GnuPG (Русский)

Состояние перевода: На этой странице представлен перевод статьи **GnuPG**. Дата последней синхронизации: 12 декабря 2015. Вы можете **помочь** синхронизировать перевод, если в английской версии произошли **изменения**.

Эта страница нуждается в сопроводителе

Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. **Команда переводчиков ArchWiki**

В соответствии с официальной **веб-страницей**:

> GnuPG - полная и свободная реализация **OpenPGP** стандарта, определенного в **RFC4880** (также известного, как PGP). GnuPG позволяет вам шифровать и подписывать данные и сообщения. Он оснащен универсальной системой управления ключами, а также модулями доступа для всех типов открытых ключей. GnuPG, также известный как GPG, это инструмент командной строки с возможностью легкой интеграции с другими приложениями. Доступен богатый выбор пользовательских приложений и библиотек. Также 2 версия GnuPG поддерживает S/MIME и Secure Shell (ssh).

Contents

[hide]

Установка

Установите пакет **gnupg**.

При этом будет установлен **pinentry** — набор простых диалоговых окон ввода PIN-кода или кодовой фразы, который использует GnuPG. Диалоговое окно *pinentry*определяется символической ссылкой `/usr/bin/pinentry`, которая по умолчанию указывает на `/usr/bin/pinentry-gtk-2`.

Настройка

Разположение каталогов

`$GNUPGHOME` используется GnuPG для определения каталога, в котором хранятся конфигурационные файлы. По умолчанию `$GNUPGHOME` не назначена и вместо этого используется `$HOME`; таким образом, вы найдете каталог `~/.gnupg` сразу после установки.

Чтобы изменить стандартное расположение, или выполните `$ gpg --homedir path/to/file`, или установите `GNUPGHOME` в одном из обычных загрузочных файлов.

Файлы конфигурации

Файлы конфигурации по умолчанию: `~/.gnupg/gpg.conf` и `~/.gnupg/dirmngr.conf`.

По умолчанию разрешения доступа каталога gnupg установлены в `700`, а файлов, которые он содержит - `600`. Только владелец каталога имеет разрешение на просмотр содержимого, радактирование и доступ к файлам. В целях безопасности, эти разрешения не должны быть изменены. В случае, если этот каталог или любые файлы внутри не следуют данной мере безопасности, вы получите предупреждение о наличии небезопасных файлов и разрешений домашнего каталога.

Добавляйте в эти файлы любые длинные опции, которые желаете. Не пишите две черты, просто имя опции и требуемые аргументы. Вы найдете шаблонные файлы (skeleton files) в `/usr/share/gnupg`. Эти файлы копируются в `~/.gnupg` во время первого запуска gpg, если их там нет. Другие примеры можно найти в **#Смотрите также**.

В дополнение, **pacman** использует различные настройки конфигурационных файлов для проверки подписи пакетов. Обратитесь к **Pacman/Package signing** для подробностей.

Стандартные настройки для новых пользователей

Если вы хотите задать какие-нибудь опции по умолчанию для новых пользователей, поместите соответствующие файлы в `/etc/skel/.gnupg`. Когда новый пользователь будет добавлен в систему, файлы отсюда будут скопированы в его домашний каталог GnuPG. Также имеется простой скрипт *addgnupghome*, с помощью которого вы можете создать новые домашние каталоги GnuPG для существующих пользователей:

```
# addgnupghome user1 user2
```

Команда добавит соответственно `/home/user1/.gnupg` и `/home/user2/.gnupg` и скопирует туда файлы из каталога шаблонов (skeleton directory). Пользователи с существующими домашними каталогами GnuPG просто будут пропущены.

Использование

Примечание: Всякий раз, когда команде требуется `user-id`, вы можете указать ваш key ID, отпечаток ключа (fingerprint), часть вашего имени, адреса электронной почты и т.д. GnuPG гибок в этом плане

Создание пары ключей

Создайте пару ключей, введя в терминал:

```
$ gpg --full-gen-key
```

Совет: Используйте опцию `--expert`, чтобы выбрать другие шифры, такие как **ECC**.

Команде потребуются ответы на несколько вопросов. Для общего использования большинство людей захотят:

- RSA (только для подписи) и RSA (только для шифрования) ключ.
- размер ключа по умолчанию (2048). Размер ключа больше 4096 "дорого обходится", но почти ничего не дает **[1]**.
- срок действия. Период в один год обычно достаточно хорош. В этом случае, даже если доступ к ключу будет потерян, он позволит другим узнать, что больше недействителен. Позже, при необходимости, срок действия может быть расширен без повторного выпуска нового ключа.
- ваше имя и адрес электронной почты. Вы можете добавить несколько идентификаторов к одному и тому же ключу позже (*например*, если у вас есть несколько адресов электронной почты, которые вы хотите связать с этим ключом).
- обязательный комментарий. Поскольку семантика поля комментария **не определена**, она имеет ограниченное значение для идентификации.
- **безопасная парольная фраза**.

Примечание: Имя и адрес электронной почты, которые вы вводите здесь, будут видеть все, кто импортирует ваш ключ.

Управление ключами

- Запуск `gpg --edit-key <идентификатор>` отобразит меню, с помощью которого вы можете выполнять большинство задач, связанных с управлением ключами. Например, для указания даты окончания срока ключа:

```
$ gpg --edit-key <идентификатор>
> key номер ключа
> expire гггг-мм-дд
> save
> quit
```

Некоторые полезные команды:

```
> passwd       # изменить пароль
> clean        # сжать все неиспользуемые идентификаторы (которые были
отозваны либо их срок действия истек)
> revkey       # отозвать ключ
> addkey       # добавить подключ к текущему ключу
> expire       # изменить дату окончания срока действия ключа
```

Tip: Если у вас несколько email аккаунтов, вы можете добавить каждый из них как идентификатор, используя комманду `adduid`. Вы можете установить один приоритетный с помощью `primary`.

- Отобразить открытый ключ в ASCII-совместимом формате (для распространения):

```
$ gpg --armor --output public.key --export <идентификатор>
```

- Зарегистрировать ключ в публичном PGP сервере ключей, чтобы другие пользователи могли получить ваш открытый ключ без необходимости напрямую с вами связываться:

```
$ gpg  --keyserver pgp.mit.edu --send-keys <идентификатор>
```

- Подписать и зашифровать файл для пользователя Bob:

```
$ gpg se -r Bob файл
```

- Создать текстовый файл со встроенной подписью:

```
$ gpg --clearsign файл
```

Экспорт подключей

Если вы планируете использовать тот же ключ на разных устройствах, вы можете вырезать мастер-ключ и оставить только минимально необходимый подключ для менее защищенных систем.

Прежде всего, определите, какой подключ вы хотите экспортировать.

```
$ gpg -K
```

Укажите, что экспортировать нужно только конкретный подключ.

```
$ gpg -a --export-secret-subkeys [subkey id]! > /tmp/subkey.gpg
```

Важно: Если вы забудете добавить !, будут экспортированы все ваши подключи.

На этом этапе вы можете закончить, но хорошей идеей будет изменить пароль. Импортируйте ключ во временный каталог.

```
$ gpg --homedir /tmp/gpg --import /tmp/subkey.gpg
$ gpg --homedir /tmp/gpg --edit-key <user-id>
> passwd
> save
$ gpg --homedir /tmp/gpg -a --export-secret-subkeys [subkey id]! >
/tmp/subkey.altpass.gpg
```

Примечание: Вы получите предупреждение о недоступности мастер-ключа и, что пароль не был изменен, но вы спокойно можете игнорировать это предупреждение, поскольку пароль для подключа был изменен.

Теперь вы уже можете использовать `/tmp/subkey.altpass.gpg` на других ваших устройствах.

Ротация подключей

Важно: Никогда не удаляйте просроченные или отозванные подключи без особой на то причины, иначе вы полностью потеряете возможность расшифровать файлы, зашифрованные старым подключом. Удаляйте **только** чужие просроченные или отозванные подключи, если хотите очистить свой список открытых ключей.

Если вы выбрали дату окончания срока действия подключа, желательно уже за несколько недель до этой даты создать новый. Таким образом, у других пользователей будет возможность вовремя обновить свой список ключей.

Примечание: Вам не обязательно создавать новый ключ только потому, что предыдущий просрочен. Вы всегда можете продлить срок ключа.

- Создайте новый подключ (запустите дважды для создания отдельных ключей для подписывания и шифрования)

```
$ gpg --edit-key <идентификатор>
> addkey
```

При этом вам будет задано несколько вопросов (рекомендованные настройки смотрите в предыдущем разделе).

- Сохраните изменения

```
> save
```

- Обновите ключ на сервере ключей.

```
$ gpg  --keyserver pgp.mit.edu --send-keys <идентификатор>
```

Примечание: Отзывать просроченные подключи абсолютно бесполезно и является плохой практикой. Если вы постоянно отзываете ключи, уверенность других пользователей в вас может быть подорвана.

Импортирование ключей

- Импортировать открытый ключ в ваш список открытых ключей:

```
$ gpg --import public.key
```

- Импортировать закрытый ключ в ваш список закрытых ключей:

```
$ gpg --import private.key
```

- Импортировать ключ с сервера ключей (если '--keyserver' не указан, будет использован сервер по умолчанию):

```
$ gpg --keyserver pgp.mit.edu --recv-keys <идентификатор>
```

Отображение ключей

- Вывести список открытых ключей:

```
$ gpg --list-keys
```

- Вывести список закрытых ключей:

```
$ gpg --list-secret-keys
```

Шифрование и дешифрование

Для шифрования и дешифрования есть возможность использовать сразу несколько закрытых ключей. В этом случае вам будет необходимо указать, какой именно ключ следует применить. Для этого используйте опцию `-u <идентификатор>` (или `--local-user <идентификатор>`), иначе будет использован ключ, выбранный по умолчанию.

Чтобы зашифровать файл:

```
$ gpg --encrypt -o secret.tar.gpg secret.tar
```

- Чтобы указать, для кого будет зашифрован файл, воспользуйтесь опцией `-r <идентификатор>` (или `--recipient <идентификатор>`).
- Вы можете использовать gnupg для шифрования важных документов, но только один файл за раз. Если вы хотите зашифровать директорию или целую файловую систему, вам стоит взглянуть на **TrueCrypt** или **EncFS**. Тем не менее, вы всегда можете заархивировать несколько файлов и затем зашифровать архив.

Чтобы расшифровать файл:

```
$ gpg --decrypt secret.tar.gpg
```

Необходимо будет ввести ваш пароль.

Шифрование пароля

Может быть полезно зашифровать какой-нибудь пароль, чтобы он не хранился в чистом виде в файле настроек. Например, пароль от вашей учетной записи электронной почты.

Первым делом создайте файл пароля, содержащий только ваш пароль и пустую строку. Обратите внимание: файл **должен** содержать одну пустую строку в конце, иначе gpg выведет сообщение об ошибке.

Теперь выполните:

```
$ gpg -e -a -r <идентификатор> файл_пароля
```

Опция `-e` обозначает режим шифрования, `-a` — для вывода в ASCII-совместимом формате, `-r` — идентификатор ключа.

После выполнения команды в текущем каталоге будет создан новый файл `файл_пароля.asc`.

gpg-agent

gpg-agent чаще всего используется как посредник для временного хранения пароля (пароль не будет запрашиваться каждый раз, когда нужен). Он полезен если GnuPG используется внешней программой — например, почтовым клиентом.

Включение

Начиная с версии **gnupg**-2.1, *gpg-agent* используется по умолчанию. Если вы используете предыдущие версии, вы можете включить его, добавив следующую строку в `gpg.conf`:

```
~/.gnupg/gpg.conf

use-agent
```

Теперь GnuPG будет использовать агент каждый раз, когда ему необходим пароль.

Запуск


Эта статья или раздел нуждается в переводе


Примечания: Перевод устарел: Переменная **#GPG_AGENT_INFO**[broken link: invalid section] устарела начиная с **gnupg**-2.1. Обратитесь к **#Unattended passphrase** за новым методом. (обсуждение: **Talk:GnuPG (Русский)#**)

Чтобы агент запускался автоматически, добавьте следующую запись в ваш файл инициализации среды (например, `.xinitrc` или `.bash_profile`). Если каталог настроек GnuPG у вас расположен не в стандартном месте (`~/.gnupg`), не забудьте при этом соответствующим образом обновить значение переменной `envfile`.

```
~/.bash_profile

envfile="$HOME/.gnupg/gpg-agent.env"
if [[ -e "$envfile" ]] && kill -0 $(grep GPG_AGENT_INFO "$envfile" | cut -
d: -f 2) 2>/dev/null; then
    eval "$(cat "$envfile")"
else
    eval "$(gpg-agent --daemon --enable-ssh-support --write-env-file
"$envfile")"
fi
export GPG_AGENT_INFO  # the env file does not contain the export
statement
export SSH_AUTH_SOCK   # enable gpg-agent for ssh
```

Перезайдите в систему и убедитесь, что *gpg-agent* запущен:

```
$ pgrep gpg-agent
```

Совет: Чтобы убедиться в том, что gpg-agent работает корректно, наберите `$ gpg-connect-agent`. Если всё сделано правильно, будет отображено приглашение для ввода команд (наберите `bye` и `quit` для того, чтобы закрыть соединение и выйти).

Запуск с помощью systemd

Также вы можете использовать **systemd в пользовательском режиме** для запуска агента. Это имеет смысл и при **gnupg** >= 2.1, поскольку так он будет запускаться вне пользовательских сессий и под контролем systemd.

Создайте файл юнита для gpg-agent:

```
~/.config/systemd/user/gpg-agent.service
```

```
[Unit]
Description=GnuPG private key agent
IgnoreOnIsolate=true

[Service]
Type=forking
ExecStart=/usr/bin/gpg-agent --daemon --homedir=%h/.gnupg
ExecReload=/usr/bin/gpg-connect-agent RELOADAGENT
Restart=on-abort

[Install]
WantedBy=default.target
```

Затем включите его в список автозапуска и запустите его командами `systemctl --user daemon-reload`, `systemctl --user enable gpg-agent` и `systemctl --user start gpg-agent`.

Примечание:

- Вам может понадобиться установить для агента какие-нибудь переменные окружения, например `GNUPGHOME`. Как это сделать, смотрите в разделе **systemd/User#Environment variables**.
- Если вы используете стандартный каталог настроек GnuPG (`~/.gnupg`), вам не нужно указывать его явно. В примере он указан исключительно для справки.
- Если вы используете SSH возможности gpg-agent'a (--enable-ssh-support), приведенный выше юнит systemd работать не будет.

Настройка

gpg-agent можно настроить в файле `~/.gnupg/gpg-agent.conf`. Все опции для настройки перечислены на странице **gpg-agent(1)**. Например, так вы можете задать время жизни для ключей в кэше с момента последнего использования:

```
~/.gnupg/gpg-agent.conf
```

```
default-cache-ttl 3600
```

Tip: Для кеширования вашего пароля на протяжении всей сессии, используйте команду:

```
$ /usr/lib/gnupg/gpg-preset-passphrase --preset XXXXXX
```

где XXXX *keygrip*. Вы можете получить это значение запустив `gpg --with-keygrip -K`. Пароль будет храниться до перезапуска `gpg-agent`. Установленное значение`default-cache-ttl` будет иметь превосходство.

Перезапуск агента

После обновления настроек, перезапустите агент, передав команду `RELOADAGENT` программе *gpg-connect-agent*:

```
$ echo RELOADAGENT | gpg-connect-agent
```

Будет выведено сообщение `OK`.

Или же, если вы используете **#Запуск с помощью systemd**, можно использовать следующую команду:

```
$ systemctl --user reload gpg-agent
```

pinentry

gpg-agent использует pinentry для отображения диалога запроса пароля. Это настраивается в файле настроек gpg-agent.

По умолчанию используется диалог **GTK+**. Однако, есть и другие варианты — смотрите подробнее в `info pinentry`. Чтобы установить другой диалог, установите опцию `pinentry-program`:

```
~/.gnupg/gpg-agent.conf

# PIN entry program
# pinentry-program /usr/bin/pinentry-curses
# pinentry-program /usr/bin/pinentry-qt
# pinentry-program /usr/bin/pinentry-kwallet
pinentry-program /usr/bin/pinentry-gtk-2
```

Совет: Чтобы использовать `/usr/bin/pinentry-kwallet` потребуется установить пакет **kwalletcli**[AUR].

После внесения изменений перезапустите gpg-agent.

Unattended passphrase

Начиная с GnuPG 2.1.0 использование gpg-agent и pinentry стало обязательным; это нарушает обратную совместимость для парольных фраз, которые передавались через входной поток с помощью опции `--passphrase-fd 0`. Чтобы иметь возможность сделать, как раньше, требуется выполнить несколько шагов.

Первым делом, отредактируйте настройки gpg-agent, разрешив режим петли (*loopback*) для pinentry:

```
~/.gnupg/gpg-agent.conf

allow-loopback-pinentry
```

Перезапустите процесс gpg-agent чтобы изменения вступили в силу.

Теперь либо запускайте GnuPG с опцией `--pinentry-mode loopback`

```
$ gpg --pinentry-mode loopback ...
```

либо добавьте ее же в файл настроек GnuPG:

```
~/.gnupg/gpg.conf
```

```
pinentry-mode loopback
```

Примечание: Разработчики отмечают, что установка `pinentry-mode loopback` в файле настроек может нарушать другую функциональность GnuPG, и, если это возможно, лучше указывать опцию через командную строку **[2]**.

Встречи для подписи ключей

Чтобы дать возможность пользователям проверить ключи в хранилищах ключей и с собственных списках (то есть, убедиться, что владелец ключа на самом деле тот, за кого себя выдает), PGP/GnuPG использует так называемую "сеть доверия". Для поддержания и развития сети периодически организуются очные встречи, на которых люди, использующие систему PGP, обмениваются своими публичными ключами.

Протокол Циммермана–Сассамана призван сделать этот процесс наиболее эффективным. **Здесь** вы можете найти подробную инструкцию по проведению встреч на русском языке.

caff

Для упрощения процедуры подписи ключей и отправку этих подписей владельцам ключей вы можете воспользоваться утилитой *caff*. Установить ее можно из **AUR** с пакетом **`caff-svn`**AUR. Там же доступен **`signing-party-svn`**AUR[ссылка недействительна: сохранено в aur-mirror] — набор удобных инструментов, включающий также *caff*. Так или иначе, потребуется установить много зависимостей из AUR. Как вариант, вы можете загрузить и установить эти зависимости из CPAN:

```
cpanm Any::Moose
cpanm GnuPG::Interface
```

Для отправки подписей владельцам вам нужен работающий агент **MTA**. Если у вас его еще нет, установите **msmtp**.

Смарт-карты

Примечание: Необходимо установить пакеты **`pcsclite`** и **`libusb-compat`**, а также **запустить** соответствующую службу systemd: `pcscd.service`.

GnuPG использует *scdaemon* как интерфейс к вашему устройству для чтения смарт-карт. Для получения дополнительной информации обратитесь к **man-странице** `scdaemon (1)`.

Настройка только для GnuPG

Если вы не планируете использовать другие карты, кроме тех, что работают на основе GnuPG, необходимо проверить параметр `reader-port` в файле `~/.gnupg/scdaemon.conf`. Значение '0' относится к первому доступному считывателю последовательного порта, а значение '32768' (по умолчанию) — к первому считывателю USB.

GnuPG вместе с OpenSC

Если вы используете смарт-карту с драйвером opensc (например, ID-карты, распространенные в некоторых странах), необходимо уделить чуть большее время настройке GnuPG. Используя стандартную конфигурацию, при запросе `gpg --card-status` вы можете получать сообщения вроде этого:

```
gpg: selecting openpgp failed: ec=6.108
```

По умолчанию scdaemon пытается подключиться к устройству напрямую. Эта попытка провалится, если считыватель карт используется другим процессом. Например, если демон pcscd используется OpenSC. Чтобы справиться с этой ситуацией, необходимо использовать тот же самый драйвер, который использует opensc — тогда они смогут работать вместе. Чтобы заставить scdaemon использовать pcscd, необходимо удалить `reader-port` из файла `~/.gnupg/scdaemon.conf`, указать путь к библиотеке `libpcsclite.so` и отключить ccid, чтобы удостовериться, что используется именно pcscd:

```
~/scdaemon.conf

pcsc-driver /usr/lib/libpcsclite.so
card-timeout 5
disable-ccid
```

Обратитесь к **man-странице** `scdaemon (1)`, если вы не используете OpenSC.

Решение проблем

Доступно недостаточно случайных байтов

При генерации ключа gpg может отработать с такой ошибкой:

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

Для проверки доступной энтропии, проверте параметры ядра:

```
cat /proc/sys/kernel/random/entropy_avail
```

Здоровая система Линукс с большим количеством доступной энтропии вернет значение близко к полным 4,096 бит энтропии. Если возвращенное значение менее 200, система имеет низкую энтропию.

Для решения этого, вам не стоит много раз создавать ключи, а лучше всего делайте то, что предлагает сообщение выше(например делайте активным диск, двигайте мышкой, редактируйте википедию - все это создает энтропию). Если это не помогает, проверьте какой сервис использует энтропию и рассмотрите временную его остановку. Если это не альтернатива, смотрите **Random number generation#Alternatives**.

su

При использовании `pinentry`, у вас должны быть корректные настройки прав доступа к устройству терминала (например `/dev/tty1`). Однако, с `su` (или `sudo`) права доступа остаются от прежнего пользователя системы. Из-за этого будут возникать проблемы с pinentry, даже при запуске от имени суперпользователя. Чтобы исправить эту проблему, назначьте нового владельца к устройству терминала до использования pinentry (например, используя gpg-agent). Используя *gpg* от суперпользователя, просто измените владельца на root перед запуском *gpg*:

```
chown root /dev/ttyN  # где N — текущий tty
```

Затем верните прежнего владельца после первого запуска *gpg*. Аналогично должно работать и для `/dev/pts`.

Примечание: Владелец tty должен совпадать с пользователем, от имени которого запущена pinentry. Добавления пользователя в группу `tty` **не** поможет решить проблему.

Agent выводит ошибку end of file

По умолчанию используется диалог pinentry-gtk-2, для правильной работы которого требуется запущенный DBus. Для получения дополнительной информации смотрите раздел **Устранение часто встречающихся неполадок#Разрешения сессии**.

Также вы можете использовать версию `pinentry-qt`. Как это сделать, смотрите в разделе **#pinentry**.

Настройка прав доступа для KGpg

Некоторые пользователи сталкивались с проблемой, когда **`kdeutils-kgpg`**[ссылка недействительна: replaced by `kgpg`] не может получить доступ к настройкам в `~/.gnupg/`. Одна из причин может быть в устаревшем файле опций. Подробности смотрите в **отчете об ошибке**.

Также наблюдались проблемы, когда *KGpg* не запускается, если права доступа к директории `~/.gnupg` отличаются от `drwxr-xr-x`. Если вы изменили права доступа таким образом, обязательно убедитесь, что все файлы внутри каталога имеют права доступа `-rw-------`! Затем, отправьте отчет об этой ошибке **разработчикам**.

mutt и gpg

Если вы хотите, чтобы пароль запрашивался только единожды за один сеанс в новой версии GnuPG 2.1, смотрите **эту ветку форума**.

"Потерявшиеся" ключи после обновления до GnuPG 2.1

Если команда `gpg --list-keys` перестала отображать какие-то ключи, а приложения ругаются на отсутствующие/поврежденные ключи, вероятно, какие-то ключи не были сконвертированы в новый формат.

Пожалуйста, прочтите **исправление ошибки Invalid packet**. Здесь говорится, что существует баг с ключами в старых файлах `pubring.gpg` и `secring.gpg`, которые были заменены файлом `pubring.kbx` и подкаталогом `private-keys-v1.d/`. Потерянные ключи можно восстановить следующими командами:

```
$ cd
$ cp -r .gnupg gnupgOLD
$ gpg --export-ownertrust > otrust.txt
$ gpg --import .gnupg/pubring.gpg
$ gpg --import-ownertrust otrust.txt
$ gpg --list-keys
```

gpg зависает на всех серверах ключей(при попытке получения ключей)

Если gpg зависает на определенном сервере ключей, когда пытается получить ключи, вам прийдется убить dirmngr для того, чтобы получить доступ к другим действительно рабочим серверам, в противном случае gpg останется зависшим для всех них.

Смарт-карта не обнаружена

Пользователь, из-под которого вы работаете, похоже, не имеет права доступа к смарт-карте, в следствие чего и возникает `card error`, даже если карта корректно настроена и установлена.

Одно из возможных решений - добавить новую группу `scard` с включением в нее пользователей, которым нужен доступ к смарт-катре.


Эта статья или раздел нуждается в переводе


Примечания: Перевод устарел: Is `MODE="664"` necessary? Assigning a group, `MODE="660"` may be enough? (обсуждение: Talk:GnuPG (Русский)#)

Дальше используйте подобное **udev** правило:

```
/etc/udev/rules.d/71-gnupg-ccid.rules

ACTION=="add", SUBSYSTEM=="usb", ENV{ID_VENDOR_ID}=="1050",
ENV{ID_MODEL_ID}=="0116|0111", MODE="664", GROUP="scard"
```

Только нужно адаптировать VENDOR и MODEL в согласии с выводом `lsusb`. Выше приведен пример для YubikeyNEO.

Смотрите также

- **The GNU Privacy Handbook**
- **A more comprehensive gpg Tutorial**
- **GnuPG FAQ**
- **gpg.conf recommendations and best practices**
- **Torbirdy gpg.conf**
- **OpenPGP subkeys in Debian**

Categories:

- Encryption (Русский)
- Email (Русский)
- GNU (Русский)
- Русский