

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini, akan dibahas mengenai pengujian penerapan sistem *file encryption* menggunakan *gnupg* pada sistem operasi Linux. Pengujian ini bertujuan untuk mengevaluasi keefektifan sistem *file encryption* dalam melindungi data pengguna dari akses yang tidak sah.

4.1 Skenario Pengujian

Pengujian dilaksanakan dengan langkah skenario sebagai berikut:

1. User 1 dan User 2 menghasilkan pasangan kunci, yaitu kunci publik dan kunci privat.
2. Masing-masing pengguna menyimpan kunci privat dan mengirimkan kunci publik kepada rekannya atau pengguna lain.
3. Memastikan bahwa kunci publik yang dibagikan diterima oleh pengguna yang ingin mengirim dan menerima pesan.
4. Setelah kunci publik dikirim oleh setiap pengguna, lakukan proses impor untuk mendapatkan kunci publik tersebut.
5. Mengirimkan data *file*/pesan dari User 1 menggunakan kunci publik dari User 2, tujuannya adalah untuk mengenkripsi *file* oleh pengirim dan dapat didekripsi oleh penerima yang memiliki kunci privat sebagai otoritas terautentikasi.
6. User 2 sebagai penerima pesan mendekripsi isi *file* dengan menggunakan kunci privat yang dimilikinya.

4.2 Pembuatan Kunci Publik

Sebelum melakukan enkripsi atau pengiriman pesan, langkah awal yang harus dilakukan adalah pembuatan sepasang kunci, yaitu kunci privat dan kunci publik. Prosedur untuk membuat pasangan kunci GPG menggunakan algoritma RSA berbasis Linux dapat dilaksanakan sesuai dengan perintah-perintah berikut.

1. Buka terminal pada Linux dengan cara **CTRL + Alt + T**

Dikarenakan paket bawaan gnupg telah ada dari sistem operasi Linux maka tidak perlu lagi menginstalasi kembali. Untuk melakukan pengujian pertama buka terminal dan ketik “*gpg --version*”, hal ini dilakukan guna untuk mengetahui versi gpg yang digunakan.

```
root@irfan:~# gpg --version
gpg (GnuPG) 2.2.27
libgcrypt 1.9.4
Copyright (C) 2021 Free Software Foundation, Inc.
License GNU GPL-3.0-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@irfan:~# █
```

Gambar 4. 1 Versi dan algoritma pada GPG

Pada gambar 4.1 diatas memperlihatkan versi gpg yang digunakan yaitu versi 2.2.27. dengan terlihat di bagian bawah terdapat algoritma PubKey (Asimetris), Cipher (Simetris), hash, dan algoritma kompresi.

2. Untuk pembuatan kunci publik dan kunci privat maka ketik perintah pada terminal “*gpg --full-generate-key*”

```
root@irfan:~# gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 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)
(14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 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

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

Real name: irfan
Email address:
Comment:
You selected this USER-ID:
"irfan"

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

Gambar 4. 2 Proses Pembuatan kunci

Pada gambar 4.2 merupakan serangkaian pembuatan sepasang kunci. Pertama terdapat pemilihan jenis algoritma enkripsi yang ada, kemudian

akan diminta memasukan panjang bit pada kunci antara 1024 bit sampai 4096 bit, semakin panjang bit yang di *input* maka semakin sulit untuk bisa di retas dari serangan *cryptanalyst*. Selanjutnya masukan masa aktif pada kunci sesuai dengan kebutuhan. Kemudian akan diminta untuk mengisi User ID yang berupa nama sedangkan untuk email dan komentar dapat kita abaikan.



Gambar 4. 3 Memasukan Passphrase

Pada gambar 4.3 merupakan pengisian passphrase. Passphrase merupakan frasa kunci yang bertujuan sebagai pengaman untuk *file* sertifikat elektronik.

3. Untuk melihat pasangan kunci yang telah dibuat gunakan perintah “*gpg --list-keys*” atau “*gpg -k*” pada terminal.

```
root@irfan:~# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid: 1  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
-----
pub  rsa4096 2024-02-04 [SC]
     FC6155DE39947CF913D3735EC2776965C83833DB
uid  [ultimate] irfan
sub  rsa4096 2024-02-04 [E]

root@irfan:~# gpg -k
/root/.gnupg/pubring.kbx
-----
pub  rsa4096 2024-02-04 [SC]
     FC6155DE39947CF913D3735EC2776965C83833DB
uid  [ultimate] irfan
sub  rsa4096 2024-02-04 [E]

root@irfan:~#
```

Gambar 4. 4 Daftar Kunci Publik

4.3 Pertukaran Kunci

Penukaran kunci, yang melibatkan ekspor dan impor kunci, diperlukan untuk menukar kunci publik antar user. Sehingga dapat saling bertukar *file* dengan aman.

4.3.1 Ekspor Kunci

Dalam kriptografi asimetris, setiap user memiliki sepasang kunci, kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi pesan atau memverifikasi tanda tangan digital, sedangkan kunci privat digunakan untuk mendekripsi pesan yang telah dienkripsi dengan kunci publik atau untuk membuat tanda tangan digital. Ekspor kunci publik adalah proses untuk menghasilkan salinan dari kunci publik yang dimiliki oleh user yang nantinya dapat digunakan untuk mengirimkan pesan terenkripsi kepada user tersebut atau memverifikasi tanda tangan digital yang dibuat oleh user tersebut.

Kunci publik akan diekspor ke dalam *file* yang ditentukan dalam format ASCII yang dapat dibaca oleh manusia. *File* ini kemudian dapat dibagikan kepada pihak lain untuk tujuan pertukaran kunci publik dan komunikasi aman.

Berikut merupakan perintah untuk mengekspor kunci publik melalui terminal:

```
root@irfan:~# gpg --armor --export Irfan > Irfan_Pubkey.asc
```



```
root@irfan:~# gpg -k
/root/.gnupg/pubring.kbx
-----
pub   rsa4096 2024-02-04 [SC]
      FC6155DE39947CF913D3735EC2776965C83833DB
uid   [ultimate] Irfan
sub   rsa4096 2024-02-04 [E]

root@irfan:~# gpg --export --armor Irfan > Irfan_Pubkey.asc
root@irfan:~#
```

Gambar 4. 5 Ekspor kunci publik

Pada gambar 4.5 dijelaskan bahwa kunci publik akan diekspor menjadi format ascii.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
mQINBGW/ZpMBEADDG/8tTFB94eFbnKDEtNLv1jsa5SC5gouRz2z4I  
hDESdK6JajBbcJLfrKIikDB46wcehtku2jhDARPFfn3fcMV0118p5t6aol  
HBVY07n21NSPkwue/FyVLdHC2C2lyM89K3Lfn88cCybgnqmUpn5un  
J/POfHKAY01t8JzR3v9HCCK/5+ZAqY/4PYTH8lg9+fFVDnpyBewl8tN  
0mejSxB5Cnt8npoVviYcnfV/8/EHtM1sp+85u8yioEtauLxhXsIchQDqx  
MKNV8AovToqSDILBzQMfO786tmTxnw7nkMTnsIuvpbCMXKXT1rq  
fK5wRr5T9g2UD803k==46lr  
-----END PGP PUBLIC KEY BLOCK-----
```

Berikut merupakan isi text yang ada pada kunci publik setelah di konversi ke ascii, pada bagian awal dan akhir merupakan batas awal dan akhir kunci publik.

4.3.2 Impor Kunci

Impor kunci publik digunakan untuk mengimpor kunci publik ke dalam keyring GPG. Ini memungkinkan menggunakan kunci publik tersebut untuk melakukan verifikasi tanda tangan digital atau untuk mengenkripsi pesan kepada pemilik kunci publik tersebut. Fungsi ini berguna terutama dalam konteks keamanan informasi, di mana pengguna dapat membagikan kunci publik mereka kepada pihak lain agar pihak tersebut dapat mengenkripsi pesan yang hanya bisa dibuka oleh pemilik kunci privat yang sesuai.

Berikut merupakan perintah untuk mengimpor kunci publik di terminal Linux dengan menggunakan perintah :

root@adiguna: ~# gpg --import Irfan_Pubkey (*nama kunci public yang telah dibuat sebelumnya*)

```
root@adiguna:/home/adiguna/Documents# ls  
Irfan_Pubkey.asc  
root@adiguna:/home/adiguna/Documents# gpg --import Irfan_Pubkey.asc  
gpg: key C2776965C83833DB: "irfan" not changed  
gpg: Total number processed: 1  
gpg:          unchanged: 1  
root@adiguna:/home/adiguna/Documents# █
```

Gambar 4. 6 Impor kunci publik

Pada gambar 4.6 merupakan proses impor kunci publik yang di kirim milik user irfan dan diimpor kedalam kunci dari pengguna adiguna. Keterangan tersebut menunjukkan bahwa GnuPG (*GNU Privacy Guard*) telah memproses satu item, dan setelah diproses, item tersebut tetap tidak berubah. Hal ini mengindikasikan bahwa tidak ada perubahan yang terdeteksi dalam item tersebut setelah diproses oleh GnuPG. Ini bisa berarti bahwa tanda tangan atau enkripsi pada item tersebut tetap valid dan tidak ada perubahan yang dilakukan pada isinya.

4.4 Enkripsi dan Dekripsi File pada GnuPG

Bagian ini akan menguraikan langkah-langkah untuk melakukan proses enkripsi dan dekripsi pada file menggunakan *Gnu Privacy Guard* (GnuPG).

4.4.1 Enkripsi

Enkripsi adalah proses mengubah teks biasa yang dapat dibaca secara langsung, yang disebut (*plaintext*), menjadi bentuk yang tidak dapat dimengerti atau dibaca secara langsung, yang disebut (*ciphertext*).

Berikut merupakan perintah untuk enkripsi di Gnupg sebagai berikut:

gpg --encrypt --armor -r adiguna (*UserID Penerima file enkripsi*)
dokumen.txt (*File yang akan di enkripsi*)

```
root@irfan:~# gpg -k
/root/.gnupg/pubring.kbx
-----
pub   rsa4096 2024-02-04 [SC]
      FC6155DE39947CF913D3735EC2776965C83833DB
uid   [ultimate] irfan
sub   rsa4096 2024-02-04 [E]

pub   rsa4096 2024-02-04 [SC]
      84D8ACEAC44939A0C5A9FF2DA541747D284A131A
uid   [ unknown] adiguna
sub   rsa4096 2024-02-04 [E]

root@irfan:~# gpg --encrypt --armor -r adiguna dokumen.txt
```

Gambar 4. 7 Enkripsi *file* ke user lain

Pada gambar 4.7 menunjukkan instruksi untuk melakukan enkripsi pada suatu *file* yang akan dikirim kepada salah user dengan menggunakan User ID penerima pesan.

```
root@irfan:~# ls
dokumen.txt  dokumen.txt.asc  Irfan_Pubkey.asc  snap
root@irfan:~# ls -l
total 16
-rw-r--r-- 1 root root  41 Feb  5 23:43 dokumen.txt
-rw-r--r-- 1 root root 927 Feb  5 23:44 dokumen.txt.asc
-rw-r--r-- 1 root root 3094 Feb  4 18:09 Irfan_Pubkey.asc
drwx----- 6 root root 4096 Jan 16 07:49 snap
root@irfan:~#
```

Gambar 4. 8 Daftar *File*

Pada gambar 4.8, terdapat daftar *file* dalam direktori yang menunjukkan adanya berkas dokumen.txt yang telah sukses dienkripsi, sehingga dikonversi menjadi dokumen.txt.asc.

Berikut merupakan perintah Untuk mengetahui isi dokumen yang telah terenkripsi dapat dilakukan dengan memasukan perintah :

root@irfan:~# more dokumen.txt.asc

```
root@irfan:~# more dokumen.txt.asc
-----BEGIN PGP MESSAGE-----

hQIMAw0sRdd1j8ZCARAAAnJsQKnprR/auCTVAD9M00BRqKagx5Mdh6jE7Zf07jqtF
iKZpH5RrInNQSE5Mf9ClW4+LGzU0H6I/n5ayredSW+pfcmYg5z00bBodMXdM0Zgq
G1bAIrPUKcjnoAyIU/on6pjNbm5CpTVnXL9DvyRVtgqTYS62Tr/EerF6PiBFX/wM
//TrnBlioSqtEq4pc2Qgiu611DfU8K3Hzq+6Tsi+/l6v/SfVeHtxh08uqak5eLca
IDmd0hKiAG8yIML7aTxkBYW+hA7jAc4XwcxHRE7dqpGbdNiR/7TGIBKj0DkGk2p6
yYvqhpvfK9BFoZ7F767vwqLFBKuW+taHiy+fDcvdeU+KdkNn38d28D8x6MxK1RA9
aXtzwemX5MJ055rV4ZMvnt9pLXclSB7ZhZGZfskobQAq/Hzav9PFzaQwf1+wwP7N
SryfEIsu0u1H0rTi4KuCJ4KwmnjLqtlbz+T04MIqsL9+oy+iQ5co2vLa7BGonNg7
cESBfW5LdXdTqG60w853qju16ePrnmBLZsMhkfnJ4yNcs43mFdi/o0NW3pYc4/Wi
BY3JlgLKsgnG7+xhLM5XD5IdvNemA6ErB+pmSvj8udHgZVeQ45PwH5VLT305YvI1
9fau5yoLHTasRDo4tnHmxZfNAI3Txw4J50v7sZD/zxzSfN92qYlcnDo3T7XTIMfS
bgGReAA9fxm3XirDijWD6+ewPlKV+zL7qA03cnQH7Fu+hdZ/Zakp6ExXcgGhr4M
C4Wj0uIXoFvosPSSWdR8nlg6zmR9P/HBwzRYC0coGMoLx0XQHygG6tKxfIuJjDOY
xHuXNGpYWBLR1WQt4UY
=YvRL
-----END PGP MESSAGE-----
```

Gambar 4. 9 Isi *file* yang telah terenkripsi

4.4.2 Dekripsi

Dekripsi adalah proses mengubah data yang telah dienkripsi yaitu dari *ciphertext* kembali ke bentuk aslinya atau *plaintext*, menggunakan kunci yang sesuai. Ini adalah kebalikan dari enkripsi. Berikut adalah perintah yang digunakan untuk mendekripsi *file* di GnuPG:

```
root@adiguna~# gpg --output hasil.txt --decrypt dokumen.txt.asc
```

atau

```
root@adiguna~# gpg -o hasil.txt -d dokumen.txt.asc
```

```
root@adiguna:/home/adiguna/Documents# ls
dokumen.txt.asc
root@adiguna:/home/adiguna/Documents# gpg --output hasil.txt --decrypt dokumen.txt.asc
gpg: encrypted with 4096-bit RSA key, ID 0D2C4437758FC642, created 2024-02-04
"adiguna"
root@adiguna:/home/adiguna/Documents# more hasil.txt
INI CONTOH DATA ENKRIPSI DARI USER IRFAN
root@adiguna:/home/adiguna/Documents#
```

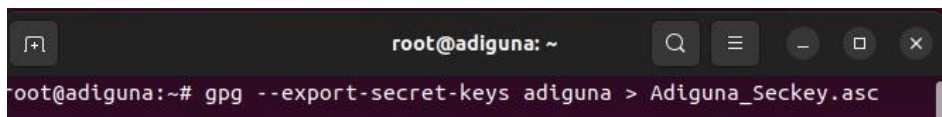
Gambar 4.10 Dekripsi *file*

Pada gambar 4.10 yang ditampilkan, terdapat instruksi untuk memulai proses dekripsi sebuah *file*. Untuk membuka data tersebut, dibutuhkan passphrase dari kunci pribadi yang dimiliki oleh penerima pesan. Dalam gambar tersebut, terlihat passphrase yang digunakan untuk mendekripsi *file* yang dimiliki oleh UserID "adiguna" yang menggunakan kunci RSA dengan panjang 4096 bit.

4.5 Ekspor Kunci Privat

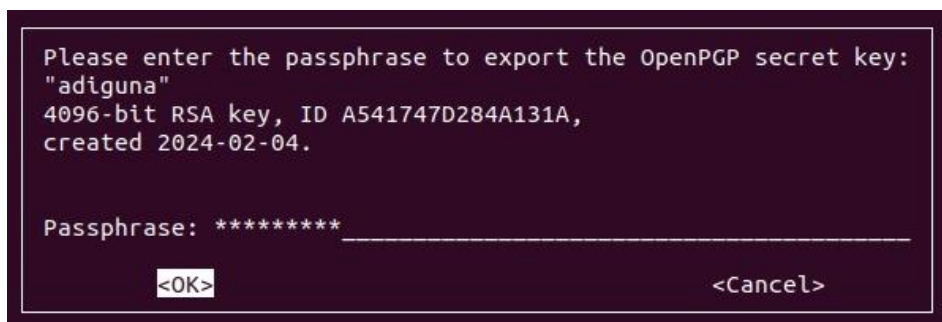
Untuk menerima *file* dan mendekripsikan isi *file* yang ada supaya pesan ataupun *file* data yang ada dapat terbaca dan terverifikasi maka perlu integrasi antara kunci privat yang sebelumnya telah terbuat pada proses gpg. Perintah untuk melakukan ekspor kunci privat pada terminal sebagai berikut :

```
root@adiguna:~# gpg --export-secret-keys adiguna > Adiguna_Seckey.asc
```



```
root@adiguna:~# gpg --export-secret-keys adiguna > Adiguna_Seckey.asc
```

Gambar 4.11 Ekpor kunci privat user adiguna



```
Please enter the passphrase to export the OpenPGP secret key:
"adiguna"
4096-bit RSA key, ID A541747D284A131A,
created 2024-02-04.

Passphrase: *****
<OK> <Cancel>
```

Gambar 4.12 Passphrase Kunci Privat adiguna

Pada gambar 4.11 terlihat proses ekspor kunci privat milik user adiguna setelah memasukan perintah pada terminal maka akan diminta untuk mengisi Passphrase yang sebelumnya telah di buat seperti yang tertera pada gambar 4.12.

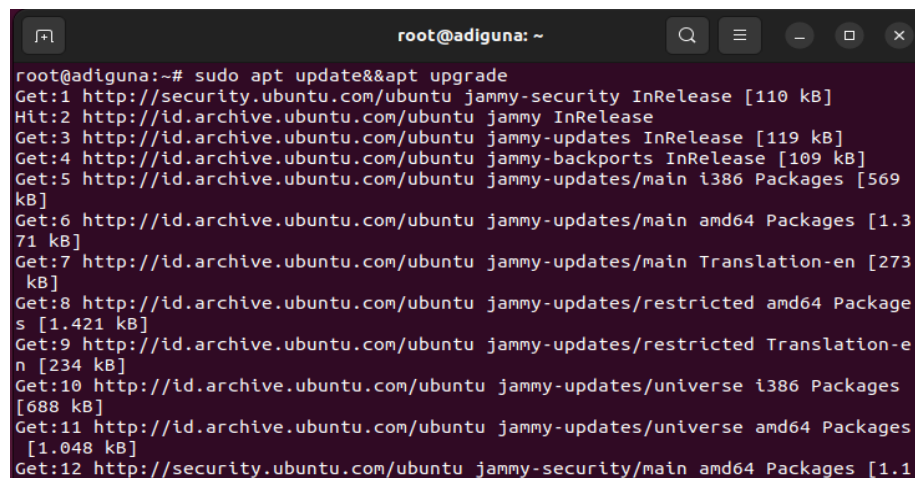
4.6 Pengujian Menggunakan *Seahorse*

Seahorse merupakan aplikasi Frontend yang bersifat GUI (Graphical User Interface) merupakan pengaman kunci dan sertifikat digital yang ada pada sistem operasi Linux, mendukung implementasi utilitas menggunakan *GNU Privacy Guard* (GPG). Selain bisa digunakan sebagai pengaman kunci enkripsi *seahorse* juga dapat memanajemen penyimpanan kunci yang ada pada SSH (Secure Shell Protocol). Sebelum melakukan pengujian menggunakan *seahorse* langkah pertama yang perlu dilakukan adalah memastikan repositori yang ada di sistem operasi Linux sudah dilakukan pembaruan sistem pada *software* yang ada.

4.6.1 Instalasi *Seahorse* dan Pembaruan Repository

Langkah yang harus dilakukan pertama kali adalah memperbarui repositori yang ada di Linux, hal ini di lakukan untuk mendapatkan paket instalasi terbaru pada software yang akan digunakan sehingga meminimalisir kerurangan paket instalasi pada software. Perintah yang perlu dilakukan adalah sebagai berikut :

```
root@adiguna:~# apt update && apt upgrade
```

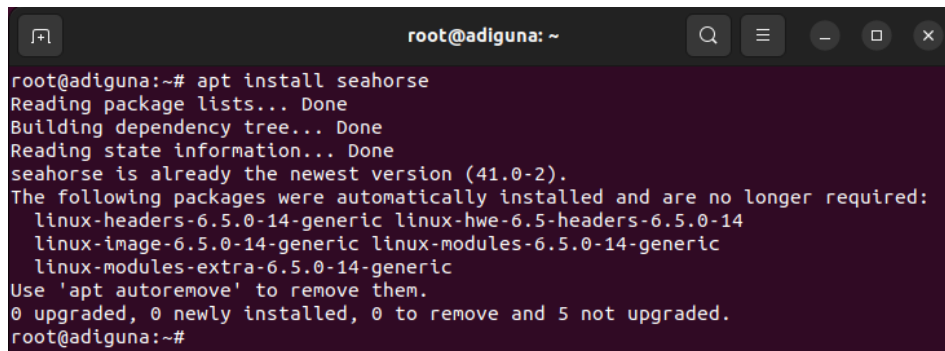


```
root@adiguna:~# sudo apt update&&apt upgrade
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://id.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://id.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://id.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://id.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [569 kB]
Get:6 http://id.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1.371 kB]
Get:7 http://id.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [273 kB]
Get:8 http://id.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1.421 kB]
Get:9 http://id.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [234 kB]
Get:10 http://id.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [688 kB]
Get:11 http://id.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1.048 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1.1
```

Gambar 4. 13 Proses Pembaruan Repository

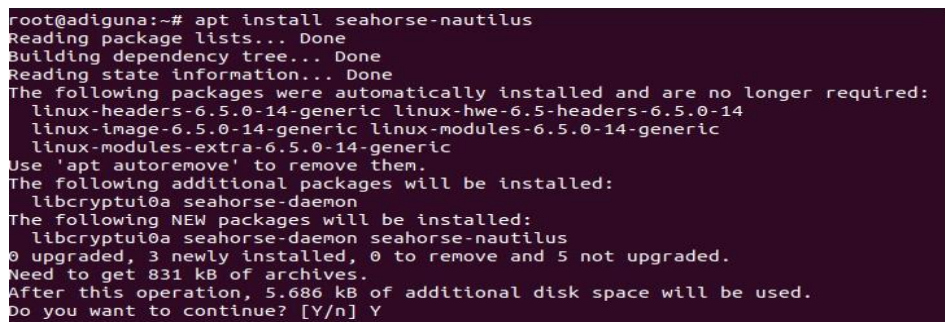
Pada gambar 4.13 menunjukkan proses *update* dan *upgrade* repositori pada sistem. Selanjutnya proses instalasi paket *Seahorse*. Perintah untuk instalasi *seahorse* melalui terminal adalah sebagai berikut :

```
root@adiguna:~# apt install seahorse
```



```
root@adiguna:~# apt install seahorse
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
seahorse is already the newest version (41.0-2).
The following packages were automatically installed and are no longer required:
  linux-headers-6.5.0-14-generic linux-hwe-6.5-headers-6.5.0-14
  linux-image-6.5.0-14-generic linux-modules-6.5.0-14-generic
  linux-modules-extra-6.5.0-14-generic
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
root@adiguna:~#
```

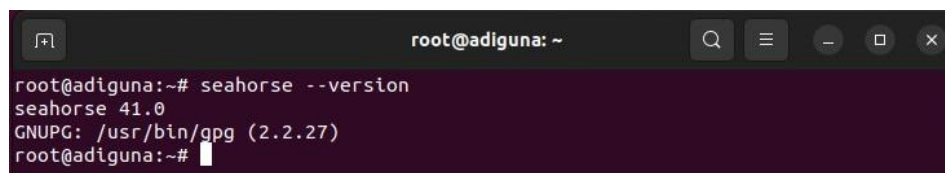
Gambar 4. 14 Instalasi *Seahorse*



```
root@adiguna:~# apt install seahorse-nautilus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-6.5.0-14-generic linux-hwe-6.5-headers-6.5.0-14
  linux-image-6.5.0-14-generic linux-modules-6.5.0-14-generic
  linux-modules-extra-6.5.0-14-generic
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  libcryptui0a seahorse-daemon
The following NEW packages will be installed:
  libcryptui0a seahorse-daemon seahorse-nautilus
0 upgraded, 3 newly installed, 0 to remove and 5 not upgraded.
Need to get 831 kB of archives.
After this operation, 5.686 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Gambar 4. 15 Instalasi Paket *Seahorse-Nautilus*

Pada gambar 4.14 dan 4.15 terlihat proses instalasi paket *Seahorse* dan Plugin *seahorse nautilus*. Setelah melakukan proses instalasi maka perlu dilakukan pengecekan versi pada *seahorse*. Perintah pada terminal untuk mengecek versi pada *seahorse* sebagai berikut :



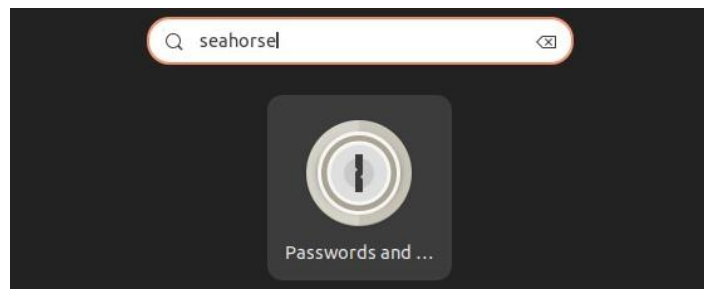
```
root@adiguna:~# seahorse --version
seahorse 41.0
GNUPG: /usr/bin/gpg (2.2.27)
root@adiguna:~#
```

Gambar 4. 16 Cek Versi *Seahorse*

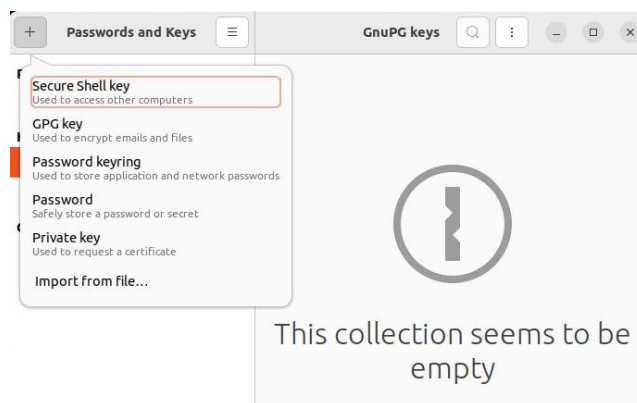
Pada gambar 4.16 merupakan pengecekan versi *seahorse* terbaru.

4.7 Impor Kunci Publik dan Privat *Seahorse*

Tahap selanjutnya Ketika sudah melakukan proses ekspor dan instalasi *Seahorse* maka perlu dilakukan proses impor kunci publik dan kunci privat ke repositori yang ada di *seahorse*. Untuk mengimpor kunci publik masuk pada menu pencarian di Linux dan cari *seahorse* atau *password and keys* pada bilah pencarian.



Gambar 4. 17 Pencarian *seahorse* yang telah terinstalasi dibilah pencarian

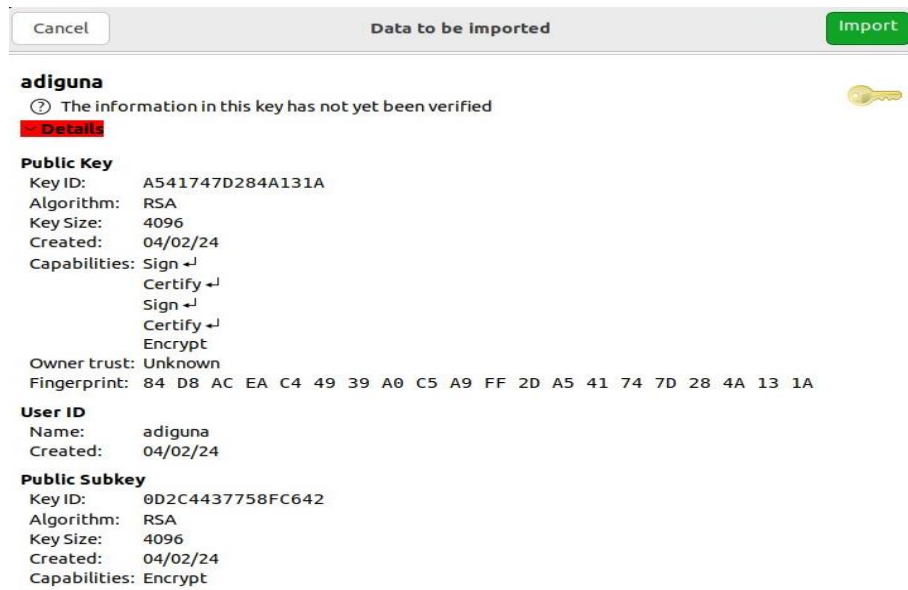


Gambar 4. 18 Impor kunci *Seahorse*

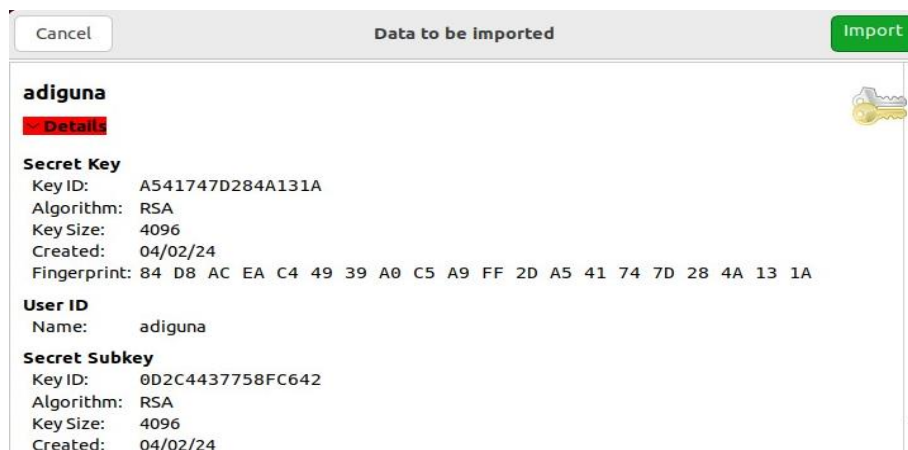


Gambar 4. 19 Kunci Publik dan Kunci Privat yang akan di Impor

Pada gambar 4.17 sampai gambar 4.19 merupakan proses impor kunci publik dan kunci privat yang sebelumnya telah di ekspor dan akan di impor menuju repositori dari *Seahorse*.



Gambar 4.20 Deskripsi Kunci Publik atas Nama User Adiguna



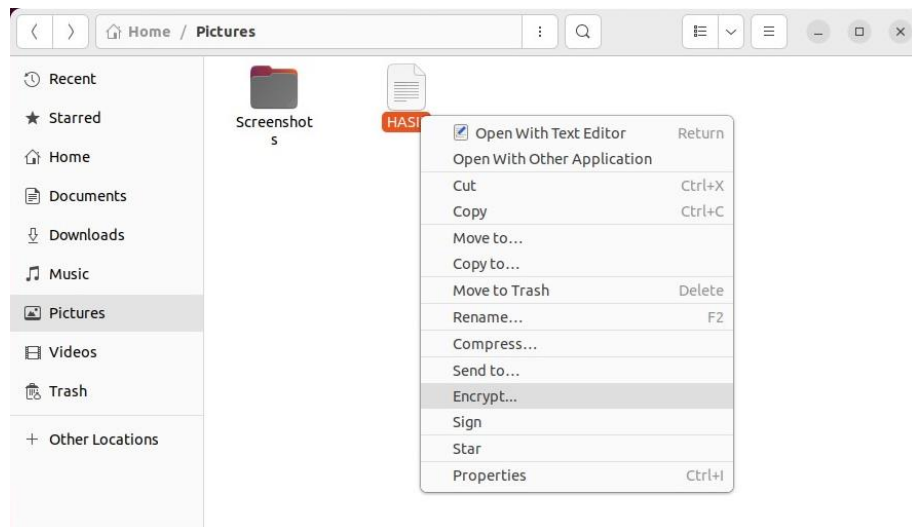
Gambar 4.21 Dekripsi Kunci Privat atas Nama User Adiguna

Pada Gambar 4.20 dan 4.21 Merupakan Detail dari sepasang kunci milik user adiguna.

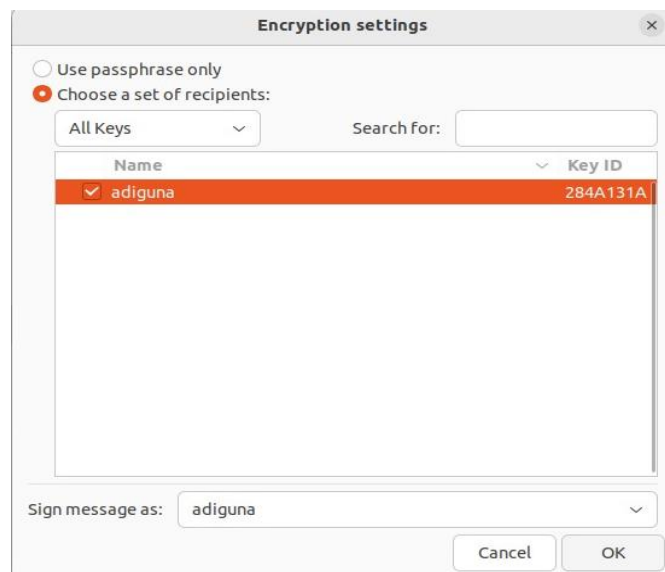
4.8 Enkripsi dan Dekripsi dengan *Seahorse*

Setelah proses penukaran kunci dan instalasi paket *seahorse* telah selesai dilakukan, pada bagian ini akan menjelaskan bagaimana melakukan enkripsi dan dekripsi dengan menggunakan *seahorse*.

4.8.1 Enkripsi

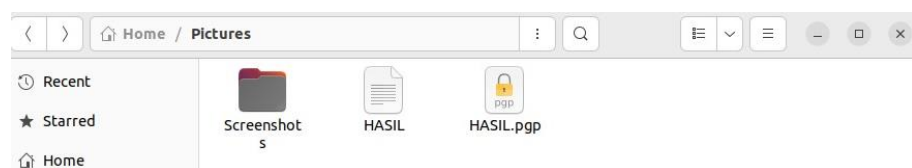


Gambar 4. 22 Enkripsi File



Gambar 4. 23 Pemilihan Kunci yang Akan Mengenkripsi File

Keterangan pada gambar 4.22 dan 4.33 Proses enkripsi file data text dengan dengan memilih ID Kunci Publik yang akan digunakan untuk mengenkripsi file tersebut.

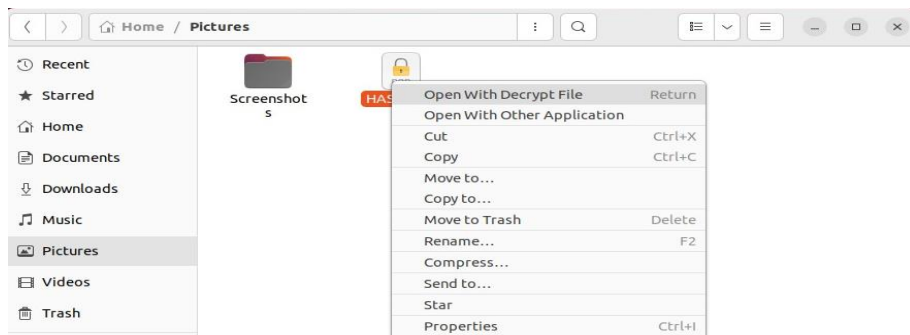


Gambar 4. 24 Hasil File Yang Telah Terenkripsi

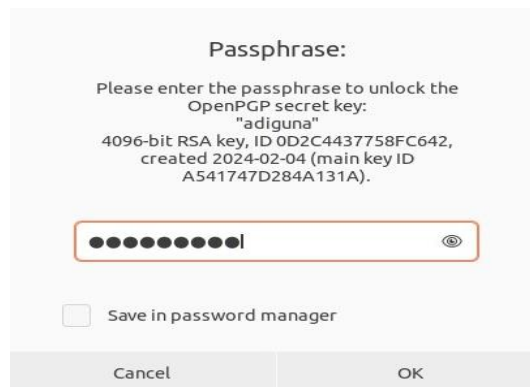
Pada gambar 4.24 telah dihasilkan *file* yang terenkripsi menggunakan kunci publik dari user adiguna yang menerapkan algoritma enkripsi RSA 4096.

4.8.2 Dekripsi

Kebalikan dari proses enkripsi, dekripsi adalah proses mengubah data *ciphertext* yang tidak dapat dibaca oleh manusia kembali menjadi *plaintext* data yang dapat dibaca oleh manusia. Pada penggunaan seahorse proses dekripsi lebih mudah.



Gambar 4. 25 Dekripsi *File* Menggunakan *Seahorse*



Gambar 4. 26 Passphrase Dekripsi *File*

Pada gambar 4.25 dan 4.26 Proses akhir dekripsi *file* dengan mengisi *passphrase* untuk mendekripsi.

4.9 Kelebihan dan Kekurangan

Penerapan algoritma keamanan dalam sistem enkripsi dengan RSA dan GPG tidak terlepas dari kelebihan dan kekurangannya. Berikut adalah beberapa kelebihan yang dapat diidentifikasi :

Kelebihan

1. RSA yang cukup kuat, Panjang kunci RSA yang lebih panjang umumnya lebih aman karena lebih sulit bagi penyerang untuk memecahkan kunci dengan teknik brute force atau faktorisasi. Sebagai contoh, kunci 2048-bit atau 4096-bit saat ini dianggap cukup aman.
2. Enkripsi RSA memungkinkan hanya penerima yang memiliki kunci privat yang sesuai yang dapat membuka pesan atau *file* yang dienkripsi. Ini menjaga privasi dan kerahasiaan data pengguna.
3. GPG adalah perangkat lunak sumber terbuka yang tersedia secara gratis untuk digunakan, memungkinkan akses universal dan partisipasi dalam pengembangan dan peningkatan algoritma enkripsi RSA.
4. GPG menyediakan fasilitas untuk pengelolaan kunci enkripsi RSA dengan efisien. Pengguna dapat menghasilkan, mengimpor, mengekspor, dan memperbarui kunci dengan mudah menggunakan GPG.

Kekurangan

1. Proses enkripsi dan dekripsi RSA dapat memakan waktu yang lebih lama daripada algoritma enkripsi simetris,
2. Kunci RSA yang digunakan dalam GnuPG lebih besar, menghasilkan overhead dalam ukuran *file* dan mempengaruhi waktu
3. Sebagian besar fungsionalitas GPG diakses melalui command line interface (CLI), Ini bisa menjadi hambatan bagi pengguna awam untuk mengadopsi dan menggunakan GPG yang mungkin tidak nyaman atau familiar bagi sebagian pengguna yang lebih suka antarmuka grafis.
4. Keamanan GPG sangat bergantung pada keamanan infrastruktur kunci publik. Jika infrastruktur ini tidak diatur dengan baik atau rentan terhadap serangan, dapat mengancam keamanan komunikasi yang dienkripsi menggunakan GPG.