Network Security Lab Exercise 02

Prepared By: Parman Sukarno, M.Sc, Ph.D

Hands-on- Email Security using GnuPG (GPG)

Introduction to GnuPG

GnuPG is the GNU project's complete and free implementation of the OpenPGP standard. This hands-on includes key generation, exchanging and verifying keys, encrypting and decrypting files, and authenticating files with digital signatures.

GnuPG employs public-key cryptography. In this system, every user has a pair of keys consisting of a private key and a public key. A user's private key is kept secret while the public key may be published.

The manual of GnuPG might be obtained by running the command below:

$ man gpg

You need to work in group to do this lab exercise.

Generating a new keypair

1. To start using GnuPG you need to first run the key generation process. List the command-line and parameters required.

The command-line option `--gen-key` is used to create a new primary keypair.

```
alice% gpg --gen-key
gpg (GnuPG) 0.9.4; Copyright (C) 1999 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
certain conditions. See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) ElGamal (sign and encrypt)
Your selection?
```

GnuPG is able to create several different types of keypairs, but a primary key must be capable of making signatures. There are therefore only three options. Option 1 actually creates two keypairs. A DSA keypair is the primary keypair usable only for making signatures. An ElGamal subordinate keypair is also created for encryption. Option 2 is similar but creates only a DSA keypair. Option 4 creates a single ElGamal keypair usable for both making signatures and performing encryption. In all cases it is possible to later add additional

subkeys for encryption and signing. For most users the default option is fine.

You must also choose a key size. The size of a DSA key must be between 512 and 1024 bits, and an ElGamal key may be of any size. GnuPG, however, requires that keys be no smaller than 768 bits. Therefore, if Option 1 was chosen and you choose a keysize larger than 1024 bits, the ElGamal key will have the requested size, but the DSA key will be 1024 bits.

```
About to generate a new ELG-E keypair.
              minimum keysize is  768 bits
              default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024)
```

The longer the key the more secure it is against brute-force attacks, but for almost all purposes the default keysize is adequate since it would be cheaper to circumvent the encryption than try to break it. Also, encryption and decryption will be slower as the key size is increased, and a larger keysize may affect signature length. Once selected, the keysize can never be changed.

Finally, you must choose an expiration date. If Option 1 was chosen, the expiration date will be used for both the ElGamal and DSA keypairs.

```
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)
```

For most users a key that does not expire is adequate. The expiration time should be chosen with care, however, since although it is possible to change the expiration date after the key is created, it may be difficult to communicate a change to users who have your public key.

You must provide a user ID in addition to the key parameters. The user ID is used to associate the key being created with a real person.

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

Real name:
```

Only one user ID is created when a key is created, but it is possible to create additional user IDs if you want to use the key in two or more contexts, e.g., as an employee at work and a political activist on the side. A user ID should be created carefully since it cannot be edited after it is created.

```
Enter passphrase:
```

There is no limit on the length of a passphrase, and it should be carefully chosen. From the perspective of security, the passphrase to unlock the private key is one of the weakest points in GnuPG (and other public-key encryption systems as well) since it is the only protection you have if another individual gets your private key. Ideally, the passphrase should not use words from a dictionary and should mix the case of alphabetic characters as well as use non-alphabetic characters. A good passphrase is crucial to the secure use of GnuPG.

2. What is the purpose of the passphrase?

You need a Passphrase to protect your private key.

Exchanging keys

1. To communicate with others you must exchange the public keys. What is the command-line used to list the keys on your public keyring ?

To communicate with others you must exchange public keys. To list the keys on your public keyring use the command-line option --list-keys.

```
alice% gpg --list-keys
/users/alice/.gnupg/pubring.gpg
------------------------------------
pub  1024D/BB7576AC 1999-06-04 Alice (Judge) <alice@cyb.org>
sub  1024g/78E9A8FA 1999-06-04
```

2. **Exporting a public key**. To send your public key to others you must first export it. List the command-line to export your public key. (You can distribute your public key using email or pen drive).

To send your public key to a correspondent you must first export it. The command-line option --export is used to do this. It takes an additional argument identifying the public key to export. As with the --gen-revoke option, either the key ID or any part of the user ID may be used to identify the key to export.

```
alice% gpg --output alice.gpg --export alice@cyb.org
```

3. How to export a public key in an ASCII-armored format?

The key is exported in a binary format, but this can be inconvenient when the key is to be sent though email or published on a web page. GnuPG therefore supports a command-line option --armor that causes output to be generated in an ASCII-armored format similar to uuencoded documents. In general, any output from GnuPG, e.g., keys, encrypted

documents, and signatures, can be ASCII-armored by adding the --armor option.

```
alice% gpg --armor --export alice@cyb.org
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v0.9.7 (GNU/Linux)
Comment: For info see http://www.gnupg.org

[...]
-----END PGP PUBLIC KEY BLOCK-----
```

4. **Importing a public key**. A public key of others may be added to your public keyring . List the command-line to import a public key to your public keyring.

A public key may be added to your public keyring with the --import option.

```
alice% gpg --import blake.gpg
gpg: key 9E98BC16: public key imported
gpg: Total number processed: 1
gpg:               imported: 1
alice% gpg --list-keys
/users/alice/.gnupg/pubring.gpg
---------------------------------
pub  1024D/BB7576AC 1999-06-04 Alice (Judge) <alice@cyb.org>
sub  1024g/78E9A8FA 1999-06-04

pub  1024D/9E98BC16 1999-06-04 Blake (Executioner) <blake@cyb.org>
sub  1024g/5C8CBD41 1999-06-04
```

5. Once a key is imported it should be validated. How to validate a public key?

Once a key is imported it should be validated. GnuPG uses a powerful and flexible trust model that does not require you to personally validate each key you import. Some keys may need to be personally validated, however. A key is validated by verifying the key's fingerprint and then signing the key to certify it as a valid key. A key's fingerprint can be quickly viewed with the --fingerprint command-line option, but in order to certify the key you must edit it.

```
alice% gpg --edit-key blake@cyb.org

pub  1024D/9E98BC16  created: 1999-06-04 expires: never      trust: -
/q
sub  1024g/5C8CBD41  created: 1999-06-04 expires: never
(1)  Blake (Executioner) <blake@cyb.org>

Command> fpr
pub  1024D/9E98BC16 1999-06-04 Blake (Executioner) <blake@cyb.org>
             Fingerprint: 268F 448F CCD7 AF34 183E  52D8 9BDE 1A08
9E98 BC16
```

A key's fingerprint is verified with the key's owner. This may be done in person or over the

phone or through any other means as long as you can guarantee that you are communicating with the key's true owner. If the fingerprint you get is the same as the fingerprint the key's owner gets, then you can be sure that you have a correct copy of the key.

After checking the fingerprint, you may sign the key to validate it. Since key verification is a weak point in public-key cryptography, you should be extremely careful and *always* check a key's fingerprint with the owner before signing the key.

```
Command> sign

pub  1024D/9E98BC16  created: 1999-06-04 expires: never      trust: -
/q
             Fingerprint: 268F 448F CCD7 AF34 183E  52D8 9BDE 1A08
9E98 BC16

     Blake (Executioner) <blake@cyb.org>

Are you really sure that you want to sign this key
with your key: "Alice (Judge) <alice@cyb.org>"

Really sign?
```

Once signed you can check the key to list the signatures on it and see the signature that you have added. Every user ID on the key will have one or more self-signatures as well as a signature for each user that has validated the key.

```
Command> check
uid  Blake (Executioner) <blake@cyb.org>
sig!       9E98BC16 1999-06-04   [self-signature]
sig!       BB7576AC 1999-06-04   Alice (Judge) <alice@cyb.org>
```

Encrypting and decrypting files

If you want to encrypt a message to your partner, you encrypt it using your partner's public key, and she decrypts it with her private key. If your partner wants to send you a message, she encrypts it using your public key, and you decrypt it with your private key.

1. Perform an encryption to any file and employ corresponding decryption to recover the original file. List the command-line.

A public and private key each have a specific role when encrypting and decrypting documents. A public key may be thought of as an open safe. When a correspondent encrypts a document using a public key, that document is put in the safe, the safe shut, and the combination lock spun several times. The corresponding private key is the combination that can reopen the safe and retrieve the document. In other words, only the person who holds the private key can recover a document encrypted using the associated public key.

The procedure for encrypting and decrypting documents is straightforward with this mental

model. If you want to encrypt a message to Alice, you encrypt it using Alice's public key, and she decrypts it with her private key. If Alice wants to send you a message, she encrypts it using your public key, and you decrypt it with your private key.

To encrypt a document the option --encrypt is used. You must have the public keys of the intended recipients. The software expects the name of the document to encrypt as input; if omitted, it reads standard input. The encrypted result is placed on standard output or as specified using the option --output. The document is compressed for additional security in addition to encrypting it.

```
alice% gpg --output doc.gpg --encrypt --recipient blake@cyb.org doc
```

The --recipient option is used once for each recipient and takes an extra argument specifying the public key to which the document should be encrypted. The encrypted document can only be decrypted by someone with a private key that complements one of the recipients' public keys. In particular, you cannot decrypt a document encrypted by you unless you included your own public key in the recipient list.

To decrypt a message the option --decrypt is used. You need the private key to which the message was encrypted. Similar to the encryption process, the document to decrypt is input, and the decrypted result is output.

```
blake% gpg --output doc --decrypt doc.gpg

You need a passphrase to unlock the secret key for
user: "Blake (Executioner) <blake@cyb.org>"
1024-bit ELG-E key, ID 5C8CBD41, created 1999-06-04 (main key ID
9E98BC16)

Enter passphrase:
```

2. The file may also be encrypted using symmetric encryption. The key used to secure the symmetric cipher is obtained from a passphrase supplied when the file is encrypted. Perform encryption and decryption using the symmetric cipher. List the command-line.

Documents may also be encrypted without using public-key cryptography. Instead, you use a symmetric cipher to encrypt the document. Symmetric encryption is useful for securing documents when the passphrase does not need to be communicated to others. A document can be encrypted with a symmetric cipher by using the --symmetric option.

```
alice% gpg --output doc.gpg --symmetric doc
Enter passphrase:
```

3. Do you use the same passphrase that you use in the key generation process? Give explanation.

The key used to drive the symmetric cipher is derived from a passphrase supplied when the document is encrypted, and for good security, it should not be the same passphrase that you use to protect your private key.

Making and verifying signatures

A digital signature certifies and timestamps a file. A verification of the signature will fail if the file is later modified. Creating and verifying signatures uses the public/private keypair are different from encryption and decryption. A signature is generated using the private key of the sender. The signature is verified using the corresponding public key.

1. List the command-line to make a digital signature.

A digital signature certifies and timestamps a document. If the document is subsequently modified in any way, a verification of the signature will fail. A digital signature can serve the same purpose as a hand-written signature with the additional benefit of being tamper-resistant. The GnuPG source distribution, for example, is signed so that users can verify that the source code has not been modified since it was packaged.

Creating and verifying signatures uses the public/private keypair in an operation different from encryption and decryption. A signature is created using the private key of the signer. The signature is verified using the corresponding public key. For example, Alice would use her own private key to digitally sign her latest submission to the Journal of Inorganic Chemistry. The associate editor handling her submission would use Alice's public key to check the signature to verify that the submission indeed came from Alice and that it had not been modified since Alice sent it. A consequence of using digital signatures is that it is difficult to deny that you made a digital signature since that would imply your private key had been compromised.

The command-line option --sign is used to make a digital signature. The document to sign is input, and the signed document is output.

```
alice% gpg --output doc.sig --sign doc

You need a passphrase to unlock the private key for
user: "Alice (Judge) <alice@cyb.org>"
1024-bit DSA key, ID BB7576AC, created 1999-06-04

Enter passphrase:
```

The document is compressed before being signed, and the output is in binary format.

2. Given a signed file, you can either check the signature or check the signature and recover the original document. List the command-line to check the signature and to verify the signature and extract the file.

Given a signed document, you can either check the signature or check the signature and recover the original document. To check the signature use the --verify option. To verify the signature and extract the document use the --decrypt option. The signed document to verify and recover is input and the recovered document is output.

```
blake% gpg --output doc --decrypt doc.sig
gpg: Signature made Fri Jun  4 12:02:38 1999 CDT using DSA key ID
BB7576AC
gpg: Good signature from "Alice (Judge) <alice@cyb.org>"
```

Reference:

The GNU Privacy Handbook, Copyright © 1999 by The Free Software Foundation, http://www.gnupg.org/gph/en/manual.html