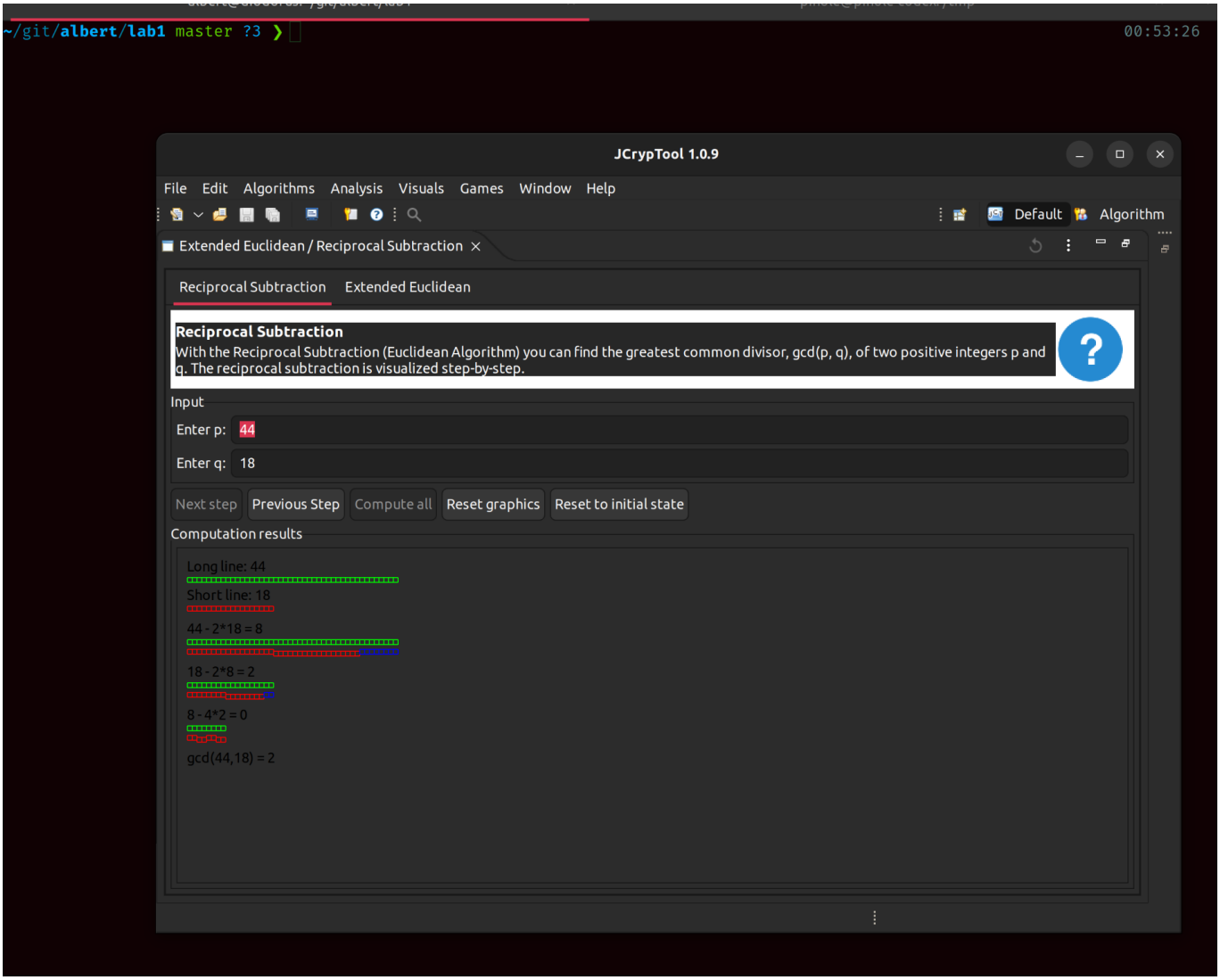
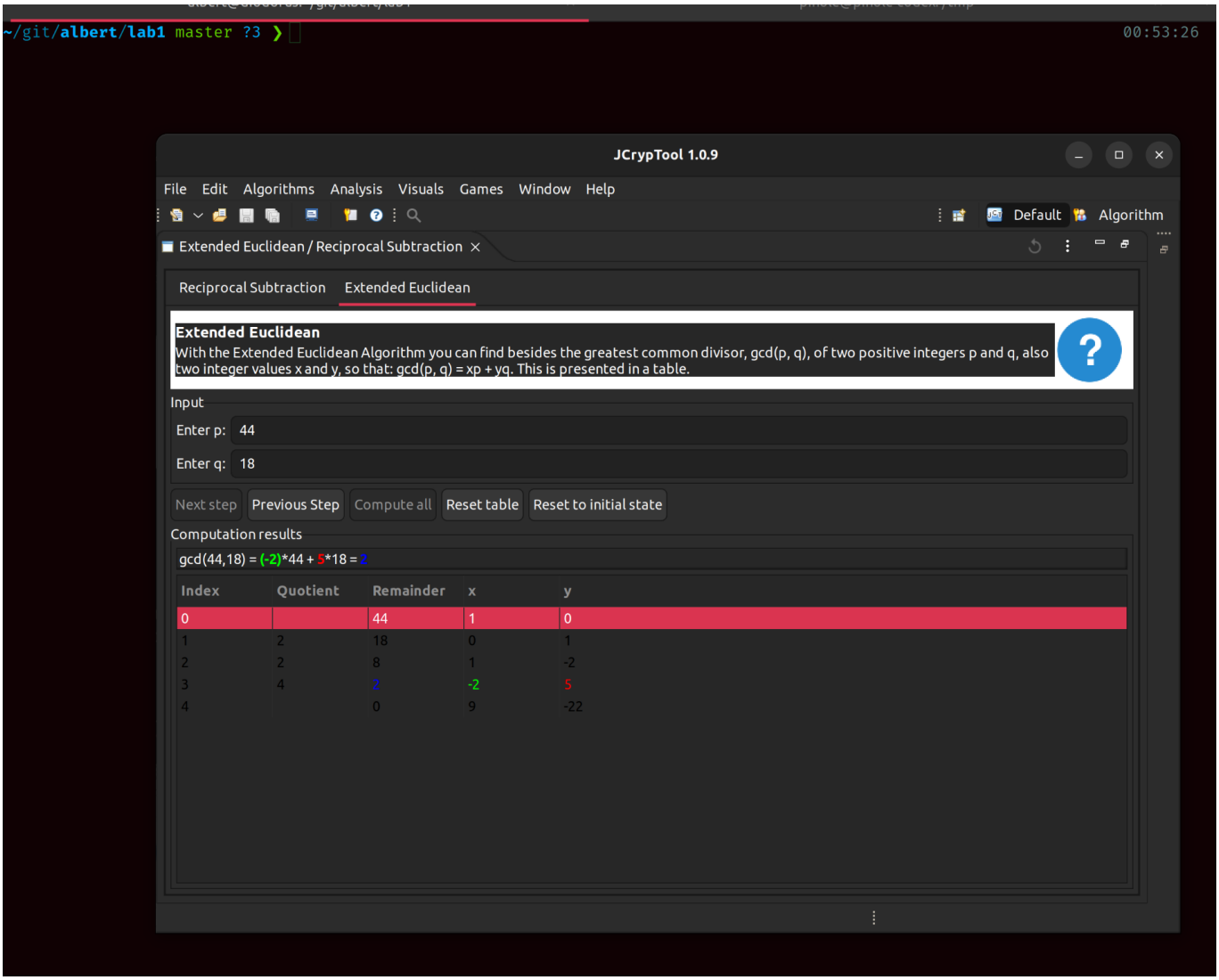
# Cryptography Lab 1

**Name:** Albert Ferguson **SID:** 13611165

## 1. Euclidian Algorithm




## 2. Extended Euclidian Algorithm




# 3. GnuPG / PGP

## Preliminaries

I'm using a debian distro and already have the dep's installed. To be certain though, I first run the recommended installs,

```
sudo apt-get install gnupg rng-tools
```

Which just tells me I already them. Moving on...

## 3.1 Quick check / practice

```
gpg --gen-key
```

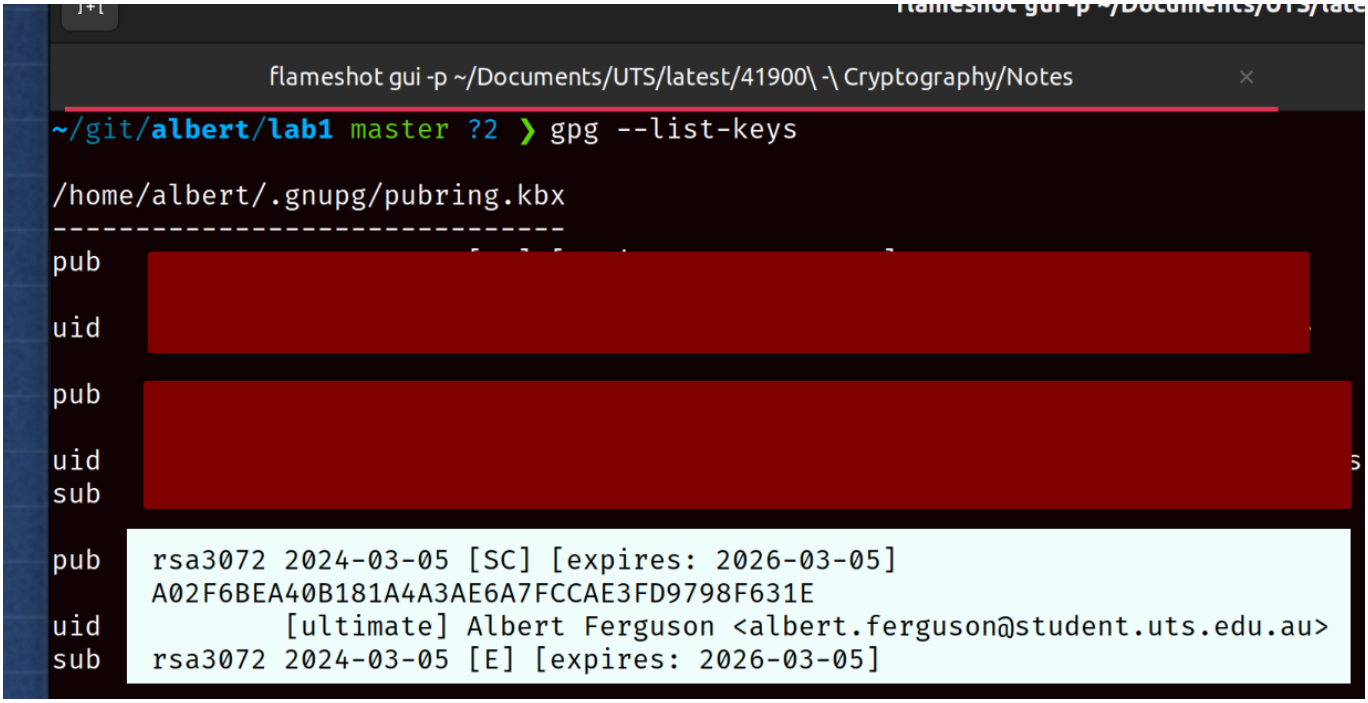
This runs a convenient interactive with the output like so,

```
gpg (GnuPG) 2.2.27; Copyright (C) 2021 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.

Note: Use "gpg --full-generate-key" for a full featured key generation
dialog.

You need a user ID to identify your key; the software constructs the user
ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Albert Ferguson
E-mail address: albert.ferguson@student.uts.edu.au
You selected this USER-ID:
    "Albert Ferguson <albert.ferguson@student.uts.edu.au>"

# excluding some filler on the entropy generation and directory creation

public and secret key created and signed.
```

I set the password to `ABC123` (nice an 'secure' so the tool warns me to change it - I do not since this is an example).

Checking the keys on my system yields the one I just created, as well as some existing keys which I've omitted,

```
gpg --list-keys
```




**Questions:**

> "Why is it necessary to sign keys? Can anyone create a key and pretend to be another person? Can you think of a way to make sure that a given key really belongs to the person listed on the key? What do you think are the benefits of signing keys?"

**Answers:**

> Why is it necessary to sign keys?

Signing keys lets any third-party understand that the signer trusts the signee.

> Can anyone create a key and pretend to be another person?

No, to pretend to be another person you would have to reproduce the same key as the target. Given the underlying key generation is random^, it is unconditionally secure and not possible to pretend to be another person.

^ *this relies the entropy of the host which is not truly ranbom - what may be known as "high entropy"*

> Can you think of a way to make sure that a given key really belongs to the person listed on the key?

Assuming that "key" here refers to the pubkey - the private key would render this mute. No, not directly. Funnily enough, gpg is kind enough to warn of this scenario when using an imported pubkey. Typically, to "prove" ownership something akin to a "certificate authority" (as with X.509 certs) or a "trust chain" would be needed to help prove ownership.

> What do you think are the benefits of signing keys?

Similar to the first question - simply put this allows the participating parties to build trust. Generally speaking, signing keys creates trust because it enables, authentication, data integrity, and non-repudiation. That last one is quite important, as users' cannot claim that they did not actually send some data.

## 3.2 Encrypt with GPG using public key

```
~/git/albert/lab1 master ?3 ❯ rm my-secrets-myname.txt                                                    00:10:57
rm: remove write-protected regular empty file 'my-secrets-myname.txt'? y
~/git/albert/lab1 master ?2 ❯ umask 0                                                                     00:11:03
~/git/albert/lab1 master ?2 ❯ touch my-secrets-myname.txt                                                 00:11:22
~/git/albert/lab1 master ?3 ❯ echo "My credit card number is 1234-5678-9012-3456" > my-secrets-myname.txt  00:11:26
~/git/albert/lab1 master ?3 ❯ cat my-secrets-myname.txt                                                   00:11:27
My credit card number is 1234-5678-9012-3456
~/git/albert/lab1 master ?3 ❯ gpg -e my-secrets-myname.txt                                                00:11:31
You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID.  End with an empty line: albert.ferguson@student.uts.edu.au

Current recipients:
rsa3072/33701764A7068089 2024-03-05 "Albert Ferguson <albert.ferguson@student.uts.edu.au>"

Enter the user ID.  End with an empty line:
~/git/albert/lab1 master ?3 ❯ ll                                                                      21s 00:12:04
total 8.0K
-rw-rw-rw- 1 albert albert  45 Mar  6 00:11 my-secrets-myname.txt
-rw-rw-rw- 1 albert albert 525 Mar  6 00:12 my-secrets-myname.txt.gpg
~/git/albert/lab1 master ?3 ❯ cat my-secrets-myname.txt.gpg                                               00:12:05
◆◆3pd◆◆◆
        ◆*◆6z◆
6◆◆◆◆◆/◆◆kH◆◆◆◆◆◆◆      f◆◆◆q^ZK◆J◆g◆5◆y◆S◆v˘◆Kg◆◆◆CI◆◆AEt◆◆d◆◆◆b◆◆T:◆eM◆9HR◆◆
  ◆◆*◆◆#|ن!:◆◆◆'◆◆\◆;&◆
◆◆◆zC◆◆"Db{qBm◆◆
                ◆◆◆◆(}◆aMr~-◆◆9◆◆◆◆◆HC◆%◆◆◆◆e◆◆◆`◆ر"S◆Om◆{P◆
                                                        Gp◆S5$◆~◆◆◆n◆řR◆B◆{g}Wl◆◆4kgfk◆◆/◆@◆d`S◆◆(◆T◆◆◆◆H:◆N◆◆9◆F◆d>we◆◆◆
4#◆n9`◆6-◆◆,
```

A picture tells a thousand words, ignore my umask command (recent backup had changed this). Inputs and outputs are as expected. Final output shows both the input and output files in my current directory.

Also encrypting with ASCII encoding, which is better for distribution, with,

```
~/git/albert/lab1 master ?3 ❯ gpg -aer albert.ferguson@student.uts.edu.au my-secrets-myname.txt          1m 10s 00:14:1
~/git/albert/lab1 master ?3 ❯ ll                                                                              00:16:5
total 12K
-rw-rw-rw- 1 albert albert  45 Mar  6 00:11 my-secrets-myname.txt
-rw-rw-rw- 1 albert albert 772 Mar  6 00:16 my-secrets-myname.txt.asc
-rw-rw-rw- 1 albert albert 525 Mar  6 00:12 my-secrets-myname.txt.gpg
~/git/albert/lab1 master ?3 ❯ cat my-secrets-myname.txt.asc                                                   00:16:5
-----BEGIN PGP MESSAGE-----

hQGMAzNwF2SnBoCJAQv/awD0lEr5ciAPQpuwBsDjALlsLZ6sXs+IE6vQXg3VXXNz
Q8f6pUBKjDMIiFctvVDDX3plPpm71FFTX2StmT+YzjgL/QizJnh3euzlAJ/o/STK
f9iac0CpvqYiQ+kH+hseFOd2mQ9d/vRZ/svH6gjfY53XXLNVMfrgqMSqxPueqkxK
/Un7hlPze7tx6/WAWFGZZDPMzltFPSvxuY8Q8FkxWALjzT2VAsZf9I749iztJRkS
Wc3+6JvukvyowTKILOSZsWbrl3k4AKknk6CgChZD+jDFNXiq2Bl6of/94qYpV788
s4/XF2vMw9jBhKcPyX0m2hTycloAS4ZYYqBUlV7mjIiKxTRmTJPDbsckIqCN65Mj
ldy2AP29qL2zAv6a1nZtfkVjKDk8Bw4use6TesWFvjDPOGx72NSUhhF1PXvfgxGL
5Yifa3NWH4jEGPPykgWJqc8ChjZz0NeLbjh/EMBSaS6lG2gCve7+VpBnlRYeob3Z
jFR6PF+NdsVT3umZ89yb0nwBg+eJSra0OLSVF3OtmGkSLT8XpGzfu9RCMFRcnnjA
UQfb7p/bINT2RPbVwHkCR7zpWIW5sQo53chhm13zR5AhFHJQ5zuloO8zEe8FV7ZC
n0XQoSdB7ByTILoXt4VfRip5QfWfVHG2rpZoinjiMOfUU+cDx8WtglZRvlr7
=5YMg
-----END PGP MESSAGE-----
```

### 3.3 Decrypting files

```
~/git/albert/lab1 master ?3 ❯ gpg my-secrets-myname.txt.asc                                                   31s 00:19:07
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: encrypted with 3072-bit RSA key, ID 33701764A7068089, created 2024-03-05
      "Albert Ferguson <albert.ferguson@student.uts.edu.au>"
File 'my-secrets-myname.txt' exists. Overwrite? (y/N) y
~/git/albert/lab1 master ?3 ❯ cat my-secrets-myname.txt                                                        8s 00:20:19
My credit card number is 1234-5678-9012-3456
~/git/albert/lab1 master ?3 ❯ gpg -e my-secrets-myname.txt.gpg                                                   00:20:22
You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID.  End with an empty line: albert.ferguson@student.uts.edu.au

Current recipients:
rsa3072/33701764A7068089 2024-03-05 "Albert Ferguson <albert.ferguson@student.uts.edu.au>"

Enter the user ID.  End with an empty line:
gpg: signal 2 caught ... exiting

~/git/albert/lab1 master ?3 ❯ gpg my-secrets-myname.txt.gpg                                          ✘ INT 13s 00:20:50
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: encrypted with 3072-bit RSA key, ID 33701764A7068089, created 2024-03-05
      "Albert Ferguson <albert.ferguson@student.uts.edu.au>"
File 'my-secrets-myname.txt' exists. Overwrite? (y/N) y
~/git/albert/lab1 master ?3 ❯ cat my-secrets-myname.txt                                                          00:20:58
My credit card number is 1234-5678-9012-3456
```

Ignore my typo in the middle, this shows decrypting both options.

## 4. Distributing and trusting keys

```
~/git/albert/lab1 master ?3 ❯ gpg --export -a --output albertferguson-key.asc albert.ferguson@student.uts.edu.au
~/git/albert/lab1 master ?3 ❯ ls
albertferguson-key.asc  my-secrets-myname.txt  my-secrets-myname.txt.asc  my-secrets-myname.txt.gpg
~/git/albert/lab1 master ?3 ❯ less albertferguson-key.asc
~/git/albert/lab1 master ?3 ❯ cat albertferguson-key.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGXnCX4BDADlCu3JgvC9CTZV/AaD7uHm2N/RkO330AGF7LKn9FIER7UeeptU
dVp/9aA3TKcyzwMMUl+hU3ZyuLDRTFYoj5pPtPQJ8/zq7Cx/jzkwSR1BckaYvsjq
2FZUh0GqfWyqxAmwZyF18qjDKshRO5tZYY82xa6mCRY2XQIKUL0rDvgp+t9fN2Ak
7VamnLmgjLgA+HDAb6rSn5NC7SfeRiwAD/TZW+f/iM+TlaiLng1ywIQF0JXueazq
HRENX911Id0Ts6SwbUkIxzTyXj4VNS+Q5EiMyGr5nc6WBfIMnqZ+O5DDqWfLYe1C
Th1RcNGgf7e/QDPZJ7vaUMa6rqTi9bLC1quS8/fizD2fIzLsiaAp5f7gDUmZkDJD
tF1xxqOcj8SWegvq3U6lPN6MWCUPP7DPhgI9aQ3FRuocdT0b5InqdQtva1e8mdZ3
HS2O/J3gSZob8K3dM8EAMVSBdMj7FX5mi8MTnYZx4b5RFfga7cClXJOMjUp7cXW+
AJJfuqJrwNQbhe8AEQEAAbQ0QWxiZXJ0IEZlcmd1c29uIDxhbGJlcnQuZmVyZ3Vz
b25Ac3R1ZGVudC51dHMuZWR1LmF1PokB1AQTAQoAPhYhBKAva+pAsYGko65qf8yu
P9l5j2MeBQJl5wl+AhsDBQkDwmcABQsJCAcCBhUKCQgLAgQWAgMBAh4BAheAAAoJ
```

Above, setup to distribute the key with another host.

## 4.1 Exporting you public key with GPG

With the exported key ready to distribute. This key is in plaintext and can potentially be intercepted. Since this is a pubkey, I am not fussed about what channel it is exposed over. That is rather the point anyway.

## 4.2 Exchanging keys

To demo the process between to hosts, I will use a raspberry pi on my network that I have remote using `ssh` and `scp`. Using my existing ssh configuration I remote into the host (`pihole`) and create a temporary directory and on my host I then transfer the pubkey using `scp`.

Having set that all up, I ssh over to the pihole and import the pubkey

```
pihole@pihole-codex:/tmp/lab-test $ ls -l
total 4
-rw-r--r-- 1 pihole pihole 2480 Mar  6 00:36 albertferguson-key.asc
pihole@pihole-codex:/tmp/lab-test $ gpg --import albertferguson-key.asc
gpg: key CCAE3FD9798F631E: public key "Albert Ferguson <albert.ferguson@student.uts.edu.au>" imported
gpg: Total number processed: 1
gpg:               imported: 1
pihole@pihole-codex:/tmp/lab-test $ gpg --list-keys
/home/pihole/.gnupg/pubring.kbx
-------------------------------
pub   rsa3072 2024-03-05 [SC] [expires: 2026-03-05]
      A02F6BEA40B181A4A3AE6A7FCCAE3FD9798F631E
uid           [ unknown] Albert Ferguson <albert.ferguson@student.uts.edu.au>
sub   rsa3072 2024-03-05 [E] [expires: 2026-03-05]

pihole@pihole-codex:/tmp/lab-test $
```

## 4.3 Encrypting a file for my host to read (from the pihole)

*To simplify things, I will skip setting up a second key and transferring it back to my host. No need to duplicate the example.*

On the pihole, I use the imported pubkey to encrypt a file. It warns me of my serious transgressions, which I choose to ignore for this demonstration. I then have the encrypted file, which I cannot decrypt.

```
pihole@pihole-codex:/tmp/lab-test $ cat some-secrets.txt
don't leak my special password 321GOverysecure
pihole@pihole-codex:/tmp/lab-test $ gpg -aer albert.ferguson@student.uts.edu.au some-secrets.txt
gpg: 33701764A7068089: There is no assurance this key belongs to the named user

sub  rsa3072/33701764A7068089 2024-03-05 Albert Ferguson <albert.ferguson@student.uts.edu.au>
 Primary key fingerprint: A02F 6BEA 40B1 81A4 A3AE  6A7F CCAE 3FD9 798F 631E
      Subkey fingerprint: 97F9 4F23 437E 32B9 B67D  68A5 3370 1764 A706 8089

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
pihole@pihole-codex:/tmp/lab-test $ ls
albertferguson-key.asc  some-secrets.txt  some-secrets.txt.asc
pihole@pihole-codex:/tmp/lab-test $ cat some-secrets.txt
don't leak my special password 321GOverysecure
pihole@pihole-codex:/tmp/lab-test $ cat some-secrets.txt.asc
-----BEGIN PGP MESSAGE-----

hQGMAzNwF2SnBoCJAQwAl2LFCFfeZgu31mQSfJTtFUjtyKyoBwVoi9xmyyaXnCJz
5J1XjgNZ7EQ9JyKibDyafLj6ylJYvxy9A/r0zkL2L4VAQsB2KunanlbY4vQavntT
18gQWOCPPPrGQ8HL6YnQcOpwtHiP4zjDRkBh3DkJhw5lt9jNotGs9TLmfSyUdnD1
5nvSrmBtJGQQZ/4QkKN+G8zXuj+QSrY/1GgLoGjNZusI/03qZEffpX+pCNmlHlzg
DlkNwZO3LPZUjAQnouO8C7Zr0pln56HmDsvPhjG0+AR2JVO/sMd2RLbU1zIYqrh1
usN/kx6uwDbWcfyHUUYqUA81+pVyMEXR4Hudbvw76mFjOkgYXUcR9YH7VU1RFXQo
h8aRiwRxC96F6PW9Fwq0CFCJuSHm4pBimjR5B+gaXagrXCBr/fe1lGRWRu4PagUb
fXTOCYsJWAGJx9C/7BoUiCAuPJ//lDSWYlJHSaliXTY6fF2OKes625ZpL71re+/E
+gs2ikIwovfJeAa63CS60nkBYleF46AWSTh+Xdp3ZgIWz65GTFqd6LThw4yAbMpJ
0FgStsjqN9HKUUSW1SJ1j/RkgXb4uiZgNJc+3qbrQ31G9Db003q3lTOKrM14Li+4
/0PjoLdn+wILRvNZo5FONSlDdCtf8lGr7ICqBnAteLssYUMGSjbxAvbl
=Ba7t
-----END PGP MESSAGE-----
pihole@pihole-codex:/tmp/lab-test $
```

Back on my host, I copy the encrypted file back to my machine with `scp` and decrypt it as before,

```
~/git/albert/lab1 master ?3 ❯ gpg some-secrets.txt.asc
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: encrypted with 3072-bit RSA key, ID 33701764A7068089, created 2024-03-05
      "Albert Ferguson <albert.ferguson@student.uts.edu.au>"
~/git/albert/lab1 master ?3 ❯ ls
albertferguson-key.asc  my-secrets-myname.txt.asc  some-secrets.txt
my-secrets-myname.txt   my-secrets-myname.txt.gpg  some-secrets.txt.asc
~/git/albert/lab1 master ?3 ❯ cat some-secrets.txt
don't leak my special password 321GOverysecure
~/git/albert/lab1 master ?3 ❯
```

## 5. Protecting Emails with PGP

**Questions:**

> "Try the following cases and summarise your observation,"

1. Select a wrong sender's private key. You may need to generate new private keys with different email addresses. You can remove a private key by running `gpg –delete-secret-key email@email.email`

2. The receiver has not imported the sender public key. You can remove an imported public key by running `gpg –delete-key email@email.email`

> "There are two operations in PGP, i.e., PGP sign and PGP Encrypt. What is the difference between them? You may search online."

**Answers:**

I'm pressed for time to install and configure the email client, I will attempt to answer these questions without having done so.

> Case 1.

By selecting the wrong sender's privkey the receiver whom has already trusted the original pubkey, will not be able to verify the signature (their key will not match).

> Case 2.

By not importing the pubkey, this is similar to case 1. This time, the receiver will not have the trusted pubkey in-memory for validation (missing key).

> There are two operations in PGP, i.e., PGP sign and PGP Encrypt. What is the difference between them? You may search online.

This refers to the difference between sending a message to a known recipient (encrypting) and allowing an arbitrary recipient with the pubkey to validate a known sender.

In the first scenario, the sender creates a message encrypted with the recipient's pubkey. The recipient then decrypts it with their privkey. In the second scenario, the senders privkey signs the message and their pubkey is somehow distributed publicly. The arbitrary receipient may then verify the original message using the (hopefully) trusted pubkey.