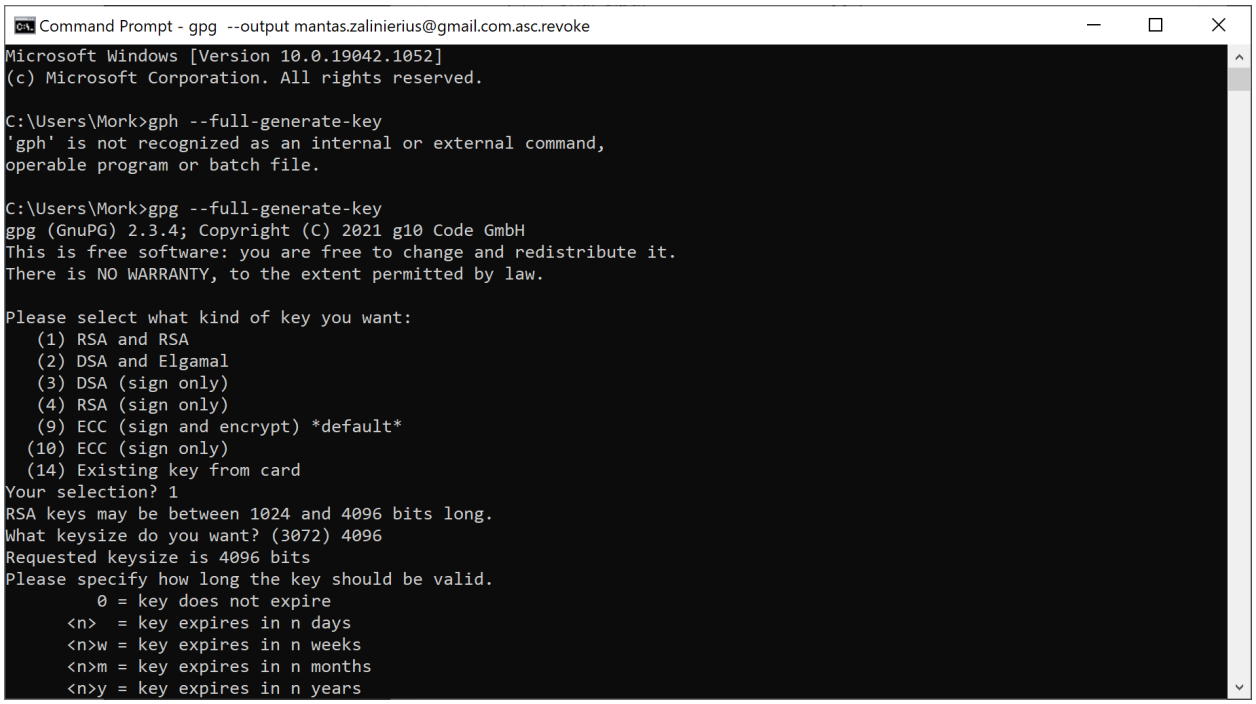
Lab 10

Part 1

STEPS 1 - 3

1. First step is to generate a key.
2. Second step is to choose what type of key I want, in this case I chose “RSA and RSA”
3. The third step details how long in bits you want your key to be. I chose 4096 bits.



```
Command Prompt - gpg  --output mantas.zalinierius@gmail.com.asc.revoke
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mork>gph --full-generate-key
'gph' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Mork>gpg --full-generate-key
gpg (GnuPG) 2.3.4; Copyright (C) 2021 g10 Code GmbH
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
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (9) ECC (sign and encrypt) *default*
  (10) ECC (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
```

STEPS 4-5

4. Fourth step is to Choose how long your key will last. For this I decide on 1 year.
5. Fifth step is to enter in your name and email you want associated with your account. For this I put in “Mantas” for the name and “mantas.zalinierius@gmail.com” for the email.

Command Prompt - gpg --output mantas.zalinierius@gmail.com.asc.revoke

```
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 1y
Key expires at 09/03/2023 10:53:46 GMT Standard Time
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Mantas
Email address: mantas.zalinierius@gmail.com
Comment:
You selected this USER-ID:
    "Mantas <mantas.zalinierius@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
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.
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: directory 'C:\\Users\\Mork\\AppData\\Roaming\\gnupg\\openpgp-revocs.d' created
gpg: revocation certificate stored as 'C:\\Users\\Mork\\AppData\\Roaming\\gnupg\\openpgp-revocs.d\\4423F029C530C1369ABBF
E31E8C23453B9CDAF37.rev'
public and secret key created and signed.

pub   rsa4096 2022-03-09 [SC] [expires: 2023-03-09]
```

STEPS 6

6. The sixth step is to see if the key is is visible via the command “gpg --list-keys”

```
Command Prompt - gpg  --output mantas.zalinierius@gmail.com.asc.revoke
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
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: directory 'C:\\Users\\Mork\\AppData\\Roaming\\gnupg\\openpgp-revocs.d' created
gpg: revocation certificate stored as 'C:\\Users\\Mork\\AppData\\Roaming\\gnupg\\openpgp-revocs.d\\4423F029C530C1369ABBF
E31E8C23453B9CDAF37.rev'
public and secret key created and signed.

pub   rsa4096 2022-03-09 [SC] [expires: 2023-03-09]
      4423F029C530C1369ABBFE31E8C23453B9CDAF37
uid                      Mantas <mantas.zalinierius@gmail.com>
sub   rsa4096 2022-03-09 [E] [expires: 2023-03-09]


C:\Users\Mork>gpg --list-keys
C:\Users\Mork\AppData\Roaming\gnupg\pubring.kbx
-------------------------------------------------
pub   rsa4096 2022-03-09 [SC] [expires: 2023-03-09]
      4423F029C530C1369ABBFE31E8C23453B9CDAF37
uid           [ultimate] Mantas <mantas.zalinierius@gmail.com>
sub   rsa4096 2022-03-09 [E] [expires: 2023-03-09]


C:\Users\Mork>gpg --output mantas.zalinierius@gmail.com.asc.revoke
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
```

STEPS 7

7. The seventh step is to create a revocation certificate.

```
Command Prompt
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mork>gpg --output mantas.zalinierius@gmail.com.asc.revoke --gen-revoke mantas.zalinierius@gmail.com

sec  rsa4096/E8C23453B9CDAF37 2022-03-09 Mantas <mantas.zalinierius@gmail.com>

Create a revocation certificate for this key? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision? 0
Enter an optional description; end it with an empty line:
>
Reason for revocation: No reason specified
(No description given)
Is this okay? (y/N) y
ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
```

STEPS 8-9

8. The eight step is to make the key public that you have created.
9. The ninth step is to import a public key from another person. For this I used a key provided by lei.

```
Command Prompt
gpg: ^C
C:\Users\Mork>gpg --output pubkey.mantas.zalinierius@gmail.com.gpg --export mantas.zalinierius@gmail.com

C:\Users\Mork>gpg --output pubkey.mantas.zalinierius@gmail.com.gpg.asc --armor --export mantas.zalinierius@gmail.com

C:\Users\Mork>gpg --import pubkey.mantas.zalinierius@gmail.com.gpg.asc
gpg: key E8C23453B9CDAF37: "Mantas <mantas.zalinierius@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1

C:\Users\Mork>gpg --import pubkey.C00236772@itcarlow.ie.gpg.asc
gpg: can't open 'pubkey.C00236772@itcarlow.ie.gpg.asc': No such file or directory
gpg: Total number processed: 0

C:\Users\Mork>gpg --import pubkey.robustness@gmail.com.gpg.asc
gpg: can't open 'pubkey.robustness@gmail.com.gpg.asc': No such file or directory
gpg: Total number processed: 0

C:\Users\Mork>gpg --keyserver=pgp.mit.edu --recv-key EC2392F2EDE74488680DA3CF5F2B4756ED873D23
gpg: key 5F2B4756ED873D23: public key "Alan Eliasen <eliasen@mindspring.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1

C:\Users\Mork>gpg --list-keys
C:\Users\Mork\AppData\Roaming\gnupg\pubring.kbx
---------------------------------------------
pub   rsa4096 2022-03-09 [SC] [expires: 2023-03-09]
      4423F029C530C1369ABBFE31E8C23453B9CDAF37
uid           [ultimate] Mantas <mantas.zalinierius@gmail.com>
sub   rsa4096 2022-03-09 [E] [expires: 2023-03-09]
```

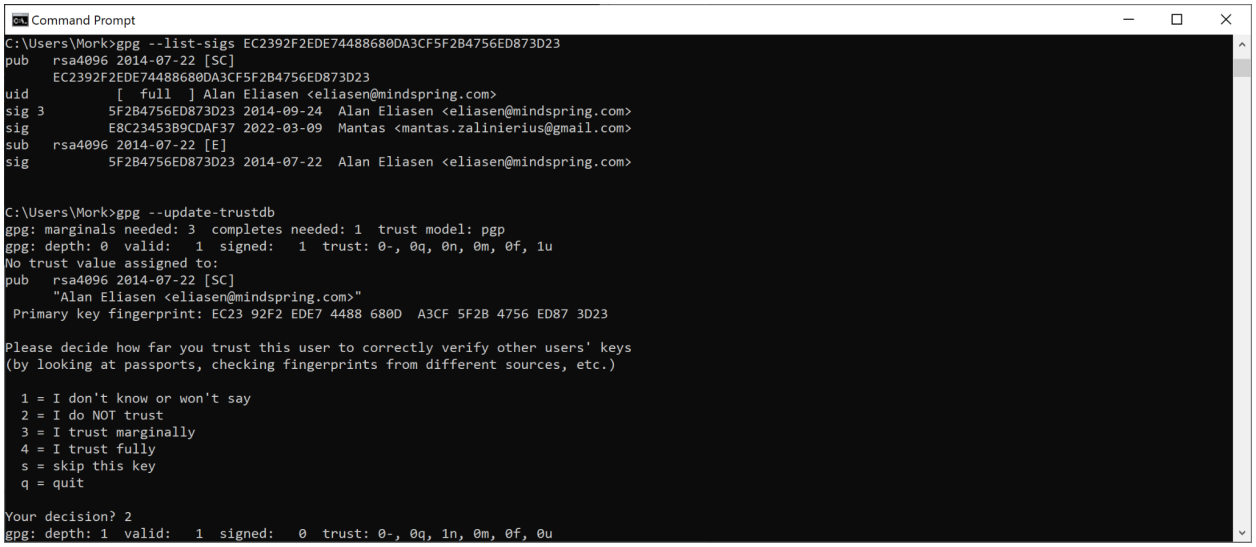
STEPS 10-11

10. The tenth step is to get a new list of keys, now there should be two keys, one you created and an imported one.
11. The eleventh step is to sign the imported key.

```
Command Prompt

C:\Users\Mork>LONG --list-keys
'LONG' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Mork>gpg --keyid-format LONG --list-keys
C:\Users\Mork\AppData\Roaming\gnupg\pubring.kbx
---------------------------------------------
pub   rsa4096/E8C23453B9CDAF37 2022-03-09 [SC] [expires: 2023-03-09]
      4423F029C530C1369ABBFE31E8C23453B9CDAF37
uid                 [ultimate] Mantas <mantas.zalinierius@gmail.com>
sub   rsa4096/8FCAA025019248CD 2022-03-09 [E] [expires: 2023-03-09]

pub   rsa4096/5F2B4756ED873D23 2014-07-22 [SC]
      EC2392F2EDE74488680DA3CF5F2B4756ED873D23
uid                 [ unknown] Alan Eliasen <eliasen@mindspring.com>
sub   rsa4096/11C2E18C5314E70B 2014-07-22 [E]


C:\Users\Mork>gpg --sign-key EC2392F2EDE74488680DA3CF5F2B4756ED873D23

pub  rsa4096/5F2B4756ED873D23
     created: 2014-07-22  expires: never       usage: SC
     trust: unknown       validity: unknown
sub  rsa4096/11C2E18C5314E70B
     created: 2014-07-22  expires: never       usage: E
[ unknown] (1). Alan Eliasen <eliasen@mindspring.com>


pub  rsa4096/5F2B4756ED873D23
```

STEPS 12-13

12. The twelfth step is to check the signatures for the imported public key. There should be two owners and mine.
13. The thirteenth step is to update your trust.



```
Command Prompt
C:\Users\Mork>gpg --list-sigs EC2392F2EDE74488680DA3CF5F2B4756ED873D23
pub   rsa4096 2014-07-22 [SC]
      EC2392F2EDE74488680DA3CF5F2B4756ED873D23
uid           [  full  ] Alan Eliasen <eliasen@mindspring.com>
sig 3        5F2B4756ED873D23 2014-09-24  Alan Eliasen <eliasen@mindspring.com>
sig          E8C23453B9CDAF37 2022-03-09  Mantas <mantas.zalinierius@gmail.com>
sub   rsa4096 2014-07-22 [E]
sig          5F2B4756ED873D23 2014-07-22  Alan Eliasen <eliasen@mindspring.com>


C:\Users\Mork>gpg --update-trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 1u
No trust value assigned to:
pub   rsa4096 2014-07-22 [SC]
      "Alan Eliasen <eliasen@mindspring.com>"
 Primary key fingerprint: EC23 92F2 EDE7 4488 680D  A3CF 5F2B 4756 ED87 3D23

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  s = skip this key
  q = quit

Your decision? 2
gpg: depth: 1  valid:   1  signed:   0  trust: 0-, 0q, 1n, 0m, 0f, 0u
```

STEPS 14-16

14. The fourteenth step is to check the signatures for the imported public key. There should be two owners and mine.
15. The thirteenth step is to update your trust.

```
Command Prompt
C:\Users\Mork\Documents>gpg --output message.txt --armor --encrypt --sign --recipient 4423F029C530C1369ABBFE31E8C23453B9CDAF37 secret-message.txt

C:\Users\Mork\Documents>gpg --decrypt secret-message.txt
gpg: decrypt_message failed: Unknown system error

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
gpg: Signature made 09/03/2022 11:20:11 GMT Standard Time
gpg:                using RSA key 4423F029C530C1369ABBFE31E8C23453B9CDAF37
gpg: Good signature from "Mantas <mantas.zalinierius@gmail.com>" [ultimate]

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
gpg: Signature made 09/03/2022 11:20:11 GMT Standard Time
gpg:                using RSA key 4423F029C530C1369ABBFE31E8C23453B9CDAF37
gpg: Good signature from "Mantas <mantas.zalinierius@gmail.com>" [ultimate]

C:\Users\Mork\Documents>gpg --output message.txt --armor --encrypt --sign --recipient 4423F029C530C1369ABBFE31E8C23453B9CDAF37 secret-message.txt
File 'message.txt' exists. Overwrite? (y/N) y

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
Hi Bob,
The IBAN for my Swiss bank account is:
CH93 1234 4567 8910 2345 6
Please transfer the funds to that account.
Regards,
```

STEPS 16-17

16. The sixteenth step is to send a message to another person using their public key.
17. The seventh step is to have the person decrypt that message.


```
Command Prompt
C:\Users\Mork\Documents>gpg --output message.txt --armor --encrypt --sign --recipient 4423F029C530C1369ABBFE31E8C23453B9CDAF37 secret-message.txt

C:\Users\Mork\Documents>gpg --decrypt secret-message.txt
gpg: decrypt_message failed: Unknown system error

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
gpg: Signature made 09/03/2022 11:20:11 GMT Standard Time
gpg:                using RSA key 4423F029C530C1369ABBFE31E8C23453B9CDAF37
gpg: Good signature from "Mantas <mantas.zalinierius@gmail.com>" [ultimate]

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
gpg: Signature made 09/03/2022 11:20:11 GMT Standard Time
gpg:                using RSA key 4423F029C530C1369ABBFE31E8C23453B9CDAF37
gpg: Good signature from "Mantas <mantas.zalinierius@gmail.com>" [ultimate]

C:\Users\Mork\Documents>gpg --output message.txt --armor --encrypt --sign --recipient 4423F029C530C1369ABBFE31E8C23453B9CDAF37 secret-message.txt
File 'message.txt' exists. Overwrite? (y/N) y

C:\Users\Mork\Documents>gpg --decrypt message.txt
gpg: encrypted with rsa4096 key, ID 8FCAA025019248CD, created 2022-03-09
      "Mantas <mantas.zalinierius@gmail.com>"
Hi Bob,
The IBAN for my Swiss bank account is:
CH93 1234 4567 8910 2345 6
Please transfer the funds to that account.
Regards,
```


STEPS 18

18. The eighteenth step is to have the other person send a reply back to me using my public key.

```
C:\Users\Mork\Documents>gpg --output reply.txt --armor --encrypt --sign --recipient EC2392F2EDE74488680DA3CF5F2B4756ED873D23 singed-reply.txt
```

STEPS 19

19. The nineteenth step is for me to decrypt that reply and have a look at the contents of the reply.

```
C:\Users\Mork\Documents>gpg --decrypt reply.txt
gpg: encrypted with rsa4096 key, ID 11C2E18C5314E70B, created 2014-07-22
      "Alan Eliasen <eliasen@mindspring.com>"
gpg: public key decryption failed: No secret key
gpg: decryption failed: No secret key
```

Part 2

How does the RSA algorithm work?

The basic idea of RSA is that it's trying to ensure that keys are as secure as possible.
The way RSA does this is by getting two very large prime numbers **X and Y** so everyone will have a hard time figuring them out. Then it calculator the value n via **n = X * Y.** Then the next step is to calculate the totient of a number that is relatively a prime number to, where 1 is relatively prime to all numbers. **Then you select a number e, such that it is a coprime of n.** A number is only coprime if the only positive integer that divides them is 1. Then you calculate d which you get via **d = 1 % totient(n).**

Example

```
// Given the plaintext P=123, the ciphertext C is :
C = (123^17) % 3233 = 855;
// To decrypt the cypher text C:
P = (855^2753) % 3233 = 123;
```