Daire O`Neill

[Company name]  [Company address]

Computer security assignment 1

Table of Contents:

Symmetric Encryption pg 2-11

Asymmetic Encryption pg 12-17

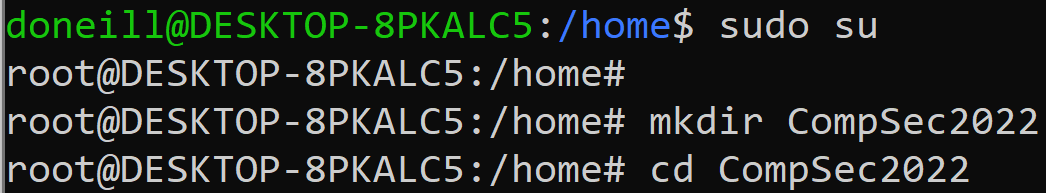
**What is Symmetric Encryption?**

Symmetric encryption is a type of encryption where only one key is used to both encrypt and decrypt `pieces of information. The people involved in the encryption and decryption must exchange the key to decrypt the information. The data is encrypted and is nonsense to those who don’t have the key.

**Steps for Symmetric Encryption**

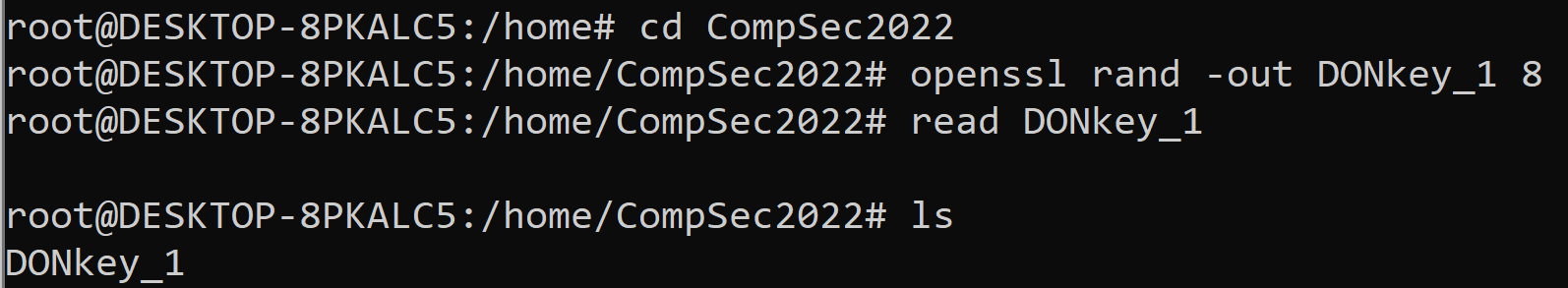
1.

I made myslef a super user, then proceeded to make the directory CompSec2022 and went into the directory created a private random key called DONkey\_1 of 8 bits, then made sure it was in the directory.

****

2a.

I created a private random key called DONkey\_1 of 8 bits, then made sure it was in the directory.

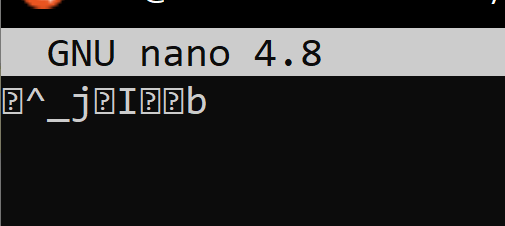


I then made 2 more key but changed the number of bits to 16 and 32.

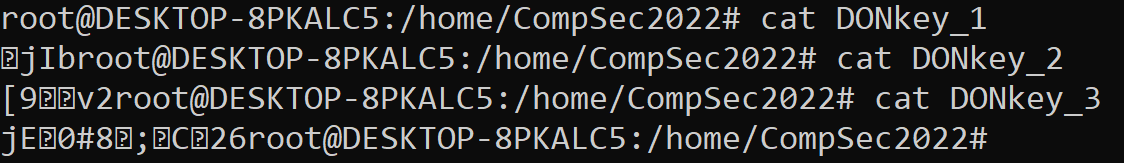
Text

Description automatically generated

I went into DONkey\_1 and found this:



The other two keys were also incomprehensible but longer.



2b.

I created and went into a text file called DONtest\_1 and wrote in some text. (I will encrypt these later)

Text

Description automatically generated

I created to other text files and made sure they were all there.

Text

Description automatically generated

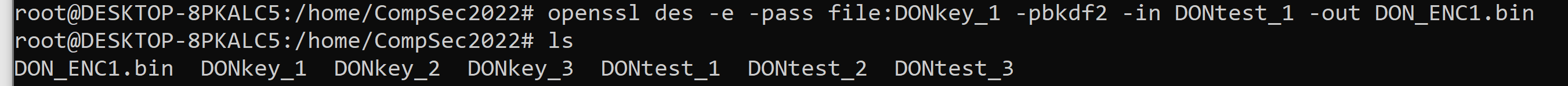
This is what I wrote in the other two files.

Text

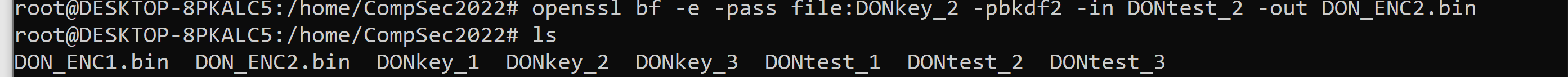
Description automatically generated

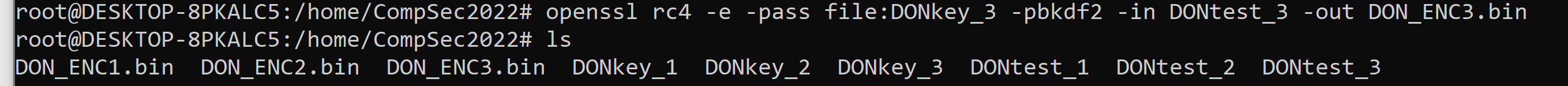
3.

Here I encrypted DONkey\_1 using des and turned it into DON\_ENC1.bin.

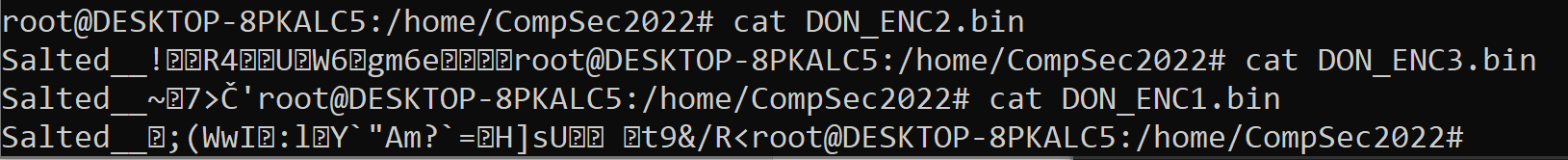


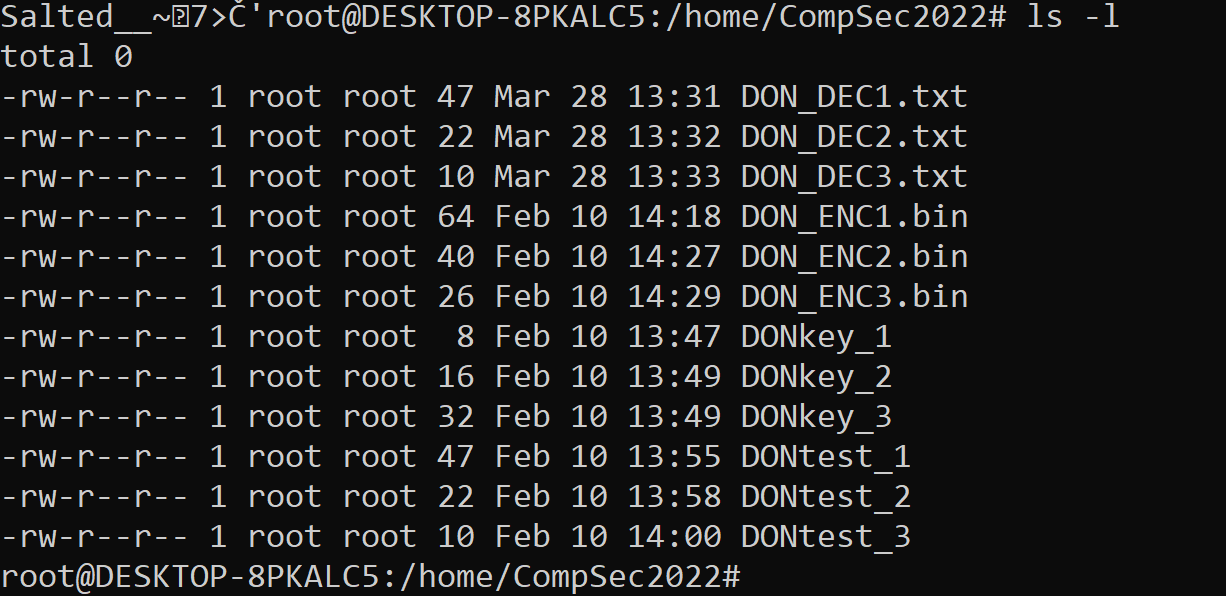
I encrypted my second test file with bf.



And my third file with rc4.

Here is what the encrypted files now produce.

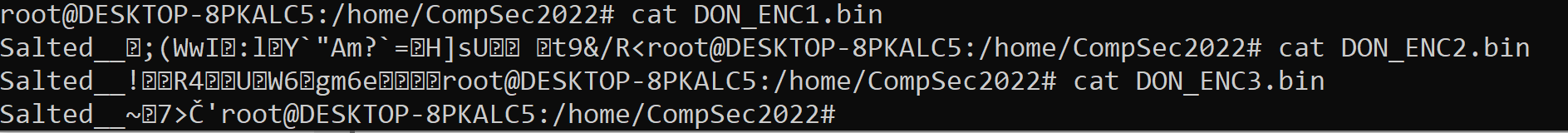


I then used ls-l. This compares the files.

4.

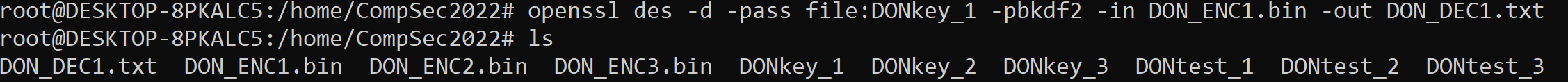
Unfortunately, due to illness I was unable to send or receive file to a lab partner so I will be decrypting my own files.

Here are the files encrypted that would have been sent by my partner

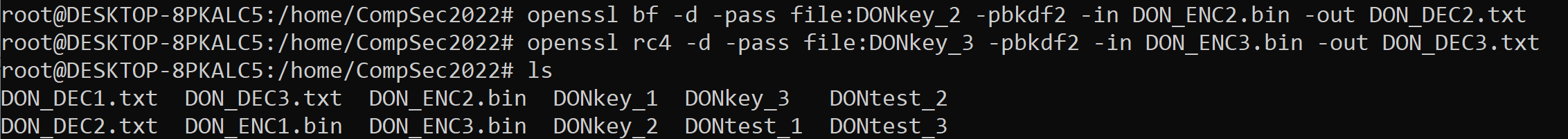


5.

Here I decrypted the files by using the key my partner would have sent me along with the encrypted files and the encryption algorithm they would have told me they encrypted them with. I put the decrypted content in a new text file.



I did the same for the other two files but decrypting them with the corresponding algorithm.



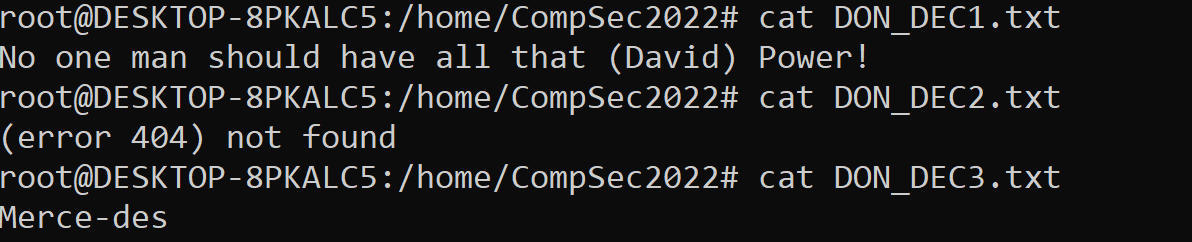
I then compared the files.

A picture containing text

Description automatically generated

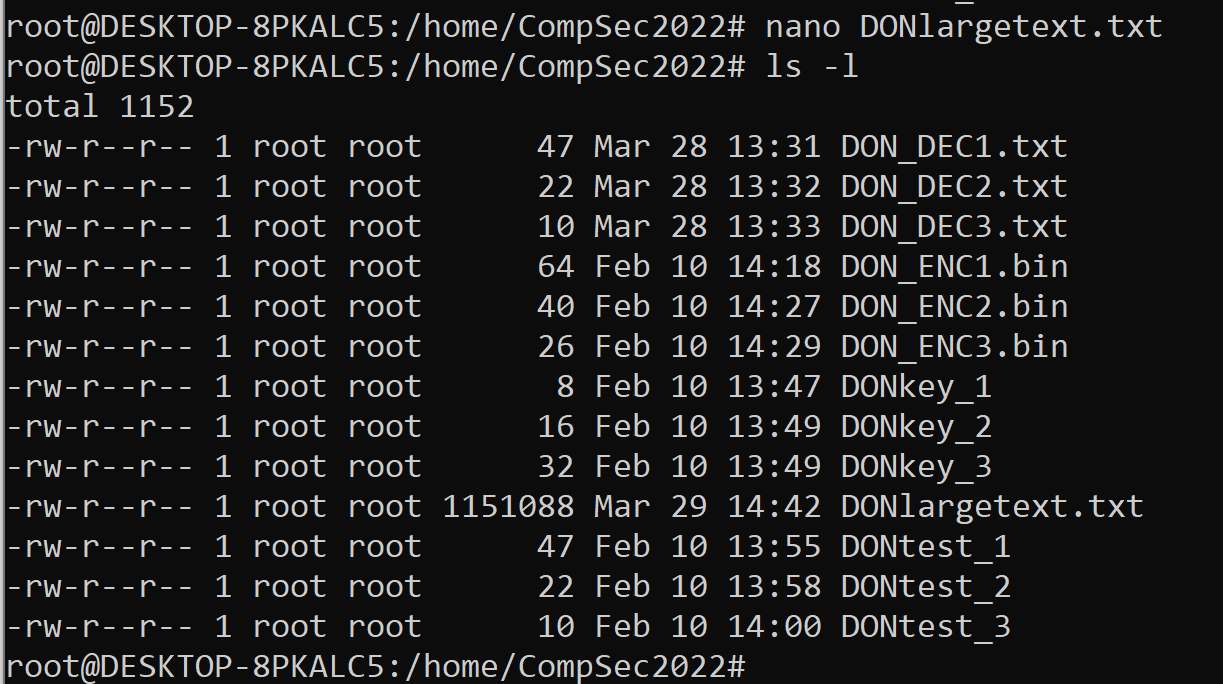
6.

I then checked the original content from the decrypted files.



7.

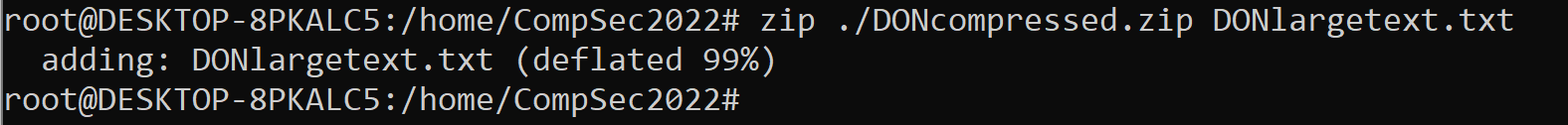
I then created a file (approximately 1mb)

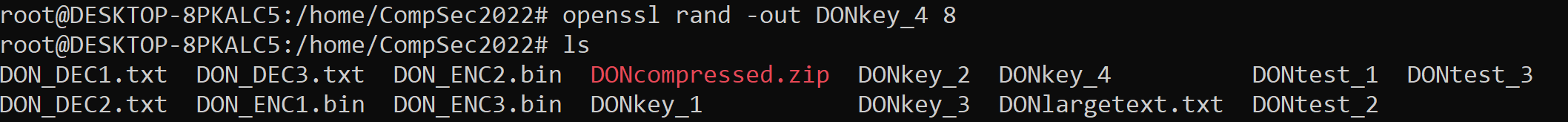
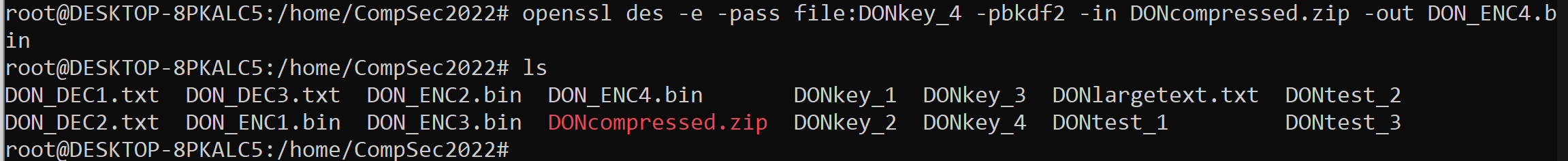


I then installed zip and compressed the file

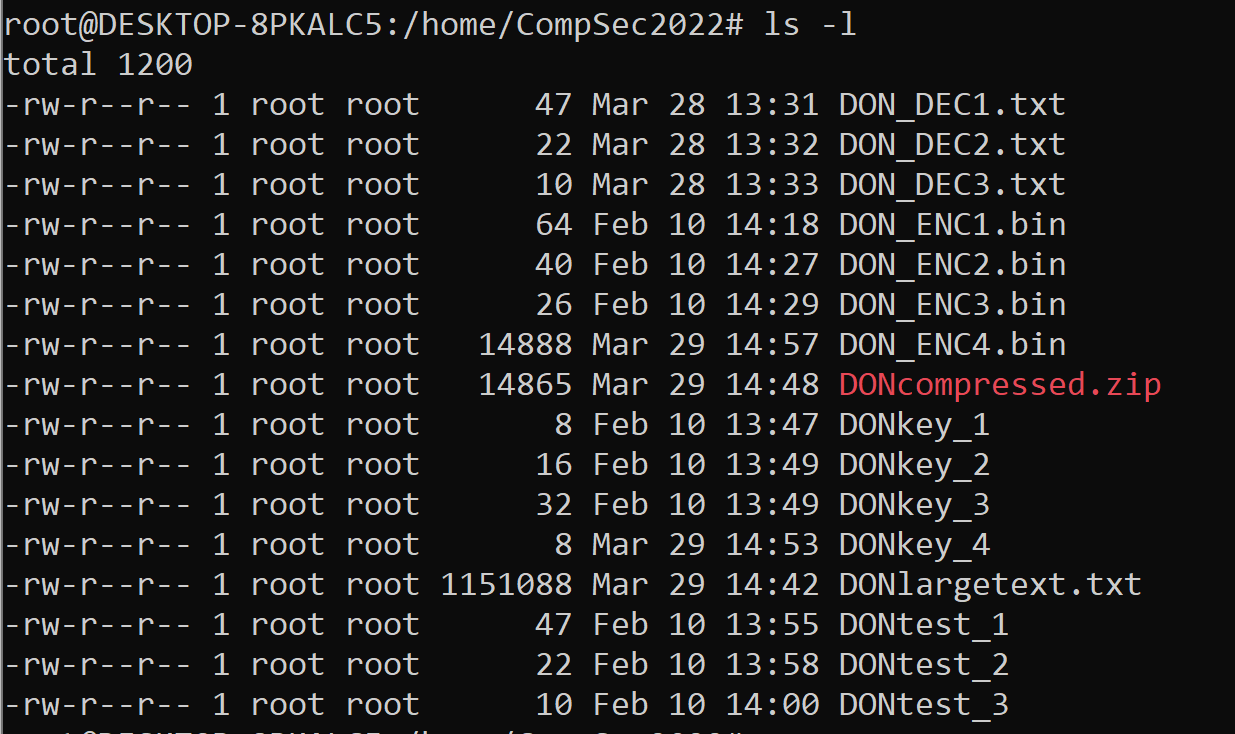
Text, logo

Description automatically generated



I then encrypted the file

Comparing the size of the three files:

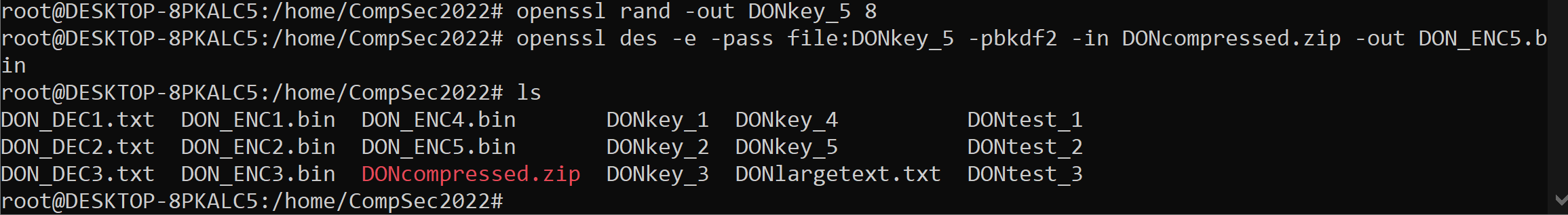


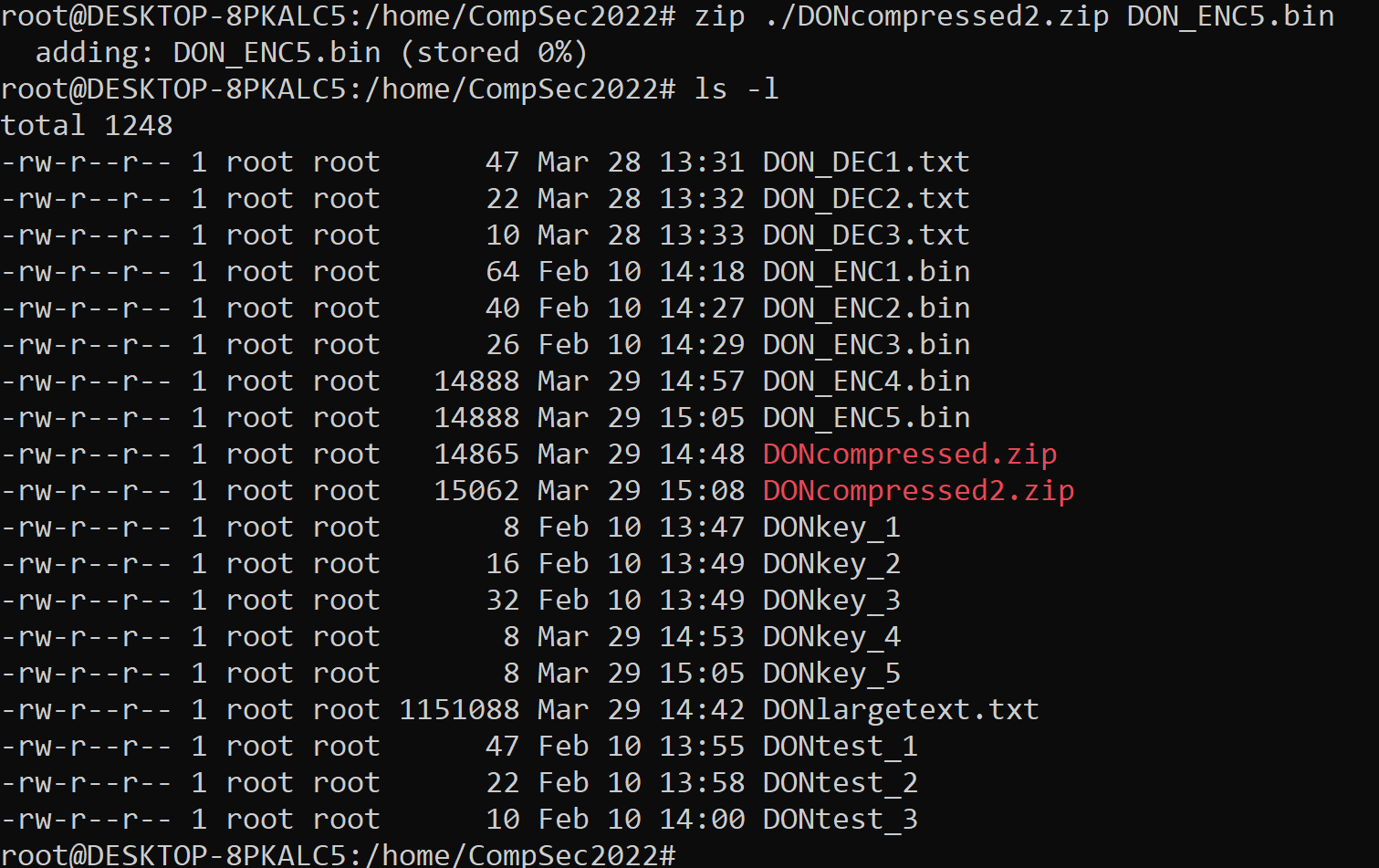
9.

From 1151080 bytes in its original state, to 14865 compressed then 14888 encrypted post compression.

8.

Going back to my original file, I then encrypted it.



Then compressed it

9.

And found that the original again being 1151088, then the encrypted being the same 14888 but compressing that actually made the file take up more bytes at 15062.

10.

Encrypting the file after it has been compressed adds a small amount of bytes.

But that is the same amount of bytes if you were to simply encrypt the file.

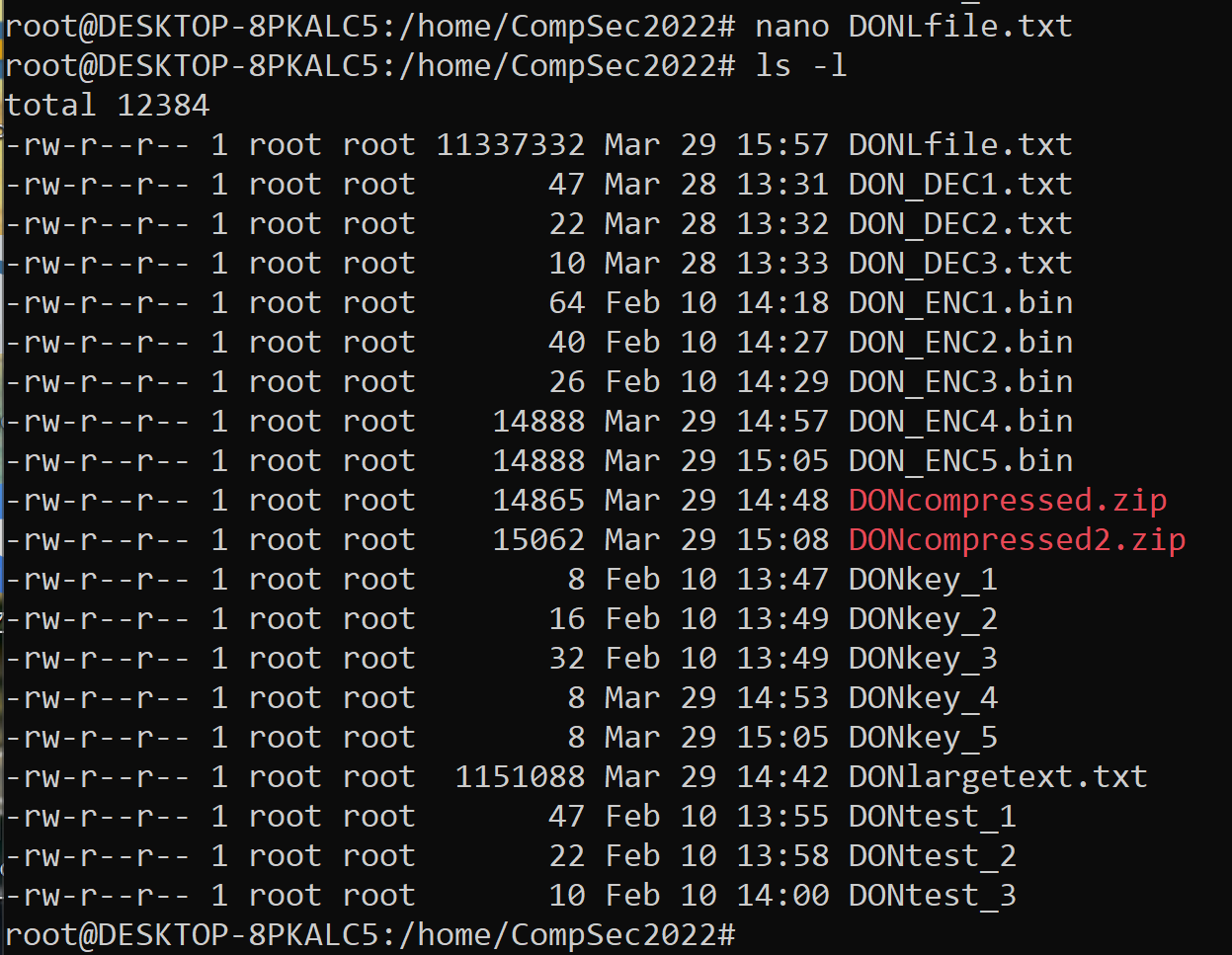
Compressing the file after it has been encrypted actually adds more bytes to it due to it losing some of the properties that make compression work.

11.

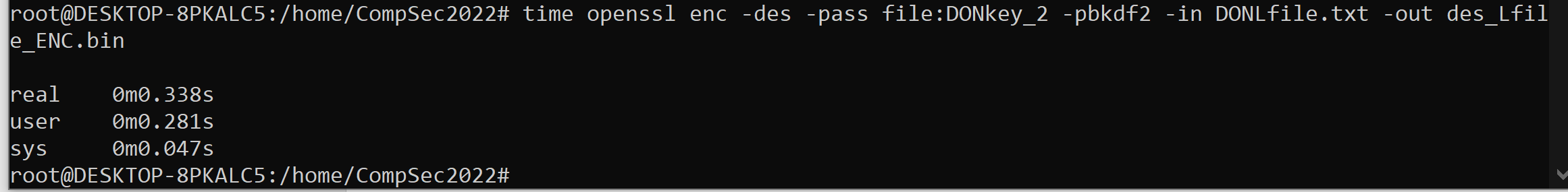
If I were to send a file to a friend, I would compress it first then encrypt it due to it taking up less bytes.

12.

I created a large file of approximately 11mb



13.

I then recorded how much time it to encrypt the large file with various encryption algorithms.

14.

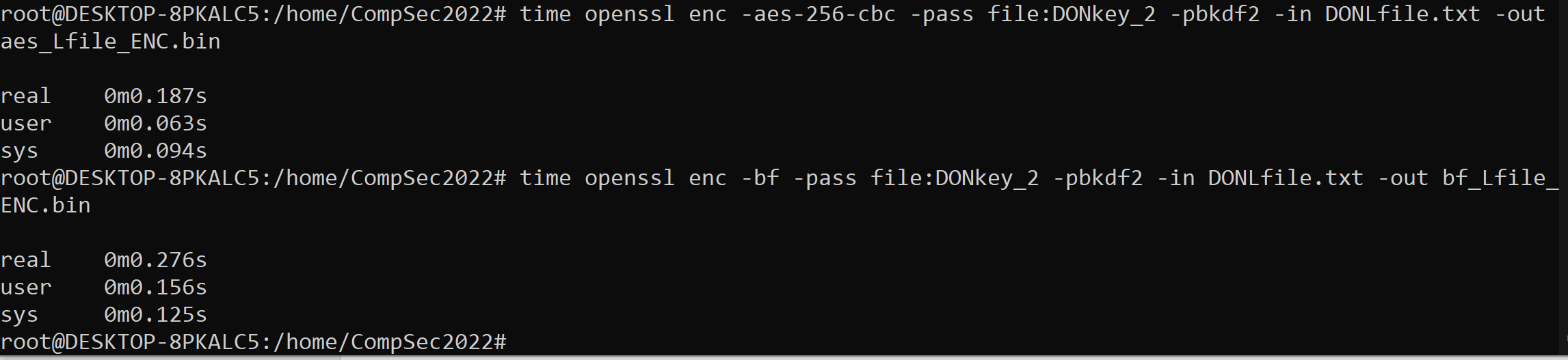
I chose a total of 5 algorithms including des, 3des, rc4, aes and bf.

Here is 3des and rc4

Text

Description automatically generated

Here is aes and bf



15.

I encrypted these files three times in order to get an accurate average for my final result.

This is my times for real

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Averages |
| des | 338 | 274 | 304 | 305.33 |
| 3des | 627 | 517 | 732 | 625.33 |
| Rc4 | 189 | 131 | 144 | 154.67 |
| Aes | 187 | 155 | 149 | 163.67 |
| Bf | 276 | 268 | 201 | 248.33 |

This is my times for user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Averages |
| des | 281 | 156 | 188 | 208.33 |
| 3des | 453 | 469 | 672 | 531.33 |
| Rc4 | 078 | 031 | 031 | 46.67 |
| Aes | 063 | 047 | 047 | 52.33 |
| Bf | 156 | 188 | 141 | 161.67 |

This is my times for sys

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Averages |
| des | 047 | 109 | 109 | 88.33 |
| 3des | 156 | 047 | 063 | 88.67 |
| Rc4 | 094 | 078 | 094 | 88.67 |
| Aes | 094 | 109 | 094 | 99 |
| Bf | 125 | 094 | 063 | 94 |

All number are in .seconds

296I made a graph of the average CPU time (which is the sum of the user time and system time)

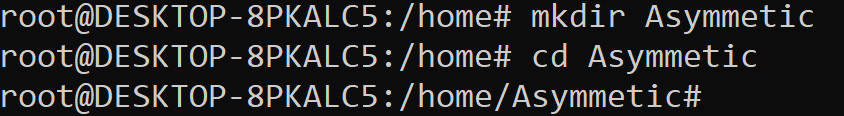
Which was

|  |  |
| --- | --- |
| Des | 296.66 |
| 3des | 620 |
| Rc4 | 135.34 |
| Aes | 151.33 |
| bf | 255.67 |

A screenshot of a computer

Description automatically generated with medium confidence

rc4 took the least time with aes close behind. Bf took substantially more time with des behind it and the standout was 3des taking over 4 times as long as rc4.

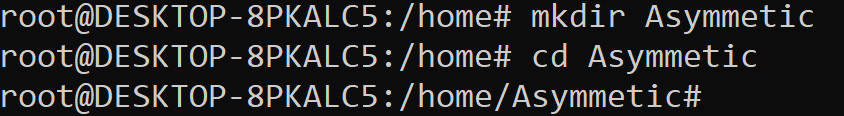


**Steps for Asymmetric Encryption**

1.

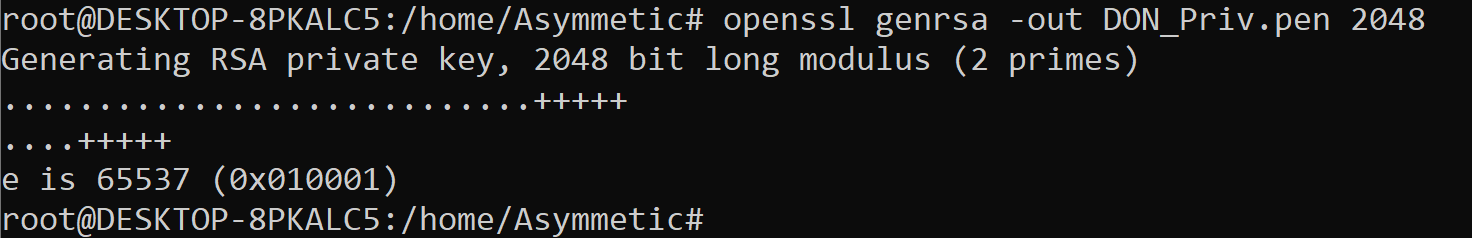
Unfortunately, due to illness I did not attend the labs and did not have a partner, so I improvised.

2.

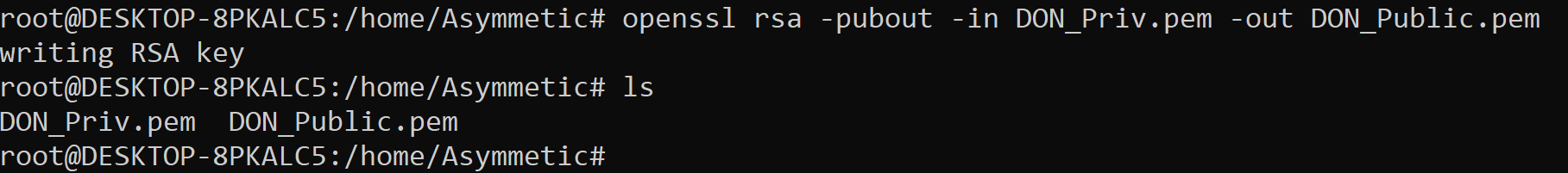


3a.

I generated a private key pair for the RSA algorithm. I decided not to give it a password, so I didn’t include aes128.

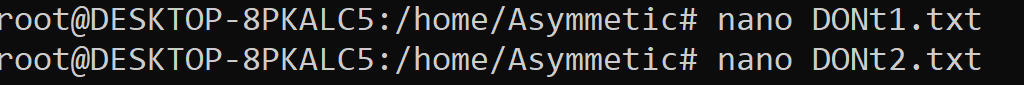


3b.

Here I extracted my public key form my private key.

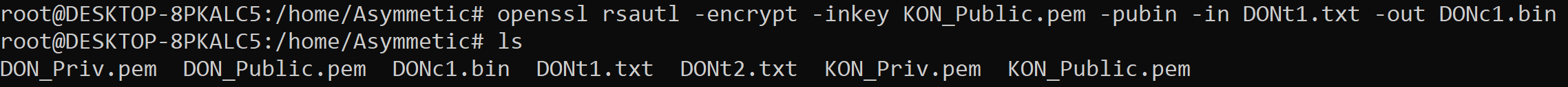
3c.

Here I created 2 plain text files called DONt1 and 2



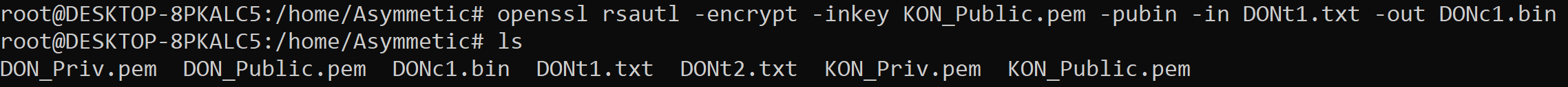
4.

Since I didn’t have a partner, I created another two keys calling them KON (for the rest of the assignment I will be acting as though Kate has given me her public key. I then encrypted my files with her public key



5.

When I received the encrypted files from Kate I then decrypted them with my own private key.



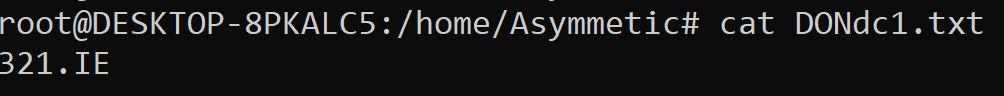
6.

I checked to see if there were any differences (there wasn’t).



7.

I then checked the contents of the file. The encryption device this supports is RSA security



8.

I then singed my second text file from earlier.

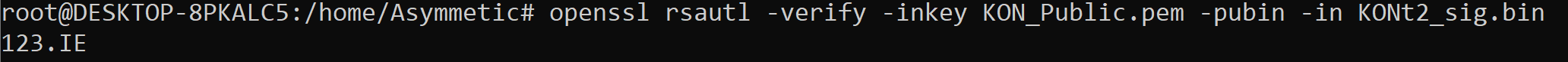


9.

I gave it to Kate and received one from her.

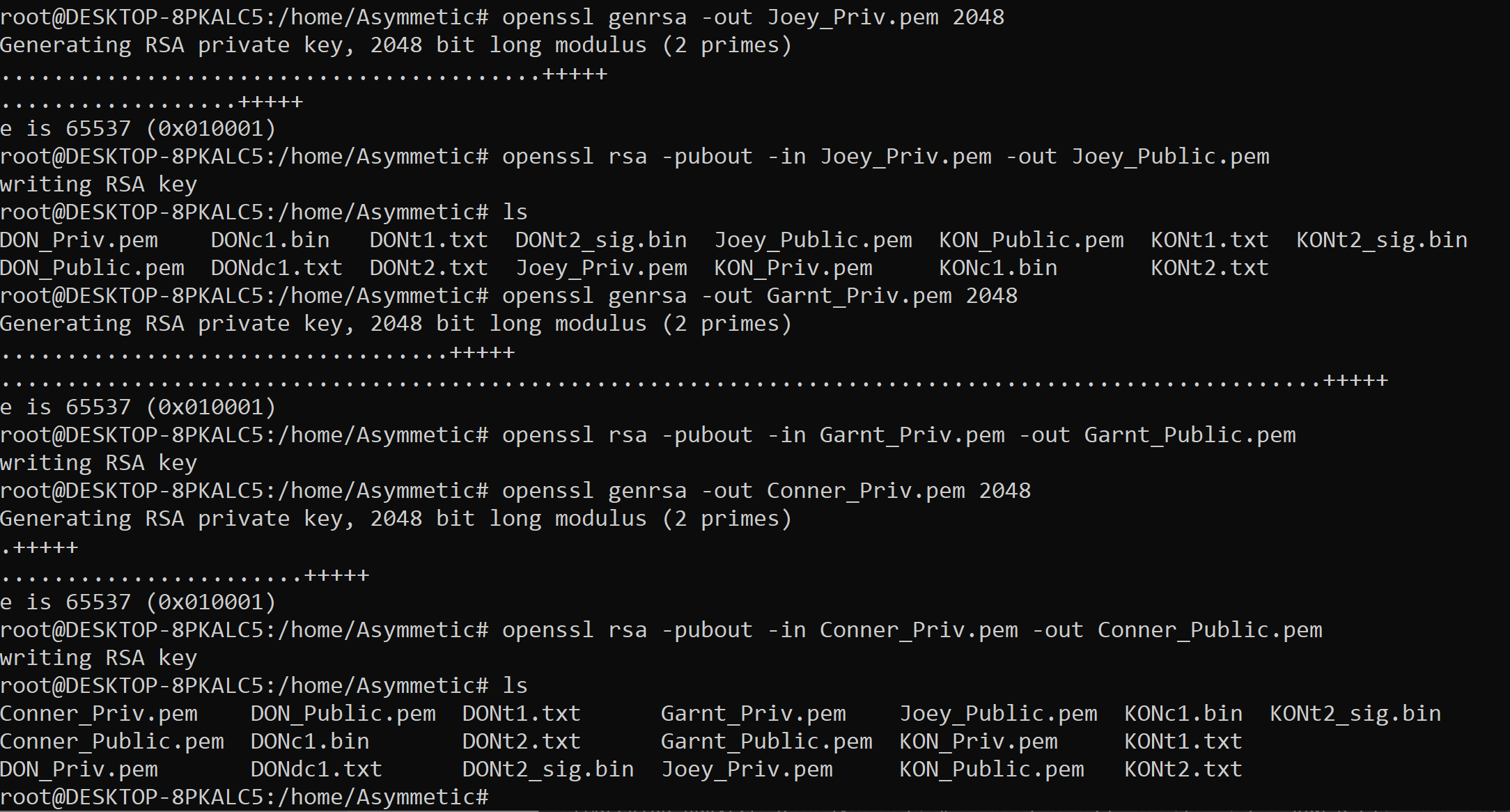
10.

I then decrypted the file using her public key.



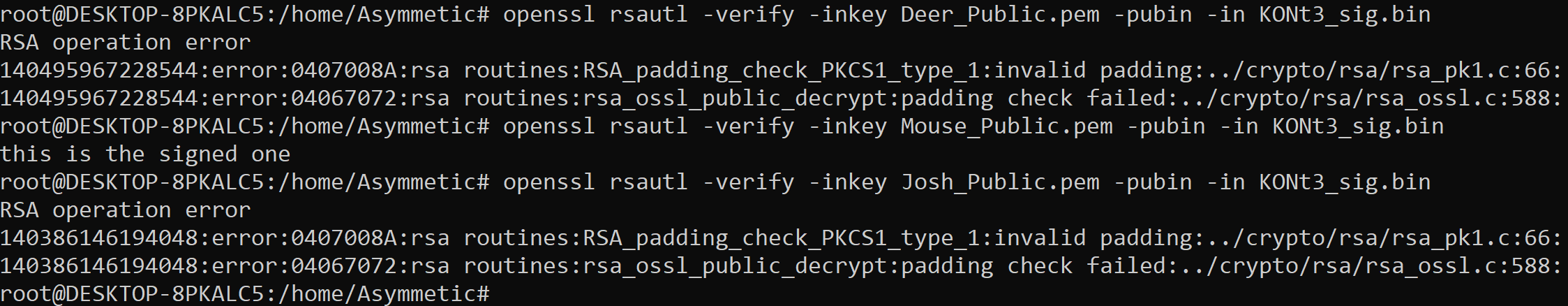
11.

I then created 3 new key pairs called Joey, Garnt and Conner.



12.

Kates three key pairs were Deer, Mouse and Josh. We exchanged public keys and a signed file, and I worked out it was signed with Mouses private key.



13.

I created 5 key of different sizes 512, 1024, 2048, 4096 and 8192

Text

Description automatically generated

Private real

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 061 | 32 | 25 | 39.33 |
| 1024| | 052 | 031 | 053 | 45.33 |
| 2048| | 076 | 069 | 084 | 76.33 |
| 4096| | 492 | 361 | 213 | 355.33 |
| 8192| | 5069 | 6448 | 7904 | 6473.67 |

Private user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 0 | 16 | 0 | 5.33 |
| 1024| | 0 | 0 | 16 | 5.33 |
| 2048| | 016 | 031 | 063 | 36.67 |
| 4096| | 422 | 313 | 125 | 286.67 |
| 8192| | 5016 | 6375 | 7781 | 6390.67 |

Private System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 0 | 0 | 0 | 0.0 |
| 1024| | 0 | 0 | 16 | 5.33 |
| 2048| | 0 | 0 | 0 | 0.0 |
| 4096| | 0 | 0 | 0 | 0.0 |
| 8192| | 016 | 0 | 078 | 31.33 |

Public real

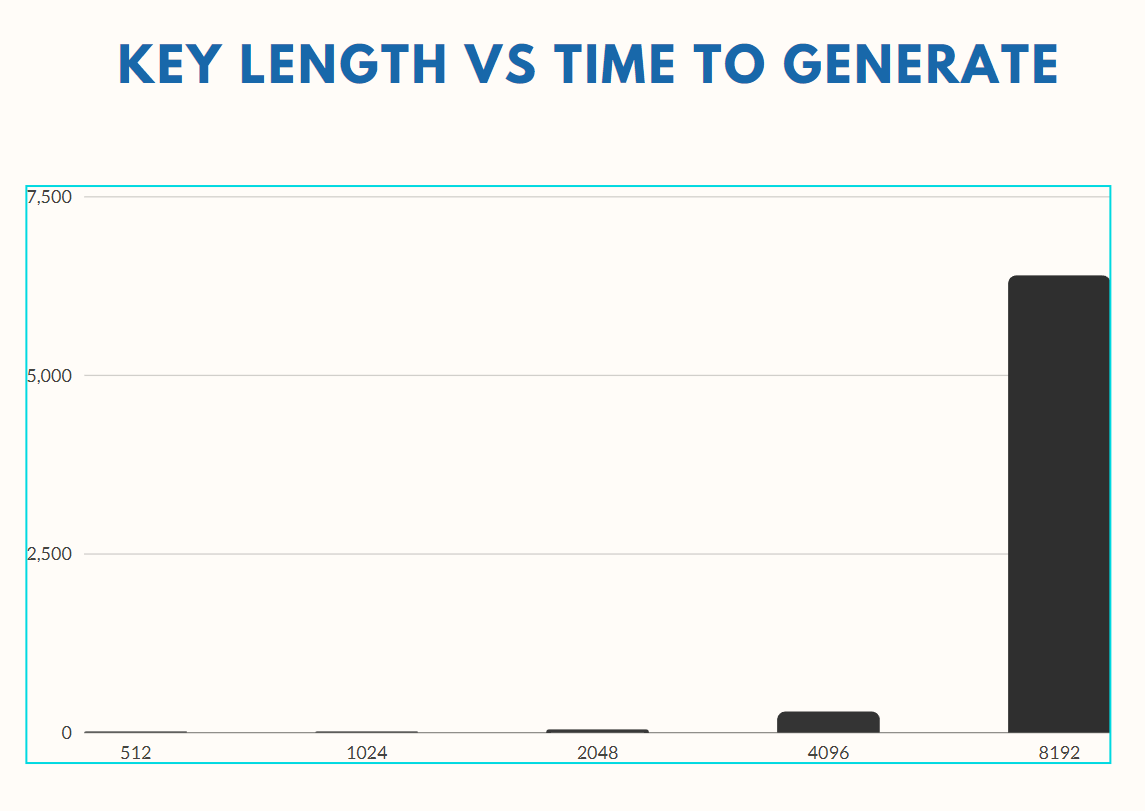
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 015 | 016 | 016 | 15.67 |
| 1024| | 19 | 11 | 12 | 14 |
| 2048| | 024 | 013 | 012 | 16.33 |
| 4096| | 012 | 014 | 012 | 12.67 |
| 8192| | 022 | 014 | 012 | 16 |

Public user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 0 | 0 | 0 | 0.0 |
| 1024| | 0 | 0 | 16 | 5.33 |
| 2048| | 016 | 016 | 0 | 10.67 |
| 4096| | 016 | 0 | 016 | 10.67 |
| 8192| | 0 | 0 | 0 | 0.0 |

Public system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1st | 2nd | 3rd | Average |
| 512| | 0 | 0 | 0 | 0.0 |
| 1024| | 16 | 0 | 0 | 5.33 |
| 2048| | 016 | 0 | 016 | 10.67 |
| 4096| | 0 | 0 | 0 | 0.0 |
| 8192| | 0 | 0 | 016 | 5.33 |



This is a graph of the private user time to generate against the key length. 512 and 2048 lengths took essentially no time. 2048 over 6 times longer than the first two but still extreme quick. 4096 took another big leap from .036.67 to 286.67 now taking a quarter of a second. 8192 took 6.4 seconds which was over 30 times longer than 4096.

Asymmetric Encryption works by having two unique but related keys, a pubic and a private one. The public one is used to encrypt and the private is used to decrypt.

Asymmetric encryption is used in email security and web security.

Generally Symmetric encryption is used over Asymmetric because it is more efficient.