

## 1. Create Policy

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	<a href="#">22975276-cloudstorage</a>	Asia Pacific (Sydney) ap-southeast-2	Error	August 31, 2022, 18:32:50 (UTC+08:00)
<input type="radio"/>	<a href="#">22975276-cloudstorage2</a>	Asia Pacific (Sydney) ap-southeast-2	Error	August 31, 2022, 18:57:12 (UTC+08:00)
<input type="radio"/>	<a href="#">22975276-cloudstorage3</a>	Asia Pacific (Sydney) ap-southeast-2	<u>Objects can be public</u>	August 31, 2022, 19:01:20 (UTC+08:00)

To test if my policy was working I applied the restrictions to my own account by using **string like** in the condition which lead to me being locked out. Somehow I accidentally made the same mistake again with my second attempt meaning I had to create a third bucket with the policy applied. Because it had locked me out in the past I knew it worked.

The final policy I used can be seen below and the code I used to apply it is also below, however the code exists within the file `cloudstorageencrypt.py` - the entire python file exists later in the document.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::22975276-cloudstorage3",
        "arn:aws:s3:::22975276-cloudstorage3/*"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:username": "22975276@student.uwa.edu.au"
        }
      }
    }
  ]
}
```

```
65 # Create bucket
66 bucket_policy = {
67     "Version": "2012-10-17",
68     "Statement": [
69         {
70             "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
71             "Effect": "Deny",
72             "Principal": "*",
73             "Action": "s3:*",
74             "Resource": [
75                 "arn:aws:s3:::22975276-cloudstorage3",
76                 "arn:aws:s3:::22975276-cloudstorage3/*"
77             ],
78             "Condition": {
79                 "StringNotLike": {
80                     "aws:username": "22975276@student.uwa.edu.au"
81                 }
82             }
83         }
84     ]
85 }
86 bucket_policy = json.dumps(bucket_policy)
87 try:
88     s3_client = boto3.client('s3', region_name=region)
89     location = {'LocationConstraint': region}
90     s3_client.create_bucket(Bucket=bucket_name, CreateBucketConfiguration=location)
91     s3_client.put_bucket_policy(Bucket=bucket_name, Policy = bucket_policy)
92 except ClientError as e:
93     logging.error(e)
94     return False
95 return True
```

## 2. AES encryption using kms

First I created my kms key using the following python code.

```
1 import json
2 import logging
3 from datetime import date, datetime
4
5 import boto3
6 from botocore.exceptions import ClientError
7
8 AWS_REGION = 'ap-southeast-2'
9
10 # logger config
11 logger = logging.getLogger()
12 logging.basicConfig(level=logging.INFO,
13                     format='%(asctime)s: %(levelname)s: %(message)s')
14
15 kms_client = boto3.client("kms", region_name=AWS_REGION)
16
17
18 def json_datetime_serializer(obj):
19     """
20     Helper method to serialize datetime fields
21     """
22     if isinstance(obj, (datetime, date)):
23         return obj.isoformat()
24     raise TypeError("Type %s not serializable" % type(obj))
25
26
27 def create_kms_key():
28     """
29     Creates a unique customer managed KMS key.
30     """
31     try:
32         response = kms_client.create_key(Description='22975276-key2',
33                                         Tags=[{
34                                             'TagKey': 'Name',
35                                             'TagValue': '22975276-key2'
36                                         }])
37
38     except ClientError:
39         logger.exception('Could not create a CMK key.')
40         raise
41     else:
42         return response
43
44
45 if __name__ == '__main__':
46     # Constants
47     logger.info('Creating a symetric CMK...')
48     kms = create_kms_key()
49     logger.info(
50         f'Symetric CMK is created with details: {json.dumps(kms, indent=4, default=json_datetime_serializer)}'
51     )
```

After running this file the output was the following.

```
lachlan@lachlan-VirtualBox:~/Cloud$ python3 create_kms.py
2022-09-06 22:45:46,507: INFO: Found credentials in shared cre
dentials file: ~/.aws/credentials
2022-09-06 22:45:46,613: INFO: Creating a symetric CMK...
2022-09-06 22:45:47,588: INFO: Symetric CMK is created with de
tails: {
  "KeyMetadata": {
    "AWSAccountId": "523265914192",
    "KeyId": "81fe7ff3-4821-47fe-a0e9-aef6065b97af",
    "Arn": "arn:aws:kms:ap-southeast-2:523265914192:key/81
fe7ff3-4821-47fe-a0e9-aef6065b97af",
```

Then I applied the policy using the following code

```
new_kms_policy.py
~/Cloud

1 import boto3
2 import boto3
3 import argparse
4 parser = argparse.ArgumentParser(description = 'Initialise to True')
5 parser.add_argument('-i',action="store_true", help="-i to create bucket")
6 args = parser.parse_args()
7 # -----
8 # CIT5503
9 #
10 # cloudstorage.py
11 #
12 # skeleton application to copy local files to S3
13 #
72 kms_client.put_key_policy(KeyId="81fe7ff3-4821-47fe-a0e9-aef6065b97af",
73                             PolicyName='default',
74                             Policy=policy)
75 pol = kms_client.get_key_policy(KeyId="81fe7ff3-4821-47fe-a0e9-aef6065b97af",
76                                 PolicyName='default')
77 print(pol)
36 policy = {
37     "Sid": "Enable IAM User Permissions",
38     "Effect": "Allow",
39     "Principal": {
40         "AWS": "arn:aws:iam::523265914192:root"
41     },
42     "Action": "kms:*",
43     "Resource": "*"
44 },
45 {
46     "Sid": "Allow access for Key Administrators",
47     "Effect": "Allow",
48     "Principal": {
49         "AWS": "arn:aws:iam::523265914192:user/22975276@student.uwa.edu.au"
50     },
51     "Action": [
52         "kms:Create*",
53         "kms:Describe*",
54         "kms:Enable*",
55         "kms:List*",
56         "kms:Put*",
57         "kms:Update*",
58         "kms:Revoke*",
59         "kms:Disable*",
60         "kms:Get*",
61         "kms:Delete*",
62         "kms:TagResource",
63         "kms:UntagResource",
64         "kms:ScheduleKeyDeletion",
65         "kms:CancelKeyDeletion"
66     ],
67     "Resource": "*"
68 }
69 ]
70 }'''
71
72 kms_client.put_key_policy(KeyId="81fe7ff3-4821-47fe-a0e9-aef6065b97af",

lachlan@lachlan-VirtualBox:~/Cloud$ python3 new_kms_policy.py
{'Policy': {'Version': '2012-10-17', 'Id': 'key-consolepolicy-3', 'Statement': [{'Sid': 'Enable IAM User Permissions', 'Effect': 'Allow', 'Principal': {'AWS': 'arn:aws:iam::523265914192:root'}, 'Action': 'kms:*', 'Resource': '*'}, {'Sid': 'Allow access for Key Administrators', 'Effect': 'Allow', 'Principal': {'AWS': 'arn:aws:iam::523265914192:user/22975276@student.uwa.edu.au'}, 'Action': ['kms:Create*', 'kms:Describe*', 'kms:Enable*', 'kms:List*', 'kms:Put*', 'kms:Update*', 'kms:Revoke*', 'kms:Disable*', 'kms:Get*', 'kms:Delete*', 'kms:TagResource', 'kms:UntagResource', 'kms:ScheduleKeyDeletion', 'kms:CancelKeyDeletion'], 'Resource': '*'}]}}
```

From here I generated a data key, encrypted my file and uploaded it to my s3 bucket. Then I downloaded it generated the data key, decrypted it and looked at the file contents. The code for both encrypting/uploading and decrypting/downloading are at the end. Also the code contains a lot of unknown library errors underlined in yellow, this is because I opened the files on my own machine which doesn't contain all the libraries, this was so I could take bigger screenshots.

The test text file for the following encryption and decryption.

```
test.txt
~/lab3/rootdir

1 Hello World!
```

The output from encrypting and decrypting, the final text appears to show that the process was successful

```
lachlan@lachlan-VirtualBox:~/Cloud$ python3 cloudstorageencrypt.py -i
The bucket exists
Uploading /home/lachlan/lab3/rootdir/test.txt

Test print statements (irrelevant output)

b'\x01\x02\x02\x00\xcf5\xbf\xdfTP\x0e\xe9\x15\xe4\xe1*\x8c\x8bR8\xd2t\xc7\xcf
3\xc159\xf6DB}\x1e:~\xbda\xe9\x01\x1eNQ\xc7\x8f\xb9\xa8.\x1d\xc6o\x1d\x81]\x
02\xea\x00\x00\x00k0i\x06\t*\x86H\x86\xf7\r\x01\x07\x06\xa0\0Z\x02\x01\x000
U\x06\t*\x86H\x86\xf7\r\x01\x07\x010\x1e\x06\t`\x86H\x01e\x03\x04\x01.0\x11\
x04\x0c\xber\xba\x0f\x96|\xdbaF\xa1\xb2\xe7\x02\x01\x10\x80(\n\xbf#g\x07n"\x
0e\xba\xab\xcf4\xfcz\x18\x80\xa7\xd1\x8f\xea\x01F\x82\xf8Ft\x06\xe7\xec\xbb\x
01m\xabn3\xb2w\x17\xa5V\r' b'SGVsbG8gV29ybGQhCg=='

Accounts
lachlan@lachlan-VirtualBox:~/Cloud$ python3 restorefromclouddecrypt.py
Downloading from s3...
b'Hello World!\n'
```

Encryption settings in bucket set to the KMS.

**Server-side encryption settings**

Edit

Server-side encryption protects data at rest. [Learn more](#)

Default encryption

Enabled

Encryption key type

AWS Key Management Service key (SSE-KMS)

AWS KMS key ARN

arn:aws:kms:ap-southeast-2:523265914192:key/1f9d88b3-2b14-4336-95ab-b9579b1ed5d5

Bucket Key

When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. [Learn more](#)

Disabled

Contents of files appear as expected.

```
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test.txt
Hello World!
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test.txt.encrypted.decrypted
Hello World!
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test.txt.encrypted
0\0Z0U00`0He.00t000S90DB}:~0a0NQI00.000]0k0i    *0H00
    0r00|0aF0000(
```

## Cloudstorageencrypt.py

```
1 import os
2 import boto3
3 import json
4 import base64
5 import logging
6 import boto3
7 from botocore.exceptions import ClientError
8 import argparse
9 parser = argparse.ArgumentParser(description = 'Initialise to True')
10 parser.add_argument('-i',action="store_true", help="-i to create bucket")
11 args = parser.parse_args()
12 # -----
13 # CITSS503
14 #
15 # cloudstorage.py
16 #
17 # skeleton application to copy local files to S3
18 #
19 # Given a root local directory, will return files in each level and
20 # copy to same path on S3
21 #
22 # -----
23
24
25 ROOT_DIR = '/home/lachlan/lab3'
26 ROOT_S3_DIR = '22975276-cloudstorage3'
27
28
29 s3 = boto3.client("s3")
30
31 bucket_config = {'LocationConstraint': 'ap-southeast-2'}
32 kms_client = boto3.client("kms", region_name = 'ap-southeast-2')
33 data_key = kms_client.generate_data_key(KeyId = "81fe7ff3-4821-47fe-a0e9-aef6065b97af",
34 def upload_file(folder_name, file, file_name):
35     print("Uploading %s" % file)
36     s3_client = boto3.client('s3')
37     bucket = "22975276-cloudstorage3"
38     print(file, "lkenis")
39     with open(file,"rb") as f:
40         file_contents = f.read()
41         print(data_key['KeyId'])
42         fileencrypt = kms_client.encrypt(KeyId = data_key['KeyId'], Plaintext = file_co
43         with open(file,"wb") as f:
44             f.write(fileencrypt['CiphertextBlob'])
45             print(fileencrypt['CiphertextBlob'], base64.b64encode(file_contents))
46         os.rename(file,file+".encrypted")
47         file = file+".encrypted"
48         try:
49             response = s3_client.upload_file(file,bucket,file)
50         except ClientError as e:
51             logging.error(e)
52             return False
53
54 def create_bucket(bucket_name, region=None):
55     """Create an S3 bucket in a specified region
56
57     If a region is not specified, the bucket is created in the S3 default
58     region (us-east-1).
59
60     :param bucket_name: Bucket to create
61     :param region: String region to create bucket in, e.g., 'us-west-2'
62     :return: True if bucket created, else False
63     """
64
65     # Create bucket
66     bucket_policy = {
67         "Version": "2012-10-17",
68         "Statement": [
69             {
70                 "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
71                 "Effect": "Deny",
72                 "Principal": "*",
73                 "Action": "s3:*",
74                 "Resource": [
75                     "arn:aws:s3:::22975276-cloudstorage3",
76                     "arn:aws:s3:::22975276-cloudstorage3/*"
77             ],
78             "Condition": {
79                 "StringNotLike": {
80                     "aws:username": "22975276@student.uwa.edu.au"
81                 }
82             }
83         ]
84     }
85     bucket_policy = json.dumps(bucket_policy)
86     try:
87         s3_client = boto3.client('s3', region_name=region)
88         location = {'LocationConstraint': region}
89         s3_client.create_bucket(Bucket=bucket_name, CreateBucketConfiguration=locat
90         s3_client.put_bucket_policy(Bucket=bucket_name, Policy = bucket_policy)
91     except ClientError as e:
92         logging.error(e)
93         return False
94     return True
95
96 # Main program
97 # Insert code to create bucket if not there
98
99 s3 = boto3.resource('s3')
100 bucket = s3.Bucket('22975276-cloudstorage3')
101 if args.i:
102     if bucket.creation_date:
103         print("The bucket exists")
104     else:
105         print("The bucket does not exist")
106         create_bucket('22975276-cloudstorage3', 'ap-southeast-2')
107
108 for dir_name, subdir_list, file_list in os.walk(ROOT_DIR, topdown=True):
109     if dir_name != ROOT_DIR:
110         for fname in file_list:
111             upload_file("%s/" % dir_name[2:], "%s/%s" % (dir_name, fname), fname)
112
113
114 print("done")
```

## restorefromclouddecrypt.py

```
1 import boto3
2 from boto3.session import Session
3 import base64
4 import os
5 import fernet
6 ACCESS_KEY_ID = 'AKIAXTVIUGVIMEGAMQ5A'
7 SECRET_KEY = 'B1Aa1pVMCe14pihaVqJTD5B+H97ou+RswSyb3k'
8
9 session = Session(aws_access_key_id=ACCESS_KEY_ID,
10 aws_secret_access_key=SECRET_KEY)
11
12 s3 = session.resource('s3')
13
14 bucket = '22975276-cloudstorage3'
15
16 my_bucket = s3.Bucket(bucket)
17 kms_client = boto3.client("kms", region_name = 'ap-southeast-2')
18 data_key = kms_client.generate_data_key(KeyId = "81fe7ff3-4821-47fe-a0e9-aef6065b97af", NumberOfBytes = 123)
19 print("Downloading from s3...")
20 filename = ""
21 for s3_files in my_bucket.objects.all():
22     #print(s3_files.key)
23     filename = s3_files.key
24     os.rename(filename, filename + ".decrypted")
25     filename = filename + ("decrypted")
26     #print(filename)
27     with open(filename,"rb") as f:
28         file_contents = f.read()
29         #print(data_key['KeyId'])
30         filedecrypt = kms_client.decrypt(KeyId = data_key['KeyId'], CiphertextBlob = file_contents)
31         #print(filedecrypt)
32         with open(filename,"wb") as f:
33             f.write(filedecrypt['Plaintext'])
34             print(filedecrypt['Plaintext'])
35
36 #initiate s3 resource
37 s3 = boto3.resource('s3')
38 # download file into current directory
39 for s3_object in my_bucket.objects.all():
40     filename = s3_object.key
41     my_bucket.download_file(s3_object.key, filename)
```

### 3. Use pycryptodome

Since the code for encryption and decryption was provided, I simply altered the previous py files to use the provided encrypt and decrypt functions instead of the data key to encrypt the files. (Once again code at the end).

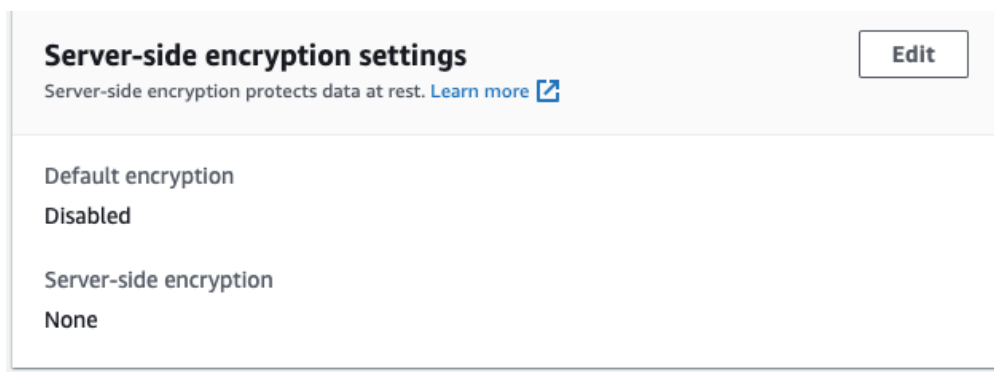
Following screenshot is of running both functions (uploading and downloading files)

```
● lachlan@lachlan-VirtualBox:~/Cloud$ python3 cloudstorageencryptQ3.py -i
The bucket exists
Uploading /home/lachlan/lab3/rootdir/test2.txt.enc
done
● lachlan@lachlan-VirtualBox:~/Cloud$ python3 restorefromclouddecryptQ3.py
Downloading from s3...
```

Exploring file contents, as you can see the encryption and decryption process worked successfully.

```
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ ls
subdir test2_dec.txt test2.txt test2.txt.enc
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test2.txt
Hello World!
lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test2.txt.enc
r0cr*)x0;0000y}E0`00f000L000lachlan@lachlan-VirtualBox:~/lab3/rootdir$ cat test
2_dec.txt
Hello World!
```

The encryption settings of the file in the bucket.





## CloudstorageencryptQ3.py

```
Users > lachlanbasi > Downloads > cloudstorageencryptQ3.py > ...
1 import os
2 import boto3
3 import json
4 import base64
5 import logging
6 from botocore.exceptions import ClientError
7 import argparse
8 parser = argparse.ArgumentParser(description = 'Initialise to True')
9 parser.add_argument('-i', action='store_true', help='-i to create bucket')
10 args = parser.parse_args()
11 import os, random, struct
12 from Crypto.Cipher import AES
13 from Crypto import Random
14 import hashlib
15 password = 'kitty and the kat'
16
17 BLOCK_SIZE = 16
18 CHUNK_SIZE = 64 * 1024
19 def encrypt_file(password, in_filename, out_filename):
20
21     key = hashlib.sha256(password.encode("utf-8")).digest()
22
23     iv = Random.new().read(AES.block_size)
24     encryptor = AES.new(key, AES.MODE_CBC, iv)
25     filesize = os.path.getsize(in_filename)
26
27     with open(in_filename, 'rb') as infile:
28         with open(out_filename, 'wb') as outfile:
29             outfile.write(struct.pack('<Q', filesize))
30             outfile.write(iv)
31
32             while True:
33                 chunk = infile.read(CHUNK_SIZE)
34                 if len(chunk) == 0:
35                     break
36                 elif len(chunk) % 16 != 0:
37                     chunk += ' '.encode("utf-8") * (16 - len(chunk) % 16)
38
39                 outfile.write(encryptor.encrypt(chunk))
40
41
42 ROOT_DIR = '/home/lachlan/lab3'
43 ROOT_S3_DIR = '22975276-cloudstorage3'
44
45 s3 = boto3.client("s3")
46
47 bucket_config = {'LocationConstraint': 'ap-southeast-2'}
48 kms_client = boto3.client("kms", region_name = 'ap-southeast-2')
49 data_key = kms_client.generate_data_key(KeyId = "81fe7f3-4821-47fe-a0e9-ae6865b97af", NumberOfBytes = 123)
50 def upload_file(folder_name, file, file_name):
51     s3_client = boto3.client('s3')
52     bucket = "22975276-cloudstorage3"
53     encrypt_file(password, file, file+".enc")
54     file = file+".enc"
55     print("Uploading %s" % file)
56     try:
57         response = s3_client.upload_file(file, bucket, file)
58     except ClientError as e:
59         logging.error(e)
60         return False
61
62 def create_bucket(bucket_name, region=None):
63     """Create an S3 bucket in a specified region
64
65     If a region is not specified, the bucket is created in the S3 default
66     region (us-east-1).
67
68     :param bucket_name: Bucket to create
69     :param region: String region to create bucket in, e.g., 'us-west-2'
70     :return: True if bucket created, else False
71     """
72
73     # Create bucket
74     bucket_policy = {
75         "Version": "2012-10-17",
76         "Statement": [
77             {
78                 "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
79                 "Effect": "Deny",
80                 "Principal": "*",
81                 "Action": "s3:*",
82                 "Resource": [
83                     "arn:aws:s3:::22975276-cloudstorage3",
84                     "arn:aws:s3:::22975276-cloudstorage3/*"
85                 ],
86                 "Condition": {
87                     "StringNotLike": {
88                         "aws:username": "22975276@student.uwa.edu.au"
89                     }
90                 }
91             }
92         ]
93     }
94     bucket_policy = json.dumps(bucket_policy)
95     try:
96         s3_client = boto3.client('s3', region_name=region)
97         location = {'LocationConstraint': region}
98         s3_client.create_bucket(Bucket=bucket_name, CreateBucketConfiguration=location)
99         s3_client.put_bucket_policy(Bucket=bucket_name, Policy = bucket_policy)
100     except ClientError as e:
101         logging.error(e)
102         return False
103     return True
104
105 s3 = boto3.resource('s3')
106 bucket = s3.Bucket('22975276-cloudstorage3')
107 if args.i:
108     if bucket.creation_date:
109         print("The bucket exists")
110     else:
111         print("The bucket does not exist")
112         create_bucket('22975276-cloudstorage3', 'ap-southeast-2')
113 for dir_name, sub_dir_list, file_list in os.walk(ROOT_DIR, topdown=True):
114     if dir_name != ROOT_DIR:
115         for fname in file_list:
116             upload_file("%s/" % dir_name[2:], "%s/%s" % (dir_name, fname), fname)
117 print("done")
```

## restorefromclouddecryptQ3.py

```
1 import boto3
2 from boto3.session import Session
3 import base64
4 import os
5 import fernet
6 import os, random, struct
7 from Crypto.Cipher import AES
8 from Crypto import Random
9 import boto3
10 import base64
11 import hashlib
12 BLOCK_SIZE = 16
13 CHUNK_SIZE = 64 * 1024
14 ACCESS_KEY_ID = 'AKIAXTVIUGVIMEGAMQ5A'
15 SECRET_KEY = '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'
16 password = 'kitty and the kat'
17 session = Session(aws_access_key_id=ACCESS_KEY_ID,
18                   aws_secret_access_key=SECRET_KEY)
19
20 s3 = session.resource('s3')
21
22 bucket = '22975276-cloudstorage3'
23
24 my_bucket = s3.Bucket(bucket)
25 def decrypt_file(password, in_filename, out_filename):
26
27     key = hashlib.sha256(password.encode("utf-8")).digest()
28
29     with open(in_filename, 'rb') as infile:
30         origsize = struct.unpack('<Q', infile.read(struct.calcsize('Q')))[0]
31         iv = infile.read(16)
32         decryptor = AES.new(key, AES.MODE_CBC, iv)
33
34         with open(out_filename, 'wb') as outfile:
35             while True:
36                 chunk = infile.read(CHUNK_SIZE)
37                 if len(chunk) == 0:
38                     break
39                 outfile.write(decryptor.decrypt(chunk))
40
41             outfile.truncate(origsize)
42
43
44 print("Downloading from s3...")
45 filename = ""
46 for s3_files in my_bucket.objects.all():
47     #print(s3_files.key)
48     filename = s3_files.key
49     decrypt_file(password, filename, "/home/lachlan/lab3/rootdir/test2_dec.txt")
50
51 #initiate s3 resource
52 s3 = boto3.resource('s3')
53 # download file into current directory
54 for s3_object in my_bucket.objects.all():
55     filename = s3_object.key
56     my_bucket.download_file(s3_object.key, filename)
57
```