

Электронная подпись в Linux. Утилита GnuPG



1. Электронная подпись

В этой главе мы рассмотрим программу GnuPG, использующуюся для безопасного хранения и передачи данных, — ведь при транспортировке файлов по Интернету от разработчика (или автора документа) к вам файлы могут быть изменены третьими лицами. GnuPG полностью соответствует стандарту OpenPGP и может применяться для проверки подлинности получаемых программ и документов.

Программа GnuPG абсолютно бесплатна, она не использует каких-либо запатентованных алгоритмов. Это одновременно является и достоинством, и недостатком программы. Приятно, конечно, что за программу ничего не нужно платить, но, к сожалению, она не поддерживает стандарт PGP2 — тогда бы GnuPG не была бесплатной.

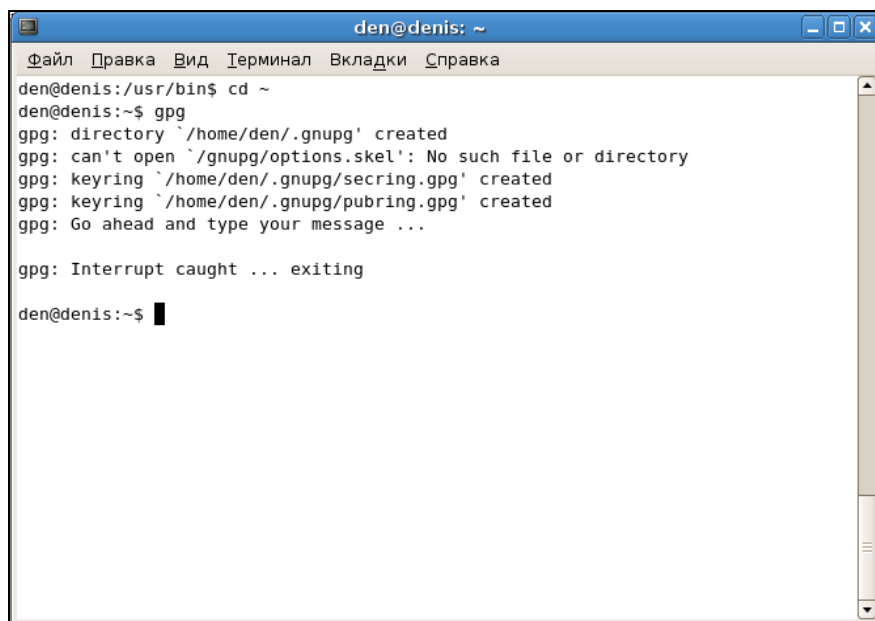
GnuPG входит в состав большинства дистрибутивов и даже устанавливается по умолчанию. Если GnuPG не установлена по умолчанию, следует установить пакет `gpg` (или `gpgv` — в зависимости от дистрибутива). Также можно установить пакеты `gprg` и `kgpg` (обратите внимание на названия пакетов!) — это оболочки GnuPG, соответственно, для GNOME и KDE.

2. Использование программы

Давайте сразу перейдем к использованию программы, чтобы не тратить время на размышления, нужна она вам или нет. А уже в процессе работы каждый для себя с этим определится.

Для тестирования работы программы нужно, чтобы в системе было хотя бы две пользовательские учетные записи. Мы попробуем зашифровать сообщение одного пользователя и передать его другому пользователю, чтобы он его расшифровал. Для определенности будем считать, что у нас есть пользователи `den` и `user2`.

Войдите в систему как пользователь `den`, перейдите в домашний каталог и введите команду: `$ gpg`. Программа создаст все необходимые подкаталоги (рис. 1) и попросит вас ввести текст сообщения. Ничего вводить не надо, просто нажмите клавиатурную комбинацию `<Ctrl>+<C>` — нам только и нужно было, чтобы програм-



```
den@denis: ~
Файл Правка Вид Терминал Вкладки Справка
den@denis:/usr/bin$ cd ~
den@denis:~$ gpg
gpg: directory `/home/den/.gnupg' created
gpg: can't open `/gnupg/options.skel': No such file or directory
gpg: keyring `/home/den/.gnupg/secring.gpg' created
gpg: keyring `/home/den/.gnupg/pubring.gpg' created
gpg: Go ahead and type your message ...

gpg: Interrupt caught ... exiting
den@denis:~$
```

Рис. 1. Первый запуск программы gpg

ма создала в домашнем каталоге все необходимые для нормальной работы подкаталоги.

Затем введите команду: `$ gpg --gen-key`.

Первым делом программа попросит выбрать алгоритм шифрования (рис. 2). По умолчанию используется первый алгоритм (DSA и Elgamal). Все алгоритмы хороши, но вам нужно знать некоторые особенности каждого алгоритма:

- ☐ RSA — служит для шифрования и для создания цифровой подписи (цифровая подпись позволяет удостовериться, что автор документа именно вы, а не кто-то другой), но GnuPG использует его только для подписи;
- ☐ DSA — служит только для создания цифровой подписи;
- ☐ Elgamal — используется только для шифрования информации;
- ☐ DSA + Elgamal — можно применять для шифрования и подписи.

Как видите, первый алгоритм более универсальный — его можно использовать как для шифрования информации, так и для цифровой подписи, поэтому смело вводите 1.

После этого программа попросит ввести длину ключа:

What keysize do you want?

Учитывая мощность современных компьютеров, минимальная длина ключа должна быть не менее 2048 битов, иначе ваше сообщение относительно быстро расшифруют.

```
den@denis:~$ 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? 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) 6m
Key expires at Bтp 09 Сен 2008 15:27:30 MSD
Is this correct? (y/N)
```

Выбор алгоритма

Длина ключа

Срок действия ключа
(6 месяцев)

Рис. 2. Настройка программы

Следующий шаг — ввод срока действия ключа:

- ☐ 0 — неограниченный срок действия;
- ☐ n — ключ действителен n дней;
- ☐ <n>w — ключ действителен n недель;
- ☐ <n>m — ключ действителен n месяцев;
- ☐ <n>y — ключ действителен n лет.

После этого вы увидите дату окончания действия ключа, и программа спросит, все ли правильно. Если все введенные данные верны, нажмите клавишу <Y>.

Далее программа попросит вас ввести ваше имя, адрес e-mail и комментарий для ключа (рис. 3).

```
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: Denis Kolisnichenko
Email address: dhsilabs@mail.ru
Comment:
You selected this USER-ID:
  "Denis Kolisnichenko <dhsilabs@mail.ru>"

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

Рис. 3. Информация о владельце ключа

Следующий шаг — это ввод пароля для ключа. Постарайтесь придумать хороший пароль. В пароле должны быть цифры, символы разного регистра и, желательно, специальные символы.

После ввода пароля и его подтверждения вам предстоит усердно понажимать клавиши на клавиатуре — это необходимо для создания случайных байтов. Нужно ввести как минимум 192 символа (рис. 4).

```
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.
+++++.++++.++++..+++++.....+++++.+++++.+++++.+++++.
+++++.++++.++++.++++.++++.+++++.+++++.+++++.+++++.>+
+++.....+++++
```

Рис. 4. Генерирование случайных байтов

Затем вы увидите сообщение, что ключ создан. Также программа отобразит все параметры ключа (рис. 5).

Самое время экспортировать ключ:

```
$ gpg --export -ao den.asc
```

```
gpg: /home/den/.gnupg/trustdb.gpg: trustdb created
gpg: key D6A721E6 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 2008-09-09
pub 1024D/D6A721E6 2008-03-13 [expires: 2008-09-09]
    Key fingerprint = 473F B009 0141 F2ED 9E73 B349 9B57 A747 D6A7 21E6
uid                               Denis Kolisnichenko <dhsilabs@mail.ru>
sub 2048g/60CA8C7E 2008-03-13 [expires: 2008-09-09]
```

Рис. 5. Ключ создан

Можно посмотреть созданный ключ (рис. 6): `cat den.asc`.

Полученный таким способом ключ называется *открытым ключом*. Вам нужно разослать его всем пользователям, с которыми вы планируете обмениваться важной информацией. Можно также выложить его на Web-сайте.

Теперь зарегистрируйтесь под вторым пользователем (user2) и аналогичным образом создайте ключ для него. Ключ нужно экспортировать в файл user2.asc.

После этого следует обменяться ключами — т. е. ключ пользователя `den` скопировать в домашний каталог пользователя `user2` и наоборот:

```
su
# cp /home/den/den.asc /home/user2
# cp /home/user2/user2.asc /home/den
```

Затем нужно подписать ключи. От имени первого пользователя (т. е. пользователя den) введите команду:

```
gpg --sign user2.asc
```

```

den@denis:~$ cat den.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)

mQGIBeFZD/URBAC5BPXFvCUNDh93wsTKfyvFid4xJiysTFpktApblcFN2o3hVyTi
TvtU0bEeUZ0U3GqGTRArFVh5Fv+TjsIOJyWAL0IQF3ssCH6YIuj+s+aCgtKg2itP
T2/YRjxtRwSoY1rH/MuHwbCgPGD1bcDoTHLW2Emw8btpo09FWSYGwFR9hwCgzJVE
YAF/2VC4j2nC8XjuhWSRSTUEAKs4Yp1Wwtylhjz4xhcaNl3zfCNglyxSnA6p1uMz
1U5XnI2MLbpPUHdSD0W1gRRzmM/5SAzU4Wmo3xBLLJ0MNVpyYiGqlyB367K0Ia5r
W6Yy7jcU+jeLW2BABIUjEgGZZspsQ1vq9QenYyZqRzgEf95+FqtyJt8yFK5U5hef
uoB9A/9E9uTEZwB0h90xVzNuMP7WEVqu4psGcZyknoj/jbm+dx8Jp9DCcyKmkQXD
gULklpFwrAdUvX3MUGuwi8gGEg7jpsdxI6T0kXdXL20+g3M2HpDAXZwzBhm7m5af
zGu1p5MRL+i3rk1NWdq0eUkLUqggHnQCnGp+yLTUEGgh2C52k7QmRGVuaXMgS29s
aXNuawNoZW5rbyA8ZGhzaWxhYnNAbWFpbC5ydT6IZgQTEQIAJgUCR9kP9QIbAwUJ
A010AAYLCQgHAWIEFQIIAwQWAgMBAh4BAheAAAJEJtXp0fWpyHmn1AAoJ2Zgu5D
dUEZgCV0wo1JZTswelzaAJ9MK//IJ3HgQG1CiFIQT3qCw35lt7kCDQRH2Q/4EAgA
wnBBzWUbGo7NSjd8URRWZatqcEepFwA6ADu4Chly/gzx0peJNFuzcI4W0ED5dhiz
dzfQwwUFTziwiSPSA+tcyyQapUTgcVhWU8Q0eQHSLa0+hm7WRk+BLiDtKS8tzpW
Nghk0VorY67mU4bvIcs125m4c3YCJ0wDT//yd6h/EIMMj60LckTyHqWKqsh0gKaA
882QE2U45hZKbdolWgGghlT1zSaHms+H0VUFKFBWu0oRf6+ajlDpqTtNDIJwwDw0
W1QaCYHF3vbHD06Mib5jzJkInAZWtEJb8KH4oZzE/lkervdSptJkstJ05TfMvS17
zdY6e3PFXrlevNodR41pAwADBQgAjh4eseJZL+WfKuHCDAKueQXX6eyqi3vcEYLH
p3oM69LMAB3fLDEIEG55svMYFL0SbrjBhkTduq4xLhgamZesH0DzzV9Bictn2lKR
OyHm4Qas+ekRWAewz0CraQn2ITeXQUcUiqyomm6PileGj+gx8ArcU8JG15FEr/HK

```

Рис. 6. Созданный ключ

Программа запросит пароль. Нужно ввести пароль, который вы вводили при создании ключа пользователя den (рис. 7).

Войдите в систему как пользователь user2 и подпишите ключ den.asc:

```
gpg --sign den.asc
```

Теперь проверим, как работает GnuPG. Пока вы находитесь в консоли под именем user2, введите команду:

```
echo "Hello, Den" > message.txt
```

Как вы уже догадались, команда создаст файл message.txt. Теперь зашифруем сообщение для пользователя den:

```
[user2@localhost user2]$ gpg -s -a dhsilabs@mail.ru message.txt
```

Вместо электронного адреса адресата можно использовать его фамилию, например,

```
[user2@localhost user2]$ gpg -s -a "Denis Kolisnichenko" message.txt
```

Программа попросит ввести пароль ключа пользователя user2.

ВНИМАНИЕ!

Будьте внимательны и не введите случайно пароль ключа пользователя den. На практике такая ситуация исключена, а вот во время эксперимента всякое может случиться.

Программа зашифрует файл и сохранит его под именем message.txt.asc. Его нужно скопировать в домашний каталог пользователя den и уже от его имени ввести команду:

```
gpg -d message.txt.asc > message.txt
```

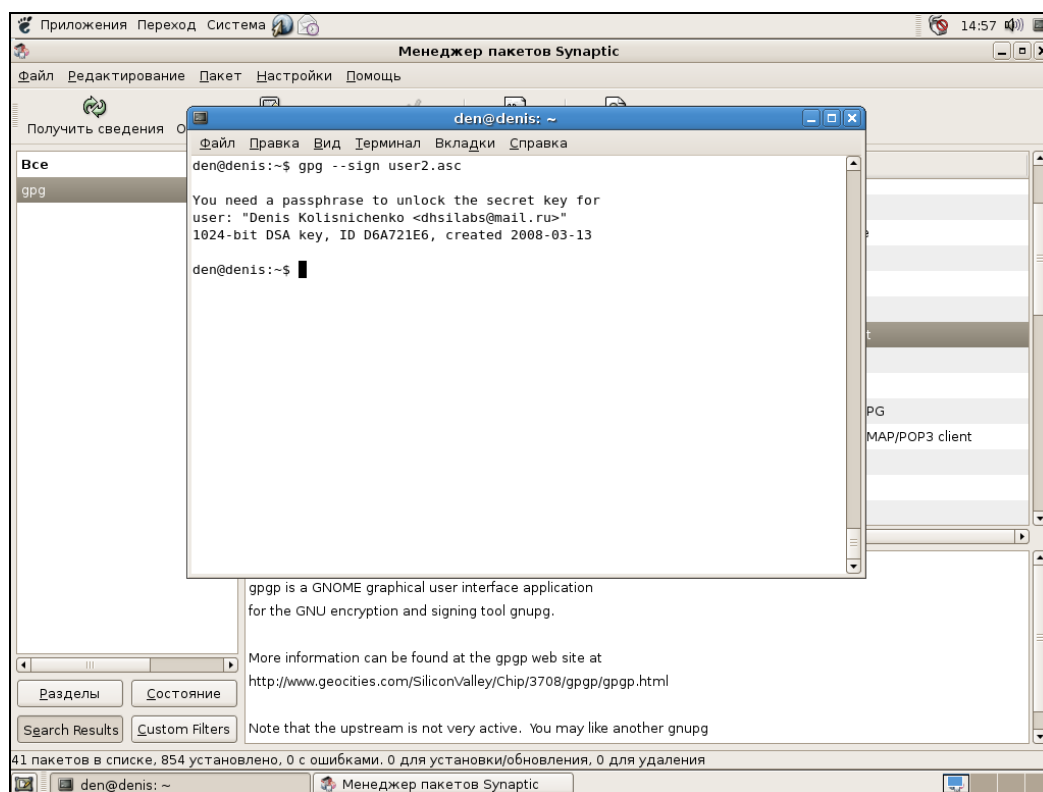


Рис. 7. Пользователь den подписал ключ user2.asc

Программа запросит пароль ключа пользователя den и расшифрует файл. В итоге в домашнем каталоге пользователя den появится файл message.txt с текстом "Hello, Den".

Как видите, GnuPG довольно удобно использовать для работы с конфиденциальной информацией. Сообщения электронной почты можно предварительно зашифровывать и передавать в уже зашифрованном виде. При наличии у получателя вашего ключа сообщение будет расшифровано.