

# 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.

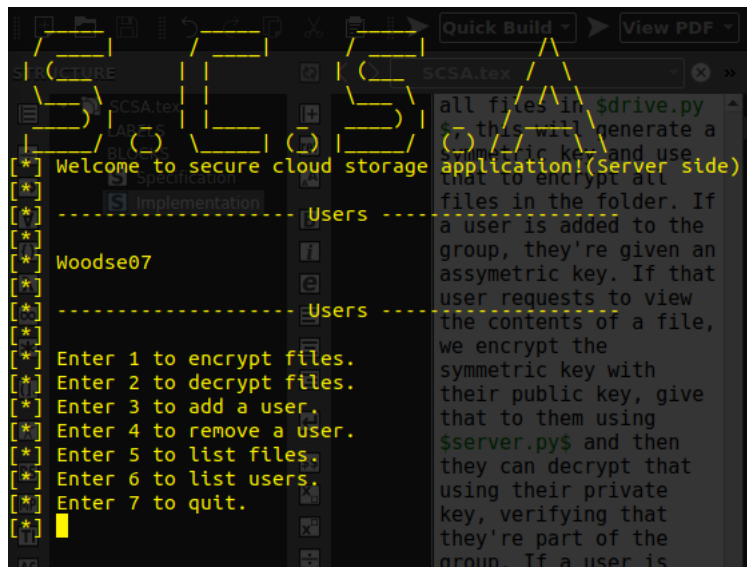
*server.py* is a flask server, which listens for *client.py* to send it a username, and will return the encrypted version of the symmetric key for that user to the client, where it can then be decrypted.

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..



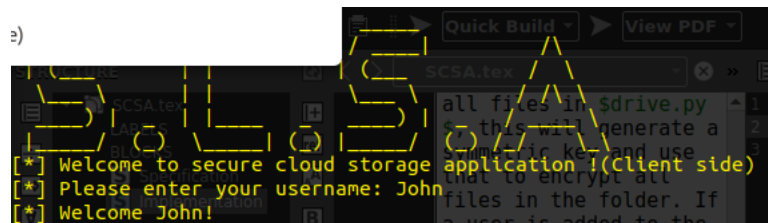
```
[*] Welcome to secure cloud storage application!(Server side)
[*] -----Users-----
[*] Woodse07
[*] -----Users-----
[*] Enter 1 to encrypt files.
[*] Enter 2 to decrypt files.
[*] Enter 3 to add a user.
[*] Enter 4 to remove a user.
[*] Enter 5 to list files.
[*] Enter 6 to list users.
[*] Enter 7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

Server side after adding user 'John'



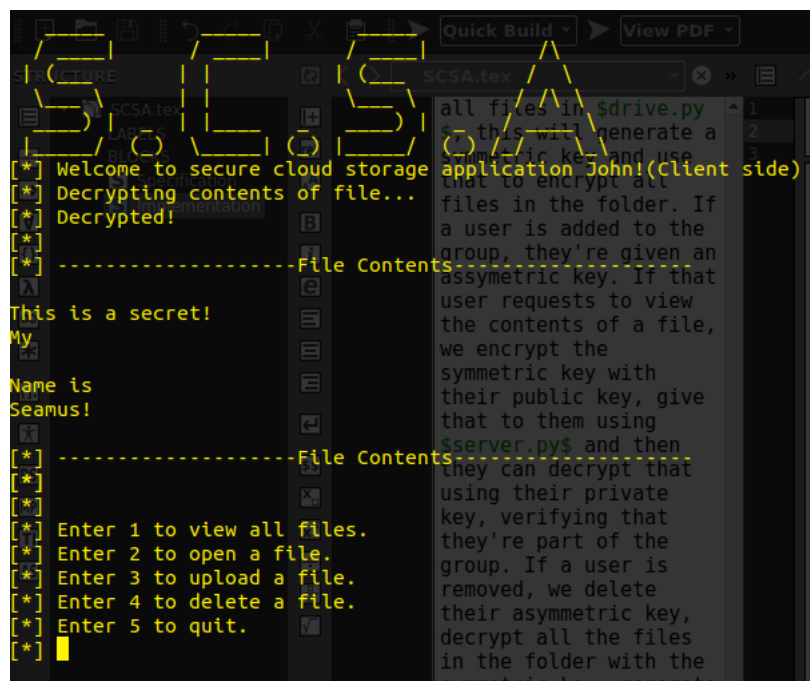
```
[*] Welcome to secure cloud storage application!(Server side)
[*] Enter username: John
[*] Adding user John..
[*] Generating keys for user..
[*] Added user John!
[*] Enter 1 to encrypt files.
[*] Enter 2 to decrypt files.
[*] Enter 3 to add a user.
[*] Enter 4 to remove a user.
[*] Enter 5 to list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

Client side after logging in as 'John'



The screenshot shows a terminal window with a dark background. At the top, there's a header with 'SCSA.tex' and some navigation buttons. Below that, the text 'Welcome to secure cloud storage application!(Client side)' is displayed. This is followed by 'Please enter your username: John' and 'Welcome John!'. The terminal also shows some code snippets from a file named 'SCSA.tex'.

Client side after requesting to view 'secret1.txt'



The screenshot shows a terminal window with a dark background. At the top, there's a header with 'SCSA.tex' and some navigation buttons. Below that, the text 'Welcome to secure cloud storage application John!(Client side)' is displayed. This is followed by 'Decrypting contents of file...' and 'Decrypted!'. Then, a section titled '-----File Contents-----' is shown, containing the text 'This is a secret!' and 'My Name is Seamus!'. Another section titled '-----File Contents-----' follows, containing a list of options: 'Enter 1 to view all files.', 'Enter 2 to open a file.', 'Enter 3 to upload a file.', 'Enter 4 to delete a file.', and 'Enter 5 to quit.'. The terminal also shows some code snippets from a file named 'SCSA.tex'.

### 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 def show_users():
13     subs = os.listdir('group/')
14     print("[*] _____ Users
15         _____")
16     print("[*]")
17     for sub in subs:
18         print("[*] " + sub)
19     print("[*] _____ Users
20         _____")
21     print("[*]")
22
23 def show_files(f_list):
24     print("[*] _____ Files
25         _____")
26     print("[*]")
27     for file in f_list:
28         print("[*] " + file['title'])
29     print("[*] _____ Files
30         _____")
31     print("[*]")
32
33 def logo():
34     os.system('cls' if os.name == 'nt' else 'clear')
35     print("
36         ")
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)")
51
52 def main():

```

```

43     logo()
44     auth = GoogleAuth()
45     auth.LocalWebserverAuth()
46     drive = GoogleDrive(auth)
47
48     try:
49         f = open('keys/symmetric_key.txt', 'r')
50         key = f.read()
51         print("[*] Your symmetric key: " + key + " ")
52     except:
53         key = Fernet.generate_key()
54         f = open('keys/symmetric_key.txt', 'w')
55         f.write(key)
56         print("[*] Your symmetric key: " + key + " ")
57
58     f = Fernet(key)
59     f_list =
        drive.ListFile({ 'q': "'1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f'
        in parents and trashed=false" }).GetList()
60
61     finished = False
62     logo()
63     print("[*]")
64     while not finished:
65         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[*] ")
66         logo()
67         print("[*]")
68         if option is 1:
69             encrypt(f, f_list)
70
71         elif option is 2:
72             decrypt(f, f_list)
73
74         elif option is 3:
75             username = raw_input("[*] Enter
            username: ")
76             if os.path.exists("group/" +
            str(username)):
77                 print("[*] Username taken.")
78                 print("[*]")

```

```

79         else:
80             print("[*] Adding user " +
81                   str(username) + "..")
82             # Creating dir for user..
83             os.mkdir("group/" +
84                     str(username))
85
86             print("[*] Generating keys for
87                   user..")
88             # Generating private key..
89             private_key =
90                 rsa.generate_private_key(
91                 public_exponent=65537,
92                 key_size=2048,
93                 backend=default_backend())
94
95             # Serialising keys for storing..
96             private_serialized =
97                 private_key.private_bytes(
98                 encoding=serialization.Encoding.PEM,
99                 format=serialization.PrivateFormat.PKCS8,
100                 encryption_algorithm=serialization.NoEncryption())
101
102             # Storing private key..
103             file = open("group/" +
104                        str(username) +
105                        "/private_key.txt", "w")
106             file.write(private_serialized)
107             file.close()
108             print("[*] Added user " +
109                   str(username) + "!!")
110             print("[*]")
111
112     elif option is 4:
113         show_users()
114         username = raw_input("[*] Enter
115                               username: ")
116         logo()
117         if os.path.exists("group/" +
118                           str(username)):
119             print("[*] Removing user..")
120             shutil.rmtree("group/" +
121                           str(username))
122             print("[*] Generating new
123                   symmetric key and

```

```

112         re-encrypting all files..)
113     decrypt(f, f_list)
114     key = Fernet.generate_key()
115     file =
116         open('keys/symmetric_key.txt',
117             'w')
118     file.write(key)
119     file.close()
120     f = Fernet(key)
121     print("[*] New symmetric key: "
122         + key)
123     encrypt(f, f_list)
124     print("[*]")
125     logo()
126     print("[*]\n[*] Deleted user "
127         + username + "!\n[*]")
128
129     else:
130         print("[*] Username does not
131             exist.. enter '6' to see
132             list of users.")
133         print("[*]")
134
135     elif option is 5:
136         show_files(f_list)
137
138     elif option is 6:
139         show_users()
140
141     elif option is 7:
142         print("[*] Thanks for using the
143             program!")
144         finished = True
145
146     else:
147         print("[*] Please pick one of the
148             options listed below..")
149         print("[*]")
150
151 if __name__ == "__main__":
152     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 # getKey() function is temporary.. will replace with some web
framework?
16 def getKey(username):
17     with open("group/" + str(username) +
18             "/private_key.txt", "rb") as key_file:
19         private_key = key_file.read()
20         private_key = load_pem_private_key(private_key, None,
21             default_backend())
22         public_key = private_key.public_key()
23         public_key = public_key.public_bytes(
24             encoding=serialization.Encoding.PEM,
25             format=serialization.PublicFormat.SubjectPublicKeyInfo)
26
27     print("[*] Requesting symmetric key from server..")
28     URL = "http://127.0.0.1:5000/get_key"
29     PARAMS = {'username': username, 'pub_key': public_key}
30     r = requests.get(url=URL, params=PARAMS)
31     encrypted = r.text
32     print("[*] Received encrypted symmetric key from
33           server..")
34
35     encrypted = base64.b64decode(encrypted)
36     symmetric_key = private_key.decrypt(
37         encrypted,
38         padding.OAEP(
39             mgf=padding.MGF1(algorithm=hashes.SHA256()),
40             algorithm=hashes.SHA256(),
41             label=None
42         )
43     )
44     print("[*] Decrypted symmetric key!")

```

```

43         return symmetric_key
44
45     def show_files(f_list):
46         print("[*] _____ Files
47             _____")
48         print("[*]")
49         for file in f_list:
50             print("[*] " + file['title'])
51         print("[*] _____ Files
52             _____")
53         print("[*]")
54     def logo(username):
55         os.system('cls' if os.name == 'nt' else 'clear')
56         print("      -----      -----      -----
57             ")
58         print(" /  ----|      /  ----|      /  ----|      /\
59             ")
60         print(" | (---      | |      | (---      / \
61             ")
62         print(" \--- \      | |      \--- \      / /\ \
63             ")
64         print(" ----) | - | |---- - ----) | - / ----
65             \ ")
66         print(" |-----/ (-) \-----| (-) |-----/ (-) /- /
67             \-\ ")
68         print("[*] Welcome to secure cloud storage application
69             "+username+"!(Client side)")
70
71
72
73
74
75     def main():
76         logo("")
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             key = getKey(username)
85             print("[*] " + key + " ")
86             f = Fernet(key)
87             f_list =

```

```

79         drive.ListFile({ 'q': "115319SNSC2qwj6wfDCMFQvXMOhX1BI0f'
80         in parents and trashed=false" }).GetList()
81
82 finished = False
83 logo(username)
84 while not finished:
85     print("[*]")
86     option = input("[*] Enter 1 to view all
87             files.\n[*] Enter 2 to open a
88             file.\n[*] Enter 3 to upload a
89             file.\n[*] Enter 4 to delete a
90             file.\n[*] Enter 5 to quit.\n[*] ")
91     logo(username)
92     print("[*]")
93     if option is 1:
94         show_files(f_list)
95
96     elif option is 2:
97         found = 0
98         show_files(f_list)
99         file_name = raw_input("[*]
100             Enter name of file: ")
101         logo(username)
102         for file in f_list:
103             if file["title"] ==
104                 file_name:
105                 found = 1
106                 encoded =
107                     file.GetContentString()
108                 print("[*]
109                     Decrypting
110                     contents of
111                     file ...")
112                 decoded =
113                     f.decrypt(encoded.encode())
114                 print("[*]
115                     Decrypted!")
116                 print("[*]")
117                 print("[*]
118                     _____File
119                     Contents_____")
120                 print("")
121                 print(decoded.decode())
122                 print("[*]
123                     _____File

```

```

107                                     Contents_____")
108                                     print(" [*]")
109     if found is 0:
110         print(" [*] No such file
111             name in drive..
112             enter '1' to see
113             list of files.")
114         print(" [*]")
115
116 elif option is 3:
117     file_name = raw_input(" [*]
118         Enter name of file to
119         upload: ")
120     file =
121         drive.CreateFile({"parents":
122             [{"kind": "drive#fileLink",
123                 "id":
124                     "1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f"}], 'title': file_name})
125     source_file =
126         open("secret_files/" +
127             file_name, "r")
128     print(" [*] Encrypting file and
129         uploading...")
130     data = source_file.read()
131     encrypted_data = f.encrypt(data)
132     file.SetContentString(encrypted_data.decode())
133     file.Upload()
134     print(" [*] Uploaded!")
135     f_list =
136         drive.ListFile({'q': "'1153l9SNSC2qwj6wfDCMFQvXMOhX1BI0f' in parents and
137             trashed=false"}).GetList()
138     print(" [*]")
139
140 elif option is 4:
141     found = 0
142     show_files(f_list)
143     file_name = raw_input(" [*]
144         Enter name of file: ")
145     logo(username)
146     for file in f_list:
147         if file["title"] ==
148             file_name:
149             found = 1
150             file.Delete()

```

```

134                                     print("[*]
                                     Deleted " +
                                     file_name +
                                     " !")
135                                     print("[*]")
136     if found is 0:
137         print("[*] No such file
                                     name in drive..
                                     enter '1' to see
                                     list of files.")
138         print("[*]")
139
140     elif option is 5:
141         print("[*] Thanks for using the
                                     program!")
142         print("[*]")
143         finished = True
144
145     else:
146         print("[*] Please pick one of
                                     the options listed below..")
147         print("[*]")
148
149
150
151
152
153
154     else:
155         print("[*] You are not part of the admin's
                                     group.. please contact admin for an invite.")
156
157 if __name__ == "__main__":
158     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 @app.route('/get_key', methods=['GET'])
20 def index():
21     username = request.args.get('username')
22     pub_key = request.args.get('pub_key').encode('ascii')
23
24     public_key =
25         serialization.load_pem_public_key(pub_key, backend=default_backend())
26
27     key = open("keys/symmetric_key.txt", "r")
28     key = key.read()
29
30     encrypted = public_key.encrypt(
31         key,
32         padding.OAEP(
33             mgf=padding.MGF1(algorithm=hashes.SHA256()),
34             algorithm=hashes.SHA256(),
35             label=None
36         )
37     )
38     encrypted = base64.b64encode(encrypted)
39
40     print("encrypted symmetric key.. sending to client..")
41     return encrypted
42
43 if __name__ == '__main__':
44     app.run(debug = True)

```