Encryption and Decryption Lab

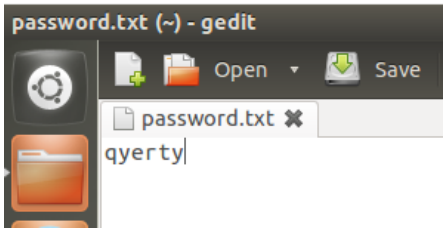
Lab Environment

We have created two accounts in the VM. The usernames and passwords are listed in the following:

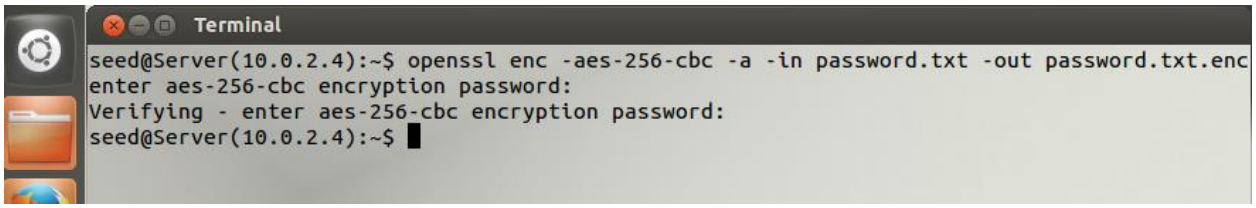
- User ID: root, Password: seedubuntu.
  - Note: Ubuntu does not allow root to login directly from the login window. You have to login as a normal user, and then use the command su to login to the root account.
- User ID: seed, Password: dees

Task 1: AES Encryption and Decryption

1. Create a file named password.txt
   1.1 Type: *gedit password.txt*

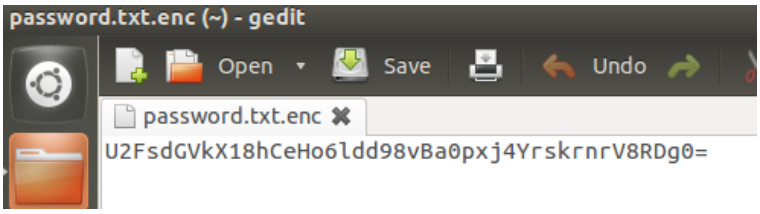


2. Encrypt with the encryption key *1234*
   2.1 Type following

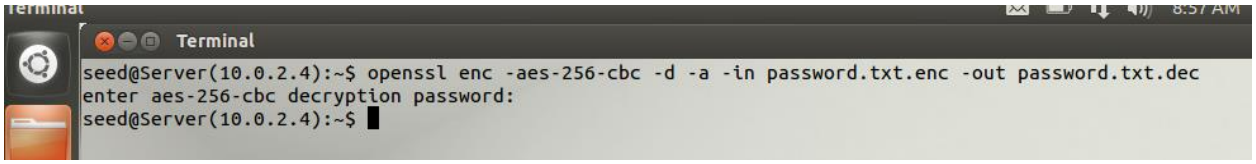


3. Questions:
   3.1 What does the option –a do? (base64-encode)
   3.2 What does the option –in do?
   3.3 What does the option –out do?
   3.4 What does the option –aes-246-cdc mean?

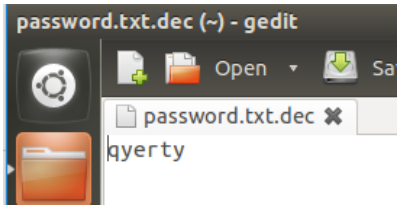
4. Show the encrypted the password



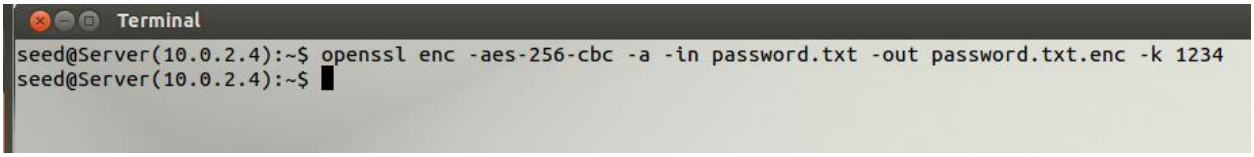
5. Decrypt with the encryption key *1234*



6. Show the decrypted the password



7. Encrypt with the option -k



8. Questions:
   - 8.1 What does the option –k mean?
   - 8.2 Decrypt the password using the option –k.

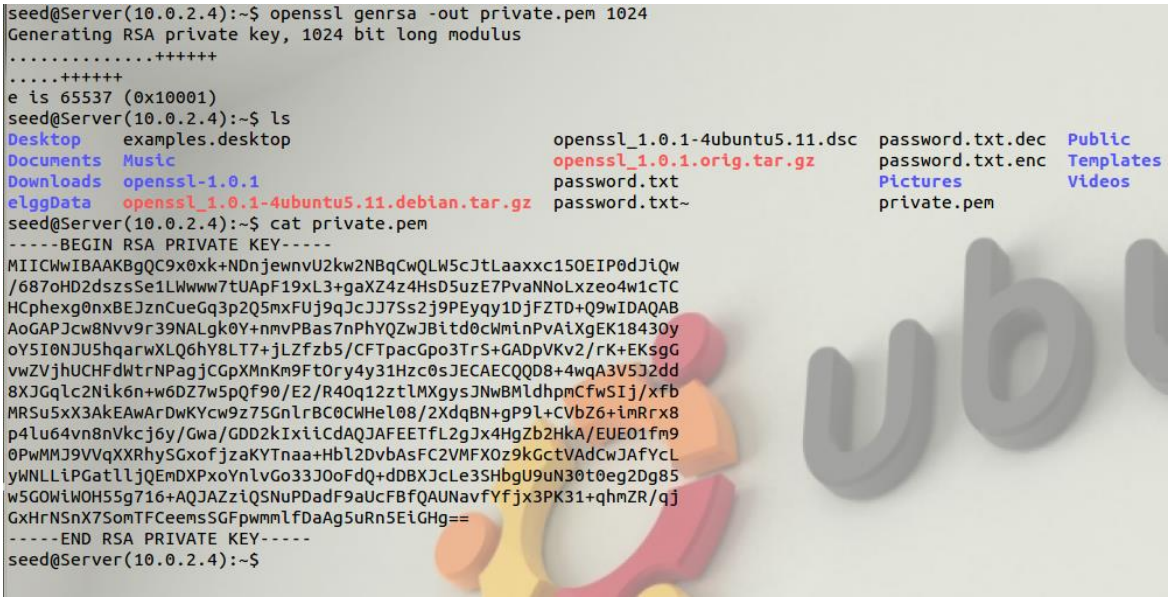
Task 2: RSA Encryption and Decryption (Public Key)

The following commands are relevant when you work with RSA keys:

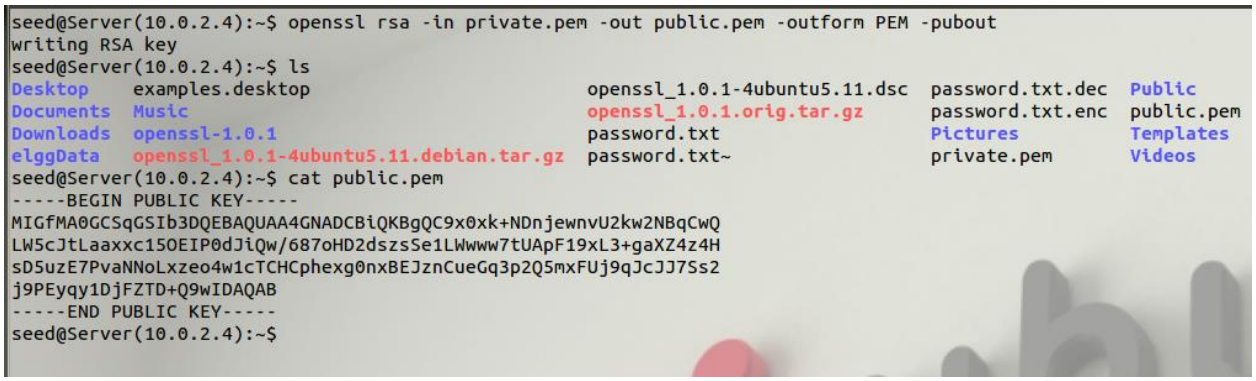
- openssl genrsa: Generates an RSA private keys.
- openssl rsa: Manage RSA private keys (includes generating a public key from it).
- openssl rsautl: Encrypt and decrypt files with RSA keys.

1. Generate the Public key.

Using OpenSSL on the command line you'd first need to generate a public and private key, you should password protect this file using the *-passout* argument. This creates a key file called ***private.pem*** that uses **1024** bits. This file actually have both the private and public keys



2. Extract the public one from this file.



3. You can freely share this with 3rd parties. You can test it all by just encrypting something yourself using your public key and then decrypting using your private key, first we need a bit of data to encrypt.

```
seed@Server(10.0.2.4):~$ echo 'mypassword' > fileRSA.txt
seed@Server(10.0.2.4):~$ ls
Desktop    examples.desktop  openssl_1.0.1-4ubuntu5.11.debian.tar.gz  password.txt~     private.pem  Videos
Documents  fileRSA.txt       openssl_1.0.1-4ubuntu5.11.dsc           password.txt.dec  Public
Downloads  Music             openssl_1.0.1.orig.tar.gz               password.txt.enc  public.pem
elggData   openssl-1.0.1     password.txt                            Pictures          Templates
seed@Server(10.0.2.4):~$ cat fileRSA.txt
mypassword
seed@Server(10.0.2.4):~$
```

4. You now have some data in *fileRSA.txt*, lets encrypt it using OpenSSL and the public key. Note that for showing the encrypted file, you need to decode it in base 64 (represent binary data in an ASCII string format by translating it into a radix-64 representation).
   - 4.1 **-inkey file:** the input key file, by default it should be an RSA private key.
   - 4.2 **–pubin:** the input file is an RSA public key.
   - 4.3 **-in filename:** This specifies the input filename to read data from or standard input if this option is not specified.

```
seed@Server(10.0.2.4):~$ openssl rsautl -encrypt -inkey public.pem -pubin -in fileRSA.txt -out file.ssl
seed@Server(10.0.2.4):~$ ls
Desktop    examples.desktop  openssl-1.0.1                           password.txt      Pictures     Templates
Documents  fileRSA.txt       openssl_1.0.1-4ubuntu5.11.debian.tar.gz password.txt~     private.pem  Videos
Downloads  file.ssl          openssl_1.0.1-4ubuntu5.11.dsc           password.txt.dec  Public
elggData   Music             openssl_1.0.1.orig.tar.gz               password.txt.enc  public.pem
seed@Server(10.0.2.4):~$ cat file.ssl | base64
a+DFmpBzJ4rN2IoKD9gQdO/HYb8gXVsTRdmF0KOrNUPA76ey0ZUppUFc3CikNqmK1IJXxclCmPCR
NF+VQ4jO968RQYtybTjquWGZ+9fGYqbBXvTOxjGRuN5NDjfmT/IXKjg3GP+92Iq5kmpVcD4XwliX
jQWPq7ieTHT11vrB5I4=
seed@Server(10.0.2.4):~$
```

ASCII - Binary Character Table

Letter	ASCII Code	Binary
B	066	01000010
C	067	01000011
D	068	01000100
E	069	01000101

5. This creates an encrypted version of *fileRSA.txt* calling it *file.ssl*, if you look at this file it's just binary junk, nothing very useful to anyone. Now you can unencrypt it using the private key.

```
seed@Server(10.0.2.4):~$ openssl rsautl -decrypt -inkey private.pem  -in file.ssl -out fileRSA.txt.dec
seed@Server(10.0.2.4):~$ ls
Desktop           fileRSA.txt      openssl_1.0.1-4ubuntu5.11.debian.tar.gz  password.txt.dec  public.pem
Documents         fileRSA.txt.dec  openssl_1.0.1-4ubuntu5.11.dsc            password.txt.enc  Templates
Downloads         file.ssl         openssl_1.0.1.orig.tar.gz                Pictures          Videos
elggData          Music            password.txt                             private.pem
examples.desktop  openssl-1.0.1    password.txt~                            Public
seed@Server(10.0.2.4):~$ cat fileRSA.txt.dec
mypassword
seed@Server(10.0.2.4):~$ cat fileRSA.txt
mypassword
seed@Server(10.0.2.4):~$
```

Reference:

- https://www.devco.net/archives/2006/02/13/public__private_key_encryption_using_openssl.php
- http://www.sis.pitt.edu/lersais/education/labs/access_control.php
- https://www.madboa.com/geek/opensslection-13537064