Lab 4 Notes

Student ID: 22994257
Name: Gaoyuan Zhang

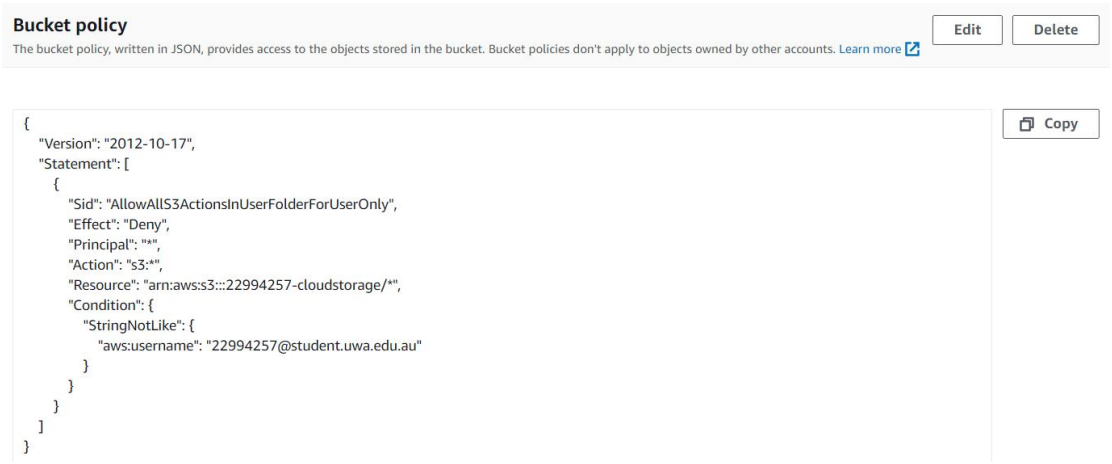
[Step 1] Apply policy to restrict permissions on bucket

1. Create step1.py and use the function put_bucket_policy to set the new policy

```python
import json
import boto3

# Create a bucket policy
bucket_name = '22994257-cloudstorage'
bucket_policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::22994257-cloudstorage/*",
            "Condition": {
                "StringNotLike": {
                    "aws:username": "22994257@student.uwa.edu.au"
                }
            }
        }
    ]
}

# Convert the policy from JSON dict to string
bucket_policy = json.dumps(bucket_policy)

# Set the new policy
s3 = boto3.client('s3')
s3.put_bucket_policy(Bucket=bucket_name, Policy=bucket_policy)
result = s3.get_bucket_policy(Bucket=bucket_name)
print(result['Policy'])
```

```
mjieli@mjieli-VirtualBox:~/lab4$ python3 step1.py
{"Version":"2012-10-17","Statement":[{"Sid":"AllowAllS3ActionsInUserFolderForUserOnly","Eff
ect":"Deny","Principal":"*","Action":"s3:*","Resource":"arn:aws:s3:::22994257-cloudstorage/
*","Condition":{"StringNotLike":{"aws:username":"22994257@student.uwa.edu.au"}}}]}
```

2. We can see the new policy in the S3 bucket permission page.



[Step 2] AES Encryption using KMS

1. Create a new program called create_key.py. Use create_key() and create_alias() functions to create a key and add the alias.

```
import boto3

kms_client = boto3.client('kms')
# Create CMK
response = kms_client.create_key(Description='22994257 key')
key_id = response['KeyMetadata']['KeyId']
key_arn = response['KeyMetadata']['Arn']

# Print the key ID and ARN
print('Key id is: ', key_id)
print('Key ARN is:', key_arn)

# Create the alias
response = kms_client.create_alias(
    AliasName='alias/22994257',
    TargetKeyId=key_id
)
print(response)
```

```
mjieli@mjieli-VirtualBox:~/lab4$ python3 create_key.py
Key id is:  31c3af87-25d8-4798-b543-b014cc29018f
Key ARN is: arn:aws:kms:ap-southeast-2:523265914192:key/31c3af87-25d8-4798-b543-b014cc29018
f
{'ResponseMetadata': {'RequestId': '6d33874c-6237-4b7d-9721-c1bef20676c9', 'HTTPStatusCode'
: 200, 'HTTPHeaders': {'x-amzn-requestid': '6d33874c-6237-4b7d-9721-c1bef20676c9', 'cache-c
ontrol': 'no-cache, no-store, must-revalidate, private', 'expires': '0', 'pragma': 'no-cach
e', 'date': 'Tue, 30 Aug 2022 02:36:12 GMT', 'content-type': 'application/x-amz-json-1.1',
'content-length': '0'}, 'RetryAttempts': 0}}
```

Key id: 31c3af87-25d8-4798-b543-b014cc29018f

ARN: arn:aws:kms:ap-southeast-2:523265914192:key/31c3af87-25d8-4798-b543-b014cc29018f

2. Create a new program called new_KMS_policy.py. Use put_key_policy() to apply the policy and get_key_policy() to get the result.

```
import json
import boto3

key_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/22994257@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": "*"
```

```
# Convert the policy from JSON dict to string
key_policy = json.dumps(key_policy)

# Set the new policy
client = boto3.client('kms')
response = client.put_key_policy(
    KeyId='31c3af87-25d8-4798-b543-b014cc29018f',
    PolicyName='default',
    Policy=key_policy
)

response = client.get_key_policy(
    KeyId='31c3af87-25d8-4798-b543-b014cc29018f',
    PolicyName='default',
)
print(response)
```

```
mjieli@mjieli-VirtualBox:~/lab4$ vim new_KMS_policy.py
mjieli@mjieli-VirtualBox:~/lab4$ python3 new_KMS_policy.py
{'Policy': '{\n  "Version" : "2012-10-17",\n  "Id" : "key-consolepolicy-3",\n  "Statement"
: [ {\n    "Sid" : "Enable IAM User Permissions",\n    "Effect" : "Allow",\n    "Principal"
 : {\n      "AWS" : "arn:aws:iam::523265914192:root"\n    },\n    "Action" : "kms:*",\n
"Resource" : "*"\n  }, {\n    "Sid" : "Allow access for Key Administrators",\n    "Effect"
: "Allow",\n    "Principal" : {\n      "AWS" : "arn:aws:iam::523265914192:user/22994257@stu
dent.uwa.edu.au"\n    },\n    "Action" : [ "kms:Create*", "kms:Describe*", "kms:Enable*", "
kms:List*", "kms:Put*", "kms:Update*", "kms:Revoke*", "kms:Disable*", "kms:Get*", "kms:Dele
te*", "kms:TagResource", "kms:UntagResource", "kms:ScheduleKeyDeletion", "kms:CancelKeyDele
tion" ],\n    "Resource" : "*"\n  }, {\n    "Sid" : "Allow use of the key",\n    "Effect" :
 "Allow",\n    "Principal" : {\n      "AWS" : "arn:aws:iam::523265914192:user/22994257@stud
ent.uwa.edu.au"\n    },\n    "Action" : [ "kms:Encrypt", "kms:Decrypt", "kms:ReEncrypt*", "
kms:GenerateDataKey*", "kms:DescribeKey" ],\n    "Resource" : "*"\n  }, {\n    "Sid" : "All
ow attachment of persistent resources",\n    "Effect" : "Allow",\n    "Principal" : {\n
  "AWS" : "arn:aws:iam::523265914192:user/22994257@student.uwa.edu.au"\n    },\n    "Action
" : [ "kms:CreateGrant", "kms:ListGrants", "kms:RevokeGrant" ],\n    "Resource" : "*",\n
 "Condition" : {\n      "Bool" : {\n        "kms:GrantIsForAWSResource" : "true"\n      }\n
    }\n  } ]\n}', 'ResponseMetadata': {'RequestId': '3f11f152-bbbc-43ba-b38d-bc0390fd0ab1',
 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '3f11f152-bbbc-43ba-b38d-bc0390
fd0ab1', 'cache-control': 'no-cache, no-store, must-revalidate, private', 'expires': '0', '
pragma': 'no-cache', 'date': 'Tue, 30 Aug 2022 03:08:48 GMT', 'content-type': 'application/
x-amz-json-1.1', 'content-length': '1616'}, 'RetryAttempts': 0}}
```

3. Generate a data key.

```python
def create_data_key(cmk_id, key_spec='AES_256'):
    # Create data key
    kms_client = boto3.client('kms')
    response = kms_client.generate_data_key(KeyId=cmk_id, KeySpec=key_spec)
    print(response['CiphertextBlob'])
    print(base64.b64encode(response['Plaintext']))

    return response['CiphertextBlob'], base64.b64encode(response['Plaintext'])
```

```
mjieli@mjieli-VirtualBox:~/lab4$ vim step2.py
mjieli@mjieli-VirtualBox:~/lab4$ python3 step2.py
b"\x01\x02\x03\x00x\x01\x00}\x7fCe&\x18AaQ\x11ZS%{[A\xbfT\x93m\xec\x88)\x90\xe6Q\xfe\xf9Z\x
a1\x01\x0f-\xf6`<lU[\xac#\xb2l\x87\x87W\x7f\x00\x00\x00~0|\x06\t*\x86H\x86\xf7\r\x01\x07\x0
6\xa0o0m\x02\x01\x000h\x06\t*\x86H\x86\xf7\r\x01\x07\x010\x1e\x06\t`\x86H\x01e\x03\x04\x01.
0\x11\x04\x0cQ\x0b\xaeo\x8c\x9a\x1a\x80Y\t\xe5\xb1\x02\x01\x10\x80;|9$\xbd\xcd\x11%'\xc7\x8
b\x82\x1d\x95~S\x90\xc5\x0f53\xbeXuP9%\xd9\xd9\xf3D\x05\x91O\xbd6\xcb\xff\xe5F\xa8%}\xffT-\
x84\xf8\xc0\xbb\x13\xeb?\xbd\xc9\xb69a1."
b'jxUM2QJzHW9QQUeLfqkro6/m+oj7UuR1Z7UwPKcHpx4='
```

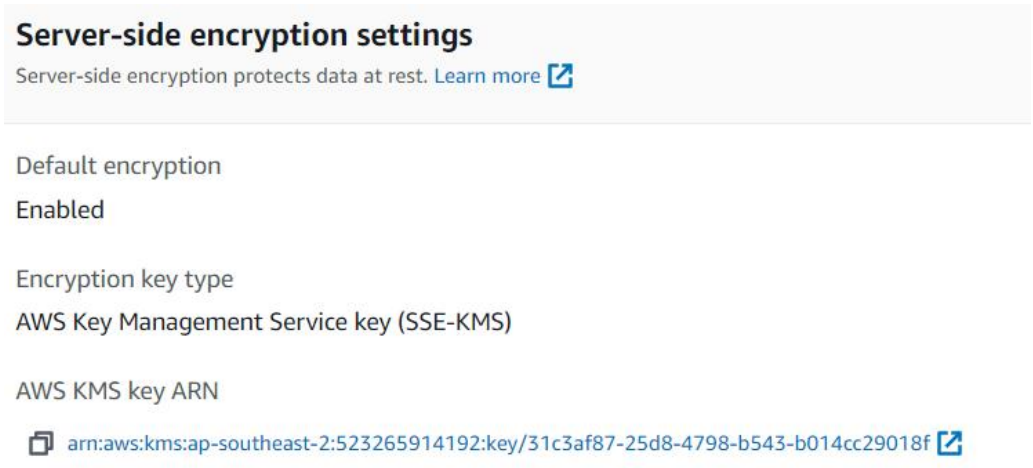
4. Use the generated data key to encrypt a local file ’kms.txt’.

```python
def encrypt_file(filename, cmk_id):
    # Read the entire file into memory
    try:
        with open(filename, 'rb') as file:
            file_contents = file.read()
    except IOError as e:
        logging.error(e)
        return False

    data_key_encrypted, data_key_plaintext = create_data_key(cmk_id)
    if data_key_encrypted is None:
        return False
    logging.info('Created new AWS KMS data key')

    # Encrypt the file
    f = Fernet(data_key_plaintext)
    file_contents_encrypted = f.encrypt(file_contents)

    try:
        with open(filename + '.encrypted', 'wb') as file_encrypted:
            file_encrypted.write(len(data_key_encrypted).to_bytes(NUM_BYTES_FOR_LEN,
                                                                  byteorder='big'))
            file_encrypted.write(data_key_encrypted)
            file_encrypted.write(file_contents_encrypted)
    except IOError as e:
        logging.error(e)
        return False
    return True
```

```
mjieli@mjieli-VirtualBox:~/lab4$ cd kms
mjieli@mjieli-VirtualBox:~/lab4/kms$ ls
kms.txt  kms.txt.encrypted
mjieli@mjieli-VirtualBox:~/lab4/kms$ cat kms.txt.encrypted
0o0m0hAa`◆He.0A◆T◆m◆)◆◆Q◆◆Z◆◆◆◆)◆DM◆◆g◆m◆~0|    *◆H◆◆
          ◆◆◆)B◆◆◆◆;◆r◆w◆◆◆G◆◆◆◆~€◆◆c6OG4x◆◆◆"◆c5\`◆Q◆◆G◆◆CwkV◆h◆◆◆◆0◆gAAAAABjDcDJF4Mj2
O3nZtBpzvF07iUs-pG4c4coYRAQMKeHOnVTd9nEolsWyjOA0kwFgQR-kLX_WbZOE5l8ZvlzVfYwljHH8Q==mjieli@m
```

5. Upload it to S3 bucket, then check it with AWS console.

```
s3 = boto3.client('s3')
s3.upload_file('kms/kms.txt.encrypted', '22994257-cloudstorage', './kms.txt.encrypted')
```



6. Download the encrypted file from cloud and decrypt it.

```
s3.download_file('22994257-cloudstorage', './kms.txt.encrypted', '/home/mjieli/lab4/kms.txt
.encrypted')
decrypt_file('kms.txt.encrypted')
```

```python
def decrypt_data_key(data_key_encrypted):
    kms_client = boto3.client('kms')
    try:
        response = kms_client.decrypt(CiphertextBlob=data_key_encrypted)
    except ClientError as e:
        logging.error(e)
        return None
    return base64.b64encode((response['Plaintext']))

def decrypt_file(filename):
    # Read the encrypted file into memory
    try:
        with open(filename, 'rb') as file:
            file_contents = file.read()
    except IOError as e:
        logging.error(e)
        return False

    data_key_encrypted_len = int.from_bytes(file_contents[:NUM_BYTES_FOR_LEN],
                                            byteorder='big') \
                             + NUM_BYTES_FOR_LEN
    data_key_encrypted = file_contents[NUM_BYTES_FOR_LEN:data_key_encrypted_len]

    # Decrypt the data key before using it
    data_key_plaintext = decrypt_data_key(data_key_encrypted)
    if data_key_plaintext is None:
        return False

    # Decrypt the rest of the file
    f = Fernet(data_key_plaintext)
    file_contents_decrypted = f.decrypt(file_contents[data_key_encrypted_len:])

    # Write the decrypted file contents
    try:
        with open(filename + '.decrypted', 'wb') as file_decrypted:
            file_decrypted.write(file_contents_decrypted)
    except IOError as e:
        logging.error(e)
        return False
    return True
```

```
mjieli@mjieli-VirtualBox:~/lab4/kms$ cat kms.txt.encrypted
0o0m0hAa`⋄He.0A⋄T⋄m⋄)⋄⋄Q⋄⋄Z⋄⋄⋄⋄)⋄DM⋄⋄g⋄m⋄~0|    *⋄H⋄⋄
          ⋄⋄⋄)B⋄⋄⋄⋄;⋄r⋄w⋄⋄⋄G⋄⋄⋄⋄~€⋄⋄c6OG4x⋄⋄⋄"⋄c5\`⋄Q⋄⋄G⋄⋄CwkV⋄h⋄⋄⋄⋄0⋄gAAAAABjDcDJF4Mj2
O3nZtBpzvF07iUs-pG4c4coYRAQMKeHOnVTd9nEolsWyjOA0kwFgQR-kLX_WbZOE5l8ZvlzVfYwljHH8Q==mjieli@m
jieli-VirtualBox:~/lab4/kms$ cd ..
mjieli@mjieli-VirtualBox:~/lab4$ cat kms.txt.encrypted.decrypted
Hi world!
```

[Step 3] AES Encryption using local python library pycryptodome

1. The 3 screenshots

```
mjieli@mjieli-VirtualBox:~/lab4$ cat afile1_dec.txt.enc
6⋄⋄U
    =⋄-⋄ƹM⋄⋄⋄T⋄⋄)S⋄\⋄xG⋄⋄.⋄_TnXY⋄w⋄Yq⋄Ğz2⋄I⋄lў̆M~⋄Y*|⋄⋄⋄|Уд⋄      n
```


```
mjieli@mjieli-VirtualBox:~/lab4$ cat afile1_dec.txt.enc
6◆◆U
    =◆-◆ƯM◆◆◆T◆◆)S◆\◆xG◆◆.◆_TnXY◆w◆Yq◆Ǧz2◆I◆lỹM~◆Y*|◆◆◆|Уд◆     nmjieli@mjieli-VirtualBox:~
/lab4$ cat afilenew_dec.txt
cloud computing
CITS5503
22994257
Gaoyuan Zhang
lab4
```

2. Question: What is the performance difference between using KMS and using the custom solution?

If I want to encrypt a file larger than 4KB using KMS, I should use a technique called "envelope encryption", which will encrypt the file in chunks. So, if I use the custom solution to do that, it’s faster.