Peter the Great St.Petersburg Polytechnic University

Department of Computer Systems & Software Engineering

**Laboratory report №1**

**Discipline: «Information Security»**

**Theme: «Encryption and Signing with GPG, Gpg4win»**

Made by student:

Volkova M.D.
Group: 13541/2

Lecturer:

Bogach N.V.

Saint-Petersburg
2018 y.

# Contents

# Laboratory work №1

## 1.1 Work purpose

Learn utilities, based on PGP technology, for encrypting and decrypting data.

## 1.2 Task

1. Study the description and launch graphic tool Kleopatra
2. Create a key pair with OpenPGP (File -> New Certificate)
3. Export Certificate (File -> Export Certificate)
4. Sign/Encrypt Files (File -> Sign/Encrypt Files)
5. Load other users certificates
6. Import a certificate, sign it
7. Verify the signature
8. Using your partner certificate encrypt, sign and send her a file
9. Accept, check and decrypt a file from your partner
10. Following the instructions in GNU Privacy handbook play with gpg by CLI,i.e. without graphic tool.

## 1.3 Work Progress

### 1.3.1 Introduction

Pretty Good Privacy (PGP) is an encryption program that provides cryptographic privacy and authentication for data communication. PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications.

GnuPG is a hybrid-encryption software program because it uses a combination of conventional symmetric-key cryptography for speed, and public-key cryptography for ease of secure key exchange, typically by using the recipient's public key to encrypt a session key which is only used once. This mode of operation is part of the OpenPGP standard and has been part of PGP from its first version.

### 1.3.2 Kleopatra frontend

#### Creating a new certificate

Let's illustrate the stage of creating a new certificate (File -> New Certificate ...). The first stage is the choice standard for a key pair. In addition to OpenPGP, Kleopatra also supports X.509:

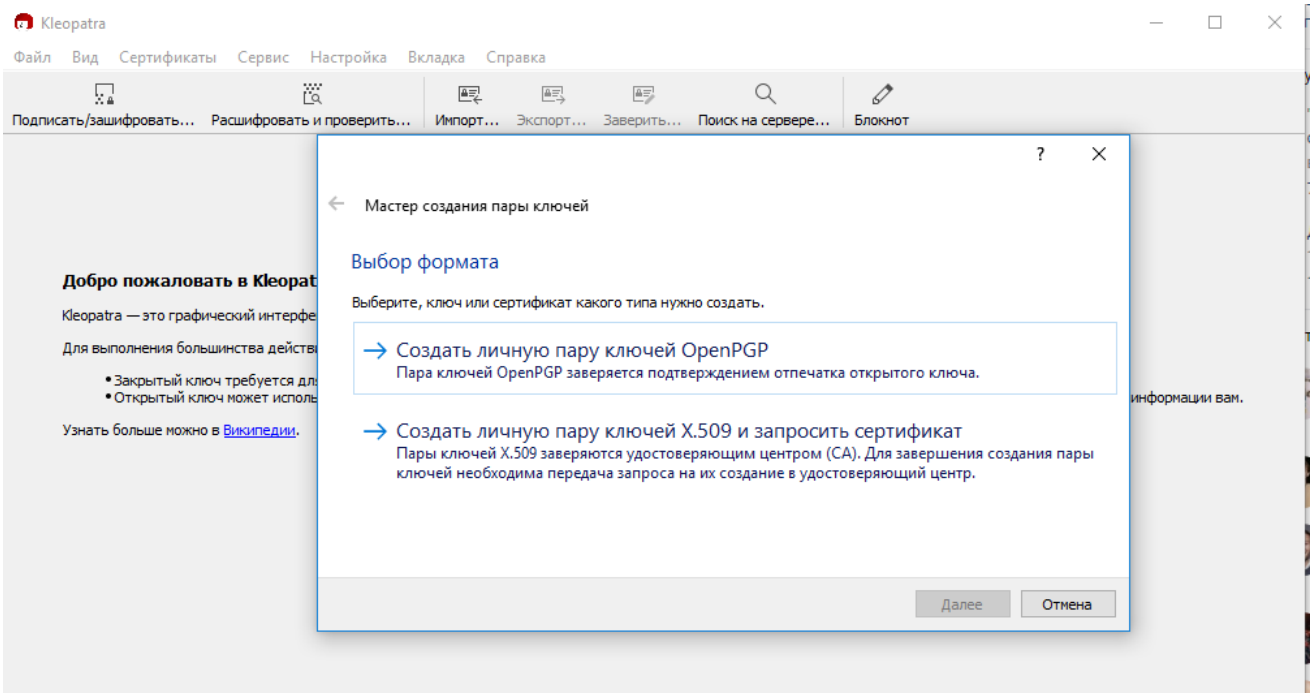


Рис. 1.1: Select a standard for the key pair

After that, the process of configuring the certificate (selection of the encryption algorithm, length key, certificate name, etc.):

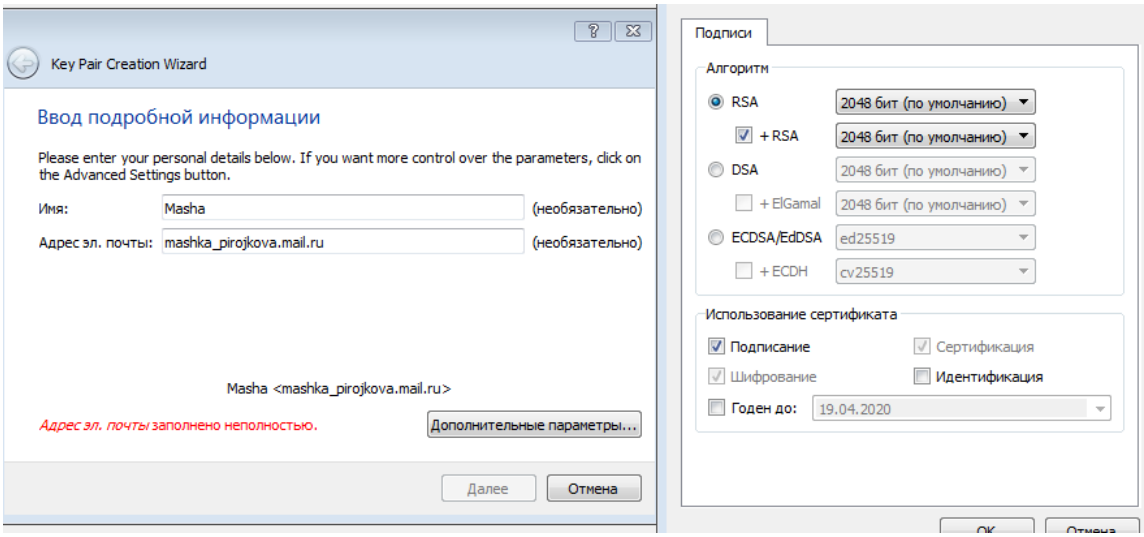


Рис. 1.2: Configure the certificate and select the encryption algorithm

Then, the process of generating a pair of keys (based on random characters and the movement of the window) occurs, and the password for accessing the private key is set:

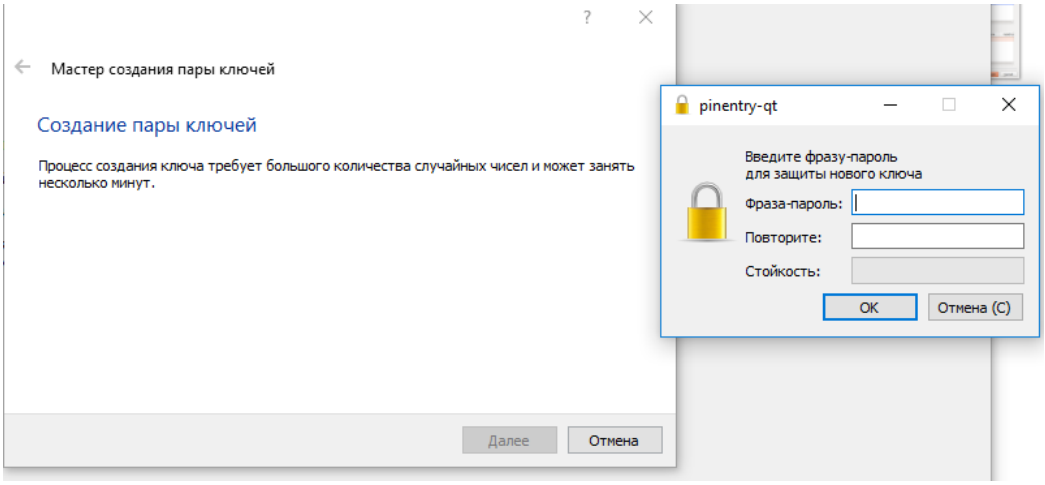


Рис. 1.3: The process of generating a key pair and setting a password for accessing a private key

The generated certificate appears in the list of certificates:

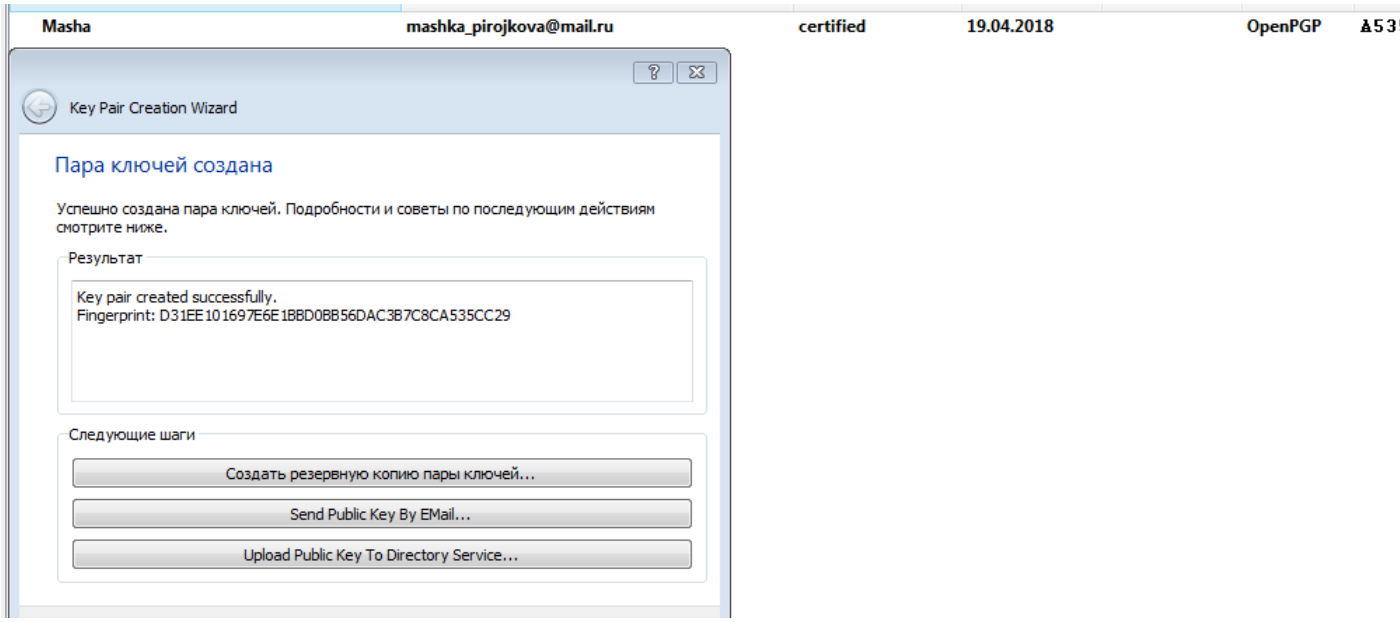


Рис. 1.4: The result of creating a certificate

To obtain a public key, use the command MRC -> Export Certificate... Open key is stored in a file with it's own extension:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBFrVAfIBCACqCIg/yIylDcJCA7P7jME/jKzjI132D4bT18WrHB7BWkvv9Oat
D/aLJdWvp4832W0UU04dV4oeesCPIUmD1AwUg88ZwrpHlZ2KU5HjQpefoprv1WXe
BiVDytA72pvw3JYY/bKFKEXSibhhhkwv5dMI3qFgDCwBZuLxFEJDB8GKic0l05vm
XqE3yFKrkx4mZy4mDZiAu9Jkvglkw7QGEzxknYktN0TpdCHlFPwdBOlGhqT96i7H
HgMWliZgmktJbyVFxraw0VsnTLdjqWlZXDkmjpTHB4WyEfGYWi6K31wtmr/6imeh
9ndz5Gia1zla8CalkrlzyFItreY0DGa2so33ABEBAAG0HFRpbXVyIDxsb3Jpc211
bGlrQHlhbmRleC5ydT6JAU4EEwEIADgWIQRLttdtwM35kel42yFjjL7iojZA6AUC
WtUB8gIbAwULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRBjjL7iojZA6PhWB/4m
PaFRddIv1gZWn2ZegSWGUWzK7dJkzznhkfnGY2O3w1Bm86msIeZw9QrEDLz8gkXZ
Tjm9BvHFRwH49qCXFwMKQFo99+wybtL9oSSLw2mu+bdqzknpDBIh1NZJ3JWSXFLA
g0avfrTxh5VZRBeDurhHqS/XZIwRJU54JpXDoDPtoGloh0vfbiFFwABIJ1tfkw2n
RN3dQvMxhephStSoTZfFG+qh/32xi8fLJ0Ij3bRheTCrZtEYVkwxNdPHLN4wdtwv
6j+sbRy29RrXXYSSDkrmGy1hJ/OvMBtXiVaC3JjsrQeqd+J2TR9mQFQbrBzacSL0
X43s4moHoVu6UeQ2ex1buQENBFrVAfIBCADAxOppqB61Mz1FW8V1S0jEgSMuzf06
Kn9PurZgVCEbTuHWI9xzdq8WbOGMjGM7L6vyjgZ1i2qouzfhZ5g4/diKinwYMn+F
7/HKYrs16eO5losD58xOs98Ni7KB4UNkqcqYlzHrs73F0NGBGNYIRxSmGQInw/59
Tx6jUfWP+rsgy2lNxZURC3z0kz5hZEtn7Qt+leCMUIVvQdYEWFSs9opmWEqkEJMv
XtXaxi6jYQqQTw/xtTY8bfmX5JCG3owJVGtg1DewrIapPmQnDxV3aCq/Zpyyol26
q13mFcchwy/Ou8YWtvxp3svTcSlSPl3Oi1BkHF7+qfMBxY13Ka03EKIFABEBAAGJ
ATYEGAEIACAWIQRLttdtwM35kel42yFjjL7iojZA6AUCWtUB8gIbDAAKCRBjjL7i
ojZA6NPzB/905kZJsfyVDS8Ji6Xz9kx+6mAgmixp5LBaagVKgujZT8rZueYDioQe
UkH0DQMruuMbQE5SKzCMgxgGmC8mGw1cFo52wt1dCh8eAeyoe1jLb0MCt9m9dcDv
VXN9ohH59pvuUsj5x81IPFArqdgj6DpVbgXecrvc21zX2pPnlk1EVvsDjCJDM7Vc
/degZClqbZ58BsySxYIseFCrM9ljZWi9SdxXoj0SsgeYHOyQvVaVCqH4aXpRYmXU
1+mKgB92WXN77C4tv1fTycrWOcDRw9XrhXP5z4sV6WHBz6DJLdCfzFF5FySlBebK
F6qY26UVCkP9hK99ZGZB6GNSVzJbKXYY
=gxTc
-----END PGP PUBLIC KEY BLOCK-----
```

Рис. 1.5

### Encrypting

On the **second** computer, we import a certificate (File -> Import Certificates ...), specifying as a parameter created public key.

To encrypt files, the command File -> Sign / Encrypt files was used ... A window appears with the choice of encryption parameters:

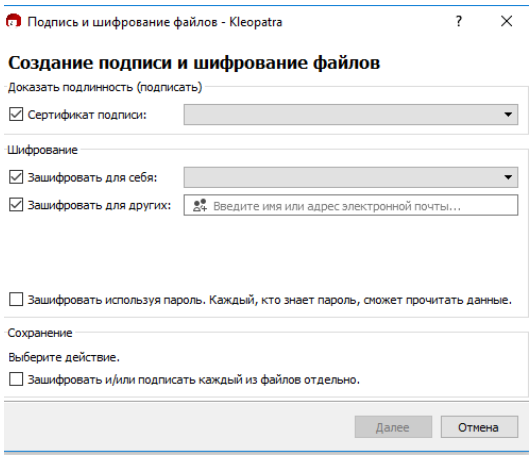


Рис. 1.6: Choosing encryption options

The result of file encryption:

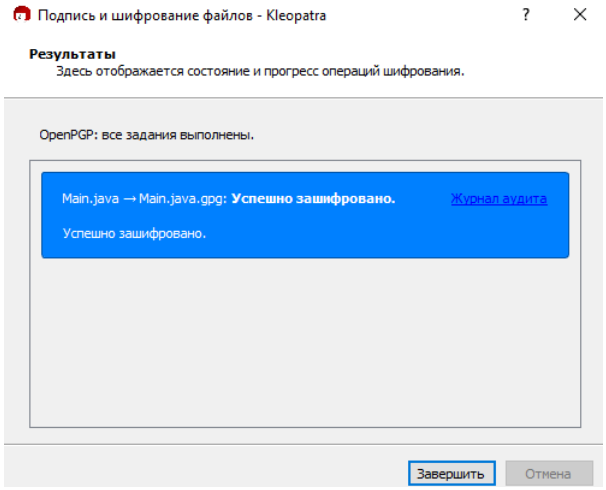


Рис. 1.7: The result of file encryption

The file was encrypted in .gpg format, the decryption of this file is possible only with a private key, so you can not decrypt it on this computer, even considering that we encrypted it.

### Decrypting

Transfer the .gpg file created on the second computer to the **first** (on which the certificate was created). To decrypt the files, we use the command File -> Decrypt / Verify files ...

The result of decrypting the file:

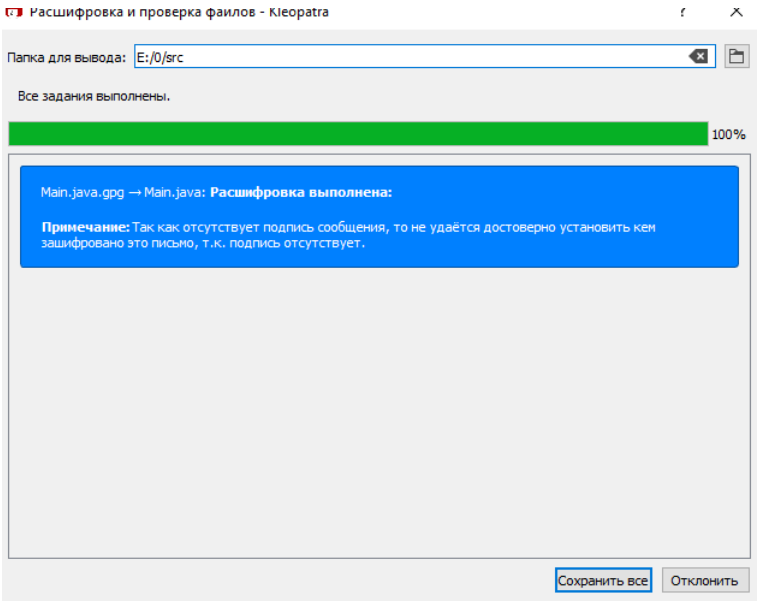


Рис. 1.8: The result of decrypting the file

The file was successfully decrypted: the name and contents of the file are the same as the original.

## 1.3.3 GPG command line

### Symmetric-key encrypting

For symmetric encryption, the -c flag is used:

```
masha@masha-VirtualBox:~/temp$ ls
SECRET.DATA

masha@masha-VirtualBox:~/temp$ cat SECRET.DATA
! My secret data !

masha@masha-VirtualBox:~/temp$ gpg -c SECRET.DATA
gpg: keyring '/home/masha/.gnupg/pubring.gpg' created

masha@masha-VirtualBox:~/temp$ ls
SECRET.DATA  SECRET.DATA.gpg

masha@masha-VirtualBox:~/temp$ cat SECRET.DATA.gpg
-> UNREADABLE SYMBOLS FOR LATEX <-
```

To decrypt a file, the combination of the -o and -d flags is used:

```
masha@masha-VirtualBox:~/temp$ ls
SECRET.DATA.gpg

masha@masha-VirtualBox:~/temp$ gpg -o SECRET.DATA -d SECRET.DATA.gpg
gpg: keyring '/home/masha/.gnupg/secring.gpg' created
gpg: AES encrypted data
gpg: encrypted with 1 passphrase

masha@masha-VirtualBox:~/temp$ ls
```

```
SECRET.DATA  SECRET.DATA.gpg

masha@masha-VirtualBox:~/temp$ cat SECRET.DATA
! My secret data !
```

**Public-key encrypting**

At the first step we creating and export the certificate by the following commands:

```
masha@masha-VirtualBox:~/temp$ gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 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? (2048) 2048
Requested keysize is 2048 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) 0
Key does not expire at all
Is this correct? (y/N) y

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: Donald
Email address: donald@trump.gov
Comment:
You selected this USER-ID:
    "Donald <donald@trump.gov>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

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 8A882796 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: public key of ultimately trusted key A6186B78 not found
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/8A882796 2017-11-05
      Key fingerprint = 0C4F 7BCD CA34 E401 77D8  07F4 EFB3 7A31 8A88 2796
uid                  Donald <donald@trump.gov>
sub   2048R/49ECD6F1 2017-11-05

masha@masha-VirtualBox:~/temp$ gpg --list-keys
/home/masha/.gnupg/pubring.gpg
------------------------------
```

```
pub   2048R/8A882796 2017-11-05
uid                  Donald <donald@trump.gov>
sub   2048R/49ECD6F1 2017-11-05

masha@masha-VirtualBox:~/temp$ gpg --armor --export Donald
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFn+gHEBCADR+ghOmcJeiDZk6TscxuurxEJBSFD7mPgaQHnPxxaj7iz9Z7am
HGxBe6pNDPbzhkRrZH7eSRC3bu94351X5kKwz39h94tCp9BbLA36VZ4j2y448Wfl
gWDrrT++cnE2xs/qcNOMB0wqd0iyPLyhcK7WP+0ZXmqoTFTZe3hY3xE6XhAYoAEB
d1j49j+K55PinZtOJa+hF8HQxig/CBRgDYeixG57fWdxU3XcNvm1OJwEpaUjHmcw
780pGTq2fAx9BEz5Dsmj0vPAWS5BfF5OMMA9CtTtHtK9ZNce5IIwdds6oahFgPWA
a0UpadDnOyLNHKRleEaTqQgFEGoWOggqL/tZABEBAAG0GURvbmFsZCA8ZG9uYWxk
QHRydW1wLmdvdj6JATgEEwECACIFAln+gHECGwMGCwkIBwMCBhUIAgkKCwQWAgMB
Ah4BAheAAAoJEO+zejGKiCeW2hUH/RNa19AKEJyHD7RQaK7B0OnzEFKUzRyje/Gi
FTn/4qhR0al0m3P8kz7m4GLVTnXhBLcxfjFa+N2LdijISe9WebeF6wG+WMGPoLL6
e98okr24aZ0i2YEgocoSHM/ySz7TBXx/yVZR8vIzjHKiqjnRn9dCrVIF/jW/rYnL
CinxUwAzb22Bz17Zt56VVJnEIGMKGcaHCMUsqxthC3Qe7h15Lt92ozIrhEQoyic5
LzzY9BYVKOyBPU4C02+wcUMsTZuaadQhcQYE3kkvhtWQBpCI8HExmIuYKokv1czX
CGMEM4PRGnmkdKEQFwb0ZAXo55PRhKEcoVLA7ccewEIBSGTRSAC5AQ0EWf6AcQEI
AKWbeRA3RN3LVkUtU79CxVtFIAMvrUJ9zZFHzQuEEefeswC4bXmWHmhlAIVTJEGS
NG1XDciIMN0qoIBvzZPMX5F4OAJg452u3T4zaaQ5R3OEm541e7b8/P/T1+iRmu3c
hCEomLqySGkzGJTcd6s6xgUFrb1XYv9xUgBZ7sKbQDRtwDCpclYhjhOmOWjLQmor
lnBP9O7btJOwcO/FaU7OXqmFYtyH4eF4vQph95gdw3YFHYTcf1iuZQVFu6trjDP+
4SuXmrhtkmftKfhLPFPy/TkHTJ1H5B6ZdPLTZkL4c8lIM7byKA86tjLlKixybEvH
fJUWPFfO/LWPxX4kV4YippkAEQEAAYkBHwQYAQIACQUCWf6AcQIbDAAKCRDvs3ox
iognlpsDB/9ooll0kv1Vu/SHk+kPZXsi452G4duGN3h1v3yndvxmhvv+lKObZm3B
uctxa7CoMXHZ14X1pd/yiurZfkxV3OSo6USzKXou8kuLGJ6O+qAPSkKCYgEWdUyM
Iwhh1gD0fJ4Dr2lZG/sJ5lvqyBKsm+0t3bdTuFJHiCGF7bYI6rfl1UYgumT9VIFX
S7ucTeUebk+P1KF5j5IRFbxZFAlrYQVrxMv5JKCuzqSM2NF+yIKrNs5e1Emd8G3Q
o5Phwy+RW6bg3rAuDViCqjYyhzjL4GqNBUfJS1r2D3YbrVLO3djB78ytWnj+v2d5
11u7VGYZbEafG58WsRrhWDoye7rTV/7d
=WxVr
-----END PGP PUBLIC KEY BLOCK-----
```

Encrypting file on the second machine by the public key:

```
masha@masha-pc:~/temp$ ls
key

masha@masha-pc:~/temp$ gpg --import key
gpg: key 8A882796: public key "Donald <donald@trump.gov>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)

masha@masha-pc:~/temp$ gpg --list-keys
/home/masha/.gnupg/pubring.gpg
------------------------------
pub   2048R/A6186B78 2017-04-23
uid                  My Super Certificate <im@the.best>
sub   2048R/D1332725 2017-04-23

pub   2048R/8A882796 2017-11-05
uid                  Donald <donald@trump.gov>
sub   2048R/49ECD6F1 2017-11-05

masha@masha-pc:~/temp$ echo "! My secret file !" > SECRET.DATA

masha@masha-pc:~/temp$ gpg -o SECRET.DATA.gpg -e -r Donald SECRET.DATA
gpg: 49ECD6F1: There is no assurance this key belongs to the named user

pub  2048R/49ECD6F1 2017-11-05 Donald <donald@trump.gov>
 Primary key fingerprint: 0C4F 7BCD CA34 E401 77D8  07F4 EFB3 7A31 8A88 2796
      Subkey fingerprint: D128 458C A1B6 EBCF 340C  752F 98ED 17FB 49EC D6F1

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

masha@masha-pc:~/temp$ cat SECRET.DATA.gpg
-> UNREADABLE SYMBOLS FOR LATEX <-
```

The result of decrypting the file on the first machine:

```
masha@masha-VirtualBox:~/temp$ ls
SECRET.DATA.gpg

masha@masha-VirtualBox:~/temp$ gpg -o SECRET.DATA -d SECRET.DATA.gpg

You need a passphrase to unlock the secret key for
user: "Donald <donald@trump.gov>"
2048-bit RSA key, ID 49ECD6F1, created 2017-11-05 (main key ID 8A882796)

gpg: encrypted with 2048-bit RSA key, ID 49ECD6F1, created 2017-11-05
      "Donald <donald@trump.gov>"

masha@masha-VirtualBox:~/temp$ ls
SECRET.DATA  SECRET.DATA.gpg

masha@masha-VirtualBox:~/temp$ cat SECRET.DATA
! My secret file !
```

## 1.4 Conclusion

In this paper, we examined asymmetric encryption using the Kleopatra program of the OpenPGP family. Asymmetric encryption has the advantage over symmetric in the ease of exchange keys, but loses in encryption speed. The encryption considered in this work is one-way. In order to carry out two-way transmission, two channels are used. In modern cryptosystems, asymmetric encryption is used for key exchange, and at the same time symmetric encryption for data exchange.