Peter the Great St.Petersburg Polytechnic University
Institute of Computer Science & Technologys

Department of Computer Systems & Software Engineering

Laboratory №1 Report

Discipline: Information Security

Theme: Encryption and Signing with GPG, Gpg4win package

Made by student of group. 13541/3 _____ D.V. Kruminsh
(signature)

Lecturer _____ N.V. Bogach
(signature)

Saint-Petersburg
2017

Contents

Encryption and Signing with GPG, Gpg4win package

1.1 Objectives

After completing this module you will be able to:

1. Create digital certificates;
2. Encrypt files;
3. Sign files.

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 (a link is in REFERENCE section in a bottom of this module) play with gpg by CLI, i.e. without graphic tool.

1.3 Theory

GnuPG is a complete and free implementation of the OpenPGP standard as defined by **RFC4880** (also known as **PGP**). GnuPG allows to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. Features:

- Full alternative to PGP;
- Does not use proprietary algorithms;
- Distributed under the GNU General Public License;
- Expansion and authentication of e-mail messages created with PGP 5, 6 and 7;
- Support for electronic signature using ElGamal, DSA, RSA and hash functions MD5, SHA-1, SHA-2, RIPE-MD-160 and TIGER;
- Work with asymmetric encryption ElGamal and RSA (key length from 1024 to 4096 bits);
- Support for block symmetric encryption algorithms AES, CAST5, 3DES, Twofish;
- Blowfish, Camellia, and IDEA using a plug-in;
- Support for compression algorithms: ZIP, ZLIB, BZIP2.

Gpg4win

Gpg4win is a Windows version of GnuPG featuring a context menu tool, a crypto manager, and an Outlook plugin to send and receive standard PGP/MIME mails.

It is maintained by the developers of GnuPG and the software included with Gpg4win are Free Software.

Work Progress

2.1 Study the description and launch graphic tool Kleopatra

Kleopatra is a certificate manager and GUI for GnuPG. The software stores OpenPGP certificates and keys. It is available for Windows and Linux.

Figure 2.1: Main window

On the main window, kleopatra suggests creating a new key pair or import it.

2.2 Create a key pair with OpenPGP (File → New Certificate)

In the following dialog, you can choose format: OpenPGP(PGP/MIME) or X.509 (S/MIME).

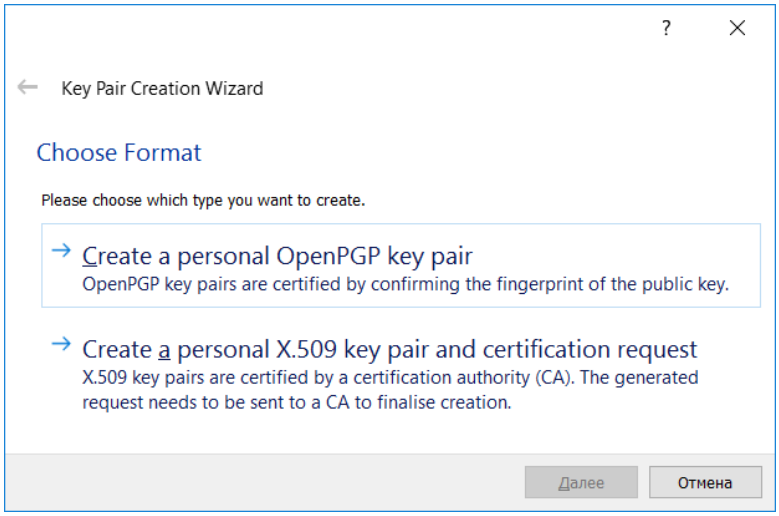
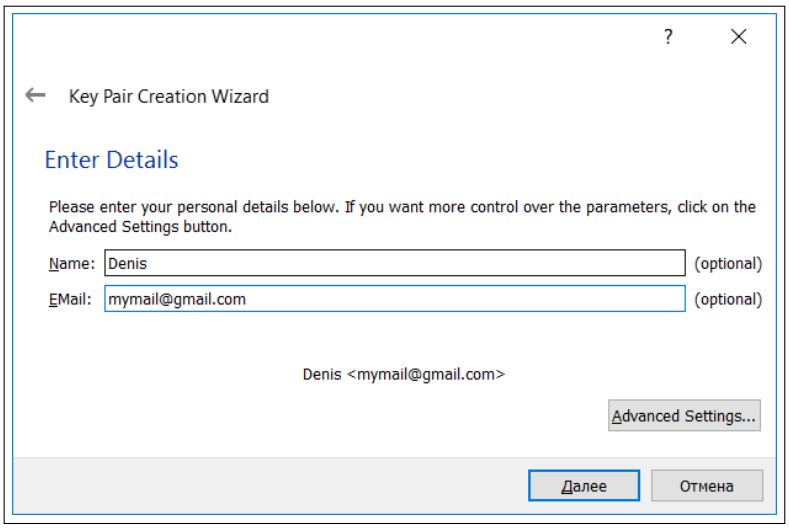
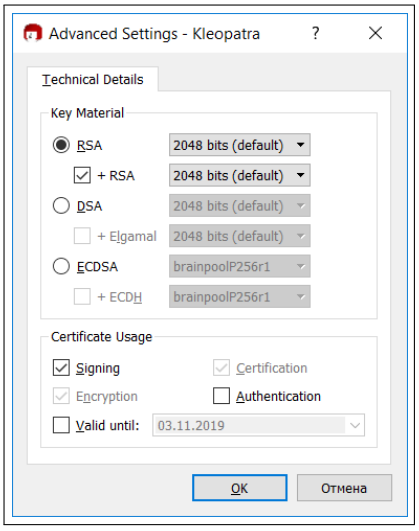


Figure 2.2: Key Pair Creation windows

After that, it is suggested to enter a name and mail, and set up additional settings, if needed.



(a) Input of name and email



(b) Advanced options

Figure 2.3: Certificate options

after click an next button, we see all inputed params.

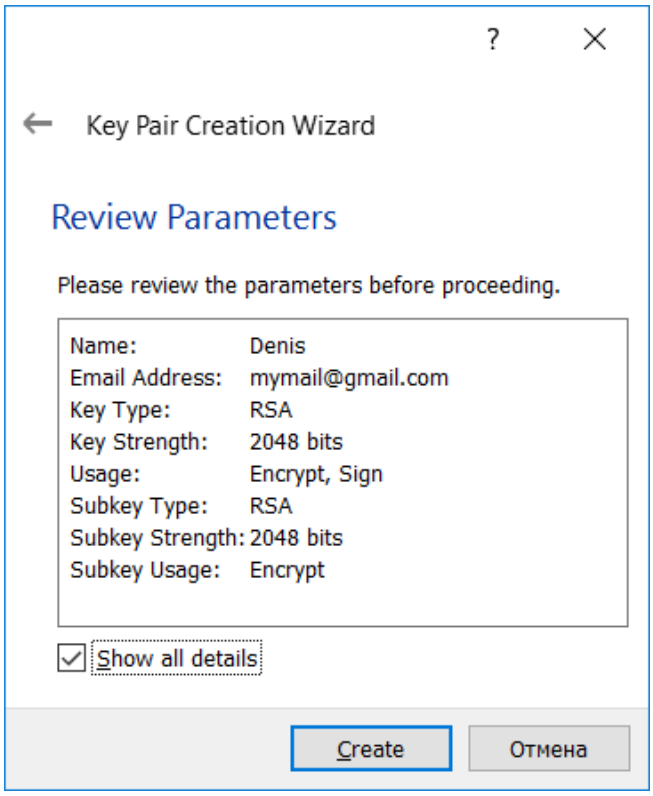


Figure 2.4: Result params

Now need to type password.

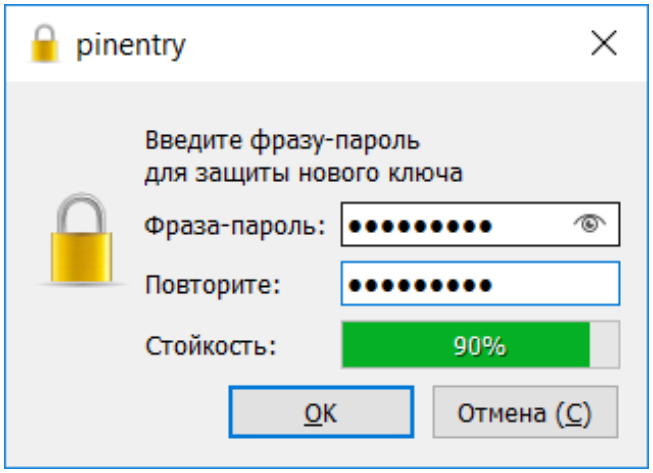


Figure 2.5: Creating passphrase

And after it, we see window with message about successfuly created key pair.

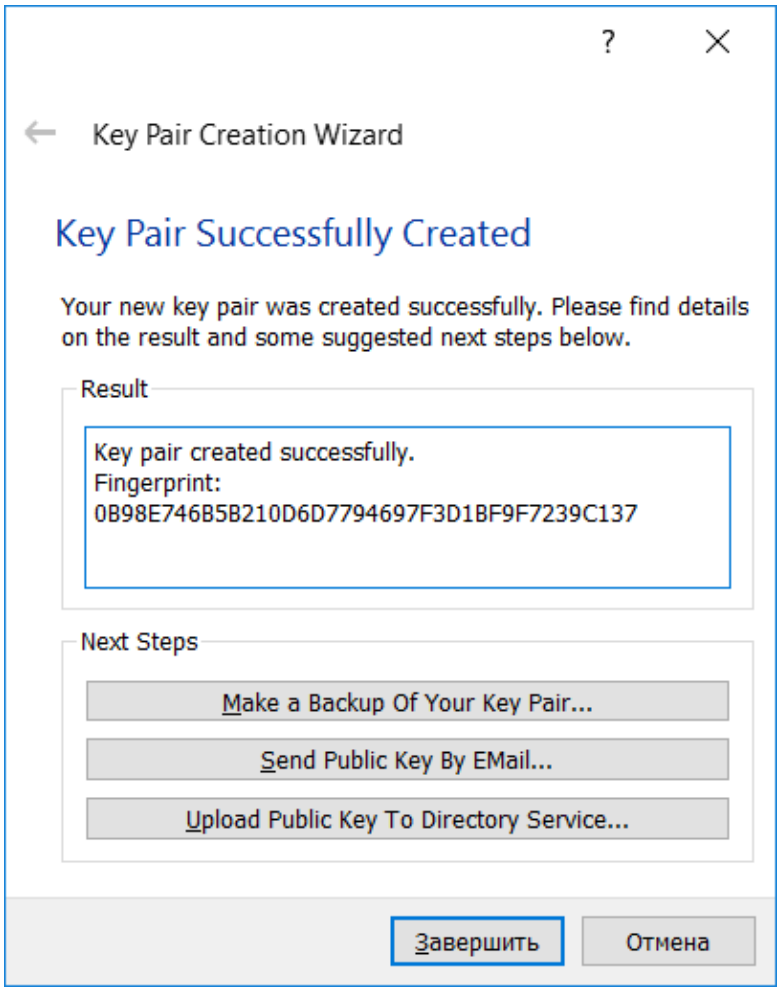


Figure 2.6: Result of creating

In result box we see **fingerprint**. Fingerprint required to identify the certificate and its owner.

2.3 Export Certificate (File → Export Certificate)

To export certificate, right click at certificate, than choose **Export...** or just **Ctrl+E**.

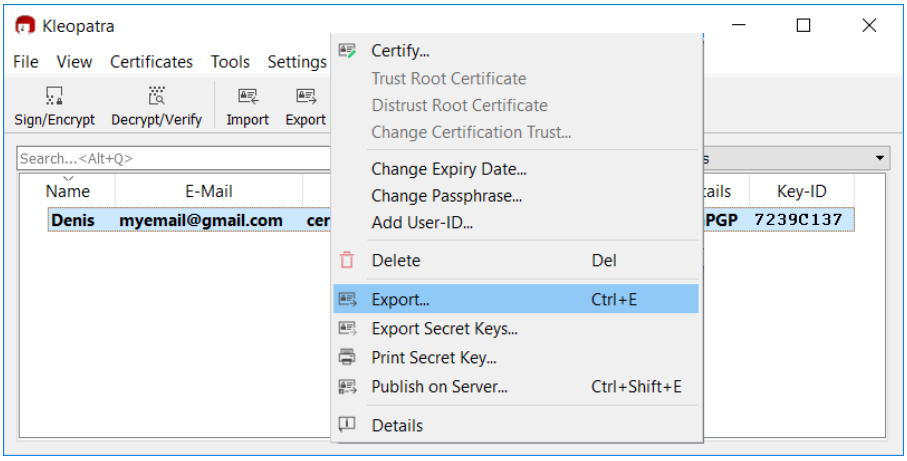


Figure 2.7: Expoting certificate

File can be exported in theese file formats:

- **asc**
- **gpg**
- **pgp**

If open it in text editor, we see following:

```
1  -----BEGIN PGP PUBLIC KEY BLOCK-----
2
3  mQENBFn8oe0BCAC0RJQ7bGdoaSX863K2NWbpx8lV5yJSYttNcoSW3Ril6SNMsrs+
4  mER1/dglsfiVNtQvehv89Va6HUlGp7EkM9saOG2ve9heBwZXrD4Xt0XxZaxLhb0C
5  NKxP2Q0PQlJbx/20PxVFkvMYOIThK2ySbPY0OdbXgC3QldfhQAL+BLFZC1yuWZk5
6  LVQuO8QAK3YF/3pnhZkX92DACmVOUVncfunObOVVCSQ3pl1jSL4xHkVVUyTSuyZU
7  +P1kX0P2NjsA8hDk+k0RfFjocUQ88wQaVq4oFV88GyWDgu1ngz1F8IR7OiuljRBW
8  wuFaP+TyI285FlpiqYyaMASiZtEQWOVJ/QHDABEBAAG0GURlbmlzIDxteWVtYWls
9  QGdtYWlsLmNvbT6JAU4EEwEIADgWIQQLmOdGtbIQ1td5Rpfz0b+fcjnBNwUCWfyh
10 7QIbAwULCQgHAgYVCAkKCwIEFgIDAQIeAQIXgAAKCRDz0b+fcjnBN7gGB/43ea3H
11 Bp57KbHcWKjh932qB0yXbOzrN/sZflEe+1/tSLI1+fOKTh0IlNA5yWrs/YZTvts9
12 FxdicUztUi7BjLKDCB1lkQVkUuKWgaCXNEoqxvTcy2aMRYAtEH+Bj9uEaEOVJt1R
13 v3os3IJeG26dJUpNlYpFNhkXhY8TTH0xLb7lgwBT8D7NMSQLhIDH3Mq7Oifr5Pgc
14 167tkstjBUAYZzwSY5uqaO5+ukMJ4KvHjDnVjSlraEZkxlrMv0W34canp7/Hk179
15 U37NXE/f2JH4Kdy/dq/zkF3XyyAPKEoysSuy9krxbKnpuLHtxIuNxdm4/hG0HlaA
16 Ufw6bRV7Xi1tzV8ruQENBFn8oe0BCADk0qPWiQtul8CibriAL0Jv8tMRqt+oveGg
17 nSi7ke2nCjkrTPcTGl3NBd8zyR7IkWOi/9FbRiFVJtd5QQSzdV5oWVOFkZZFLX8U
18 klaRvtuBibbZT6brJTkxI9Rw2XTmQVZOw6yJnO57LXMU3rFDzP5PoxpK8mQKEs6Z
19 3vZG7Gd5FYEGo+Ts+TciZ3oLvn0p94/CXwPRt4ri+kRHD0jGO0Vfe/txr3ZJSSht
20 m+Zw95oreZv7TWeS/nyLTvtDVVad2+gTrOVuIduZNT1sV3Kmbq15lqOFJqQ392KW
21 sO1DECPa/ar8kxkTFf16wCvrjp22pbpTBO8hWW5HkFNVoA6obfH5ABEBAAGJATYE
22 GAEIACAWIQQLmOdGtbIQ1td5Rpfz0b+fcjnBNwUCWfyh7QIbDAAKCRDz0b+fcjnB
23 N0cUB/429at3bHAfK2dw7AkCiFzIrjjZS+bW8zJBSHzrTrVplb5MKHPSEvkRQFI
24 6WolkQD3bqhefNY/rzcby2q6EiME1a//CDOzOEFRkKTvQJFsk66SHd5t4tgTb5s9
25 BbFzh9fAasygpyAP94+MMTQpzEcaqvf+XxUE4b9vQDoEAjMUPqYQxILwKYURr8v9
26 poFcWIT5iyerl7SQQfCKofzpnJ8iR0zuTNG+vhhunRVHfSbLwYLCOb9A7OTte7nX
27 VdGnGbTxI0RcZhFKovvxQkfLefJZT2uEIJpGLI/pAvRu8B52OpP/vLFfqESPss6l
28 cOOZvHGi2deYH4uLonHgDZe8FStK
29 =RJzf
30 -----END PGP PUBLIC KEY BLOCK-----
```

Listing 2.1: 0B98E746B5B210D6D7794697F3D1BF9F7239C137.asc

With this certificate we can sign and encrypt files.

2.4 Sign/Encrypt Files (File → Sign/Encrypt Files)

Let's create new file, with following text:

```
1  My secret message
```

Listing 2.2: test.txt

And choose this file to encrypt

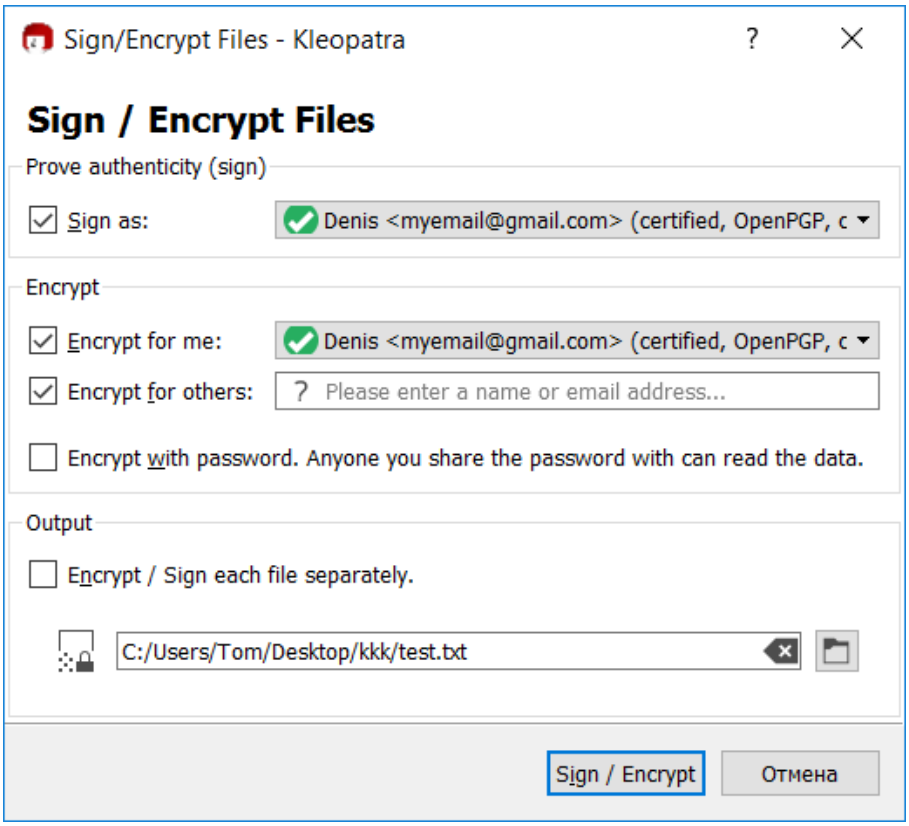


Figure 2.8: Sign/Encrypt Files

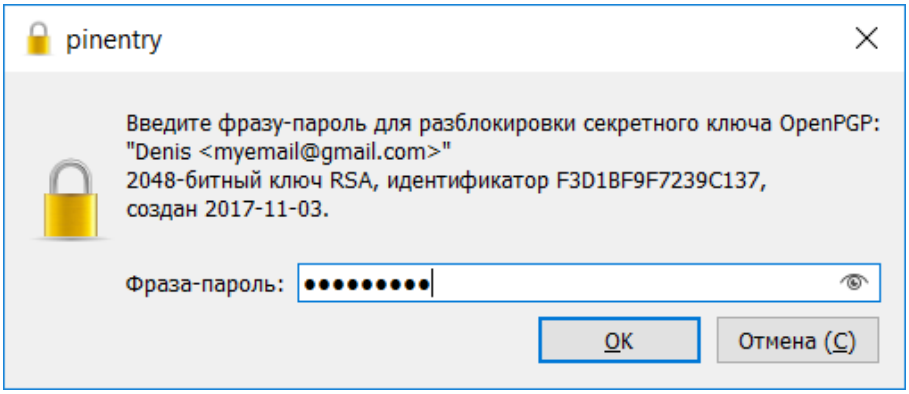


Figure 2.9: Password input

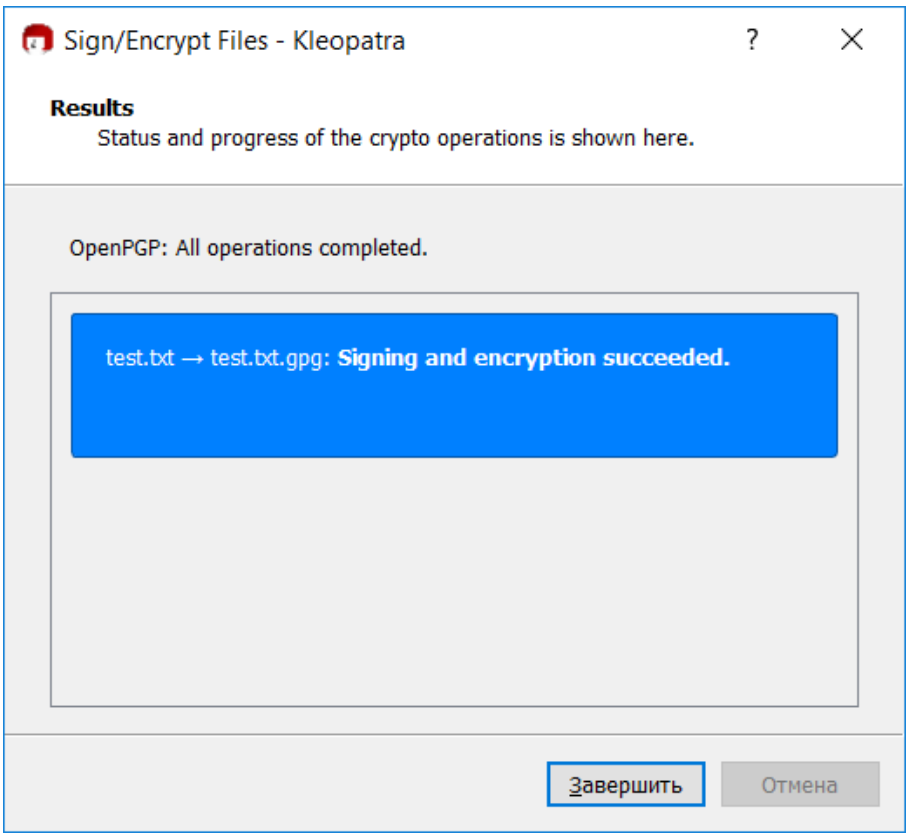


Figure 2.10: Signing and encryption succeeded

Now let's see what we got:

```
 1 8501 0c03 8c02 7204 1e1f 91b2 0108 0087
 2 114c a623 1617 88db 2f47 0a91 288c 0caf
 3 2b8c 59f1 3eb1 f47d d62e 0577 c4af f4f0
 4 b846 3f55 11b9 1d68 a4c7 92e8 7a22 d8f0
 5 ceb9 59e2 d342 062a 3e34 f2f1 9268 24e1
 6 71e5 4c13 2c00 82c5 544b b7ca e74c 78d6
 7 1879 4913 6dcc 6a87 f18f 014f 5088 d3bf
 8 cd10 30ea 6eda cd04 d3f2 8e31 6364 b61d
 9 6aa3 95bf 4c37 2890 47c1 9702 5802 42a5
10 6be4 a291 8cd0 65c1 b9d4 a673 5ced f965
11 817d a47d 5168 0f41 4711 75f8 377c a8d7
12 d04a f1a9 7a94 80fb a2c9 bb31 f5a4 7743
13 e723 0e63 099c 6dd3 d57e 4618 e689 4728
14 5b9c 6c9c e914 4a5d c7b2 7c8d db4b de10
15 c192 7d18 eace 1c1a 8daa f1d1 a8e6 6395
16 a9da a087 9421 4b9f 0500 0939 d8ca f4e0
17 2781 a8b4 aa6c 458b 1a7d f126 b5fc ffd2
18 c0d2 016c a6e5 3a13 3b90 7733 2eb0 cea9
19 695f f43a b0d3 7979 4e57 d94d 8eee 2f8f
20 387a bd77 d4a6 d5e2 11f4 60bb f4f4 7b7d
21 ee5b be68 2496 1d76 dc74 c239 4b4f 0eea
22 31c5 c477 704a 23eb 1d9f 4922 ec93 f48e
23 92c5 4242 7794 efd1 bf6b 898d 0a1f 378f
24 0574 804f 9815 e9ac 91cd def0 08db 8fa9
25 80bc 0b8d 4f05 abc1 4b05 200a 1437 d179
```

```
345d 1f98 ea26 e559 f1a8 7da8 d630 9429
645a 4057 90e6 1fe6 8096 0f13 e2a1 16ac
bca2 c61a 5f56 16db c018 acaf a94a 4cf1
f834 05c5 7d41 0042 4ab0 a653 ef94 7c76
a25c 1d45 03d2 d178 af2e 0ce1 c2c3 e190
0d52 5afc e83d fcee d66a 54e6 af4a 14f9
6556 39dd ad9b 8181 05cf 2df9 9ef0 6b0d
13e4 b170 696f 82c7 25b6 7d0c f8ed 30e9
16ba 79a5 e1d2 8d0e 80f4 6e04 57d9 0604
9f47 c010 ccb1 cc61 dbf4 b4bc 0019 a0e8
fcea d6c2 2219 8c65 c294 9280 d755 721b
907d 0b3e fdb5 bf14 bde6 5ac6 6c95 9a30
b82b 9fd6 d8e8 4a16 89bc 052e 9e42 9fff
74b5 b5aa 6aeb 0d94 00f6 ae27 e937 1e2f
4896 c303 9725 a4f3 1f8f 5a4c c0db 571b
d8b1 daf9 f4d6 245a 03a3 6073 70a3 ab10
a11e 736c 83df 0acc 7ba7 49ca 5493 cd5d
113f 4c8b
```

Listing 2.3: test.txt.gpg

As expected the message was encrypted.

2.5 Load other users certificate, import, sign and verify it

Click at **import** button and choose other user certificate.

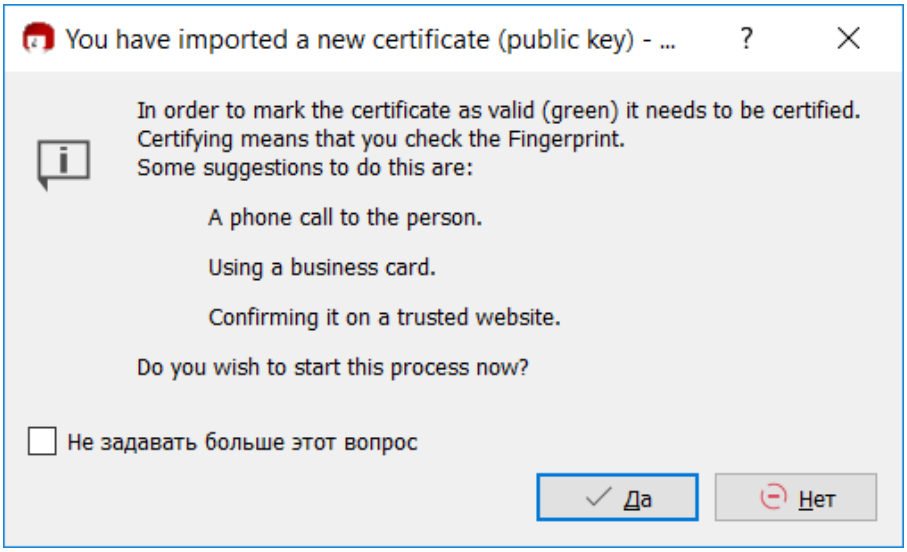


Figure 2.11: Importing new certificate

We see his name, email and fingerprint.

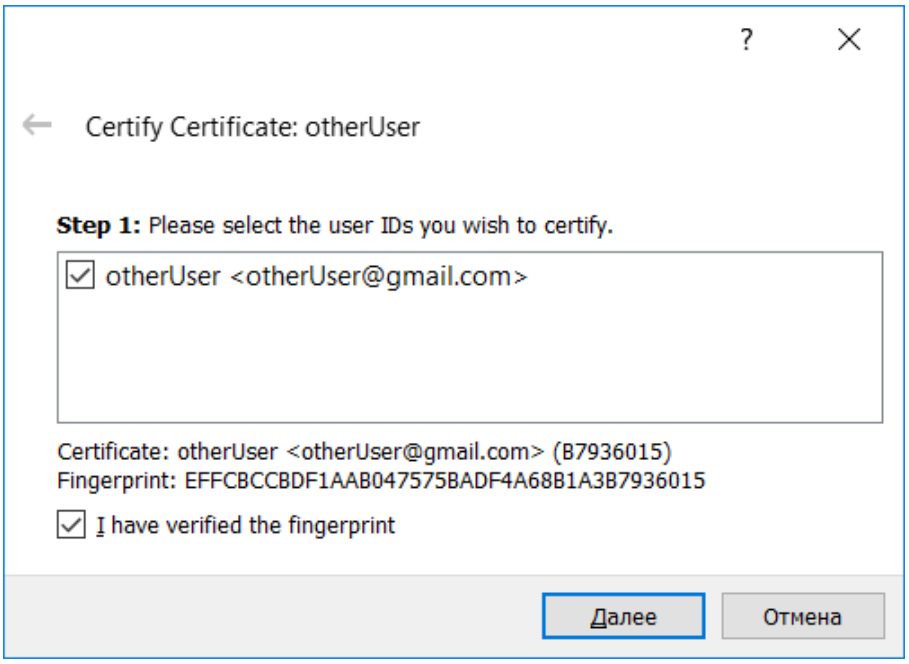


Figure 2.12: Certificate information

Now we choose for whom we certify this.

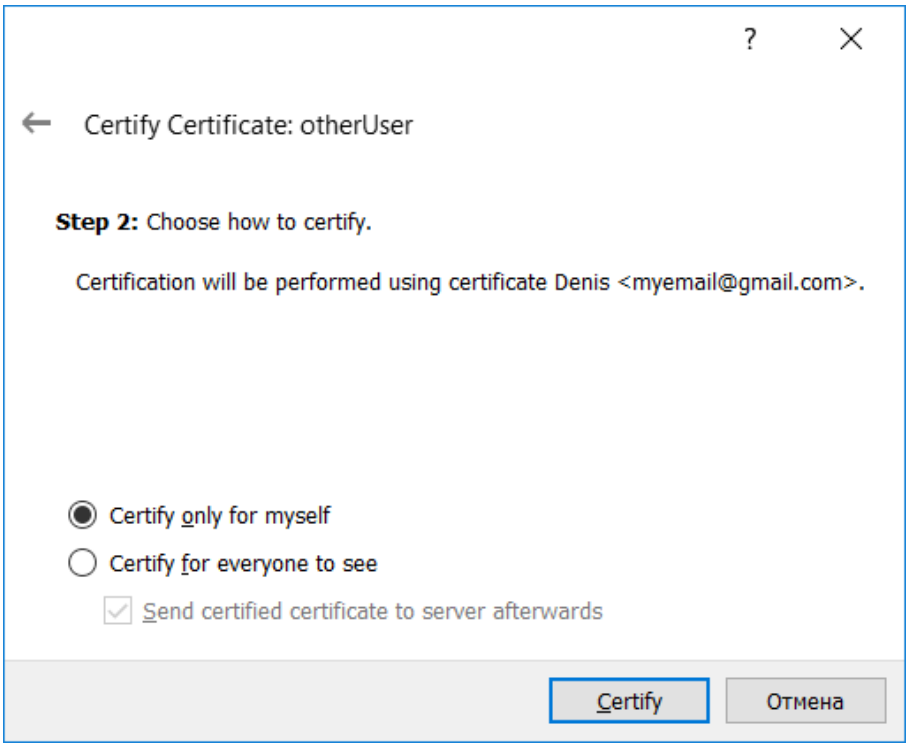


Figure 2.13: Type of certification

As result, we see fully trusted certificate.

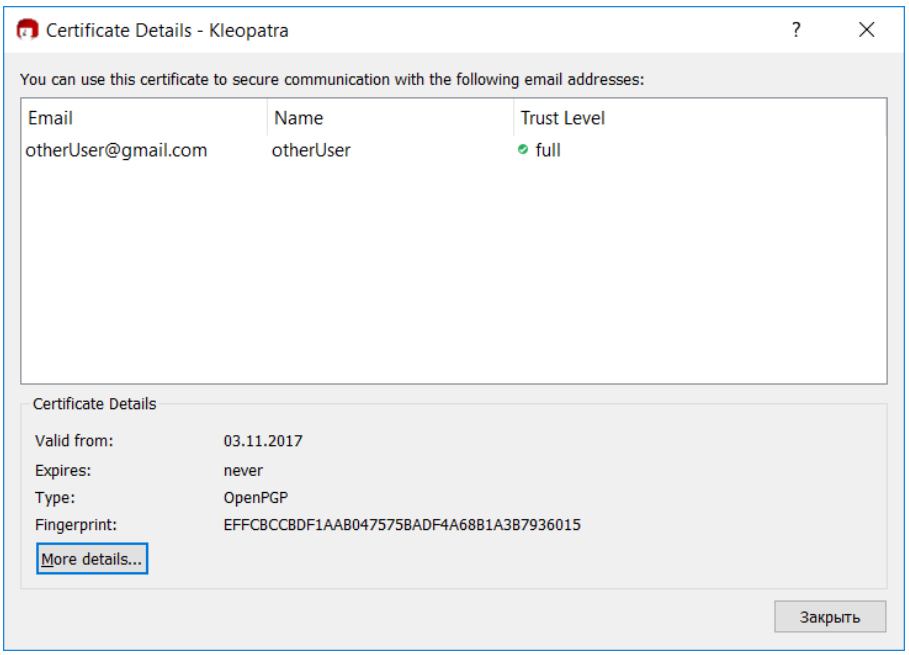


Figure 2.14: Result

2.6 Using your partner certificate encrypt, sign and send her a file

2.7 Accept, check and decrypt a file from your partner

Let's decrypt file(test.txt.gpg) from paragraph 2.4.
Click File → Decrypt/Verify...

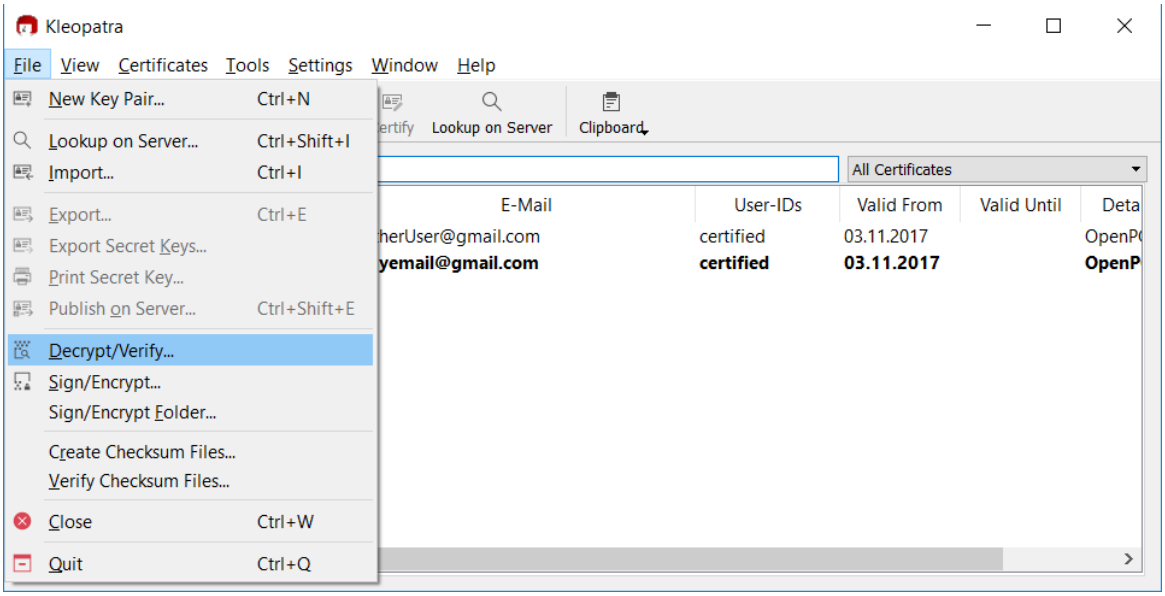


Figure 2.15: Decrypt/Verify

After choosing file, kleopatra using known certificate decrypt this, and than we got following message.

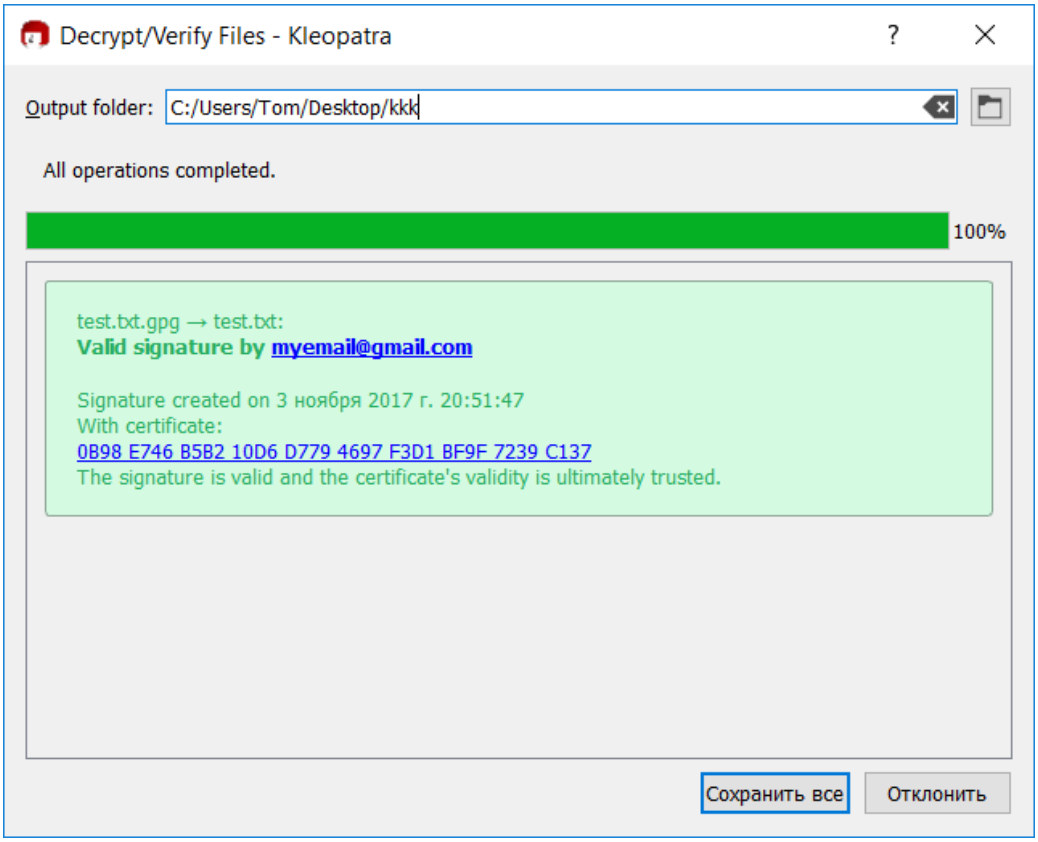


Figure 2.16: Successful decrypt

2.8 Following the instructions in GNU Privacy handbook, play with gpg by CLI, i.e. without graphic tool.

In this part of the report, i will use kali linux as virtual machine. First check version of gpg, and then generate key, using following command:

gpg –gen-key

Below is a full action log.

```
root@kali:~/Desktop/testFolder2# gpg --version
gpg (GnuPG) 2.2.0
libgcrypt 1.7.9
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/
    ↪ licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /root/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

```
root@kali:~/Desktop/testFolder2# gpg --gen-key
gpg (GnuPG) 2.2.0; 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.

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

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

Real name: Denis
Email address: myemail@gmail.com
You selected this USER-ID:
    "Denis <myemail@gmail.com>"

Change (N)ame, (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.
```

Listing 2.4: terminal log

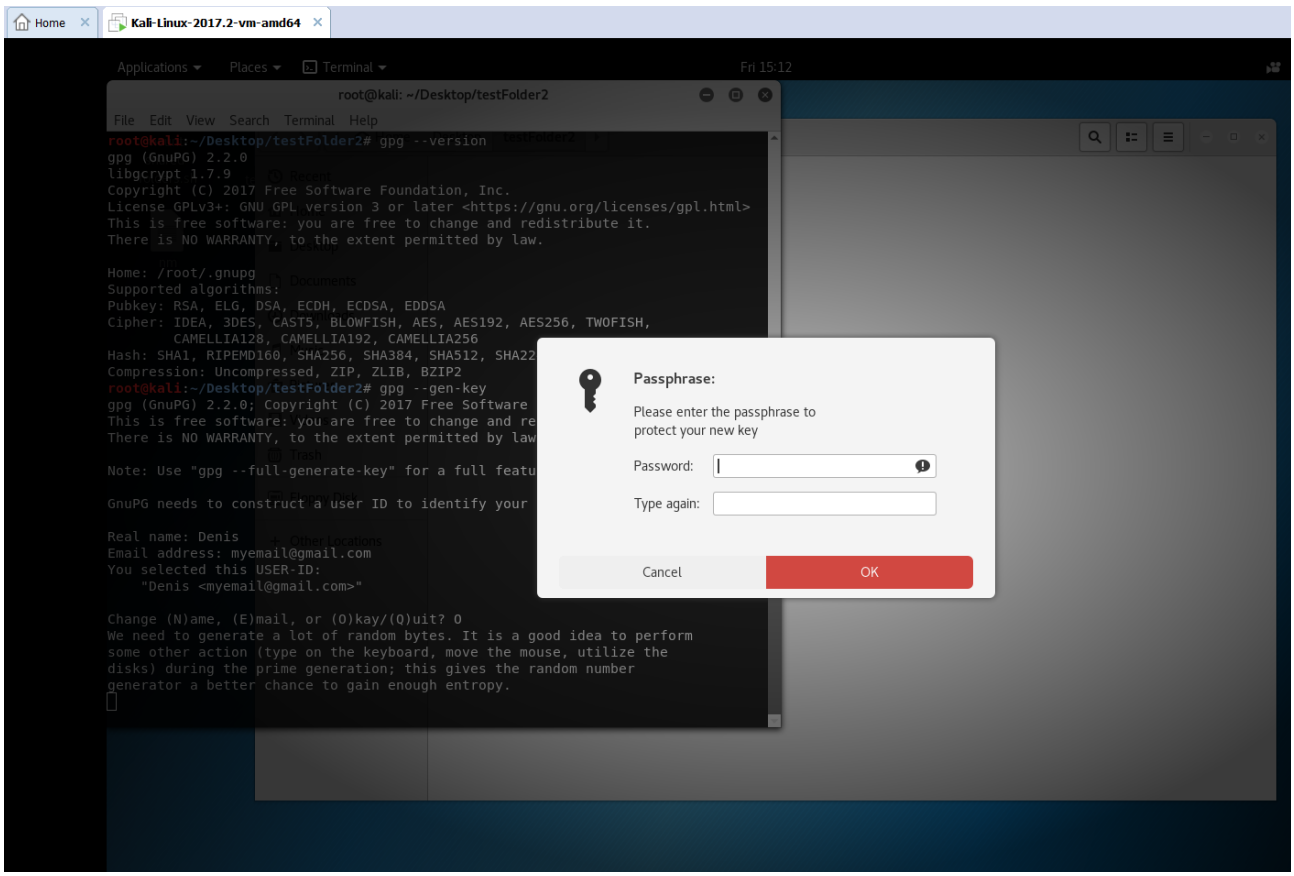


Figure 2.17: Password input

After some action's with mouse moving, we see following:

```
1
2  gpg: /root/.gnupg/trustdb.gpg: trustdb created
3  gpg: key 0F3B233F4F373F3A marked as ultimately trusted
4  gpg: directory '/root/.gnupg/openpgp-revocs.d' created
5  gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs
     ↪ .d/AF3AA2D6F72A7B38B5ED151B0F3B233F4F373F3A.rev'
6  public and secret key created and signed.
7
8  pub   rsa3072 2017-11-03 [SC] [expires: 2019-11-03]
9        AF3AA2D6F72A7B38B5ED151B0F3B233F4F373F3A
10 uid                      Denis <myemail@gmail.com>
11 sub   rsa3072 2017-11-03 [E] [expires: 2019-11-03]
```

Listing 2.5: successful key generation

My key was successfully created, now to export it, need to type following command:

```
1  root@kali:~/Desktop/testFolder2# gpg --armor --output DENIS.asc --
     ↪ export myemail@gmail.com
```

Listing 2.6: export

Key was identified by email, and now in current directory we have exported certificate file.

```
1  root@kali:~/Desktop/testFolder2# ls -l
2  total 4
3  -rw-r--r-- 1 root root 2444 Nov  3 15:29 DENIS.asc
```

Listing 2.7: directory

For import we can use following command:

gpg –import someCert.asc

Conclusion

As result in this report i learned how to use encryption tool's with GUI(kleopatra) and with console(pgp). Kleopatra is easy to use, because of the intuitive interface. In the console it was more difficult, but using **The GNU Privacy Handbook** helped me to understand how it work's.

Encryption is extremely important in the modern world, especially when transferring over the Internet important files or data. So, using of encryption in everything(web, email, text messanger etc) is considered normal.