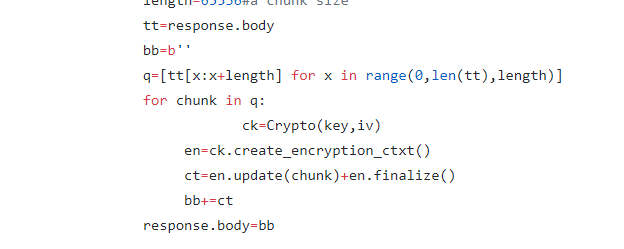
# Weekly goal

1.Fix the error when downloading large-sized object

2.Generate the encryption-key automatically while GET objects and store these keys in database.

## 1.Download process for large-sized objects

Now for GET method, response.body will be cut into chunks, and decrypted chunk by chunk(65536K).



## 2.Set up a database for key storage

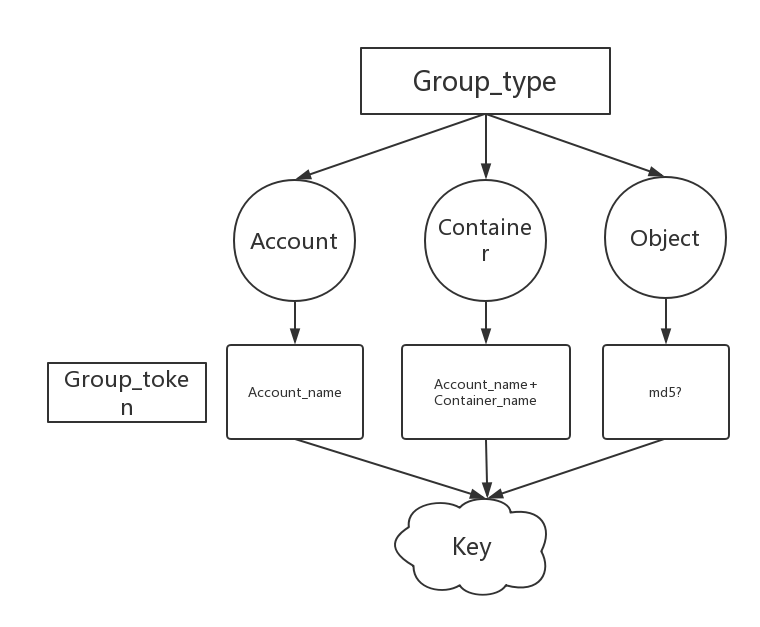
OpenStack use MySQL as a service for database. The information is stored in table ”keytable” in database “keyencry”.

Logic structure for “keytable”

Fields: group\_type (varchar 10 not null)

group\_token (varchar ? not null unique)

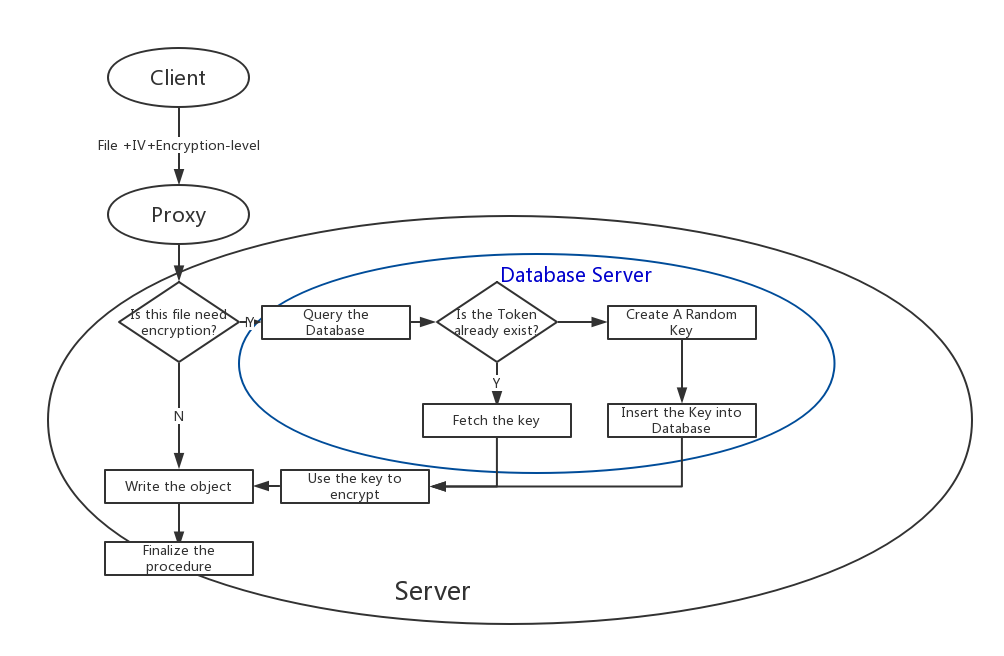
keytext (char 32 not null)



The encryption can be divided into three levels as the image shows above. For now, the default setting is encrypting all objects in one account by the same key(account-level). For next steps, this setting should be made on client-end and be kept in objects’ metadata. Account-level use account-name as token and container join account-name and container-name as its token.

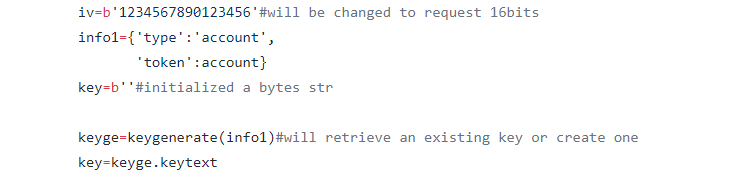
A question is how to process object-level encryption, because we might want to identify two files share the same content, maybe it’s good if we use its md5 as token?

A conclusion image shows the whole uploading process.

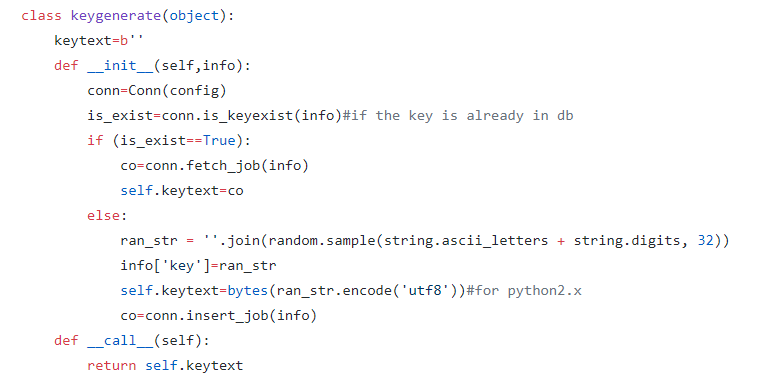


Source code:

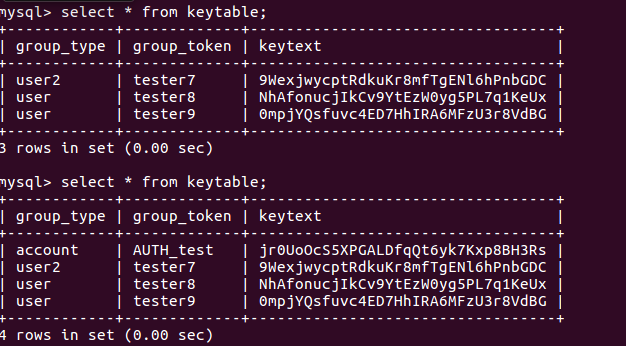
In server.py ,call class keygenrate to get a key(retrieve or create)



The class is defined in keyprocess.py



A screenshot token before and after a new file uploaded. The server generate a new key to encrypt the object and the key will be stored in database for use of decryption.



Next Steps:

1.Improve the process between proxy and client, such as letting the client choose which level the file should be encrypted.

2.Improve the process between proxy and server, such as letting the proxy generate the IV and send it to the server.

3.The database is set on the server-client and maybe we can use OpenStack Identify service in order to secure the database.