

## INTRODUCTION

Throughout this lab, we will be practicing hashing techniques while also demonstrating and observing encryption in action in later parts.

### SECTION 1: HANDS ON DEMONSTRATION

#### Part 1: Create a text file on Linux.

In this section, I accessed the client and created a text file on the system.

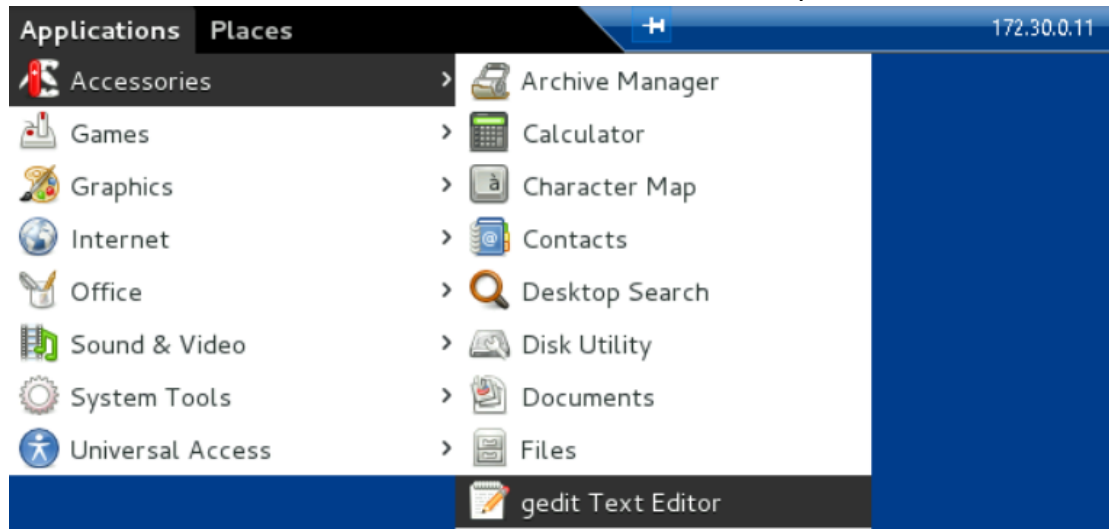


Figure 1: Accessing the client and accessing the text editor.

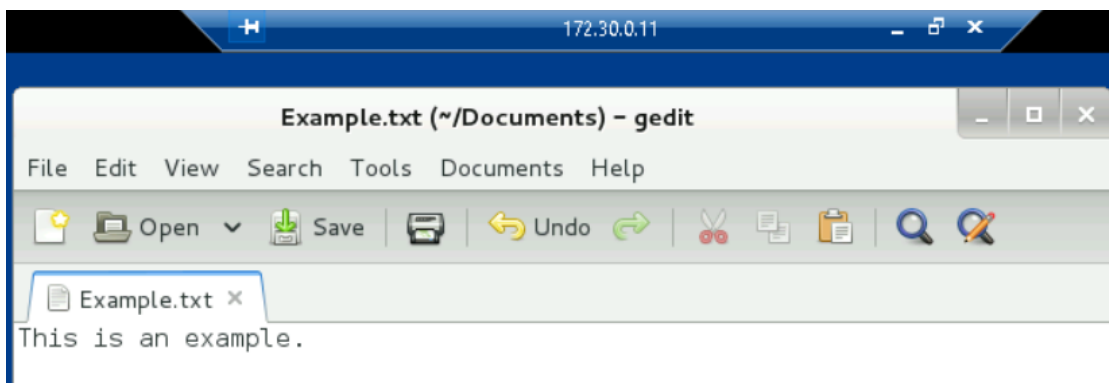


Figure 2: Setting up the file and inputting text.

## **Part 2: Create a MD5sum and a SHA1sum Hash String.**

### **Segment 1: MD5sum Hash String**

Next, I navigated to the terminal and used the `md5sum --help` command to get an idea of what the command does.

```
student@TargetLinux01:~$ md5sum --help
Usage: md5sum [OPTION]... [FILE]...
Print or check MD5 (128-bit) checksums.
With no FILE, or when FILE is -, read standard input.
```

*Figure 3: the md5sum --help command with the beginning output.*

After that, I used multiple commands to navigate to the Example.txt file listed earlier and display its contents. I also found the md5sum for the file.

```
student@TargetLinux01:~$ cd Documents
student@TargetLinux01:~/Documents$ ls -l
total 4
-rw-r--r-- 1 student student 20 Feb  1 18:20 Example.txt
student@TargetLinux01:~/Documents$ cat Example.txt
This is an example.
student@TargetLinux01:~/Documents$ md5sum Example.txt
46edc6541babd006bb52223c664b29a3  Example.txt
student@TargetLinux01:~/Documents$ █
```

*Figure 4: screenshot of the commands I typed followed by their outputs.*

Following that, I created the md5 and made sure the md5 hash was the same as the above hash.

```
student@TargetLinux01:~/Documents$ ls
Example.txt  Example.txt.md5
student@TargetLinux01:~/Documents$ cat Example.txt.md5
46edc6541babd006bb52223c664b29a3  Example.txt
```

*Figure 5: The output for the Example.txt.md5 file.*

### **Segment 2: SHA1sum Hash String**

First, I used the `sha1sum --help` to get an understanding of the hashing.

```
The sums are computed as described in FIPS-180-1. When checking, the input
should be a former output of this program. The default mode is to print
a line with checksum, a character indicating type ('*' for binary, ' ' for
text), and name for each FILE.
```

```
Report shasum bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'shasum invocation'
student@TargetLinux01:~/Documents$ shasum Example.txt
a6f153801c9303d73ca2b43d3be62f44c6b66476  Example.txt
student@TargetLinux01:~/Documents$ █
```

*Figure 6: Identifying the SHA1 hash for the Example.txt file.*

Next, I created the sha1 file and went ahead and made sure the sha1sum was the same as the other file.

```
student@TargetLinux01:~/Documents$ shasum Example.text > Example.txt.sha1
shasum: Example.text: No such file or directory
student@TargetLinux01:~/Documents$ shasum Example.txt > Example.txt.sha1
student@TargetLinux01:~/Documents$ ls
Example.txt  Example.txt.md5  Example.txt.sha1
student@TargetLinux01:~/Documents$ cat Example.txt.sha1
a6f153801c9303d73ca2b43d3be62f44c6b66476  Example.txt
student@TargetLinux01:~/Documents$ █
```

*Figure 7: Verifying that the Example.txt.sha1 file has the same SHA1 hash.*

The last step was to make sure that the sha1 hash wasn't altered in some way.

```
student@TargetLinux01:~/Documents$ shasum -c Example.txt.sha1
Example.txt: OK
student@TargetLinux01:~/Documents$ █
```

*Figure 8: Checking to ensure the file wasn't manipulated.*

### **Part 3: Modify a File and Verify Hash Values.**

In this part, we will be modifying the file to include our name and comparing the new hash to the two previous hashes.

```
student@TargetLinux01:~/Documents$ echo Javonnie Rutherford >> Example.txt
student@TargetLinux01:~/Documents$ cat Example.txt
This is an example.
Javonnie Rutherford
student@TargetLinux01:~/Documents$ md5sum Example.txt
9b5c691e3aa76330393024801f5d7044  Example.txt
student@TargetLinux01:~/Documents$ cat Example.txt.md5
46edc6541babd006bb52223c664b29a3  Example.txt
student@TargetLinux01:~/Documents$ shasum Example.txt
651c04dc14ef9369d507c3a2c1f1f0ef261ff8bc  Example.txt
student@TargetLinux01:~/Documents$ cat Example.txt.sha1
a6f153801c9303d73ca2b43d3be62f44c6b66476  Example.txt
student@TargetLinux01:~/Documents$ █
```

*Figure 9: Modifying the file and comparing the different hashes to the previous versions.*

## **Part 4: Generate GnuPG Keys.**

In this part, I will be using GnuPG to encrypt a message using RSA.

First, I started the program using the `gpg --gen-key` command and answering a series of questions.

```
student@TargetLinux01:~/Documents$ gpg --gen-key
gpg (GnuPG) 1.4.12; Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory `/home/student/.gnupg' created
gpg: new configuration file `/home/student/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/student/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/home/student/.gnupg/secring.gpg' created
gpg: keyring `/home/student/.gnupg/pubring.gpg' created
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
```

*Figure 10: Initiating the gpg process.*

After that, I used the `./entropy_loop.sh` command on a separate terminal to generate enough byte combinations for the key generation. It was a little bit of a weird process, but I think it works by taking “inspiration” from the byte combinations currently being created whenever you execute a command or an action, and then uses that to help fuel the encryption.

```
We need to generate a lot of random bytes. It is a good idea
to perform some other action (type on the keyboard, move the
mouse, utilize the disks) during the prime generation; this
gives the random number generator a better chance to gain
enough entropy.
.+++++
.+++++
gpg: /home/student/.gnupg/trustdb.gpg: trustdb created
gpg: key B692CA5D marked as ultimately trusted
public and secret key created and signed.
```

*Figure 11: Output showing it needed more entropy.*

Next, I output the created key to a file to ensure I have it for later use.

```
student@TargetLinux01:~/Documents$ gpg --export -a > student.pub
student@TargetLinux01:~/Documents$ ls
Example.txt  Example.txt.md5  Example.txt.shal  student.pub
```

*Figure 12: Outputting the key into a file.*

Lastly, I repeated all the previous steps, but instead for the instructor account.

```
Instructor@TargetLinux01:~$ gpg --export -a > instructor.pub
Instructor@TargetLinux01:~$ ls
Desktop  Documents  Downloads  instructor.pub  Music  Pictures  Public  Templates  Videos
```

*Figure 13: The last output to verify I repeated the steps.*

## Part 5: Share a GnuPG Key.

For this step, the main goal is to exchange the keys by making a copy and importing.

First, I copied the key to the student documents directory. Next, I made sure it worked before importing it.

```
student@TargetLinux01:~/Documents$ cp /home/Instructor/instructor.pub /home/student/Documents/instructor.pub
student@TargetLinux01:~/Documents$ ls
Example.txt  Example.txt.md5  Example.txt.sha1  instructor.pub  student.pub
student@TargetLinux01:~/Documents$ gpg --list-keys
/home/student/.gnupg/pubring.gpg
-----
pub   1024R/B692CA5D 2024-02-03
uid           Student <student@securelabsondemand.com>
sub   1024R/FF9A5183 2024-02-03

student@TargetLinux01:~/Documents$ gpg --import instructor.pub
gpg: key 9E815A4E: public key "Instructor <instructor@securelabsondemand.com>" imported
gpg: Total number processed: 1
gpg:         imported: 1 (RSA: 1)
```

Figure 14: Copying the key, verifying it was copied, and importing it.

After I imported the key, I verified that it showed up under the list.

```
student@TargetLinux01:~/Documents$ gpg --list-keys
/home/student/.gnupg/pubring.gpg
-----
pub   1024R/B692CA5D 2024-02-03
uid           Student <student@securelabsondemand.com>
sub   1024R/FF9A5183 2024-02-03

pub   1024R/9E815A4E 2024-02-03
uid           Instructor <instructor@securelabsondemand.com>
sub   1024R/DCFB5457 2024-02-03
```

Figure 15: Checking to make sure the key was now on the list.

## Part 6: Encrypt and Decrypt a ClearText Message.

For this step, we will be encrypting and decrypting a cleartext message with the keys we generated earlier.

First, I created the cleartext.txt file. After that I went ahead and encrypted the file and made sure it was there.

```
63 echo "this is a clear-text message from Javonnie Rutherford" > cleartext.txt
64 cat cleartext.txt
65 gpg -e cleartext.txt
66 ls
67 cat cleartext.txt.gpg
```

Figure 16: I had to use history since after displaying the encryption it cleared my screen.

After that, I displayed the contents of the file to make sure it was properly encrypted.

```
000$400060000000Sk<UmB0[2Z700%Qs00[000f*01#UM0f[9P8T0a=0AY00@
00f0X0jH000}bc10P}0r0R-0u00w[0000Q10Dh900H0fs0r
qi[9000z.[00000^0\6[00w00[00?00^00!00000500000T'00sS,1L/[000[00[00bn[00[00`0*H00X[00
x01:~/Documents$ █
```

Figure 17: Displaying the contents of the encrypted file.

Following displaying the contents, I went ahead and “sent” the file to the Instructor, and changed the ownership to ensure that they can open the message.

```
student@TargetLinux01:~/Documents$ sudo cp cleartext.txt.gpg /home/Instructor/
[sudo] password for student:
student@TargetLinux01:~/Documents$ cd /home/Instructor/
student@TargetLinux01:/home/Instructor$ sudo chown Instructor:Instructor cleartext.txt.gpg
student@TargetLinux01:/home/Instructor$ ls -la
total 124
drwxr-xr-x 20 Instructor Instructor 4096 Feb  3 14:28 .
drwxr-xr-x  4 root          root      4096 Mar 27  2017 ..
-rw-r----- 1 Instructor Instructor   89 Feb  3 14:14 .bash_history
-rw-r--r--  1 Instructor Instructor  220 Mar 27  2017 .bash_logout
-rw-r--r--  1 Instructor Instructor 3392 Mar 27  2017 .bashrc
drwx----- 4 Instructor Instructor 4096 Nov 23  2020 .cache
-rw-r--r--  1 Instructor Instructor  263 Feb  3 14:28 cleartext.txt.gpg
```

*Figure 18: changing ownership of the file and making sure the file is present.*

Next, I switched users, typed in the secret key, and decrypted the message! This wraps up section 1.

```
student@TargetLinux01:/home/Instructor$ su Instructor
Password:
Instructor@TargetLinux01:~$ gpg -d cleartext.txt.gpg

You need a passphrase to unlock the secret key for
user: "Instructor <instructor@securelabsondemand.com>"
1024-bit RSA key, ID DCFB5457, created 2024-02-03 (main key ID 9E815A4E)

gpg: encrypted with 1024-bit RSA key, ID DCFB5457, created 2024-02-03
      "Instructor <instructor@securelabsondemand.com>"
this is a clear-text message from Javonn Rutherford
```

*Figure 19: decrypting the file and checking the output.*



## SECTION 2: APPLIED LEARNING

### Part 1: Create a text file on Linux.

In this first part, the process was pretty simple. All I did was ensure I was logged into the student account, changed directories, and create a text file.

```
student@TargetLinux01:~$ whoami
student
student@TargetLinux01:~$ cd Documents
student@TargetLinux01:~/Documents$ vi Example2.txt
student@TargetLinux01:~/Documents$ cat Example2.txt
This file is from Javonnie Rutherford.
student@TargetLinux01:~/Documents$
```

Figure 20: creating the .txt file.

### Part 2: Create a MD5sum and a SHA1sum Hash String.

#### Segment 1: MD5sum hash

In this segment, I look at the md5sum hash for the Example2.txt, output the md5sum hash into a .md5 file, and then cat the file to make sure the hash is the same.

```
student@TargetLinux01:~/Documents$ md5sum Example2.txt
6af3517496e43fcbdd002e15c5d44231 Example2.txt
student@TargetLinux01:~/Documents$ md5sum Example2.txt > Example2.txt.md5
student@TargetLinux01:~/Documents$ ls
cleartext.txt      Example2.txt.md5  Example.txt.sha1
cleartext.txt.gpg  Example.txt      instructor.pub
Example2.txt       Example.txt.md5  student.pub
student@TargetLinux01:~/Documents$ cat Example2.txt.md5
6af3517496e43fcbdd002e15c5d44231 Example2.txt
student@TargetLinux01:~/Documents$
```

Figure 21: observing the md5 hash and outputting it into a file.

#### Segment 2: SHA1sum hash

In this segment, I look at the sha256sum hash for the Example2.txt, output the SHA256sum has into a .sha256 file, and the cat the file to make sure the hash is the same. Ignore some of the errors at the end, I got a little confused for a second.

```
student@TargetLinux01:~/Documents$ sha256sum Example2.txt
146b812a8dbc054c0be97ee4775aa95521b9fab39132174e6ed706229c878ab Example2.txt
student@TargetLinux01:~/Documents$ sha256sum Example2.txt > Example2.txt.sha256
student@TargetLinux01:~/Documents$ ls
cleartext.txt      Example2.txt.md5  Example.txt      instructor.pub
cleartext.txt.gpg  Example2.txt.sha1 Example.txt.md5  student.pub
Example2.txt       Example2.txt.sha256 Example.txt.sha1
student@TargetLinux01:~/Documents$ sha256 -c Example2.txt
-bash: sha256: command not found
student@TargetLinux01:~/Documents$ sha256sum -c Example2.txt
sha256sum: Example2.txt: no properly formatted SHA256 checksum lines found
student@TargetLinux01:~/Documents$ sha256sum -c Example2.txt.sha256
Example2.txt: OK
```

Figure 22: observing the sha256sum hash and outputting it into a file.

### **Part 3: Modify a File and Verify Hash Values.**

In this part, we modify the file and overwrite the two previous hash files, and we can verify the different hash values by comparing them to the previous hashes captured in the above screenshot.

```
student@TargetLinux01:~/Documents$ echo "This example is testing hash values." >
> Example2.txt
student@TargetLinux01:~/Documents$ cat Example2.txt
This file is from Javonnie Rutherford.
This example is testing hash values.
student@TargetLinux01:~/Documents$ md5sum Example2.txt > Example2.txt.md5
student@TargetLinux01:~/Documents$ sha256 Example2.txt > Example2.txt.sha256
-bash: sha256: command not found
student@TargetLinux01:~/Documents$ sha256sum Example2.txt > Example2.txt.sha256
student@TargetLinux01:~/Documents$ cat Example2.txt.md5
f28f16eda25536038457b043bb08f2ca Example2.txt
student@TargetLinux01:~/Documents$ cat Example2.txt.sha256
1a494e17d20750bc55d0c9cf1869ba44d77d91f936b0cd176bab532cf579012e Example2.txt
student@TargetLinux01:~/Documents$
```

*Figure 23: modifying the .txt file and replacing the hash values.*

### **Part 4: Generate GnuPG Keys.**

First, I started the process by answering some questions needed to begin.

```
student@TargetLinux01:~/Documents$ gpg --gen-key
gpg (GnuPG) 1.4.12; Copyright (C) 2012 Free Software Foundation, Inc
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
```

*Figure 24: starting the gpg key generation process.*

Next, I opened a separate tab to generate enough entropy for the encryption. After, I exported the key to a file.

```
Not enough random bytes available. Please do some other work to give
the OS a chance to collect more entropy! (Need 73 more bytes)
.+++++
gpg: key 151C9CEC marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 2 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 2u
pub 2048R/151C9CEC 2024-02-03
    Key fingerprint = 28B1 94A7 943D 3F87 8ED4 606C 29E8 3649 151C 9CEC
uid                               Student2 <student@securelabsondemand.com>
sub 2048R/579C237E 2024-02-03

student@TargetLinux01:~/Documents$ gpg --export -a > student2.pub
```

*Figure 25: exporting the key into a file.*



After that, I listed the keys to ensure the student2 key is present.

```
student@TargetLinux01:~/Documents$ gpg --list-keys
/home/student/.gnupg/pubring.gpg
-----
pub      1024R/B692CA5D 2024-02-03
uid                               Student <student@securelabsondemand.com>
sub      1024R/FF9A5183 2024-02-03

pub      1024R/9E815A4E 2024-02-03
uid                               Instructor <instructor@securelabsondemand.com>
sub      1024R/DCFB5457 2024-02-03

pub      2048R/151C9CEC 2024-02-03
uid                               Student2 <student@securelabsondemand.com>
sub      2048R/579C237E 2024-02-03
```

Figure 26: verifying the importing of the key.

Finally, I just repeat all the previous steps, but with the Instructor profile.

```
Instructor@TargetLinux02:~$ gpg --export -a > instructor2.pub
Instructor@TargetLinux02:~$ gpg --list-keys
/home/Instructor/.gnupg/pubring.gpg
-----
pub      2048R/AA461EB7 2024-02-04
uid                               Instructor <instructor@securelabsondemand.com>
sub      2048R/4D175F43 2024-02-04
```

Figure 27: repeating the process with the instructor key.

## Part 5: Share a GnuPG Key.

So, first, I moved the instructor2.pub document from the Instructor path, to the Administrator/Documents path. Afterwards, I moved it from the Administrator/Documents path to the student/Documents path.

C:\Users\Administrator\Documents					/home/student/Documents				
Name	Size	Type	Changed		Name	Size	Changed	Rights	Owner
..		Parent directory	2/3/2024 4:37:30 PM		cleartext.txt	1 KB	2/3/2024 2:21:37 PM	rw-r--r--	student
instructor2.pub	2 KB	PUB File	2/3/2024 4:27:04 PM		cleartext.txt.gpg	1 KB	2/3/2024 2:23:20 PM	rw-r--r--	student
					Example.txt	1 KB	2/3/2024 1:13:41 PM	rw-r--r--	student
					Example.txt.md5	1 KB	2/1/2024 6:38:26 PM	rw-r--r--	student
					Example.txt.sha1	1 KB	2/1/2024 7:05:47 PM	rw-r--r--	student
					Example2.txt	1 KB	2/3/2024 3:42:05 PM	rw-r--r--	student
					Example2.txt.md5	1 KB	2/3/2024 3:42:43 PM	rw-r--r--	student
					Example2.txt.sha1	1 KB	2/3/2024 3:43:02 PM	rw-r--r--	student
					Example2.txt.sha256	1 KB	2/3/2024 3:43:22 PM	rw-r--r--	student
					instructor.pub	2 KB	2/3/2024 2:15:42 PM	rw-r--r--	student
					instructor2.pub	2 KB	2/3/2024 4:27:04 PM	rw-r--r--	student
					student.pub	2 KB	2/3/2024 1:34:08 PM	rw-r--r--	student
					student2.pub	4 KB	2/3/2024 3:55:56 PM	rw-r--r--	student

Figure 28: moving the key over to the /student/Documents directory.

Next, I changed the permissions to ensure that the student user can open the file.

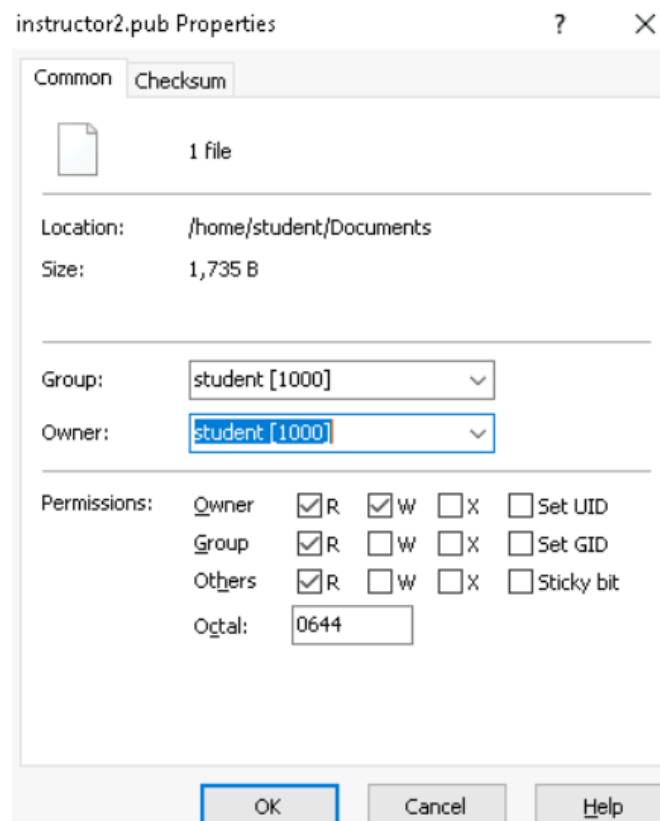


Figure 29: changing the permissions on the file.

Finally, I listed the keys. I forgot to add the 2 at the end, so I had to repeat some earlier steps since I found out on the next step you can't just pick one and the system pulls the older of the two.

```
student@TargetLinux01:~/Documents$ gpg --list-key
/home/student/.gnupg/pubring.gpg
-----
pub   1024R/B692CA5D 2024-02-03
uid           Student <student@securelabsondemand.com>
sub   1024R/FF9A5183 2024-02-03

pub   1024R/9E815A4E 2024-02-03
uid           Instructor <instructor@securelabsondemand.com>
sub   1024R/DCFB5457 2024-02-03

pub   2048R/151C9CEC 2024-02-03
uid           Student2 <student@securelabsondemand.com>
sub   2048R/579C237E 2024-02-03

pub   2048R/AA461EB7 2024-02-04
uid           Instructor <instructor@securelabsondemand.com>
sub   2048R/4D175F43 2024-02-04

pub   2048R/18F80615 2024-02-04
uid           Instructor2a <instructor@secureondemandlabs.com>
sub   2048R/59754D42 2024-02-04
```

Figure 30: making sure the keys were on the student machine.

## Part 6: Encrypt and Decrypt a ClearText Message.

Again, messed up earlier steps so instead Instructor2 is Instructor2a for me. In this step, we will be encrypting the message.

```
Enter the user ID. End with an empty line: Instructor2a
gpg: 59754D42: There is no assurance this key belongs to the named user

pub 2048R/59754D42 2024-02-04 Instructor2a <instructor@secureondemandlabs.com>
Primary key fingerprint: 286E 86DF 2CED C4E6 ED94 FFB7 71E6 4179 18F8 0615
Subkey fingerprint: 1E39 90D4 D6EB A45E 582F F062 E3DF BE15 5975 4D42

It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y

Current recipients:
2048R/59754D42 2024-02-04 "Instructor2a <instructor@secureondemandlabs.com>"

Enter the user ID. End with an empty line:
student@TargetLinux01:~/Documents$
```

Figure 31: encrypting the file.

Next, we display the contents of the file

```
student@TargetLinux01:~/Documents$ cat cleartext2.bt.gpg
B a
C: I 8! ; ; tXtqf l " !Y BG) v4Y student@TargetLinux01:~/Documents$
```

Figure 32: contents of encrypted file.

After that, I transferred the file from the /student/Documents directory, to the /Administrator/Documents directory, and from there to the /Instructor/Documents directory.

C:\Users\Administrator\Documents					/home/Instructor				
Name	Size	Type	Changed		Name	Size	Changed	Rights	Owner
..		Parent directory	2/3/2024 5:05:09 PM		..		3/27/2017 1:40:10 PM	rwxr-xr-x	root
cleartext2.bt.gpg	1 KB	GPG File	2/3/2024 5:02:23 PM		Desktop		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
instructor2.pub	2 KB	PUB File	2/3/2024 4:27:04 PM		Documents		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
instructor2a.pub	4 KB	PUB File	2/3/2024 4:56:19 PM		Downloads		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					Music		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					Pictures		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					Public		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					Templates		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					Videos		3/27/2017 2:09:21 PM	rwxr-xr-x	Instruc...
					cleartext2.bt.gpg	1 KB	2/3/2024 5:02:23 PM	rw-r--r--	root
					entropy_loop.sh	1 KB	4/12/2017 10:32:17 AM	rwxrwxrwx	Instruc...
					instructor2.pub	2 KB	2/3/2024 4:27:04 PM	rw-r--r--	Instruc...
					instructor2a.pub	4 KB	2/3/2024 4:56:19 PM	rw-r--r--	Instruc...

Figure 33: copying the file to the /home/Instructor directory.

Next, I changed the permissions to ensure the Instructor user can open the file.

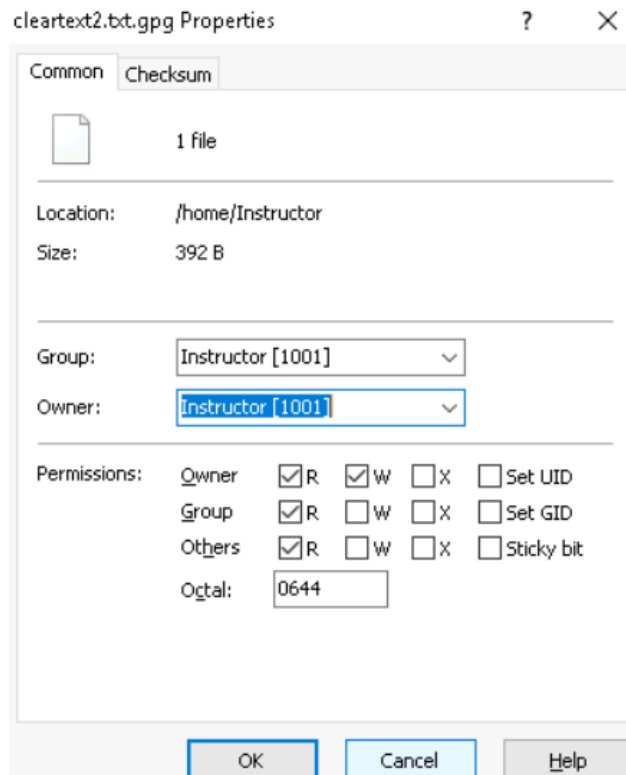


Figure 34: changing the permissions on the file.

Lastly, I decrypted the file with the password and viola: it's clear text again.

```
Instructor@TargetLinux02:~$ gpg -d cleartext2.txt.gpg

You need a passphrase to unlock the secret key for
user: "Instructor2a <instructor@secureondemandlabs.com>"
2048-bit RSA key, ID 59754D42, created 2024-02-04 (main key ID 18F80615)

gpg: encrypted with 2048-bit RSA key, ID 59754D42, created 2024-02-04
      "Instructor2a <instructor@secureondemandlabs.com>"
This clear-text message is from Javonnie Rutherford
```

Figure 35: decrypting the file and verifying the contents.

## SECTION 3: LAB CHALLENGE AND ANALYSIS

### Part 1: Analysis and Discussion

One key difference between RSA and ECDSA is what they are both based on mathematically. The RSA algorithm is based on the factorization of prime numbers, while ECDSA based models are based on the elliptical curve. This makes RSA algorithms much less complex to implement, at the cost of it requiring much longer strings, which in turn does impact their performances. On the other hand, ECDSA models don't require as long strings due to the complexity of the algorithm itself. Additionally, they perform much better in comparison to the RSA algorithms.

Thakkar, Jay. "Ecdsa vs RSA: Everything You Need to Know." *InfoSec Insights*, 9 June 2020, sectigostore.com/blog/ecdsa-vs-rsa-everything-you-need-to-know/#:~:text=One%20of%20the%20earliest%20methods%20of,is%20a%20less%20adopted%20encryption%20algorithm.&text=One%20of%20the%20earliest,less%20adopted%20encryption%20algorithm.&text=the%20earliest%20methods%20of,is%20a%20less%20adopted

### Part 2: Tools and Commands

In this section I will be redoing the md5sum hash and encrypting-decrypting process but with little to no assistance. So, let's begin.

First, I "redid" the first part by creating the .txt file.

```
student@TargetLinux01:~$ cd Documents
student@TargetLinux01:~/Documents$ echo "My name is Javonnie Rutherford" > Send.txt
student@TargetLinux01:~/Documents$ ls
cleartext2.txt      Example2.txt.md5      Example.txt.sha1      student2.pub
cleartext2.txt.gpg  Example2.txt.sha1     instructor2a.pub       student.pub
cleartext.txt       Example2.txt.sha256   instructor2.pub
cleartext.txt.gpg   Example.txt           instructor.pub
Example2.txt        Example.txt.md5       Send.txt
student@TargetLinux01:~/Documents$ md5sum Send.txt
a616876a604a749c66e8a8a0ed1fccee  Send.txt
student@TargetLinux01:~/Documents$ md5sum Send.txt > send.txt.md5
```

Figure 36: creating the document.

Next, I generated the keys and made sure to export them to a file. I also repeated the same process on the instructor machine.

```
student@TargetLinux01:~/Documents$ gpg --export -a > student_tc.pub
[1]+  Done                  gpg --export -a > studentt
student@TargetLinux01:~/Documents$ gpg --list-keys
/home/student/.gnupg/pubring.gpg
-----
```

Figure 37: exporting the key.

```
pub 2048R/C22869A4 2024-02-04
uid          t&c part2 <student@securelabsondemand.com>
sub 2048R/15EF33ED 2024-02-04
```

Figure 38: verifying the key (used name t&c for the part title "Tools & Command")

Following that, I went ahead and exchanged the keys for the two machines (instructor\_tc.pub > /student/Documents and student\_tc.pub > /home/Instructor)

instructor.pub	2 KB	2/3/2024 2:15:42 PM	rw-r--r--	student
instructor_tc.pub	5 KB	2/4/2024 11:58:10 AM	rw-r--r--	student
instructor2.pub	2 KB	2/3/2024 4:27:04 PM	rw-r--r--	student

Figure 39: moving the instructor\_tc file to the student machine.

instructor2a.pub	4 KB	2/3/2024 4:56:19 PM	rw-r--r--	Instruc...
student_tc.pub	9 KB	2/4/2024 11:53:29 AM	rw-r--r--	Instruc...

Figure 40: moving the student\_tc file to the instructor machine.

Moreover, I made sure to import the key on the student machine and verify that it was imported properly.

```
student@TargetLinux01:~/Documents$ gpg --import instructor_tc.pub
gpg: key AA461EB7: "Instructor <instructor@securelabsondemand.com>" not changed
gpg: key 18F80615: "Instructor2a <instructor@secureondemandlabs.com>" not change
d
gpg: key FE33B38A: public key "instructor_tc <instructor@secureondemandlabs.com>
" imported
```

Figure 41: importing the instructor key onto the student machine.

```
pub 2048R/C22869A4 2024-02-04
uid          t&c part2 <student@securelabsondemand.com>
sub 2048R/15EF33ED 2024-02-04

pub 2048R/FE33B38A 2024-02-04
uid          instructor_tc <instructor@secureondemandlabs.com>
sub 2048R/5FAOE0BA 2024-02-04
```

Figure 42: checking to make sure it was imported.

Finally, I encrypt the message, transfer the document to the instructor machine, and then decrypt the message.



```

student@TargetLinux01:~/Documents$ gpg -e Send.txt
You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID. End with an empty line: Instructor_tc
gpg: 5FA0E0BA: There is no assurance this key belongs to the named user

pub 2048R/5FA0E0BA 2024-02-04 instructor_tc <instructor@secureondemandlabs.com>
Primary key fingerprint: 2FC4 AFE3 FD48 3B4B 2E89 9E6F F917 C3CA FE33 B38A
Subkey fingerprint: 41F5 1500 ADBD 27CC 2CB2 AEF5 F8F9 74EF 5FA0 E0BA

It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y

Current recipients:
2048R/5FA0E0BA 2024-02-04 "instructor_tc <instructor@secureondemandlabs.com>"

```

Figure 43: encrypting the .txt file.

/home/Instructor				
Name	Size	Changed	Rights	Owner
..		3/27/2017 1:40:10 PM	rw-r-xr-x	root
Videos		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Templates		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Public		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Pictures		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Music		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Downloads		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Documents		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Desktop		3/27/2017 2:09:21 PM	rw-r-xr-x	Instruc...
Send.txt.gpg	1 KB	2/4/2024 12:09:34 PM	rw-r--r--	Instruc...

Figure 44: copying the file to the instructor machine and changing permissions.

```

Instructor@TargetLinux02:~$ gpg -d Send.txt.gpg

You need a passphrase to unlock the secret key for
user: "instructor_tc <instructor@secureondemandlabs.com>"
2048-bit RSA key, ID 5FA0E0BA, created 2024-02-04 (main key ID FE33B38A)

gpg: encrypted with 2048-bit RSA key, ID 5FA0E0BA, created 2024-02-04
      "instructor_tc <instructor@secureondemandlabs.com>"
My name is Javonnie Rutherford
Instructor@TargetLinux02:~$

```

Figure 45: decrypting the file to verify its contents.

### Part 3: Challenge Exercise

For this part, I will be doing the same exercise, but with a new encryption that I don't know the syntax of.

First thing first, like any normal person, I looked up “aes256 encryption with gpg” and got this amazing answer from Copilot.

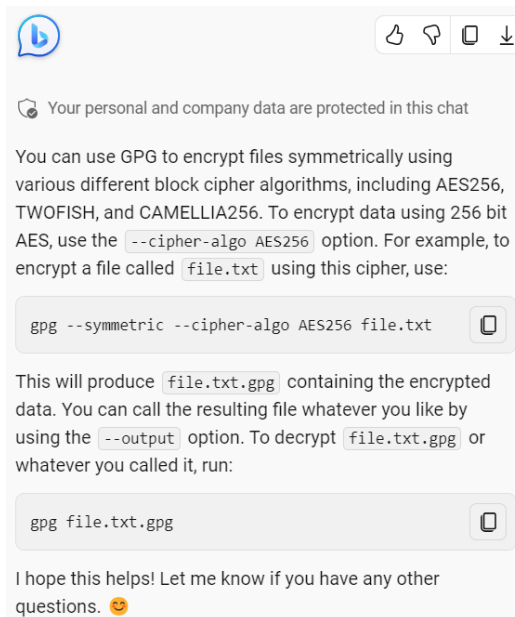


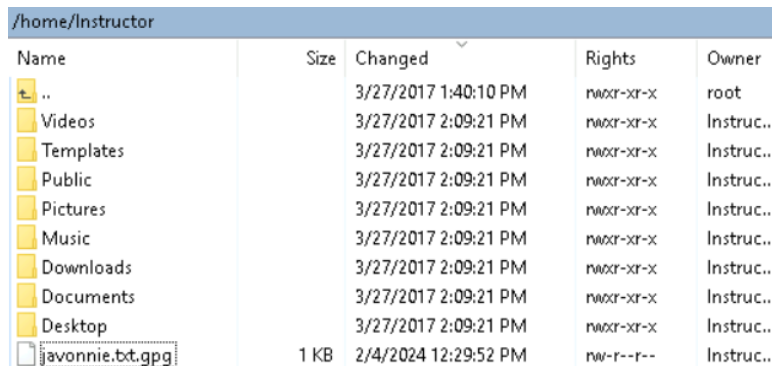
Figure 46: where I found the syntax.

Next, I tested it out and to my surprise it worked (my Copilot experience is usually it's off just by a hair, but so far so good).

```
student@TargetLinux01:~/Documents$ echo "This is a test of AES256 encryption" > javonnie.txt
student@TargetLinux01:~/Documents$ gpg --symmetric --cipher-algo AES256 javonnie.txt
student@TargetLinux01:~/Documents$ ls
clear.txt          Example2.txt.sha1  instructor2.pub    Send.txt.gpg
clear.txt2.txt.gpg Example2.txt.sha256 instructor.pub      send.txt.md5
clear.txt.txt      Example.txt         instructor.tc.pub  student2.pub
clear.txt.txt.gpg  Example.txt.md5    javonnie.txt      student.pub
Example2.txt       Example.txt.sha1   javonnie.txt.gpg  studentt
Example2.txt.md5   instructor2a.pub   Send.txt          student_tc.pub
student@TargetLinux01:~/Documents$ cat javonnie.txt.gpg
-----BEGIN PGP MESSAGE-----
Version: 1.0
mQIwHQAQpI H
GfUQ-Ir[sof# , \S3r<U Kgtl^XU,0cstudent@
```

Figure 47: encrypting the file.

After, I went ahead and transferred the file from the student machine to the instructors.



Name	Size	Changed	Rights	Owner
..		3/27/2017 1:40:10 PM	rw-r--r--	root
Videos		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Templates		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Public		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Pictures		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Music		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Downloads		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Documents		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
Desktop		3/27/2017 2:09:21 PM	rw-r--r--	Instruc...
javonnie.txt.gpg	1 KB	2/4/2024 12:29:52 PM	rw-r--r--	Instruc...

Figure 48: copying the file and changing permissions.

Finally, I decrypted the file and outputted the next (I repeated it twice, hence the overwrite, because I got confused on why it didn't immediately output).

```
Instructor@TargetLinux02:~$ gpg javonnie.txt.gpg
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
File 'javonnie.txt' exists. Overwrite? (y/N) y
Instructor@TargetLinux02:~$ cat javonnie.txt
This is a test of AES256 encryption
Instructor@TargetLinux02:~$
```

Figure 49: decrypting the file.

## Conclusion

I thought this lab was amazing and incredibly thought provoking. Encryption is one of those areas (like Networking for me, but vastly more confusing) where you're expected to have at least a decent idea of how it works, so being able to experiment and witness it firsthand was an amazing experience. My one "gripe" was how repetitive it was, especially at the end – when I saw that the Challenge Activities was doing it all over again, I wanted to die – however, I was pleasantly surprised at how straight forward the AE256 encryption process was. Additionally, because I got to basically do the md5 how I wanted on the challenge activity, I got to skip a few redundant steps, such as modifying the hash. Overall, this was an amazing lab and really reinforced a niche field that, admittedly, more cybersecurity professionals should be aware of. Thank you so much for the opportunity and I'm super excited for the next lab (configuring firewalls is DEFINITELY going on the resume!).