Advanced Telecommunications - Securing the cloud

Owen Burke, 15316452

April 9, 2019

High-level description of the protocol design and implementation

See images below for proof of work

The aim of this project is to develop a secure cloud storage application for Dropbox, Box, Google Drive, Office365etc. For example, your application will secure all files that are uploaded to the cloud, such that only people that are part of your "Secure Cloud Storage Group" will be able to decrypt your uploaded files. To all other users the files will be encrypted.

You are required to design and implement a suitable key management system for your application that will allow you to share files securely, and add or remove users from your group. You are free to implement your application for a desktop or mobile platform and make use of any open source cryptographic libraries.

High-level description :

The first step was to establish a way for a single user to encrypt and decrypt their own files. This was done using RSA and AES. The AES symmetric key would be used to encrypt and decrypt all files. (See the encryptFile and decryptFile functions).
This was relatively simple. The symmetric key would be encrypted with the user's public key and when the symmetric key was needed, it would be decrypted with the user's private key. The choice was made to encrypt the symmetric key for each user because the symmetric key would be shared (across dropbox in this case) and to have the symmetric key unencrypted in public view would be a security hole.
So, in this system, each user has an encrypted version of the symmetric key ready for them to access. The user pulls it down and decrypts it with their private rsa key and goes about using it. The person who added that user will take the symmetric key for themselves, decrypt it and then encrypt it with the new user's public key. (each user's public key will also be made available to all users, as they are needed if they want to encrpyt the symmetric key for somebody as described above.)

Firstly, when the program is started, the user is prompted for their dropbox access token (in order to access their account). They are also asked for their name. They are then asked for the name of the shared project. For this, the relevant sub-folders are made (original files, encrypted files, decrypted files, etc). This allows users to have multiple shared projects/groups by providing a different shared project name with many different users as each project has an individual members list.

If the user does not currently have any public keys on their dropbox, they are assumed to be making a new shared project (as you can see around line 170). They make a RSA key pair for themselves and encrypt a 16 byte symmetric key (see the genSymKey and generateRSA_pairFiles functions) and upload these to dropbox.
They are then prompted for user input. They can encrypt a file (see encryptFile), decrypt (see decryptFile), add a user, remove a user, encrypt a symmetric key for somebody, or quit the program.
Encrypting a file is done as described above. They take their private key. Use this to decrypt their version of the encrypted symmetric key and then use this symmetric key to encrypt the file and upload it to dropbox.
Decrypting the file is a similar process in reverse. They take their private key, decrypt the symmetric key for them and then use this key to decrypt the file.

To add a user, they simply provide a username and this is added to a shared members file (listing the members of the group). When the program is started, if the user is not initially making a project, the members file is checked. If they are not a member, they are told this and the program exits. If they are a member, but don't have any keys, then they make themselves an RSA key pair and share the public key (the new user must do this on their own system themselves rather than the person that added them, due to security issues regarding the sharing of private keys). They are then told to wait for a symmetric key to be encrypted for them. If they already have all their keys, they are told this and prompted for input.
For removing a user, they provided username is removed from the members file and when they aforementioned user tries to use them system, the program closes as they are not on the list anymore. When the user is removed from the list, all the files on the system are decrypted and re-encrypted with a new symmetric key. All the users that are still in the group will also receive new encrypted symmetric keys (to decrypt old and new files, while the removed user can no longer decrypt any files on the system. This re-encryption is all done in the removeUser function).
The user can also encrypt a symmetric key for a recently added user. This is done by taking their own version of the encrpyted symmetric key and decrpyting it. They then take the new user's public key and encrpyt the symmetric key with it and push it to the dropbox, so that the new user can encrypt and decrypt files like anybody else.

Code :

```
1 from Crypto.PublicKey import RSA
2 from Crypto.Random import get_random_bytes
3 from Crypto.Cipher import AES, PKCS1_OAEP
4 import Crypto
```

```
5  import dropbox
6  import os
7  import zipfile
8
9  project = ""
10
11
12 def generateRSA_pairFiles(username):    # generate a
       public and private key pair for the given username and
        save in the relevant .pem files
13     key = RSA.generate(2048)
14     encrypted_key = key.exportKey(pkcs=8, protection="
           scryptAndAES128-CBC")
15     with open('./' + project + '/' + username + '
           _private_rsa_key.pem', 'wb') as f:
16             f.write(encrypted_key)
17     with open('./' + project +'/Public_Keys/' + username
           + '_rsa_public.pem', 'wb') as f:
18             f.write(key.publickey().exportKey())
19
20
21 #
       ###################################################################

22
23
24 def encryptFile(username, fileToEncrypt): #encrypts file
       with sym key
25     with open('./' + project + '/Encrypted_Files/' + "
           Encrypted_" + fileToEncrypt, 'wb') as out_file:
           # create the intial encrypted file
26         with open('./' + project + '/' + username + '
               _private_rsa_key.pem', 'rb') as f:
                                   # get the user's
               private key
27                 data = f.read()
28                 key = RSA.importKey(data)
29                 cipher = PKCS1_OAEP.new(key)
30                 plainSymKey = cipher.decrypt(open('./' +
                       project + '/Sym_Keys/' + username + '
                       _SymKey.bin', 'rb').read())  # decrypt
                        the symmetric key for the user
31                 cipher = AES.new(plainSymKey, AES.
                       MODE_ECB)
32                 data = open('./' + project + '/
                       Original_Files/' + fileToEncrypt, 'rb'
```

```
                    ).read()
33                  data = Crypto.Util.Padding.pad(data, 16,
                        style = 'pkcs7')        # pad the file
34                  ciphertext = cipher.encrypt(data)       #
                        encrypt the file with the decrypted
                        symmetric key
35                  out_file.write(ciphertext)      # write
                        to the encrypted file
36
37
38  #
    ###################################################################
39
40
41  def decryptFile(username, fileToDecrypt, dbx): #decrypts
        the file with the sym key
42      with open('./' + project + '/Encrypted_Files/' +
            fileToDecrypt, 'rb') as fobj:       # open the
            encrypted file
43          with open('./' + project + '/' + username + '
                _private_rsa_key.pem', 'rb') as f:          #
                get the users private key
44                  dbx.files_download_to_file('./' + project
                        + '/Sym_Keys/' + username + '_SymKey.
                        bin', '/'+project+'/Sym_Keys/' +
                        username + '_SymKey.bin') # get the
                        most up to date sym keys
45                  data = f.read()
46                  key = RSA.importKey(data)
47                  cipher = PKCS1_OAEP.new(key)
48                  plainSymKey = cipher.decrypt(open('./' +
                        project + '/Sym_Keys/' + username + '
                        _SymKey.bin', 'rb').read())         #
                        decrypt the sym key for the user with
                        their private key
49                  data = fobj.read()
50                  cipher = AES.new(plainSymKey, AES.
                        MODE_ECB)
51                  padDecrypt = cipher.decrypt(data)
                                                            #
                        decrypt the encrypted file with the
                        sym key
52                  orig  = Crypto.Util.Padding.unpad(
                        padDecrypt, 16, style = 'pkcs7')
                        # unpad the data
```

```
53                  with open('./' + project + '/
                        Decrypted_Files/' + "Decrypted_" +
                        fileToDecrypt.split("Encrypted_")[1],
                        'wb') as f:
54                      f.write(orig)       # write to the
                            decrypted file
55
56
57  #
        ############################################################


58
59
60  def addUser(username, dbx):
61          dbx.files_download_to_file('./' + project + '/
                members.txt', '/'+project+'/members/members.
                txt') # pull the members file
62          f= open("./" + project + "/members.txt","a")
63          f.write(username + "\n")        # write the new
                user to the file
64          f.close()
65          dbx.files_upload(open("./" + project + "/members.
                txt", 'rb').read(), '/'+project+'/members/
                members.txt', mode = dropbox.files.WriteMode('
                overwrite')) # push the updated members file
66
67
68  #
        ############################################################


69
70
71  def encryptSymKeyForUser(username, adminName): # encrypt
        the symmetric key for a new user
72          encryptZip = dbx.files_download_zip_to_file("./"
                + project + "/Public_Keys/pubKeyZip.zip", '/'+
                project+'/Public_Keys')
73          zip_ref = zipfile.ZipFile("./" + project + "/
                Public_Keys/pubKeyZip.zip", 'r')
74          zip_ref.extractall("./" + project)
75          zip_ref.close()
76          os.remove("./" + project + "/Public_Keys/
                pubKeyZip.zip")                 # retrieve all
                the most up to date public keys from dropbox
77
78
```

```
79          admin_private_key = RSA.import_key(open('./' +
                project + '/' + adminName + '_private_rsa_key.
                pem', 'rb').read()) # get the private key for
                the user that is doing the encrypting
80          cipher = PKCS1_OAEP.new(admin_private_key)
81          plainSymKey = cipher.decrypt(open('./' + project
                + '/Sym_Keys/' + adminName + '_SymKey.bin', '
                rb').read()) # decrypt the symmetric key
82
83          key = RSA.importKey(open('./' + project + '/
                Public_Keys/' + username + '_rsa_public.pem').
                read()) # get the new users public key
84          cipher = PKCS1_OAEP.new(key)
85          encryptedSymKey = cipher.encrypt(plainSymKey)   #
                 encrypt the symmetric key with the new users
                public key
86          with open('./' + project + '/Sym_Keys/' +
                username + '_SymKey.bin', 'wb') as out_file:
87                  out_file.write(encryptedSymKey) # write
                        out this encrypted sym key
88
89
90  #
        ###################################################################
91
92
93  def genSymKey(adminName): # makes rsa keys and sym key
        for the first user
94          generateRSA_pairFiles(adminName) # make rsa keys
95          session_key = get_random_bytes(16) # make the
                symmetric key
96          key = RSA.importKey(open('./' + project + '/
                Public_Keys/' + adminName + '_rsa_public.pem')
                .read()) # get the users public key
97          cipher = PKCS1_OAEP.new(key)
98          encryptedSymKey = cipher.encrypt(session_key)   #
                 encrypt the symmetric key
99          with open('./' + project + '/Sym_Keys/' +
                adminName + '_SymKey.bin', 'wb') as out_file:
100                 out_file.write(encryptedSymKey) # write
                        out the encrypted symmetric key
101
102
103 #
        ###################################################################
```

```
104
105
106 def removeUser(username, superUser, dbx) :
107         dbx.files_download_to_file('./' + project + '/
                members.txt', '/'+project+'/members/members.
                txt') # pull the members file
108         f = open("./" + project + "/members.txt","r")
109         people = f.read()
110         people = people.split("\n")     # get the names
111         try:
112                 people.remove(username)          # remove
                        the user
113                 f = open("./" + project + "/members.txt",
                        "w")
114                 people = "\n".join(people)
115                 f.write(people) # write back to the file
116                 f.close()
117                 dbx.files_upload(open("./" + project + "/
                        members.txt", 'rb').read(), '/'+
                        project+'/members/members.txt', mode =
                         dropbox.files.WriteMode('overwrite'))
                         # push the updated members list
118
119                         ################ RE-ENCRYPT STUFF
                                WITH NEW SYM KEY
                            ##############
120
121                 encryptZip = dbx.
                        files_download_zip_to_file("./" +
                        project + "/Encrypted_Files/encryptZip
                        .zip", '/'+project+'/Encrypted_Files')
122                 zip_ref = zipfile.ZipFile("./" + project
                        + "/Encrypted_Files/encryptZip.zip", '
                        r')
123                 zip_ref.extractall("./" + project)
124                 zip_ref.close()
125                 os.remove("./" + project + "/
                        Encrypted_Files/encryptZip.zip")
126                 ####Pull encrypted files from dbx
127                 directory = "./" + project + "/
                        Encrypted_Files/"
128                 for file in os.listdir(directory):
129                         filename = os.fsdecode(file)
130                         decryptFile(superUser, filename,
                                dbx)   # decrypt all files
```

```
131
132                 session_key = get_random_bytes(16) # make
                        the new symmetric key
133                 key = RSA.importKey(open('./' + project +
                        '/Public_Keys/' + superUser + '
                        _rsa_public.pem').read()) # get the
                        users public key
134                 cipher = PKCS1_OAEP.new(key)
135                 encryptedSymKey = cipher.encrypt(
                        session_key)   # encrypt the symmetric
                         key
136                 with open('./' + project + '/Sym_Keys/' +
                         superUser + '_SymKey.bin', 'wb') as
                        out_file:
137                     out_file.write(encryptedSymKey) #
                            write out the encrypted
                            symmetric key
138
139                 directory = "./" + project + "/
                        Decrypted_Files/"       # re-encrypt
                        all files
140                 for file in os.listdir(directory):
141                     filename = os.fsdecode(file)
142                     encryptFile(superUser, filename
                            [10:])
143                     dbx.files_upload(open("./" +
                            project + "/Encrypted_Files/"
                            + "Encrypted_" + filename
                            [10:], 'rb').read(), '/'+
                            project+'/Encrypted_Files/' +
                            "Encrypted_" + filename[10:],
                            mode = dropbox.files.WriteMode
                            ('overwrite'))
144
145
146                 encryptZip = dbx.
                        files_download_zip_to_file("./" +
                        project + "/Public_Keys/pubKeyZip.zip"
                        , '/'+project+'/Public_Keys')
147                 zip_ref = zipfile.ZipFile("./" + project
                       + "/Public_Keys/pubKeyZip.zip", 'r')
148                 zip_ref.extractall("./" + project)
149                 zip_ref.close()
150                 os.remove("./" + project + "/Public_Keys/
                        pubKeyZip.zip")                 #
                        retrieve all the most up to date
```

```
                    public keys from dropbox
151
152                 directory = "./" + project + "/
                        Public_Keys/"       # re-encrypt all
                        files
153                 for file in os.listdir(directory):
154                         filename = os.fsdecode(file)
155                         name = filename.split('_')[0]
156                         if name not in people.split("\n")
                                :
157                                 continue
158                         key = RSA.importKey(open('./' +
                                project + '/Public_Keys/' +
                                name + '_rsa_public.pem').read
                                ()) # get the new users public
                                 key
159                         cipher = PKCS1_OAEP.new(key)
160                         encryptedSymKey = cipher.encrypt(
                                session_key)   # encrypt the
                                symmetric key with the new
                                users public key
161                         with open('./' + project + '/
                                Sym_Keys/' + name + '_SymKey.
                                bin', 'wb') as out_file:
162                                 out_file.write(
                                        encryptedSymKey) #
                                        write out this
                                        encrypted sym key
163                         dbx.files_upload(open('./' +
                                project + '/Sym_Keys/' + name
                                + '_SymKey.bin', 'rb').read(),
                                 "/"+project + "/Sym_Keys/" +
                                name + "_SymKey.bin", mode =
                                dbx.files.WriteMode('
                                overwrite'))
164
165         except:
166                 print("This user is not in the group")
167
168 #
    ###################################################################
169
170
171 def getUserInput(username, command, dbx): # deals with
        all the user input
```

```
172             if command.lower() == 'encrypt': # encrypt a file
173                     filename = input("Give a File Name :\n\t"
                            )
174                     try:
175                             encryptFile(username, filename)
176                             dbx.files_upload(open("./" +
                                    project + "/Encrypted_Files/"
                                    + "Encrypted_" + filename, 'rb
                                    ').read(), '/'+project+'/
                                    Encrypted_Files/' + "
                                    Encrypted_" + filename)
177                     except FileNotFoundError:
178                             print("File does not exist")
179
180             elif command.lower() == 'decrypt': # decrypt a
                    file
181                     ####Pull encrypted files from dbx
182                     try:
183                             encryptZip = dbx.
                                    files_download_zip_to_file("./
                                    " + project + "/
                                    Encrypted_Files/encryptZip.zip
                                    ", '/'+project+'/
                                    Encrypted_Files')
184                             zip_ref = zipfile.ZipFile("./" +
                                    project + "/Encrypted_Files/
                                    encryptZip.zip", 'r')
185                             zip_ref.extractall("./" + project
                                    )
186                             zip_ref.close()
187                             os.remove("./" + project + "/
                                    Encrypted_Files/encryptZip.zip
                                    ")
188                             ####Pull encrypted files from dbx
189
190                             filename = input("Give a File
                                    Name :\n\t")
191                             decryptFile(username, "Encrypted_
                                    " + filename, dbx)
192                     except:
193                             print("No files have been
                                    encrypted yet")
194
195             elif command.lower() == 'add user':
196                     newUser = input("Provide a new user name
                            :\n\t")
```

```
197                 addUser(newUser, dbx) #add new user name
                        to members file
198                 print("Wait for them to make their rsa
                        keys.")
199
200         elif command.lower() == 'remove user': # remove a
                user from the members file
201                 userToRemove = input("What user do you
                        want to remove? : \n\t")
202                 removeUser(userToRemove, username, dbx)
203
204         elif command.lower() == "sym key for user": #
                encrpyt a sym key for a new user
205                 newKeyRecv = input("Who are you
                        encrypting the keys for? :\n\t")
206                 try:
207                         encryptSymKeyForUser(newKeyRecv,
                            username)
208                         dbx.files_upload(open('./' +
                            project + '/Sym_Keys/' +
                            newKeyRecv + '_SymKey.bin', '
                            rb').read(), "/"+project + "/
                            Sym_Keys/" + newKeyRecv + "
                            _SymKey.bin") # push the new
                            sym key
209                         os.remove('./' + project + '/
                            Sym_Keys/' + newKeyRecv + '
                            _SymKey.bin')
210                 except:
211                         print("This user is not in the
                            group")
212
213         elif command.lower() == 'quit': # quit the
                program
214                 exit(1)
215
216         else:
217                 print("Invalid input. Try again") #
                        invalid input
218
219
220 #
    ##########################################################################

221
222 def isInGroup(username, dbx): # check user is in group
```

```
223         dbx.files_download_to_file('./' + project + '/
                members.txt', '/'+project+'/members/members.
                txt') # check they have the members file
224         people = open("./" + project + "/members.txt","r"
                ).read()
225         people = people.split("\n")
226         if username in people:  # they are a members
227                 return True
228         return False
229
230
231 #
    ##################################################################
232 #### MAIN ####
233 #
    ##################################################################

234
235
236 apiKey = input("Provide your access token :\n\t") # get
        access token
237 username = input("Provide your name :\n\t") # get name
238 dbx = dropbox.Dropbox(apiKey)
239 dbx.users_get_current_account() # initialise dropbox
240 project = input("Provide your shared project name :\n\t")
241
242 isAdmin = False
243 try:
244         meta = dbx.files_get_metadata("/"+project+"/
                Public_Keys") # check are they making an
                initial project
245 except:
246         print("You're the admin") # they are the first
                user
247         isAdmin = True
248         os.mkdir(project)
249         os.mkdir("./" + project + "/Original_Files")
250         os.mkdir("./" + project + "/Public_Keys")
251         os.mkdir("./" + project + "/Sym_Keys")
252         os.mkdir("./" + project + "/Encrypted_Files")
253         os.mkdir("./" + project + "/Decrypted_Files")
254         genSymKey(username) # make their keys
255
256         directory = "./" + project + "/Public_Keys/"
257         for file in os.listdir(directory):
```

```
258                     filename = os.fsdecode(file)
259                     dbx.files_upload(open(directory +
                            filename, 'rb').read(), '/'+project+'/
                            Public_Keys/' + filename) # upload the
                             keys
260
261             directory = "./" + project + "/Sym_Keys/"
262             for file in os.listdir(directory):
263                     filename = os.fsdecode(file)
264                     dbx.files_upload(open(directory +
                            filename, 'rb').read(), '/'+project+'/
                            Sym_Keys/' + filename) # upload the
                            keys
265
266             f= open("./" + project + "/members.txt","w+")
267             f.write(username + "\n")
268             f.close()
269             dbx.files_upload(open("./" + project + "/members.
                    txt", 'rb').read(), '/'+project+'/members/
                    members.txt') # make the members file
270
271
272     if isAdmin == False: # they are not making an initial
            project
273             try:
274                     os.mkdir(project)
275                     os.mkdir("./" + project + "/
                            Original_Files")
276                     os.mkdir("./" + project + "/Public_Keys")
277                     os.mkdir("./" + project + "/Sym_Keys")
278                     os.mkdir("./" + project + "/
                            Encrypted_Files")
279                     os.mkdir("./" + project + "/
                            Decrypted_Files")
280             except:
281                     None
282             dbx.files_download_to_file('./' + project + '/
                    members.txt', '/'+project+'/members/members.
                    txt') # check they have the members file
283
284             #check is in members
285             people = open("./" + project + "/members.txt","r"
                    ).read()
286             people = people.split("\n")
287             if username in people:  # they are a members
288                     try:
```

```
289                             meta = dbx.files_get_metadata("/"
                                    +project+"/Sym_Keys/" +
                                    username + "_SymKey.bin") #
                                    check do they have keys
290                     except:
291                             ## make keys
292                             generateRSA_pairFiles(username)
                                                # they don't have
                                    keys - make their public and
                                    private keys and push the
                                    public key
293                             for file in os.listdir("./" +
                                    project + "/Public_Keys/"):
294                                     filename = os.fsdecode(
                                            file)
295                                     dbx.files_upload(open("./
                                            " + project + "/
                                            Public_Keys/" +
                                            filename, 'rb').read()
                                            , '/'+project+'/
                                            Public_Keys/' +
                                            filename, mode =
                                            dropbox.files.
                                            WriteMode('overwrite')
                                            )
296                             print("Made rsa keys")
297                             print("Wait for a sym key to be
                                    encrypted for you") # wait for
                                     a symmetric key to be
                                    encrypted for you
298                             exit(-1)
299             else :
300                     print("No!!! You ain't in the group!!!")
                            # the user is not a member
301                     exit(-1)
302             print("You are in the group and you have keys") #
                    they are a member and have their keys
303
304
305     while 1:
306             command = input("Enter a command (encrypt,
                    decrypt, add user, remove user, sym key for
                    user) or 'quit':\n\t") # get the user input
307             if isInGroup(username, dbx):
308                     getUserInput(username, command, dbx)
309             else:
```

```
310                     print("You've been removed from the group")
311                     exit(-1)
312
313
314 ##########################################################
```

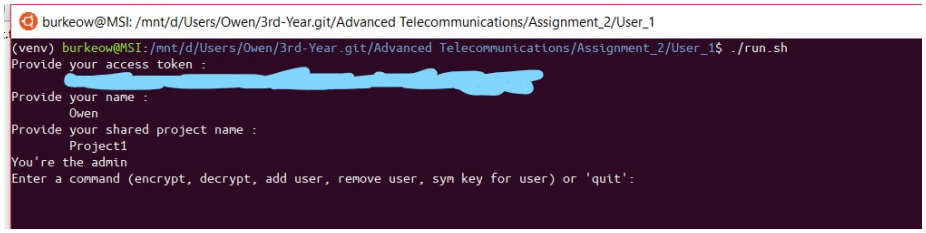


Figure 1: First user - initial project

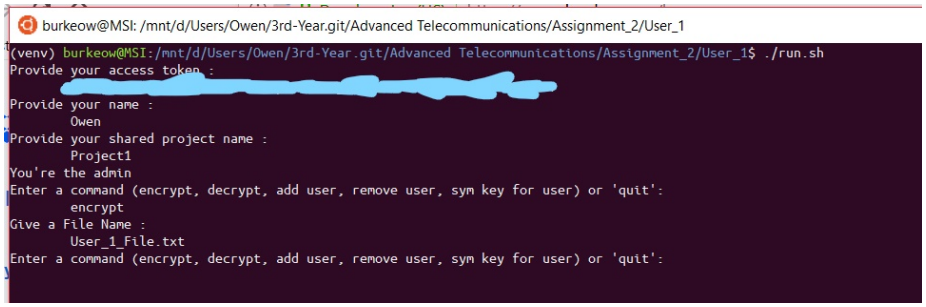


Figure 2: Encrypt a file

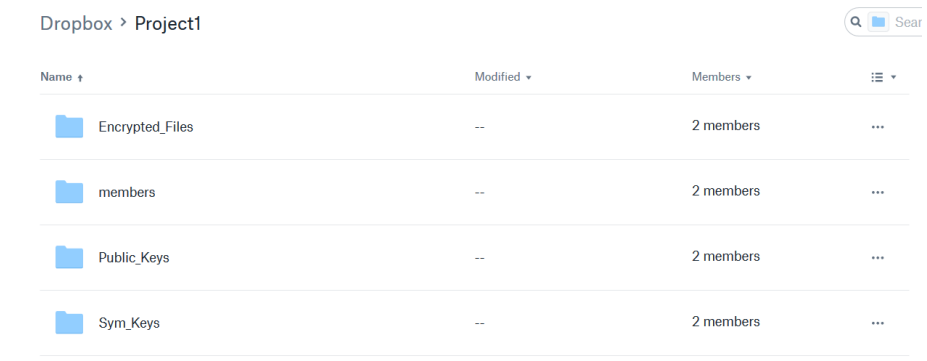


Figure 3: After an encryption

Figure 4: An encrypted file

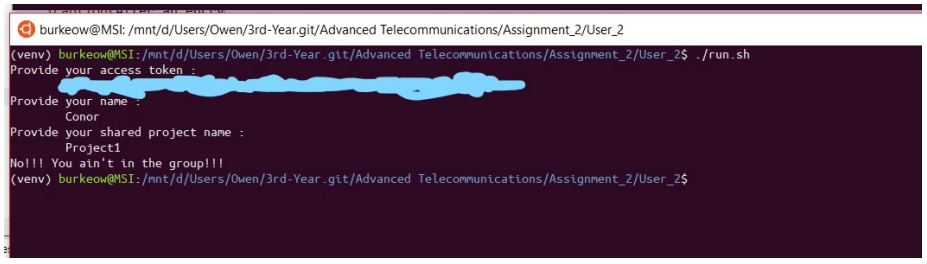


Figure 5: When a user is not a member

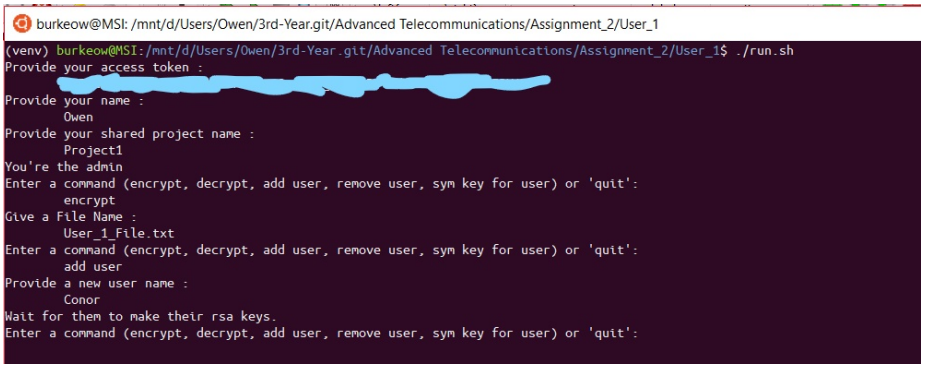


Figure 6: When a user is added

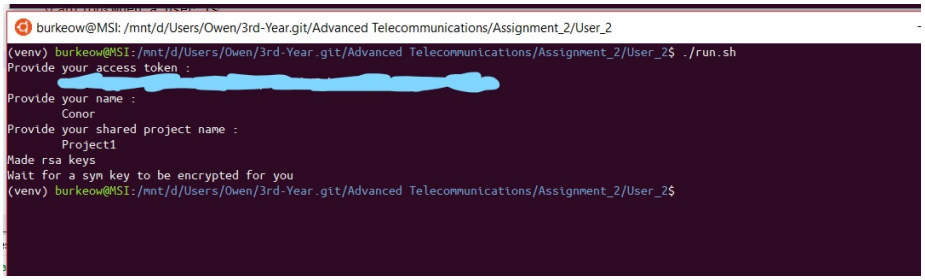


Figure 7: When a user has been added, they make their RSA keys

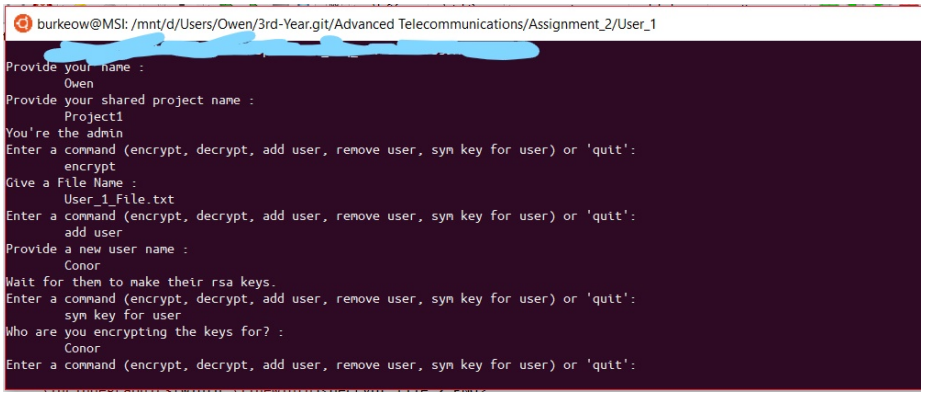


Figure 8: Encrypting a symmetric key for a new user

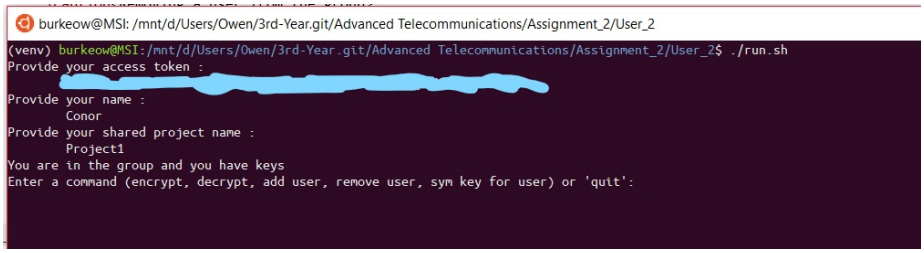


Figure 9: A new user has been added and their keys made

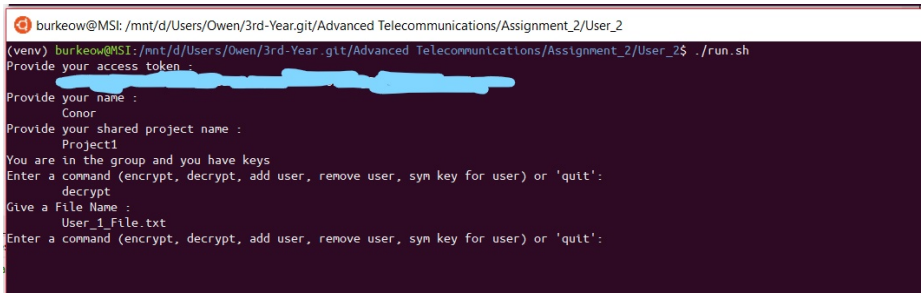


Figure 10: Decrypting a file from another user

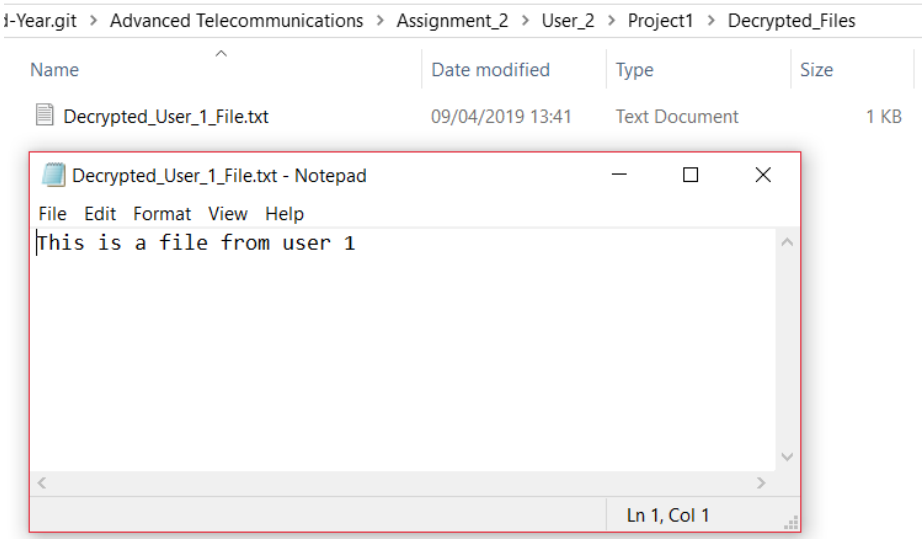
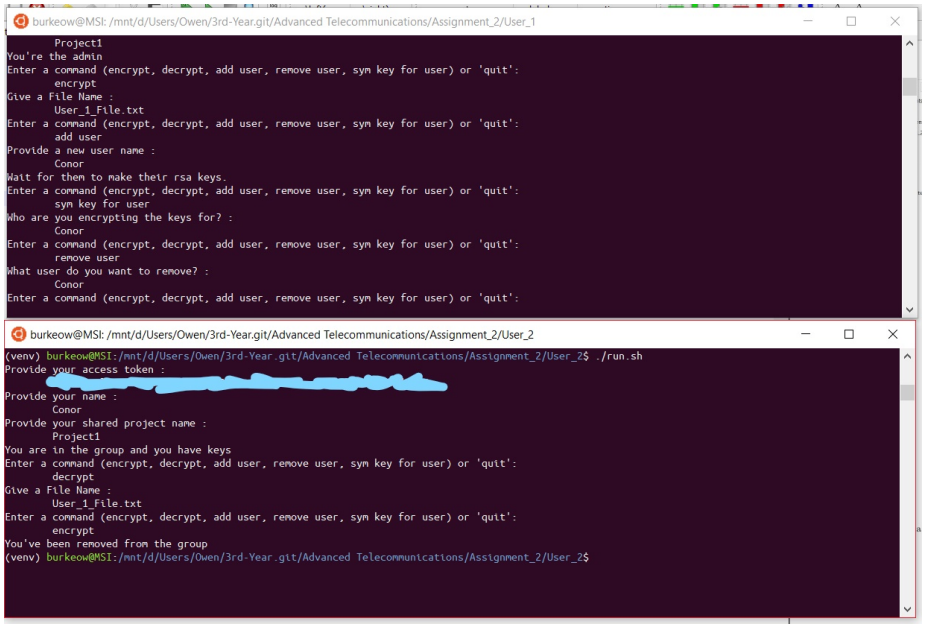


Figure 11: The decrypted file from another user



a user.jpg

Figure 12: Removing a user from the group