PGP key operation Hands on Exercise.

JPCERT/CC

1. Before starting to learn.
   i. Check the version of your GnuPG
      ① Check your GnuPG version.

```
$
$ gpg --version
gpg (GnuPG) 1.4.6
Copyright (C) 2006 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.

Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
$
```

Note: Please check the “Supported algorithms” and find what you can do.
This document is written with GnuPG 1.4.6

      ② Use “gpg –help” or “man gpg” for manuals

```
Commands:

 -s, --sign [file]             make a signature
     --clearsign [file]        make a clear text signature
 -b, --detach-sign             make a detached signature
 -e, --encrypt                 encrypt data
 -c, --symmetric               encryption only with symmetric cipher
 -d, --decrypt                 decrypt data (default)
     --verify                  verify a signature
     --list-keys               list keys
     --list-sigs               list keys and signatures
     --check-sigs              list and check key signatures
     --fingerprint             list keys and fingerprints
 -K, --list-secret-keys        list secret keys
     --gen-key                 generate a new key pair
     --delete-keys             remove keys from the public keyring
     --delete-secret-keys      remove keys from the secret keyring
     --sign-key                sign a key
     --lsign-key               sign a key locally
     --edit-key                sign or edit a key
     --gen-revoke              generate a revocation certificate
     --export                  export keys
     --send-keys               export keys to a key server
     --recv-keys               import keys from a key server
     --search-keys             search for keys on a key server
     --refresh-keys            update all keys from a keyserver
     --import                  import/merge keys
     --card-status             print the card status
     --card-edit               change data on a card
     --change-pin              change a card's PIN
     --update-trustdb          update the trust database
     --print-md algo [files]   print message digests
```

2. Create public & private key pairs for GnuPG.

Summary of steps

① Type “gpg --gen-key”

```
$
$ gpg --gen-key
gpg (GnuPG) 1.4.6; Copyright (C) 2006 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)
   (5) RSA (sign only)
Your selection?
```

Find the above screen and choose “algorithm” of the encryption.

At this time, we’ll choose “DSA and Elgamal” as a default.

② Some people say that 1024 bit not strong enough anymore. So we’ll choose 2048bit for this time. After that we’ll have to think about the expire date of the key pars.

```
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
ELG-E 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)
```

Note: It is important to select expire period. It is basically up to your security policy to decide this one. Several organization operate with 1 year. If you choose one year for this, you have to notify to users about the changing of the keys.

③ Type your “Real name” and “e-mail address” for this.

```
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
ELG-E 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) 1y
Key expires at 2009年02月20日 19時55分52秒 JST
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: Keisuke Kamata
Email address: k-kamata@jpcert.or.jp
Comment:
You selected this USER-ID:
    "Keisuke Kamata <k-kamata@jpcert.or.jp>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

Note: Please keep in mind that anyone can make your keys of e-mail address. So what is the way that you can make sure that your key belongs your key ? The answer is “fingerprint”.

④ Enter passphrase for 1st time

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

Enter passphrase:
```

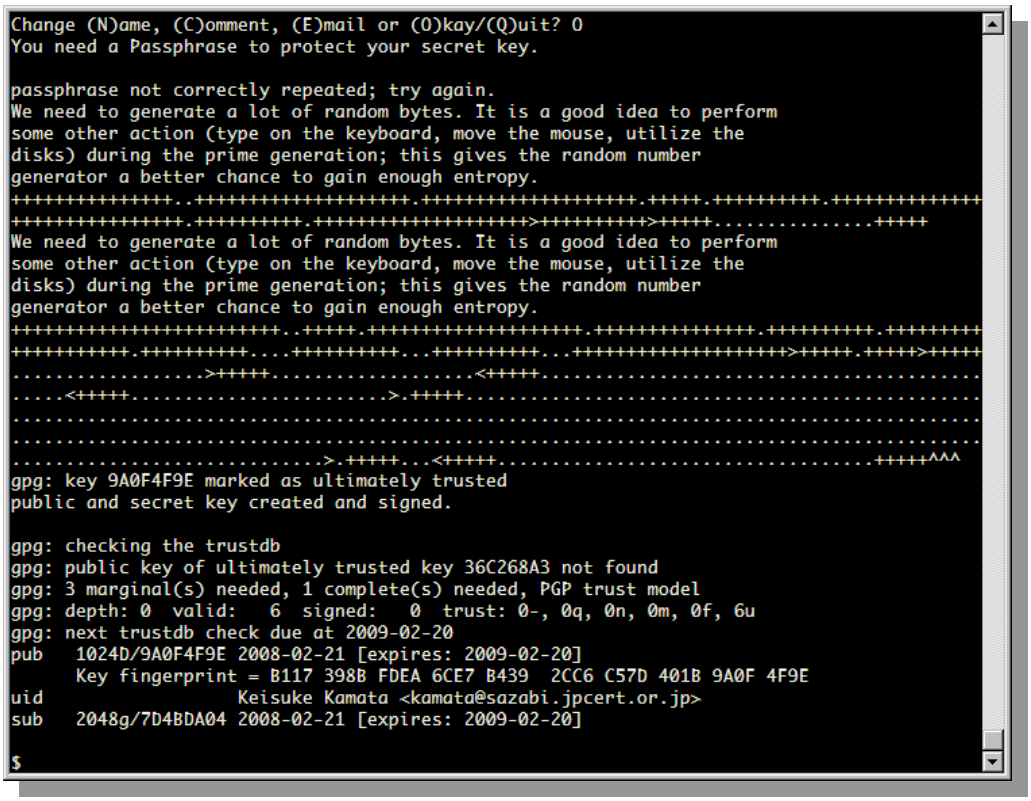
And 2nd time

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

Repeat passphrase:
```

Note: Please do not forget this password and make sure the password is strong enough for brute forcing.

⑤ GnuPG automatically generate keys.



Note1: When generating the key pairs, the operating system needs many random numbers. It is recommended to do something on the system for that.

Note2: Read these messages carefully and should know the contents below

- Key ID
- What is the “trust”
- Key Length
- Expires date
- Key fingerprint

⑥ See your keys

```
$ gpg --list-keys 9A0F4F9E
pub   1024D/9A0F4F9E 2008-02-21 [expires: 2009-02-20]
uid                  Keisuke Kamata <kamata@sazabi.jpcert.or.jp>
sub   2048g/7D4BDA04 2008-02-21 [expires: 2009-02-20]

$ gpg --list-keys kamata@sazabi.jpcert.or.jp
pub   1024D/22A2313F 2005-08-25
uid                  Keisuke KAMATA <kamata@sazabi.jpcert.or.jp>
sub   2048g/6A7FFC0A 2005-08-25

pub   1024D/9A0F4F9E 2008-02-21 [expires: 2009-02-20]
uid                  Keisuke Kamata <kamata@sazabi.jpcert.or.jp>
sub   2048g/7D4BDA04 2008-02-21 [expires: 2009-02-20]

$
```

Note: Please remember the option "gpg --list-keys" you can list keys in your keyrings. And you can use both Key ID and e-mail address. But, sometimes e-mail address can not determine unique Keys. Because of e-mail confliction problem like below:

k-kamata@jpcert.or.jp vs kamata@jpcert.or.jp

you can not determine these two by using kamata@jpcert.or.jp

⑦ Where is the key files ?

```
$
$ cd ~
$ cd .gnupg/
$ ls -l
合計 100
-rw-------  1 kamata kamata  8084 2005-08-25 17:19 gpg.conf
-rw-------  1 kamata kamata 70430 2008-02-21 20:09 pubring.gpg
-rw-------  1 kamata kamata   600 2008-02-21 20:09 random_seed
-rw-------  1 kamata kamata  5054 2008-02-21 20:09 secring.gpg
-rw-------  1 kamata kamata  1800 2008-02-21 20:09 trustdb.gpg
$
```

Just under the ".gnupg" directory of your home directory.

Public keys stored in : pubring.gpg

Private keys are stored in : secring.gpg

You can choose your favorite option in : gpg.conf

➔ Please see the manual for more detail ☺

3. Sign messages and verify it.

Let's try to sign some text message with your privte key.

① Create file for encryption

```
$ echo "This is a test message." > test_sign
$ echo "I hope I can sign well." >> test_sign
$ cat test_sign
This is a test message.
I hope I can sign well.
$
$
```

② Let's sign the file

1. Type as below and type your long passphrase too.

```
$ gpg --clearsign test_sign

You need a passphrase to unlock the secret key for
user: "Keisuke Kamata <kamata@test>"
1024-bit DSA key, ID 89FF169E, created 2008-02-21

Enter passphrase:
```

2. After typing your passphrase correctly, please try the "ls –l" and find the file "test_sign.asc". That is a signed file. Lt's see the inside of file.

```
$ gpg --clearsign test_sign

You need a passphrase to unlock the secret key for
user: "Keisuke Kamata <kamata@test>"
1024-bit DSA key, ID 89FF169E, created 2008-02-21

$ ls -l
合計 8
-rw-r--r--  1 keycre users  48 2008-02-21 20:55 test_sign
-rw-r--r--  1 keycre users 284 2008-02-21 20:58 test_sign.asc
$ cat test_sign.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a test message.
I hope I can sign well.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQFHvWduyap2O4n/Fp4RAmk5AKCuS0y0TZEEfXG0kIxXRgyaUdByxgCgscM8
6opYfzI1sHHhWWb3nnn90t0=
=O7Ai
-----END PGP SIGNATURE-----
$
```

3. The verify process.

```
$ gpg --verify test_sign.asc
gpg: Signature made 2008年02月21日 20時58分38秒 JST using DSA key ID 89FF169E
gpg: Good signature from "Keisuke Kamata <kamata@test>"
$
```

Note: Please find the message “Good signature from” and that is a message that gpg command can successfully verify the message. That means the file is surely signed by your private keys.

4. If the file is not correctly signed…

```
$ gpg --verify test_sign.asc
gpg: Signature made 2008年02月21日 20時58分38秒 JST using DSA key ID 89FF169E
gpg: BAD signature from "Keisuke Kamata <kamata@test>"
$
```

Note: You may find the message “BAD signature from” that means the file may be altered by someone. Do you want to see the inside ?

```
$ cat test_sign.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

saaThis is a test message.
I hope I can sign well.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQFHvWduyap2O4n/Fp4RAmk5AKCuS0y0TZEEfXG0kIxXRgyaUdByxgCgscM8
6opYfzI1sHHhWWb3nnn90t0=
=O7Ai
-----END PGP SIGNATURE-----
$
```

See and find the difference of the file contents.

4. Import public keys for your key rings.

   If you want to send some encrypted messages, you need to get the public key of that e-mail address. How to get ?

   ① Searching by PGP key server

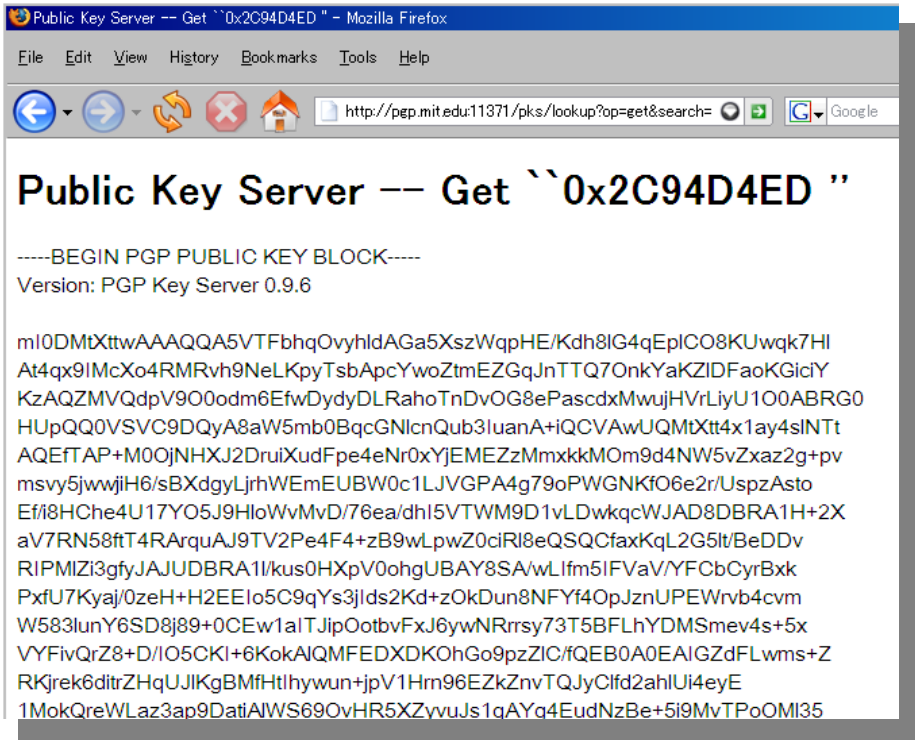
      1. http://pgp.mit.edu
      2. http://pgp.nic.ad.jp

         ※ These key servers are typically synchronized.

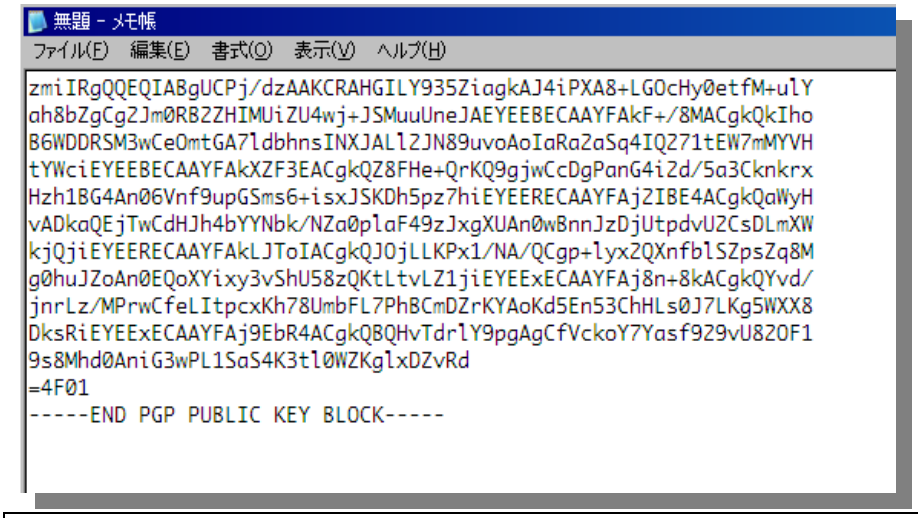
   ② Try to find key for info@jpcert.or.jp with pgp.mit.edu

Which is suitable for you ? Try to click the “2C94D4ED”

③ Get the key data in “ASCII” characters.



④ Type Ctrl-A and Ctrl-C on the browser, then open your notepad on your windows, then Ctrl-V. You’ll find that you’ve got the key data in your buffer



Note: The characters ‘-----END PGP PUBLIC KEY BLOCK----‘ is a special sign for telling that is the end of PGP key data. It is important to include this part if you import to your key ring.

⑤ There are two main ways for importing the public key.

1. Type “gpg --import” and paste the copied key data to the terminal.

```
$
$ gpg --import
Public Key Server -- Get ``0x2C94D4ED ''

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP Key Server 0.9.6

mI0DMtXttwAAAQQA5VTFbhqOvyhldAGa5XszWqpHE/Kdh8lG4qEplCO8KUwqk7Hl
At4qx9IMcXo4RMRvh9NeLKpyTsbApcYwoZtmEZGqJnTTQ7OnkYaKZlDFaoKGiciY
KzAQZMVQdpV9O0odm6EfwDydyDLRahoTnDvOG8ePascdxMwujHVrLiyU1O0ABRG0
HUpQQ0VSVC9DQyA8aW5mb0BqcGNlcnQub3IuanA+iQCVAwUQMtXtt4x1ay4slNTt
AQEfTAP+M0OjNHXJ2DruiXudFpe4eNr0xYjEMEZzMmxkkMOm9d4NW5vZxaz2g+pv
msvy5jwwjiH6/sBXdgyLjrhWEmEUBW0c1LJVGPA4g79oPWGNKfO6e2r/UspzAsto
Ef/i8HChe4U17YO5J9HloWvMvD/76ea/dhI5VTWM9D1vLDwkqcWJAD8DBRA1H+2X
aV7RN58ftT4RArquAJ9TV2Pe4F4+zB9wLpwZ0ciRl8eQSQCfaxKqL2G5lt/BeDDv
RIPMlZi3gfyJAJUDBRA11l/kus0HXpV0ohgUBAY8SA/wLIfm5IFVaV/YFCbCyrBxk
PxfU7Kyaj/0zeH+H2EEIo5C9qYs3jIds2Kd+zOkDun8NFYf4OpJznUPEWrvb4cvm
W583lunY6SD8j89+0CEw1aITJipOotbvFxJ6ywNRrrsy73T5BFLhYDMSmev4s+5x
VYFivQrZ8+D/IO5CKI+6KokAlQMFEDXDKOhGo9pzZlC/fQEB0A0EAIGZdFLwms+Z
RKjrek6ditrZHqUJlKgBMfHtIhywun+jpV1Hrn96EZkZnvTQJyClfd2ahlUi4eyE
1MokQreWLaz3ap9DatiAlWS69OvHR5XZyvuJs1qAYg4EudNzBe+5i9MvTPoOMl35
```

Note: Type Ctrl-D after pasting the copy buffer. You’ll get message as below

```
=wPpd
-----END PGP PUBLIC KEY BLOCK-----

gpg: key 2C94D4ED: public key "JPCERT/CC <office@jpcert.or.jp>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
$
```

2. Import from file. Please remember that [FILENAME] should be written in real filename.

```
$ cat hoge
$ gpg --import [FILENAME]
gpg: key 2C94D4ED: "JPCERT/CC <office@jpcert.or.jp>" 3 new signatures
gpg: Total number processed: 1
gpg:          new signatures: 3
gpg: public key of ultimately trusted key 36C268A3 not found
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   6  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 6u
gpg: next trustdb check due at 2009-02-20
$

$
```

⑥ Please find that the public key is really imported.

```
$ gpg --list-keys 2C94D4ED
pub   1024R/2C94D4ED 1997-01-10
uid                  JPCERT/CC <office@jpcert.or.jp>
uid                  JPCERT/CC <info@jpcert.or.jp>

$
```

Note: Make sure that the fingerprint is right.
Use " $ gpg --fingerprint [KeyID]" to show the fingerprint.

5. Encrypt messages.

① Make sure which e-mail address you want to send. You are going to send to ww-info@jpcert.or.jp you can get the key from the URL below:
http://pgp.nic.ad.jp/pks/lookup?op=get&search=0x298F386F

```
$ gpg --list-keys
/irt/home/keycre/.gnupg/pubring.gpg
---------------------------------
pub   1024R/2C94D4ED 1997-01-10
uid                  JPCERT/CC <office@jpcert.or.jp>
uid                  JPCERT/CC <info@jpcert.or.jp>

$
```

② Make some file for encryption.

```
$ echo 'This is a file for encryption' > test_encrypt
$ echo 'Can you read me ?' >> test_encrypt
$
```

③ Try to encrypt with you key.

```
$ gpg --encrypt --armor -r ww-info@jpcert.or.jp test_encrypt
gpg: 8A5F3658: There is no assurance this key belongs to the named user

pub  2048g/8A5F3658 2005-08-25 JPCERT/CC WW Group <ww-info@jpcert.or.jp>
 Primary key fingerprint: 470F F413 3DCC 5D38 7CAC  3500 80C4 944B 298F 386F
      Subkey fingerprint: 8B43 64B7 795E BA1E D827  A0A4 CCB5 14A9 8A5F 3658

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
$
```

Note: Please type “y” only if you are sure to encrypt with that key.

④ Try to see the encrypted file.

```
$ cat test_encrypt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.6 (GNU/Linux)

hQIOA8y1FKmKXzZYEAf6AqALpgrs0f35e8KyDN4Qk8Vx0K5Z0NfhfZw9eKudoLiU
L1eVnScauq6/qcL/3PTNRxCzsIwzdgwKBJZDPAQnNyD2chM/E/JKeiUjMrSw9LO6
AUhKocMyqzRtDLwb+3bQbFV1fI6dQUIbrwGxYUPk9E0Jdnc7bt14Hd0Vi/0aDFfd
1a7EQWB5XdRJPsE/xzkKBz5A3P578fsYNpjjekkyFcpqLusMAKlCkf/PfG4rUNv+
dOGnO6RixRvVGoVFYxkKOSu5PimJEOo2qGPhmmD/gOjN+qAW2zAhwzTofaybLcqy
DLx+foosrlbnpSLhxQ5tBiFow66JAKSu1BaSimksTQf9GacBBtSXU71SIqV84Zcl
WXEG2aBy9KEq8fl2DH3he3ReS+hJF64S/ZSA537ZopISB7CcsQXEDFaXOqs3ORqO
MBic7UOuS5cszR1JAZtdLEyERxuwpcV/QYRS8kKxRfqJRCl/ZLxeWfm+u3KgNWJj
79sIP2HcZStGHAIC4NdZAsKlR7LXZEGqvsZLdSMcOu3/AgrPIFs0t1/fqzLOe1FT
jVZYfhcFdd+hBKvCYQ0JmFrrUDZMq8XvmfBR9HI0WAZxX6QuSf0rXEjaxEtIuBiH
OYZAMY9AwU7SZ8HBZzz/oGzhU5yFqNejgobvzXryK3+OpRxJlUllS1kmmvJ0ENlf
YdJwAUaXATcUupZMQQ/8vOSquTJkV4cBmHTuOCpddFcW2DvAfkJsGvETGz0/U9iL
Tu12Ti0oyEd4lUDI3x2IQAgbp7myuE9WhpcuA6DY4+E794UOpQ37KLls4Ga7EzY6
YAxYGkYB8ebajYkGz2uWlHRg6Q==
=NsHT
-----END PGP MESSAGE-----
$
```

Note: You can not decrypt this file. Do you know why? There is a way to make encrypted message that you can decrypt with command as below

gpg --encrypt --armor -r ww-info@jpcert.or.jp -r kamata@test test_encrypt

Note: You can use "-r" option to encrypt multiple public keys as many as you want.

6. Sign and encrypt

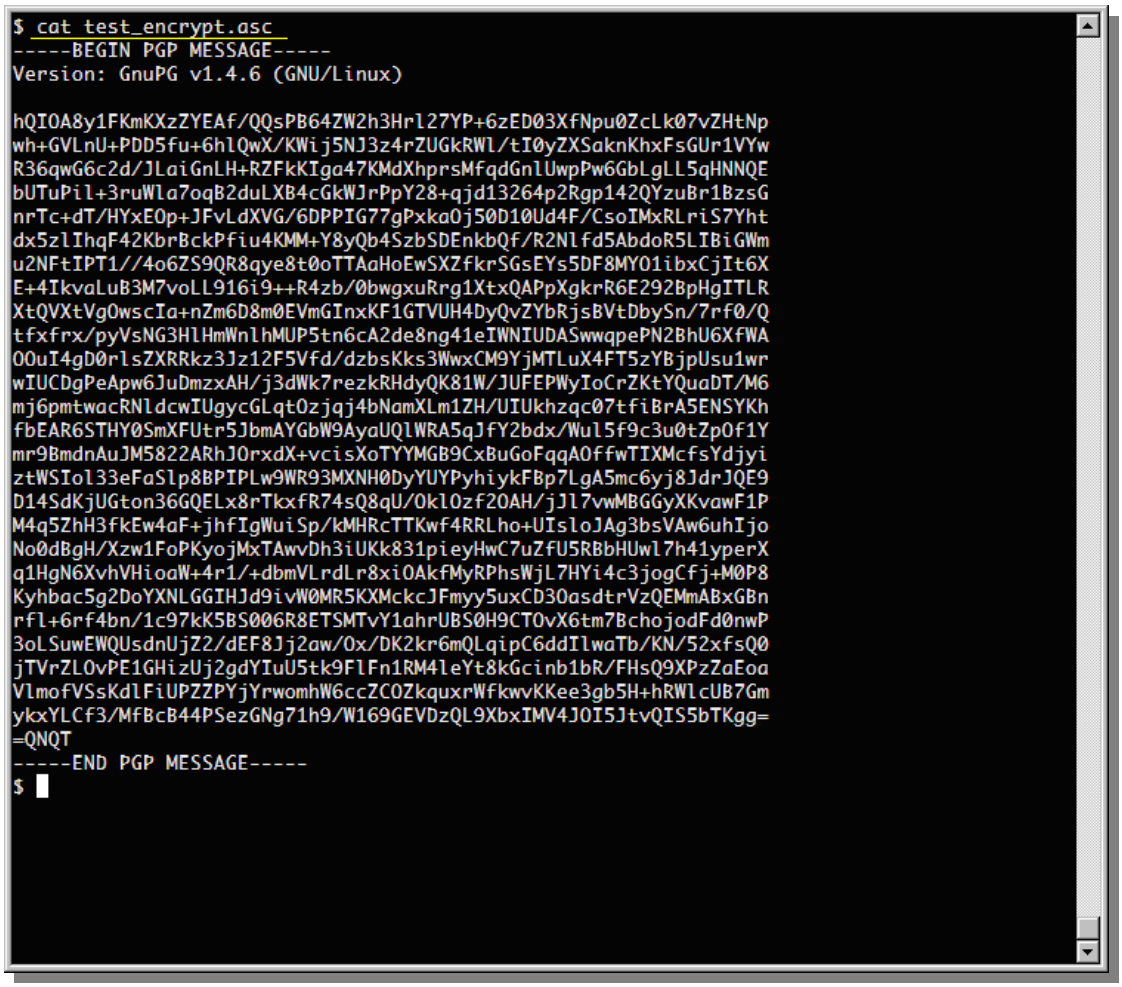
   It should be very easy for you now.

```
$ echo "test test test" > test_encrypt_sign
$ gpg --encrypt --armor --sign -r ww-info@jpcert.or.jp -r kamata@test test_encrypt
gpg: error checking usability status of 89FF169E
gpg: key 89FF169E: secret key without public key - skipped

You need a passphrase to unlock the secret key for
user: "Keisuke Kamata <kamata@test>"
1024-bit DSA key, ID 45E7C146, created 2008-02-21

Enter passphrase:
```

   You can get signed and encrypted message.


$ cat test_encrypt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.6 (GNU/Linux)
hQIOA8y1FKmKXzZYEAf/QQsPB64ZW2h3Hrl27YP+6zED03XfNpu0ZcLk07vZHtNp
wh+GVLnU+PDD5fu+6hlQwX/KWij5NJ3z4rZUGkRWl/tI0yZXSaknKhxFsGUr1VYw
R36qwG6c2d/JLaiGnLH+RZFkKIga47KMdXhprsMfqdGnlUwpPw6GbLgLL5qHNNQE
bUTuPil+3ruWla7oqB2duLXB4cGkWJrPpY28+qjd13264p2Rgp142QYzuBr1BzsG
nrTc+dT/HYxEOp+JFvLdXVG/6DPPIG77gPxkaOj50D10Ud4F/CsoIMxRLriS7Yht
dx5zlIhqF42KbrBckPfiu4KMM+Y8yQb4SzbSDEnkbQf/R2Nlfd5AbdoR5LIBiGWm
u2NFtIPT1//4o6ZS9QR8qye8t0oTTAaHoEwSXZfkrSGsEYs5DF8MYO1ibxCjIt6X
E+4IkvaLuB3M7voLL916i9++R4zb/0bwgxuRrg1XtxQAPpXgkrR6E292BpHgITLR
XtQVXtVgOwscIa+nZm6D8m0EVmGInxKF1GTVUH4DyQvZYbRjsBVtDbySn/7rf0/Q
tfxfrx/pyVsNG3HlHmWnlhMUP5tn6cA2de8ng41eIWNIUDASwwqpePN2BhU6XfWA
OOuI4gD0rlsZXRRkz3Jz12F5Vfd/dzbsKks3WwxCM9YjMTLuX4FT5zYBjpUsu1wr
wIUCDgPeApw6JuDmzxAH/j3dWk7rezkRHdyQK81W/JUFEPWyIoCrZKtYQuaDT/M6
mj6pmtwacRNldcwIUgycGLqtOzjqj4bNamXLm1ZH/UIUkhzqc07tfiBrA5ENSYKh
fbEAR6STHY0SmXFUtr5JbmAYGbW9AyaUQlWRA5qJfY2bdx/Wul5f9c3u0tZpOf1Y
mr9BmdnAuJM5822ARhJOrxdX+vcisXoTYYMGB9CxBuGoFqqAOffwTIXMcfsYdjyi
ztWSIol33eFaSlp8BPIPLw9WR93MXNH0DyYUYPyhiykFBp7LgA5mc6yj8JdrJQE9
D14SdKjUGton36GQELx8rTkxfR74sQ8qU/OklOzf2OAH/jJl7vwMBGGyXKvawF1P
M4q5ZhH3fkEw4aF+jhfIgWuiSp/kMHRcTTKwf4RRLho+UIsloJAg3bsVAw6uhIjo
No0dBgH/Xzw1FoPKyojMxTAwvDh3iUKk831pieyHwC7uZfU5RBbHUwl7h41yperX
q1HgN6XvhVHioaW+4r1/+dbmVLrdLr8xiOAkfMyRPhsWjL7HYi4c3jogCfj+M0P8
Kyhbac5g2DoYXNLGGIHJd9ivW0MR5KXMckcJFmyy5uxCD3OasdtrVzQEMmABxGBn
rfl+6rf4bn/1c97kK5BS006R8ETSMTvY1ahrUBS0H9CTOvX6tm7BchojodFd0nwP
3oLSuwEWQUsdnUjZ2/dEF8Jj2aw/Ox/DK2kr6mQLqipC6ddIlwaTb/KN/52xfsQ0
jTVrZLOvPE1GHizUj2gdYIuU5tk9FlFn1RM4leYt8kGcinb1bR/FHsQ9XPzZaEoa
VlmofVSsKdlFiUPZZPYjYrwomhW6ccZCOZkquxrWfkwvKKee3gb5H+hRWlcUB7Gm
ykxYLCf3/MfBcB44PSezGNg71h9/W169GEVDzQL9XbxIMV4JOI5JtvQIS5bTKgg=
=QNQT
-----END PGP MESSAGE-----
$

HOME WORK

1. Find some friend and exchange each public key.
   Hint: Export your publickey with “gpg --export --armor KeyID”

```
$ gpg --export --armor kamata@test
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)

mQGiBEe9bLARBAD7M0uG9Uzl/88VyHpSXLyabPDP6oi6GZTx4KF1WiE2BPtEMKdS
p3u+V2SAzCKWKavfOBdhumHdlQu2kHBEd8dkN2VlPPQ02KUeD9bIiKuNkXLspzHA
vuCo6wcO4xVQW8mOnPxl8iuk7S0h17Y1pcYWU81truI+GT8qNMSCpaS7vwCgqR28
se/ZNhaWhnbLUjKbVcjGDi8EAMbiVTl1edu/cQa0tfsCME4hKuBGc4bpF+KHV8wE
yGqMwEwztMCWLk2H6xG3IyPkf79RhQYgQ4v2a15oJ54H1plQ/LvjZTDKO/22oH7c
pA8Pfi+iQ4vjGLL6ATy/XrZfHK7c9tam+A7DietE9L9icspDBu0xARQVANwoJ5Zg
/Wh3BACts2tOsmlGfCrn1SKsnYcv8/zj8hlnhKuQ4NnPRaBu5s4F88ljsW1waQ3V
MJA6aWZfmp/px7xrv9eBOD+vYEiv0amUUTcqNNXHdUFDAFzni21KiE4Z0hmL+4tw
opnk0BqBmdgOpg84zdq3g2fnuzyIF4TeItohFx30BesZI5wrMLQcS2Vpc3VrZSBL
YW1hdGEgPGthbWF0YUB0ZXN0PohmBBMRAgAmBQJHvWywAhsDBQkB4TOABgsJCAcD
```

2. Try to encrypt with your friend’s public key and sign with your private key

3. Give the text data to your friend in some way (ex. E-mail)

4. Get the signed and encrypted message from your friend.

5. Try to decrypt with you private key and verify the message.

   To decrypt the file: gpg –decrypt [filename]
   To verify the file: gpg –verify [filename]

ADVANCED

1. SIGN someone's public key

   $ gpg --sign-key

```
$ gpg --sign-key
usage: gpg [options] --sign-key user-id
$ gpg --sign-key ww-info@jpcert.or.jp

pub  1024D/298F386F  created: 2005-08-25  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  2048g/8A5F3658  created: 2005-08-25  expires: never       usage: E
[ unknown] (1). JPCERT/CC WW Group <ww-info@jpcert.or.jp>

gpg: error checking usability status of 89FF169E
gpg: key 89FF169E: secret key without public key - skipped

pub  1024D/298F386F  created: 2005-08-25  expires: never       usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: 470F F413 3DCC 5D38 7CAC  3500 80C4 944B 298F 386F

     JPCERT/CC WW Group <ww-info@jpcert.or.jp>

Are you sure that you want to sign this key with your
key "Keisuke Kamata <kamata@test>" (45E7C146)

Really sign? (y/N)
```

2. SIGN for some binaries with detached signature file.

```
$ echo "hoge"> test_detach
$ gpg -b test_detach
gpg: error checking usability status of 89FF169E
gpg: key 89FF169E: secret key without public key - skipped

You need a passphrase to unlock the secret key for
user: "Keisuke Kamata <kamata@test>"
1024-bit DSA key, ID 45E7C146, created 2008-02-21

$ ls -l test_detach*
-rw-r--r--  1 keycre users  5 2008-02-21 21:58 test_detach
-rw-r--r--  1 keycre users 65 2008-02-21 21:59 test_detach.sig
$
```

Note: The .sig file is a detached signature file. See how to verify.

```
$ gpg --verify test_detach.sig  test_detach
gpg: Signature made 2008年02月21日 21時59分03秒 JST using DSA key ID 45E7C146
gpg: Good signature from "Keisuke Kamata <kamata@test>"
$
```

3. Symmetric cipher encryption

```
$ echo hoge > test_symmetric
$ gpg --symmetric test_symmetric
Enter passphrase:
```

To decrypt

```
$ gpg --decrypt test_symmetric.gpg
gpg: CAST5 encrypted data
gpg: encrypted with 1 passphrase
hoge
gpg: WARNING: message was not integrity protected
$
```