Wroclaw

Secure Systems and Networks

Laboratory No. 3

Wiktor Lagiewka 219226

- Generate your own GPG key (and sign it)

To generate GPG key I used a command from CLI: `# gpg --full-generate-key`
When the command was executed program ask about a kind of cryptography key.

```
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

RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
```

In this section is opportunity to define the time of key expiration:

```
wiktor@wiktor-Lenovo-B51-80:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 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
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
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

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

Real name: wiktor
Email address: 219226@student.pwr.edu.pl
Comment: Task1 GPG key
You selected this USER-ID:
    "wiktor (Task1 GPG key) <219226@student.pwr.edu.pl>"

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: key 5B1B3EC166A93FE2 marked as ultimately trusted
gpg: revocation certificate stored as '/home/wiktor/.gnupg/openpgp-
revocs.d/C575D38B212F7E94D78559D55B1B3EC166A93FE2.rev'
public and secret key created and signed.

pub   rsa4096 2019-01-27 [SC]
      C575D38B212F7E94D78559D55B1B3EC166A93FE2
uid                 wiktor (Task1 GPG key) <219226@student.pwr.edu.pl>
sub   rsa4096 2019-01-27 [E]
```

In the next step I use command : gpg --edit-key wiktor to choose my previous generated GPG key and signed it with command sign.

```
gpg> sign
"wiktor.lagiewka (My first generatet key PGP) <219226@student.pwr.edu.
```

- **Export your public key to ASCII format so you can share it with other people**

  To accomplished this task I used gpg command with two option –-armor to convert to ASCII format and –-export to my email

```
wiktor@wiktor-Lenovo-B51-80:~$ gpg --armor --export 219226@student.pwr.edu.pl >mykey.asc
```

- **Import GPG keys of other users (including teacher's key)**

```
wiktor@wiktor-Lenovo-B51-80:~$ gpg --import 42391952.key.asc

gpg: key AE49624442391952: public key "Tomasz R Surmacz
<tsurmacz@pwr.wroc.pl>" imported gpg:
Total number processed: 1 gpg:
imported: 1

root@kali:~/Pulpit# gpg --import 42391952.key.asc
gpg: key 19D6931E770023F1: public key "Jan Kiepski <johnnypoor@gmail.com>" imported

gpg: Total number processed: 1 gpg:                    imported: 1
```

- **Sign a message with your GPG key**

```
wiktor@wiktor-Lenovo-B51-80:~$ gpg --clearsign message.txt >message.txt.asc
```

–clearsign make legible output without compression.

- **Sign keys of other people, acquire other people signatures for your own key**

To export signed key I used `# gpg --output signed.key.asc --export --armor` *command*