Email Security
PGP / Pretty Good Privacy

PacNOG19

28th November - 2nd December 2016

Nadi, Fiji

Issue Date: [31-12-2015]

Revision: [V.1]

APNIC

Security issues for E-mail

- Confidentiality
  - Network admin can read your e-mail.
  - Webmail provider can read your e-mail.
  - LAN user may read your e-mail by monitoring tool.
  - Even in some hotel, I could have chance to read other rooms internet traffic.
- Integrity
  - E-mail contents may be changed by some attacker on the network.
- Authenticity
  - Easy to set any e-mail headers like “From”.
  - Any other e-mail headers can be set anything you want.
  - Difficult to know it is true.

Targeted Attack

- Attacks on information security which seek to affect a specific organization or group, rather than indiscriminately. Some may be customized for a specific target organization or group.
  - An e-mail with suspicious file attached
  - Executable binary
  - Word document file
  - Database application file

Targeted Attack

To: your e-mail address

From: Fakrul Alam fakrul@dhakacom.com

Subject: my request

Hello,

I have been looking for someone who can answer questions of the attached file. I hope you can do that and reply me.

Thanks !

Example of Spoof Mail

```
    by spamwall.dhakacom.com (Postfix) with ESMTP id 70EF3BA13F8
    for <fakrul@dhakacom.com>; Tue, 12 Jun 2012 04:39:04 +0600 (BDT)
Received: (qmail 62722 invoked from network); 12 Jun 2012 05:34:59 +0700
X-Spam-Status: No, hits=7.6 required=8.0
    tests=MISSING_HEADERS,RAZOR2_CF_RANGE_51_100,RAZOR2_CF_RANGE_E8_51_100,RAZOR2_CHECK SUBJ_ALI
X-Spam-Check-By: smtp3.dnet.net.id
Received: from smtp3.dnet.net.id (HELO newwebmail.dnet.net.id) (202.148.1.233)
    by smtp3.dnet.net.id (qpsmtpd/0.84) with ESMTP; Tue, 12 Jun 2012 05:34:54 +0700
Received: from 94.41.250.182
        (SquirrelMail authenticated user raphael@dnet.net.id)
        by newwebmail.dnet.net.id with HTTP;
        Tue, 12 Jun 2012 05:34:54 +0700 (WIT)
Message-ID: <751e890ca8b32f6b6ec8b26dd484fb71.squirrel@newwebmail.dnet.net.id>
Date: Tue, 12 Jun 2012 05:34:54 +0700 (WIT)
Subject: ACCOUNT TERMINATION
From: "Dhakacom.com Mail Manager" <webadmin@dhakacom.com>
User-Agent: SquirrelMail/1.4.16
MIME-Version: 1.0
Content-Type: text/plain;charset=iso-8859-1
Content-Transfer-Encoding: 8bit
X-Priority: 3 (Normal)
Importance: Normal
X-Virus-Checked: Checked by ClamAV on smtp3.dnet.net.id
X-dhakacom-MailScanner-ID: 70EF3BA13F8.7A916
X-dhakacom-MailScanner: Found to be clean
X-dhakacom-MailScanner-From: webadmin@dhakacom.com
CC: undisclosed-recipients:;
```

Cryptography

- Symmetric and Asymmetric (public-key)
- The latter is widely accepted
- PGP is based on Asymmetric (Public-Key) Encryption

Symmetric Encryption

- Involves only one key, which is used by both the sender for encrypting and the recipient for decrypting
- Symmetric algorithms: blowfish, Triple-DES, AES (Advanced Encryption Standard), CAST (Carlisle Adams and Stafford Tavares) , IDEA (International Data Encryption Algorithm, legally restricted, but the other algorithms may be freely used)
- Problem: the means of distributing the key

Asymmetric (Public-Key) Encryption

- Solves the problem of distributing keys by using one pair of complimentary keys, one public and the other private.
- Public: freely exchanged to others without fear of compromising security.
- Private: only you have access, should be carefully protected.
- A message is encrypted to a recipient using the recipient's public key, and it can only be decrypted using the corresponding private key.

Asymmetric Encryption Refresher

- One key mathematically related to the other.
- Public key can be generated from private key. But NOT vice versa.
- If you encrypt data with the public key, you need to private key to decrypt
- You can sign data with the private key and verify the signature using the public key

Keys

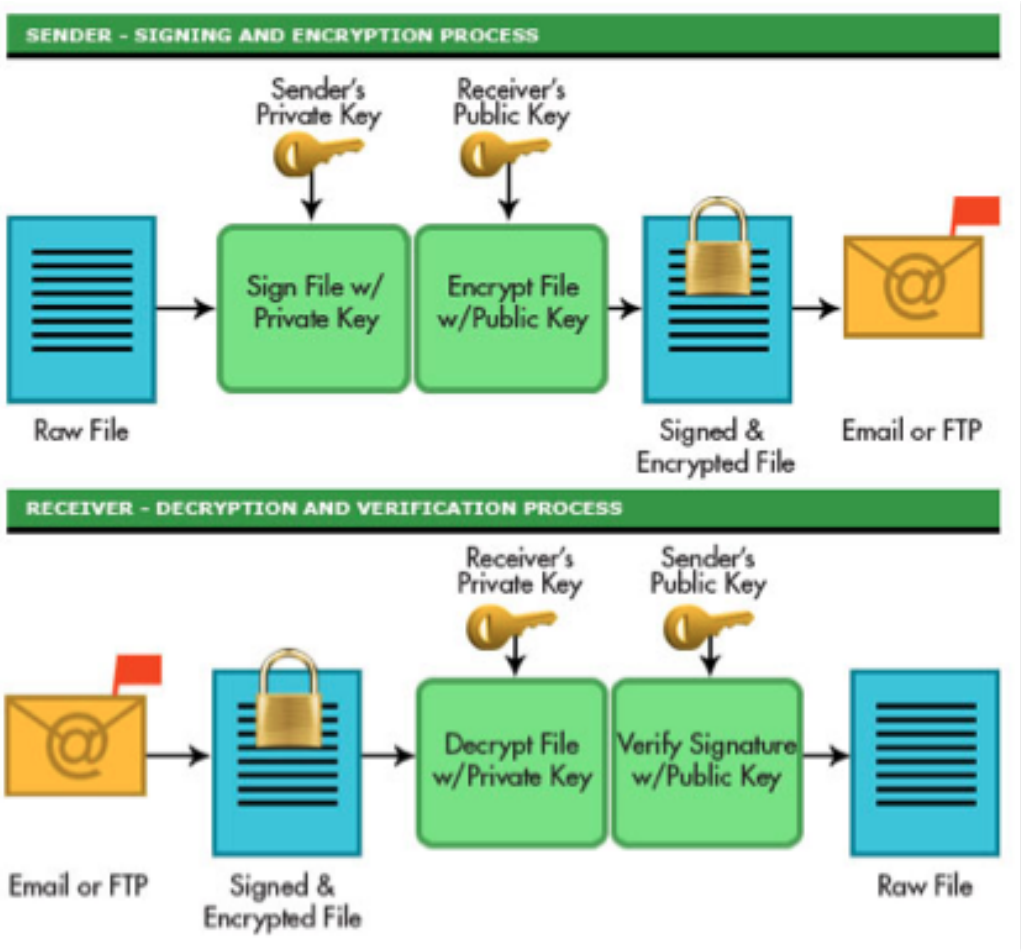
- Private key is kept SECRET.
- You should encrypt your private key with a symmetric passphrase.

- Public key is distributed.
- Anyone who needs to send you confidential data can use your public key

Signing & Encrypting

- Data is encrypted with a public key to be decrypted with the corresponding private key.
- Data can be signed with the private key to be verified by anyone who has the corresponding public key.
- Since public keys are data they can be signed too.

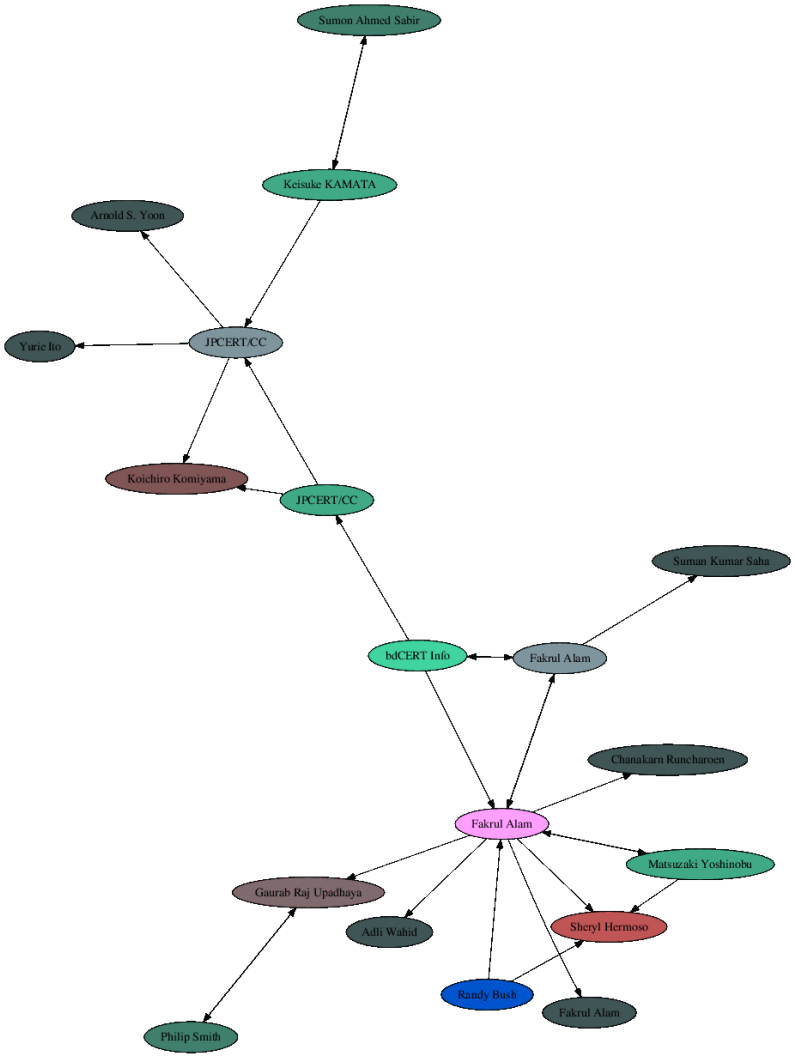
How PGP Works



Trust

- Centralized / hierarchal trust – where certain globally trusted bodies sign keys for every one else.
- Decentralized webs of trust – where you pick who you trust yourself, and decide if you trust who those people trust in turn.
- Which works better for what reasons?

Sample Web of Trust



http://localhost/pubring.gif

PGP by GnuPG

- **Create your keys**
  - Public key
  - Private key (secret key)
- **Identify key by**
  - Key ID (like 0x23AD8EF6)
- **Verify others' public key by**
  - Key fingerprint
- **Find keys on PGP key servers**
  - Like http://pgp.mit.edu

Key Management

- Using graphical tools based on what you installed above:
  - GPG Keychain Access for OS X
  - Kleopatra or GPA for windows
- Using the command line:
  - gpg --list-keys

Key Management

- On printed media: published book or business cards:
- Digitally in email or using sneaker-net
- Online using the openpgp key servers.
- Still does not tell you if you trust the key.

Key Management

- Expiry dates ensure that if your private key is compromised they can only be used till they expire.
- Can be changed after creating the key.
- Before expiry, you need to create a new key, sign it with the old one, send the signed new one to everyone in your web of trust asking them to sign your new key.

Key Management - Revocation

- Used to mark a key as invalid before its expiry date.
- Always generate a revocation certificate as soon as you create your key.
- Do not keep your revocation certificate with your private key.
- gpg --gen-revoke IDENTITY

Key Management - Partying

- Key signing parties are ways to build webs of trust.
- Each participant carries identification, as well as a copy of their key fingerprint. (maybe some $ as well ☺)
- Each participant decides if they’re going to sign another key based on their personal policy.
- Keys are easiest kept in a keyring on an openpgp keyserver in the aftermath of the party.

Installing GnuPG Software

- Core software either commercial from pgp or opensource from gnupg.
  - https://www.gpg4win.org/ for windows
  - https://www.gpgtools.org/ for OS X
- Your package manager for Linux/UNIX
  - Source code from https://www.gnupg.org/

How PGP Works

- Check your GnuPG version

```
fakrul@rnd:~$
fakrul@rnd:~$ gpg --version
gpg (GnuPG) 1.4.12
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://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: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128,
        CAMELLIA192, CAMELLIA256
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
fakrul@rnd:~$
```

How PGP Works

- 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
```

Create Public & Private key pairs for GnuPG.

- Create Public & Private key pairs for GnuPG.

```
fakrul@rnd:~$ gpg --gen-key
gpg (GnuPG) 1.4.12; Copyright (C) 2012 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.

gpg: directory `/home/fakrul/.gnupg' created
gpg: new configuration file `/home/fakrul/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/fakrul/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/home/fakrul/.gnupg/secring.gpg' created
gpg: keyring `/home/fakrul/.gnupg/pubring.gpg' created
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?
```

Find the above screen and choose “algorithm” of the encryption. At this time, we’ll choose “RSA and RSA” as a default.

Create public & private key pair

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

```
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)
```

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.

Create public & private key pair

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

```
Key is valid for? (0) 1y
Key expires at Wed 30 Apr 2014 05:45:23 PM BDT
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: Fakrul Alam
Email address: fakrul@dhakacom.com
Comment: Fakrul Alam / PGP Key
You selected this USER-ID:
    "Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>"

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”.

Create public & private key pair

- Enter passphrase for 1st time & repeat it.

```
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.

Create public & private key pair

- GnuPG automatically creates the keys

```
Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 284 more bytes)
.+++++
.....+++++
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.

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 92 more bytes)
.+++++

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 111 more bytes)
..+++++
gpg: /home/fakrul/.gnupg/trustdb.gpg: trustdb created
gpg: key B2CF94E5 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2014-04-30
pub   2048R/B2CF94E5 2013-04-30 [expires: 2014-04-30]
      Key fingerprint = 0302 768A C6F3 8EB3 3ED2  C511 FE72 5A7A B2CF 94E5
uid                  Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>
sub   2048R/33D42A92 2013-04-30 [expires: 2014-04-30]
```

Note 1: When generating the key pairs, the operating system needs many random numbers. It is recommended to do something on the system for that.

Note 2: Read these messages carefully and should know the contents below

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

Create public & private key pair

- List your keys

```
fakrul@rnd:~$ gpg --list-keys B2CF94E5
pub   2048R/B2CF94E5 2013-04-30 [expires: 2014-04-30]
uid                  Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>
sub   2048R/33D42A92 2013-04-30 [expires: 2014-04-30]

fakrul@rnd:~$ gpg --list-keys fakrul@dhakacom.com
pub   2048R/B2CF94E5 2013-04-30 [expires: 2014-04-30]
uid                  Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>
sub   2048R/33D42A92 2013-04-30 [expires: 2014-04-30]
```

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.

Create public & private key pair

- Where is the key files

```
fakrul@rnd:~$ cd .gnupg/
fakrul@rnd:~/.gnupg$ ls -lah
total 40K
drwx------  2 fakrul fakrul 4.0K Apr 30 18:00 .
drwxr-xr-x 34 fakrul fakrul 4.0K Apr 30 17:43 ..
-rw-------  1 fakrul fakrul 9.0K Apr 30 17:43 gpg.conf
-rw-------  1 fakrul fakrul 1.2K Apr 30 17:57 pubring.gpg
-rw-------  1 fakrul fakrul 1.2K Apr 30 17:57 pubring.gpg~
-rw-------  1 fakrul fakrul  600 Apr 30 17:57 random_seed
-rw-------  1 fakrul fakrul 2.6K Apr 30 17:57 secring.gpg
-rw-------  1 fakrul fakrul 1.3K Apr 30 17:57 trustdb.gpg
fakrul@rnd:~/.gnupg$
```

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

Sign messages & verify it

- Create file for encryption

```
fakrul@rnd:~/.gnupg$
fakrul@rnd:~/.gnupg$ echo "This is a test message." > test_sign
fakrul@rnd:~/.gnupg$ echo "Hope we can sign it." >> test_sign
fakrul@rnd:~/.gnupg$ cat test_sign
This is a test message.
Hope we can sign it.
fakrul@rnd:~/.gnupg$
```

- Sign the file

```
fakrul@rnd:~/.gnupg$ gpg --clearsign test_sign

You need a passphrase to unlock the secret key for
user: "Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>"
2048-bit RSA key, ID B2CF94E5, created 2013-04-30

Enter passphrase:
```

Sign messages & verify it

- After typing your passphrase correctly, please try the “ls –l” and find the file “test_sign.asc”. That is a signed file. Let’s see the inside of file.

```
fakrul@rnd:~/.gnupg$ ls
gpg.conf  pubring.gpg  pubring.gpg~  random_seed  secring.gpg  test_sign  test_sign.asc  trustdb.gpg
fakrul@rnd:~/.gnupg$ cat test_sign.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a test message.
Hope we can sign it.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.12 (GNU/Linux)

iQEcBAEBAgAGBQJRf7QcAAoJEP5yWnqyz5TlrY4H/i4eft0bBu310tUvG+4cAvGl
OVj7vLkB/Ty8jkaCFIpzP1lYrhaqcjTVSwCwXQ77SEa5hrRN7Wa/sfDbLsXBLJpK
OHqzDSgTErUbT2tjhFmrVtvmfqzuE52RqZkF4YjjSJX+cysdgY/WydnVWakLFBhs
4wqcXU5lV2pPJ08HGpSwalaF21VbnyLrseYdTXAwuqn6OIybh+7gSDOVCeT9/YPu
jbZniQEhBA7fdi18juTcwP61GZ2A/gLAyPaBKrHgsyABqN/7YnFbKnAXVPUiTZc5
gF/nkqFPR5wQ9kuPCsK2Uy24WrVU+gyDdBzFtBQPFDKZR2pCXG7HIbcxcuhXG7I=
=lV5Q
-----END PGP SIGNATURE-----
```

Sign messages & verify it

- Verification Process

```
fakrul@rnd:~/.gnupg$ gpg --verify test_sign.asc
gpg: Signature made Tue 30 Apr 2013 06:07:56 PM BDT using RSA key ID B2CF94E5
gpg: Good signature from "Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>"
fakrul@rnd:~/.gnupg$
```

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.

```
fakrul@rnd:~/.gnupg$
fakrul@rnd:~/.gnupg$ gpg --verify test_sign.asc
gpg: Signature made Tue 30 Apr 2013 06:07:56 PM BDT using RSA key ID B2CF94E5
gpg: BAD signature from "Fakrul Alam (Fakrul Alam / PGP Key) <fakrul@dhakacom.com>"
fakrul@rnd:~/.gnupg$
```

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 ?

Export / Import Public Key

- Export your public key

```
fakrul@rnd:~/.gnupg$ gpg -a --export fakrul@dhakacom.com
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.12 (GNU/Linux)

mQENBFF/sO8BCAC827VLM+1PbztyPPWKTSIlOMpq45qlakdBAccZZQe9GX4YuUPi
epP3VxKAMgTKk2lAu6kRiA9VmNGhXJ4waB44VzGhyjigfuztL3RH8Olu+CJRraGj
e+76KajST4gy+hJCiDSUwU3+OJNLajVHUzPmSu6/v3LzVcxuQnhcgyD85zPqacjI
jgFVu76j6DEjrhzjd2U1fSdNoBhltfasDo5Mr6loyekXNForEljI3X7foz26aKuN
UueUICRH60CvOn4xVaLU71R76aTxZigaCQUgUCoBwdyfgsvBhQh2GgxD6mj9pGIs
/nROw6PCbaBllrxLOTHTdDlDVuvJWTsirGGhABEBAAG0OUZha3J1bCBbGFtIChG
YWtydWwgQWxhbSAvIFBHUCBLZXkpIDxmYWtydWxAZGhha2Fjb20uY29tPokBPgQT
AQIAKAUCUX+w7wIbAwUJAeEzgAYLCQgHAwIGFQgCCQoLBBYCAwECHgECF4AACgkQ
/nJaerLPlOW9qgf/X9vT9vzfX59zd4isY0xoGEzsaXvNtLi1a+Gu7kMUXNAUGxwz
gLlKJLoKN5Y9/uxCRjm+EdiEkPTJwxiKq/gVqR5fRzHkhmI4vZ4GfAHmkHlJ6BxL
6GfCOjwiLcNdffNP0eONSg9io6Q1RAAVu4PSYBxHg7i5Ohx1FzR+/Ecl7J/Pr408
cbqjeHC3LTQWZgWlBPtQbrvtfPYUcQtcq1Bal69F1ywKz7Cf3jhnTpSxZJMQL+aK
LcTDRHNfQl20rpJb81rNtvolfdANB5JjgPOGDC+szxzOSyPwCttr5sanJy3DHVaT
1wqT7d9IQCi8fVcMdDOcjbp4Om/JO/AUY1gchLkBDQRRf7DvAQgAs44En2oAcLAM
IuQuhp83D7uX4Zsp4EJGnDtCX3o+2QU4staVmVeGAZIBeOd+iIvRlsLmKmvSdDCa
4ZTU3Yrk+lVcJjUbBldFIdf3n0X1JbrJh/e0a+gWglSJVJ3OlTK0olZ6Y/tOkxkh
OU/HIbUtNrTrqPrqmXwiD08JnwaG1N3FppUQ/9mLCmHJ5vGjF0wg8ototota2acHH8
y9OgaopPsi3AEYdfgdHyHON0gxzK4BQ324wbiDppUoFR/0/OsqgYKURaWmiPr6Y0
T1WwJ3dijCHb84JuoVPvsy4XtvOJSUtfuazZ13JKR2qQo4qKkaZxYG6++3DGAujk
Nm2sa8+0QwARAQABiQElBBgBAgAPBQJRf7DvAhsMBQkB4TOAAAoJEP5yWnqyz5Tl
Q38H/00lOX71iyXONoFyn/fSUjXNHxhnEMtlA7MKHla4EdgsbpW92/hTPWEfI7qz
VR0ZFbJUjXgWOFLrd5I0BeDxTpsdhbKELGX3S1MimlogEilM4i/z1zCpKvMFG1rf
1vkUYz+oD0WxT2jsvavnE+fYM229r4bxk2ulxVgoSb6/7FwBWsUXVHrvcjHRPgHG
H/d6fWJGpIRzofOSeKm1WPvzOz/rN9/1JkEpJAmws1pTiC6ClqXAVG2X3E5oHTCL
ay8DwOWRUD+tgHwTFR6F+FpHC/4cBoI65npaXnRUeHlhggJ2NKLybDdmL9L2N/tX
2ADI8ibfMZYWKnLvgRIJmAHBpic=
=tZiU
-----END PGP PUBLIC KEY BLOCK-----
```

Note: You can export key to a file using:

```
gpg -a --export fakrul@dhakacom.com > fakrul_public.key
```

Export / Import Public Key

- Import Key

```
fakrul@rnd:~/.gnupg$ gpg --import fakrul_bdhub.key
gpg: key 109C56FC: public key "Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1   (RSA: 1)
fakrul@rnd:~/.gnupg$
```

- Find the imported key

```
fakrul@rnd:~/.gnupg$ gpg --list-key 109C56FC
pub   2048R/109C56FC 2013-02-05 [expires: 2020-02-05]
uid                  Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>
uid                  [jpeg image of size 10334]
sub   2048R/F66ACECA 2013-02-05 [expires: 2020-02-05]
```

Export / Import Public Key

- Make sure fingerprint is right

```
fakrul@rnd:~/.gnupg$ gpg --fingerprint 109C56FC
pub   2048R/109C56FC 2013-02-05 [expires: 2020-02-05]
      Key fingerprint = 94EA 86AD 428C 4072 7995  9150 E338 712B 109C 56FC
uid                  Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>
uid                  [jpeg image of size 10334]
sub   2048R/F66ACECA 2013-02-05 [expires: 2020-02-05]
```

Encrypt Message

- Make some file to encrypt

```
fakrul@rnd:~/.gnupg$ echo "This is a file for encryption" > test_encrypt
fakrul@rnd:~/.gnupg$ echo "Can you read me" >> test_encrypt
fakrul@rnd:~/.gnupg$ cat test_encrypt
This is a file for encryption
Can you read me
fakrul@rnd:~/.gnupg$
```

- Encrypt the file

```
# gpg --encrypt --armor -r RECEIVER_EMAIL_ID -u
SENDER_EMAIL_ID test_encrypt
```

```
fakrul@rnd:~/.gnupg$ gpg --encrypt --armor -r fakrul@bdhub.com -u fakrul@dhakacom.com test_encrypt
gpg: F66ACECA: There is no assurance this key belongs to the named user

pub  2048R/F66ACECA 2013-02-05 Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>
 Primary key fingerprint: 94EA 86AD 428C 4072 7995  9150 E338 712B 109C 56FC
      Subkey fingerprint: E2FB 4B8C E12A C043 A578  DB66 CAA0 09C8 F66A CECA

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
```

Encrypt Message

- Try to read encrypted message

```
fakrul@rnd:~/.gnupg$ cat test_encrypt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.12 (GNU/Linux)

hQEMA8qgCcj2as7KAQf/bMc79wwCaEl1UbdW13Dz6YEOUDaaMG9dBbNY8iK+ijfA
usow8AZJBH/L94HY83t+OzmWbMMhyXwCn3DN6VtqhfAtuNk1QFiqiTQ+njHg33cW
TT3pcwsDmqVhJlD+WuKqezY59HSWy1kNJLS7t4Tyw3ROLyj1Eyg2Og3Bv4VE2sBM
Pr3nHub6TweVHdmp7kQeW6LrLe93pjnXWtShVfvuvRuhfoV3XPfUqIX+XH679ZdU
1vZYaX1hg1rVJoV6rOgWA/IYPUon/e/n4CcEETu2TqPoTwvbs96qmSwB8FeF0dHC
QeaEHddd1zO4IO112xnGfJ3BmXuJ4s3s/dHmNDepL9JuAboumgGemMckLA1b1RIX
ClITHIX+wLA8zWj9u0Z8t9sGOS1uPNnj1IZWUH3Cc1ptT+jtEd15oPMrfx+I0bac
6gzFEbOa2OaG/hq2sUXPz+CD0FSR4xREaxAy1cNctnuCKZSyOLMGnVBMy6O9qNY=
=fB9B
-----END PGP MESSAGE-----
```

Decrypt Message

- Decrypt the file.

```
# gpg --output OUTPUT_FILE_NAME --
decrypt ENCRYPTED_FILE_NAME
```

```
FakrulMac:Downloads rapappu$ gpg --output test1.txt --decrypt test.txt.asc

You need a passphrase to unlock the secret key for
user: "Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>"
2048-bit RSA key, ID F66ACECA, created 2013-02-05 (main key ID 109C56FC)

gpg: encrypted with 2048-bit RSA key, ID F66ACECA, created 2013-02-05
      "Fakrul Alam (bdHUB pgp key) <fakrul@bdhub.com>"
```

- Read the file

```
FakrulMac:Downloads rapappu$ cat test1.txt
This is an encrypted message.
Let see if you can decrypt it.
FakrulMac:Downloads rapappu$
```

PGP Extension for Webmail

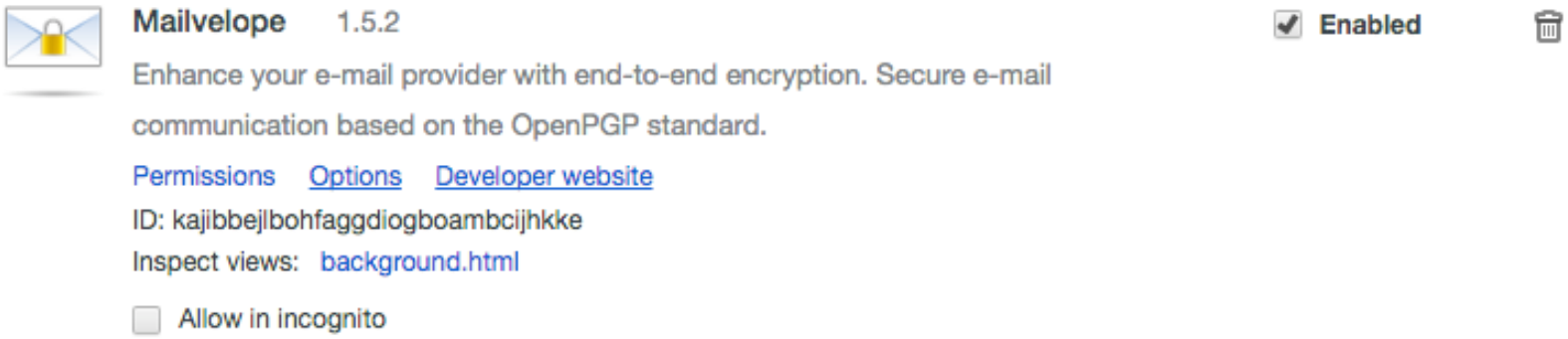
Mailvelope

APNIC

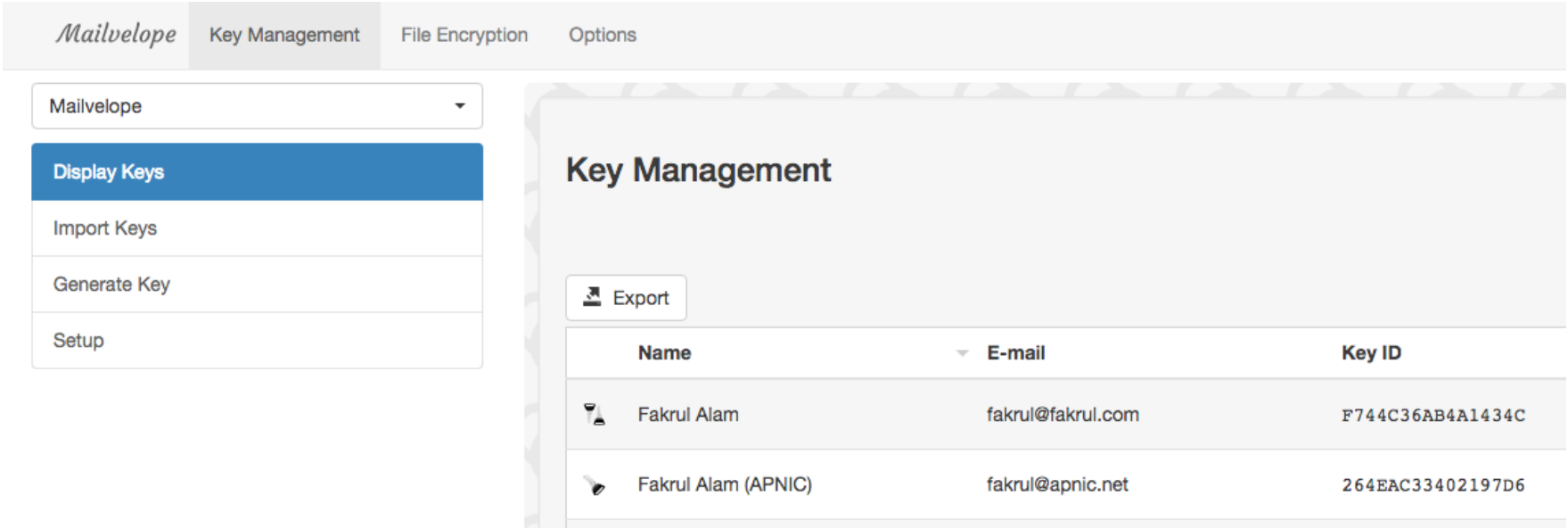
Mailvelope

- https://www.mailvelope.com/
- OpenPGP Encryption for Webmail
- Mailvelope integrate directly into the Webmail user interface
- Supports
  - Gmail
  - Yahoo
  - Outlook
  - GMX
- Browser
  - Chrome Extension
  - Firefox Addon

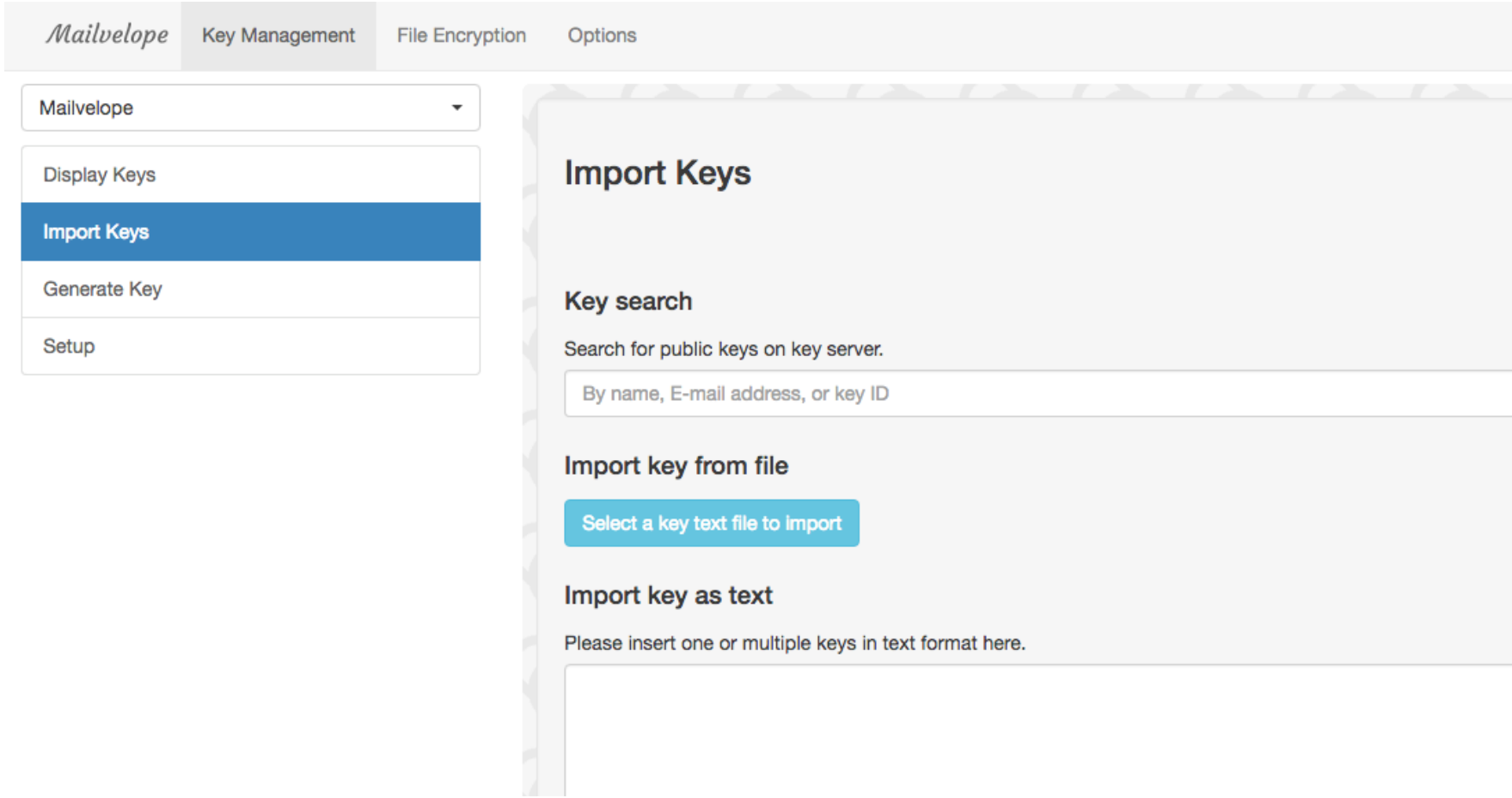
Mailvelope : Chrome Extension



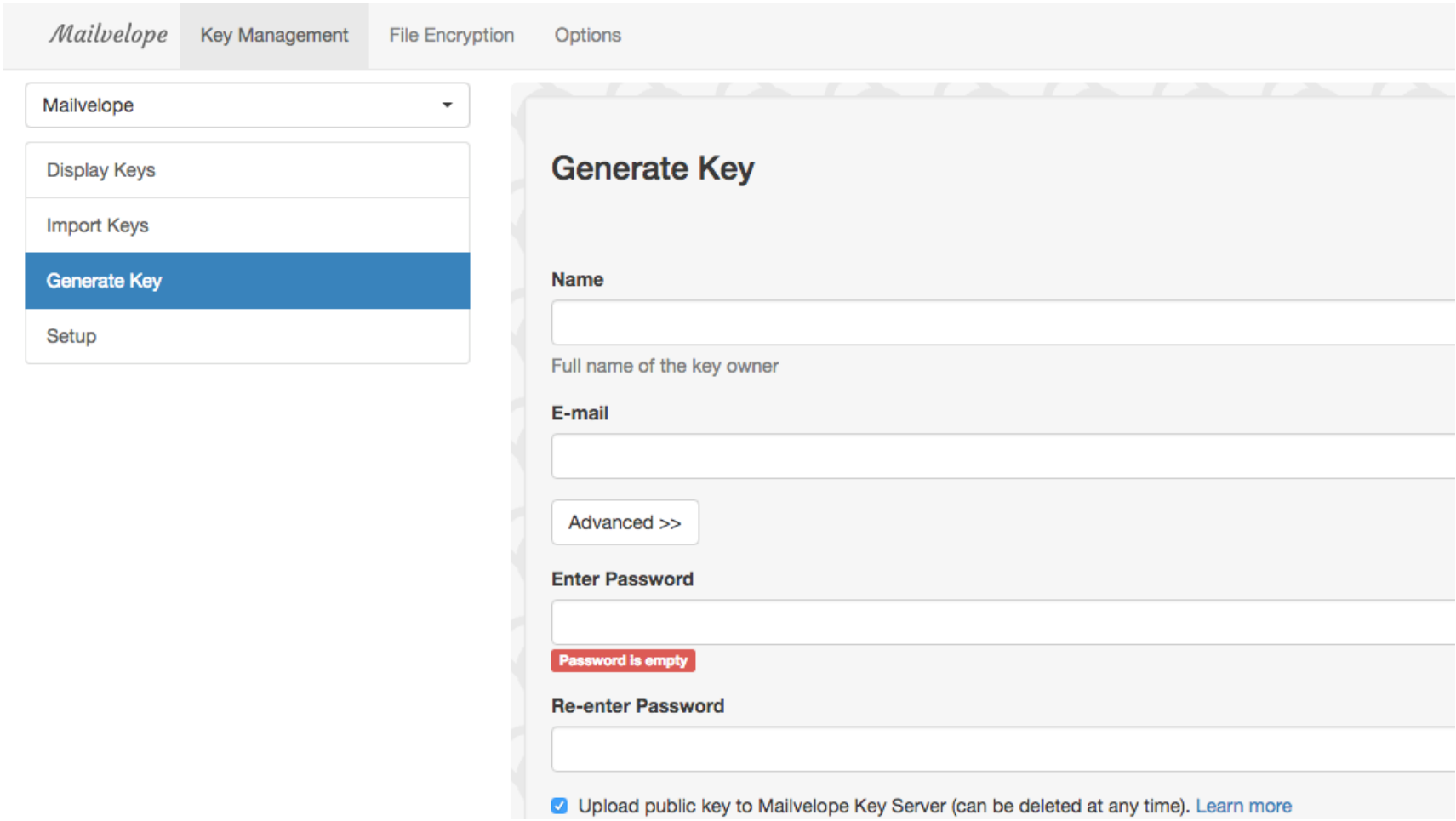
Mailvelope : Display Keys



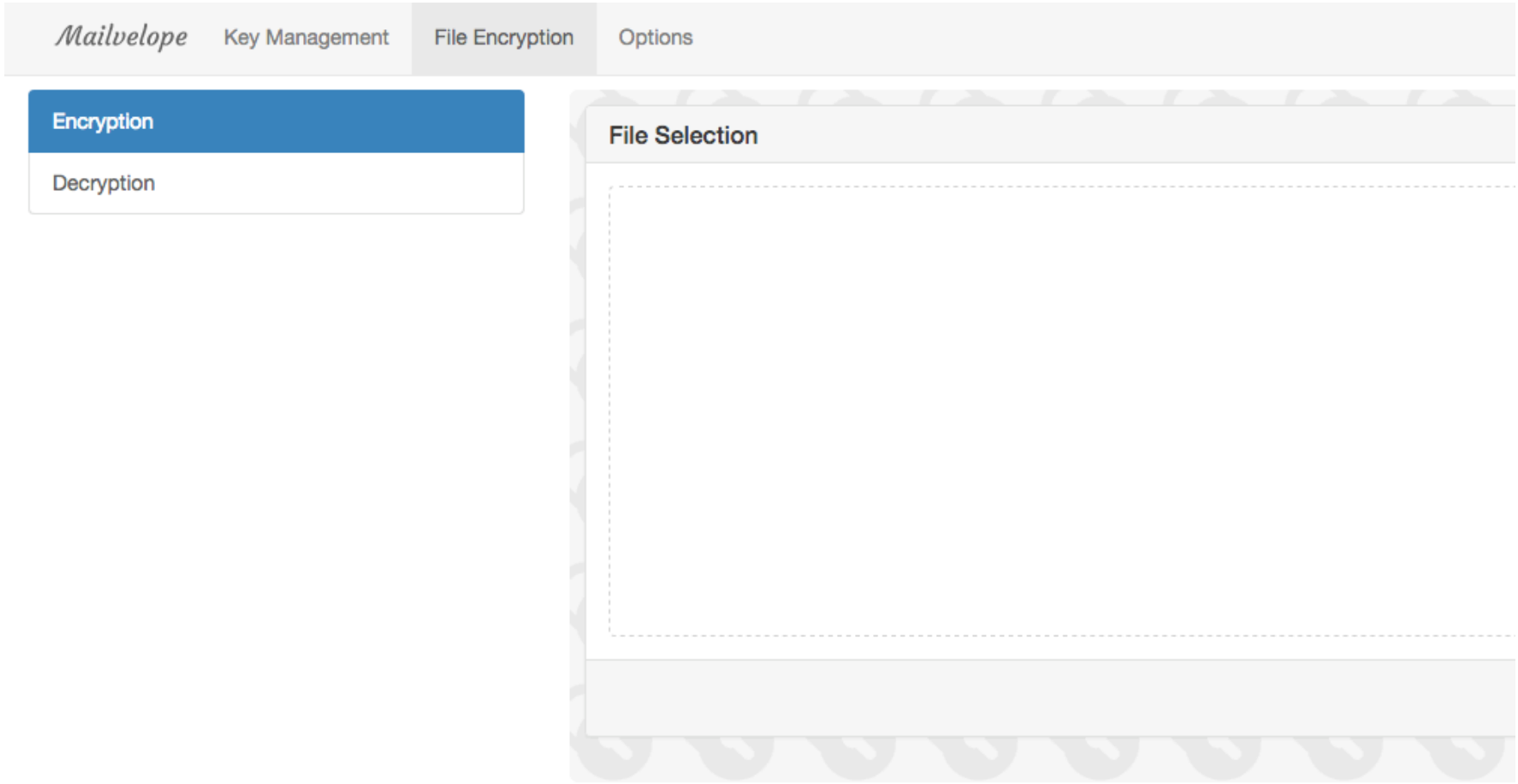
Mailvelope : Import Keys



Mailvelope : Generate Key

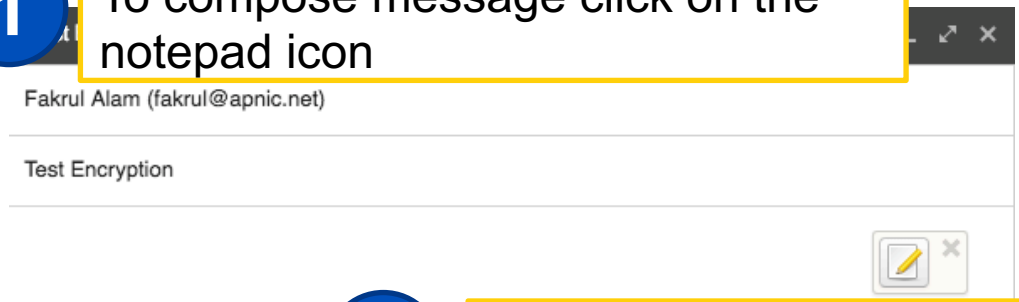


Mailvelope : File Encryption/Decryption

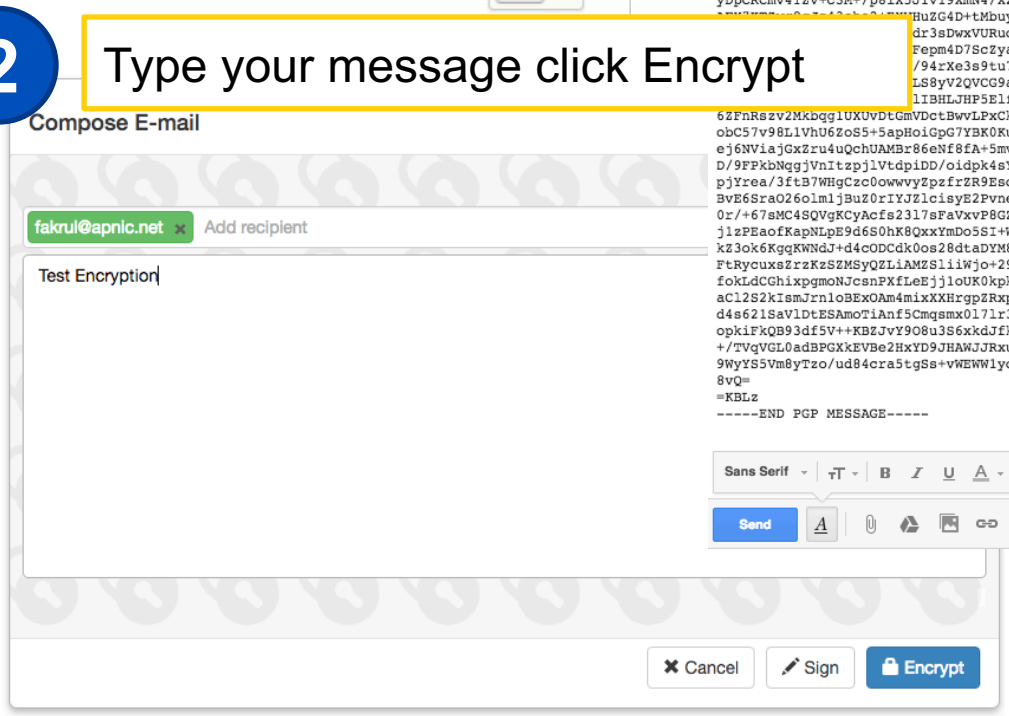


Mailvelope : Encrypt Mail

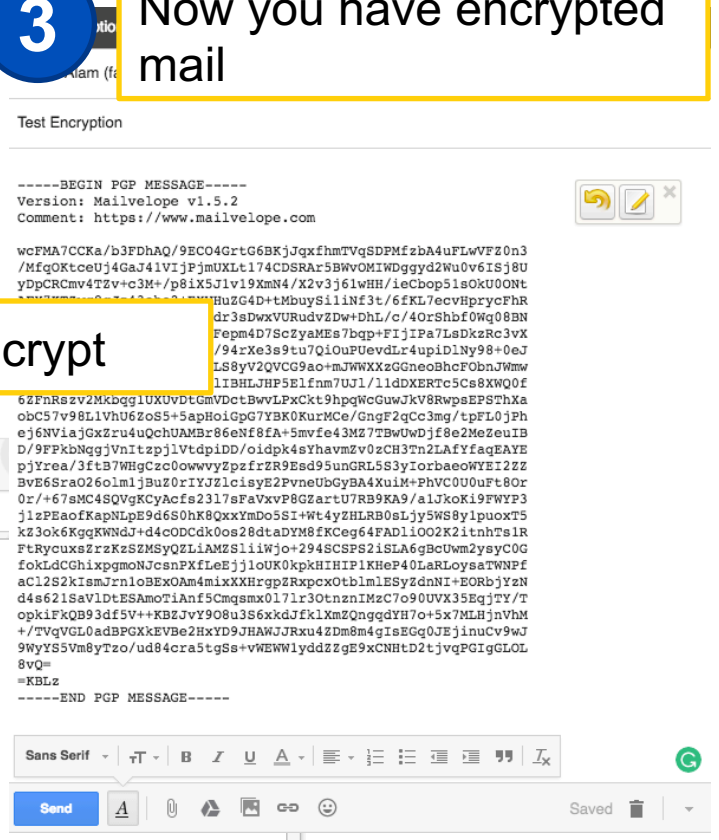
1 To compose message click on the notepad icon



2 Type your message click Encrypt

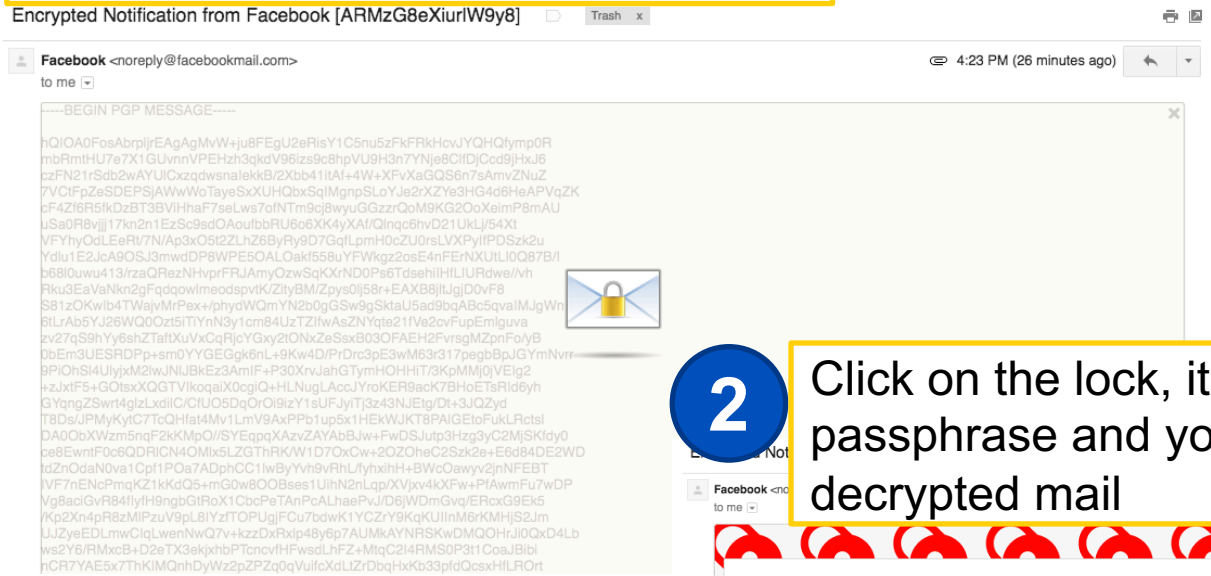


3 Now you have encrypted mail

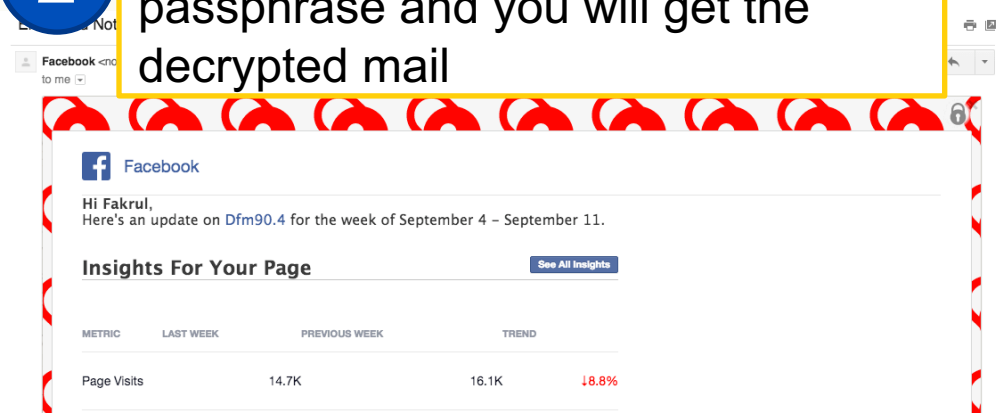


Mailvelope : Dencrypt Mail

1 You will see encrypted mail



2 Click on the lock, it will ask for passphrase and you will get the decrypted mail

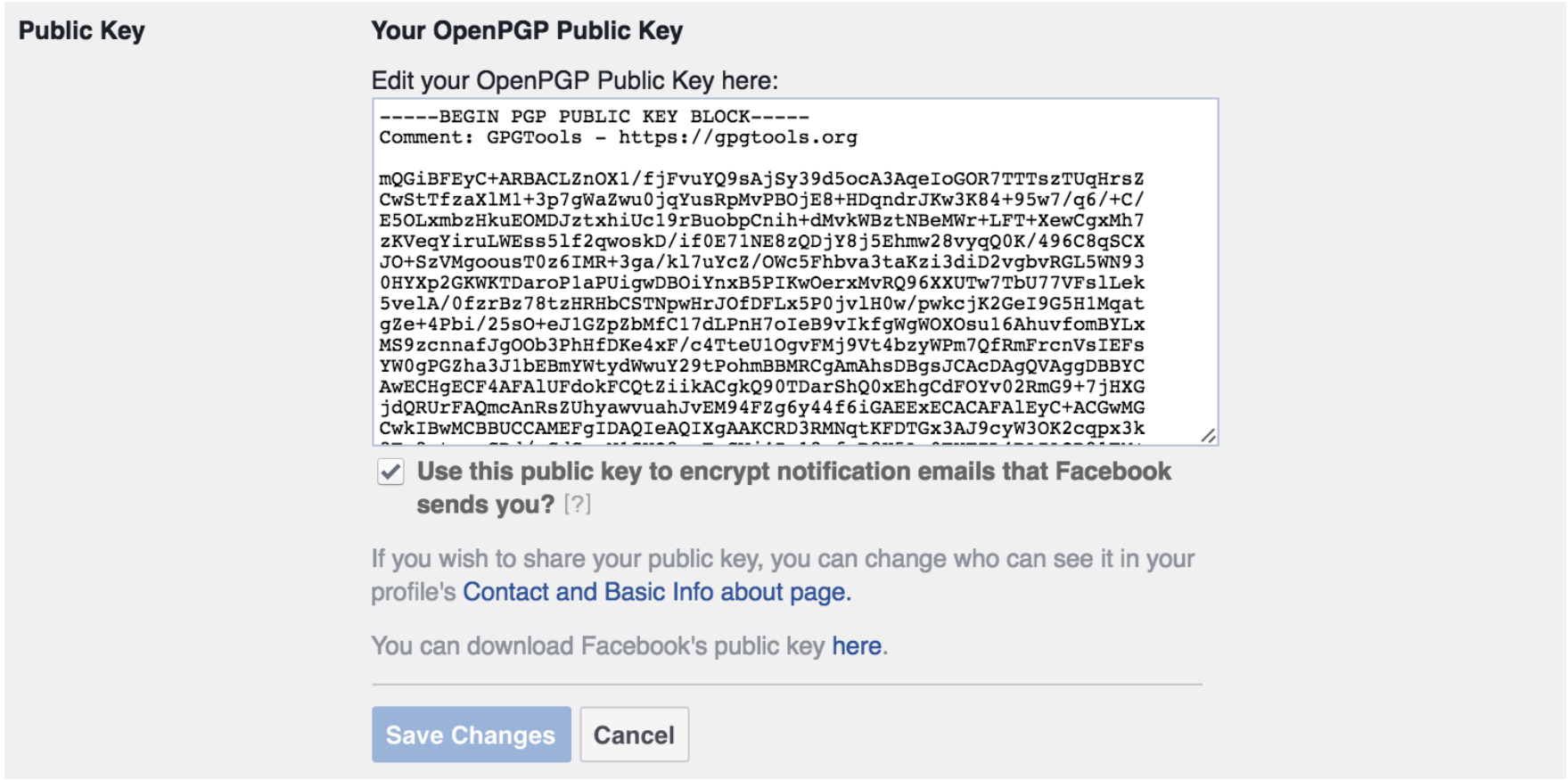


For others

- Enigmail + GnuPG for on Thunderbird

Encrypt Facebook Communication

- settings > security > public key



Encrypt Facebook Communication

Encrypted Notification from Facebook [ARMauaNGN5I_Bon3] Trash x

Facebook <noreply@facebookmail.com> Aug 4 (2 days ago)

to me

This message has been deleted. Restore message

```
-----BEGIN PGP MESSAGE-----

hQIOA0FosAbrpljrEAgA26U5Dlcquf17+LEUx9hpDGmjyIybsFlTuOr+9fm4vtWr
WDaE4YPjbhRGQmWrkAktwM2RjmQRjJG5WU2Nx2olueMOGpoZvbNjy5zvU/B2p1xH
7rb3ws9yq5dh+fV6zsZd9UsvcQYAz7a0VONfs2XxCvxGvb8OGwIe122nnIEyLYVH
lbOzaKL7OB1i9SuiUcGn49V6GJlJ22esDG0aacDedaJdbcf19kuKFQ/Bhbo6O2ng
9XM+OeT2lR1MF9V3gH6Tqz6ZSdSwdUtmorWPFVnrV49offq+SL3Jg2BH7Fa+kSgs
LHkT4kSttObu/tfkDU4bV+gnUGm3HuvI0bt92L2BMgf/a3E/HigG5OL5OWdOi2cC
V5y/sOdKKIo4iOC6mummhH2yfue1dqa8YvLZXTZAUgVH/hlbOHuZ66oqcWAFPqfW
6U8r+DzIIPSUle4WtWaDNozNbeWcdsT3yplWMRhvNN5ilRD/V0k79TMvRiKL8KTX
iXr4bFzmUgd+TeglvrfEfL32G1hDkAU+RXlv/ZiU0liyQveV3mnFhqExW2rKoZRs
171QbUI01lOYpyVxSOiJVkbKggfIrsjtQ0QcP10fjC4NzllbxGGZtvekgpwkCGYd
wFmfpKpeYDJafQ0WFKa/addnRcIthhQMcgR1A44P6n4AVKSoDcNCOnn0dktb3aU
ANLrAeIZHtY4cNbqyP0L/nKAIIVbqEwmqJqiLgzZAJy45KLQHNjIuY4Q9otqQZI6
K+2dclnibAZj/3kdy2jAA8XaNgwB+NOadhE25YJ1Lbw79wvAz8peAusuY8JBSh0j
3W1RSkw9V7cz7JWSX+S3VARKQXw/ufEp7TqeRq/5OJegF9T3JmoXvxISED7mG+H0
EdK4w6yIS11smIeRMan1WmqDoSQSSg0WKzUNT3RbHFT58Ei0mzQ8Eus5HJ384kei
67DcWDffG19SD+tuqCdRI7vEhSKWrq8O8/oG9y3UD6Xp90vjHDHGoin6OOsgOuVT
j+9mufC4EpTZ7I1CYCZv8JSQOJvqJ+Xdy6OkhJASJCqUliPUxx64Ab13SXFCijSd
n9OKRz+Qzs7F9aT/Vv6oVAD+aWB4tHuaJik7TrDH4XvLHFS92O35U2/2zzC6oqcj
cOzZ27W3eG6hgTpFEGgL6FrUPDjEnWSww2RQytuUXGGqq1ZXEHOEWd1IAVEM5T+D
GQstT9WTn6ER2/ALIO1SiP9vgJi/0S0h6L3PfQChnaUTMvfErf209+LF8UJwGwza
fq60DBkDSJxc2bbEDEtUEqvAOfxNtig516rorFzm8scg6xNs04LRELkURPMzfsaf
u7Lgz+iBHGzmvvQGQ5Pi5Z8Pn8xqreao+AE8LyEORxDByzLNVfRRrzMkoS/PpaW5
XUEB60j030EdII7XjyOGxTv4VIftj6yQAX/QHyJRMsbmLmQ1cwLqMYE13xE8mTjr
1UG5FI2TsxEvG/nb2OemIiJ5rB7tNydyZALlK9mfkqRCd1JAXmw6Tm93wzKn1GoR
KxQSIszv7+pb+43B/cSVvL/sU3XD22Iny5y8w2M+BL0BcATFiJL6norEMKsnZ5eI
TBqi4DiWxViD8tp7eypDb9jCTAvLn7920YLNo4rLmn2WYAGuh9HfEQaV14zh2omk
csZ0gsildAfoM8HRiT7Z1zXjATFAgYtcYt0voU228/QDdXvwUHVoQtSfcIlau7L6
oKpR3qqvOHCB6DbmFcpz88iMQza+Qxz08qQK5pT7j+dmJNkhk5Gy5pgPjYNaRu6Q
oPKVoT1rFA8sJjq1PpdEmNXE6VbpMQOAkk6baiQxGJ0bcK3yKt7ZoF2mb2NyS+4I
NJ14tY/Z0GqupkrireszhvnmA3iAaGkYMLGbxhbUN5hh/kz4qTI18sxg/crmNNWM
FeRbGtlu5zZMiHYA4fT+SkfIqu7EGpfh90CDf6vCEr6c67OEeA0fYGzr3JvUkI7u
FPRYKvdps2tOmSnhJVEVQgitkknNmcshfwF7/f9YSmCAkejh+FoLih1RV2J7Lbwv
```

LAB