decrypt(f, f_list)
139             key = Fernet.generate_key()
140             file =
141                 open('keys/symmetric_key.txt',
142                     'w')
143             file.write(key)
144             file.close()
145             f = Fernet(key)
146             print("[*] New symmetric key: "
147                 + key)
148             encrypt(f, f_list)
149             print("[*]")
150             logo()
151             print("[*]\n[*] Deleted user "
152                 + username + "!\n[*]")
153
154     else:
155         print("[*] Username does not
156             exist.. enter '6' to see
157             list of users.")

```



```

141                                     print("[*]")
142
143                                     # Shows all files in the folder of the drive..
144                                     elif option is 5:
145                                         show_files(f_list)
146
147                                     # Shows all users in the admin's group..
148                                     elif option is 6:
149                                         show_users()
150
151                                     # Exit()
152                                     elif option is 7:
153                                         print("[*] Thanks for using the
154                                             program!")
155                                         finished = True
156
157                                     else:
158                                         print("[*] Please pick one of the
159                                             options listed below..")
160                                         print("[*]")
161
162 if __name__ == "__main__":
163     main()

```

0.4 Client.py

```

1 from cryptography.hazmat.primitives.serialization import
   load_pem_private_key
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.hazmat.primitives import serialization
4 from cryptography.hazmat.primitives import hashes
5 from cryptography.hazmat.primitives.asymmetric import padding
6 from pydrive.auth import GoogleAuth
7 from pydrive.drive import GoogleDrive
8 from cryptography.fernet import Fernet
9 from encrypt import encrypt
10 from decrypt import decrypt
11 import base64
12 import requests
13 import os
14
15 # Takes in a username and returns the encrypted version of the
   symmetric key specifically
16 # for that user.
17 def getKey(username):

```

```

18     # Grabbing private key..
19     with open("group/" + str(username) +
20             "/private_key.txt", "rb") as key_file:
21         private_key = key_file.read()
22         private_key = load_pem_private_key(private_key, None,
23             default_backend())
24
25     # Generating public key..
26     public_key = private_key.public_key()
27     public_key = public_key.public_bytes(
28         encoding=serialization.Encoding.PEM,
29         format=serialization.PublicFormat.SubjectPublicKeyInfo)
30
31     # Sending request to flask server..
32     print("[*] Requesting symmetric key from server..")
33     URL = "http://127.0.0.1:5000/get_key"
34     PARAMS = {'username': username, 'pub_key': public_key}
35     r = requests.get(url=URL, params=PARAMS)
36     encrypted = r.text
37     print("[*] Received encrypted symmetric key from
38         server..")
39
40     # Decrypting response from flask server..
41     encrypted = base64.b64decode(encrypted)
42     symmetric_key = private_key.decrypt(
43         encrypted,
44         padding.OAEP(
45             mgf=padding.MGF1(algorithm=hashes.SHA256()),
46             algorithm=hashes.SHA256(),
47             label=None
48         )
49     )
50     print("[*] Decrypted symmetric key!")
51
52     return symmetric_key
53
54 # Shows all files in the folder of the drive..
55 def show_files(f_list):
56     print("[*] _____ Files
57         _____")
58     print("[*]")
59     for file in f_list:
60         print("[*] " + file['title'])
61     print("[*]")
62     print("[*] _____ Files

```

```

59         _____")
60     print("[*]")
61     # Clears terminal and prints logo
62     def logo(username):
63         os.system('cls' if os.name == 'nt' else 'clear')
64         print("      -----      ")
65         print("    /  ----|      /  ----|      /  ----|      /\
        ")
66         print("  | ( ---      | |      | ( ---      /  \
        ")
67         print(" \ --- \      | |      \ --- \      / /\ \
        ")
68         print("  ----) | - | | ---- - ----) | - / ----
        \ ")
69         print(" | ----/ (-) \ ----| (-) | ----/ (-) /- /
        \ - \ ")
70         print("[*] Welcome to secure cloud storage application
        "+username+"!(Client side)")
71
72
73
74     def main():
75         logo("")
76         # Prompts client for username and checks if they're a
        valid user..
77         username = raw_input("[*] Please enter your username: ")
78         if os.path.exists("group/" + str(username)):
79             print("[*] Welcome " + str(username) + "!")
80             auth = GoogleAuth()
81             auth.LocalWebserverAuth()
82             drive = GoogleDrive(auth)
83
84             # Get symmetric key..
85             key = getKey(username)
86             print("[*] ' " + key + " '")
87             f = Fernet(key)
88             f_list =
                drive.ListFile({'q': "'1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f'
                in parents and trashed=false"}).GetList()
89
90             finished = False
91             logo(username)
92             while not finished:

```

```

93     print(" [*]")
94     # Present user with options...
95     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[*] ")
96     logo(username)
97     print(" [*]")
98
99     # Shows all files in the folder of the
        drive..
100    if option is 1:
101        show_files(f_list)
102
103    # Opens the contents of a file..
        basically grabs contents of the file
        in the
104    # drive and drcrypts it using the
        symmetric key.
105    elif option is 2:
106        found = 0
107        show_files(f_list)
108        file_name = raw_input(" [*]
            Enter name of file: ")
109        logo(username)
110        for file in f_list:
111            if file["title"] ==
                file_name:
112                found = 1
113                encoded =
                    file.GetContentString()
114                print(" [*]
                    Decrypting
                    contents of
                    file ...")
115                decoded =
                    f.decrypt(encoded.encode())
116                print(" [*]
                    Decrypted!")
117                print(" [*]")
118                print(" [*]
                    _____File
                    Contents_____")
119    print("")

```

```

120                                     print(decoded.decode())
121                                     print(" [*]
                                     _____File
                                     Contents_____")
122                                     print(" [*]")
123
124     if found is 0:
125         print(" [*] No such file
126             name in drive..
127             enter '1' to see
128             list of files.")
129         print(" [*]")
130
131     # Uploads a file.. just takes in the
132     # name of a file form secret_files/ and
133     # encrypts it before uploading..
134     elif option is 3:
135         file_name = raw_input(" [*]
136             Enter name of file to
137             upload: ")
138         file =
139             drive.CreateFile({"parents":
140                 [{"kind": "drive#fileLink",
141                     "id":
142                         "1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f"}], 'title': file
143                 source_file =
144                     open("secret_files/" +
145                         file_name, "r")
146         print(" [*] Encrypting file and
147             uploading...")
148         data = source_file.read()
149         encrypted_data = f.encrypt(data)
150         file.SetContentString(encrypted_data.decode())
151         file.Upload()
152         print(" [*] Uploaded!")
153         f_list =
154             drive.ListFile({'q':"1153l9SNSC2qwj6wfDCMFQvXMO
155                 in parents and
156                 trashed=false"}).GetList()
157         print(" [*]")
158
159     # Removes a file.. self explanatory.
160     elif option is 4:
161         found = 0
162         show_files(f_list)
163         file_name = raw_input(" [*]

```

```

147         Enter name of file: ")
148     logo(username)
149     for file in f_list:
150         if file["title"] ==
151             file_name:
152                 found = 1
153                 file.Delete()
154                 print("[*]
155                     Deleted " +
156                     file_name +
157                     " !")
158                 print("[*]")
159     if found is 0:
160         print("[*] No such file
161             name in drive..
162             enter '1' to see
163             list of files.")
164         print("[*]")
165
166     # Exit.
167     elif option is 5:
168         print("[*] Thanks for using the
169             program!")
170         print("[*]")
171         finished = True
172
173     else:
174         print("[*] Please pick one of
175             the options listed below..")
176         print("[*]")
177
178     # If user is not a valid user, print this.
179     else:
180         print("[*] You are not part of the admin's
181             group.. please contact admin for an invite.")
182
183 if __name__ == "__main__":
184     main()

```

0.5 Server.py

```

1 from cryptography.hazmat.primitives.serialization import
    load_pem_private_key
2 from cryptography.hazmat.backends import default_backend
3 from cryptography.hazmat.primitives import serialization
4 from cryptography.hazmat.primitives import hashes
5 from cryptography.hazmat.primitives.asymmetric import padding
6 from pydrive.auth import GoogleAuth
7 from pydrive.drive import GoogleDrive
8 from cryptography.fernet import Fernet
9 from encrypt import encrypt
10 from decrypt import decrypt
11 import requests
12 import base64
13 import os
14
15 from flask import Flask
16 from flask import request
17 app = Flask(__name__)
18
19 # Create route for get_key function
20 @app.route('/get_key', methods=['GET'])
21 def index():
22     # Grab username and public_key for that user.
23     username = request.args.get('username')
24     pub_key = request.args.get('pub_key').encode('ascii')
25     public_key =
        serialization.load_pem_public_key(pub_key, backend=default_backend())
26
27     # Grab the unencrypted symmetric key..
28     key = open("keys/symmetric_key.txt", "r")
29     key = key.read()
30
31     # Encrypt the symmetric key..
32     encrypted = public_key.encrypt(
33         key,
34         padding.OAEP(
35             mgf=padding.MGF1(algorithm=hashes.SHA256()),
36             algorithm=hashes.SHA256(),
37             label=None
38         )
39     )
40
41     # Return the symmetric key specifically encrypted for
        that user...
42     encrypted = base64.b64encode(encrypted)

```

```
43
44         print("encrypted symmetric key.. sending to client..")
45         return encrypted
46
47 if __name__ == '__main__':
48     app.run(debug = True)
```