

Доклад на тему: Система PGP

Дисциплина: Основы информационной безопасности

Мишина Анастасия Алексеевна НПИбд-02-22

Содержание

1	Введение	5
2	Основные принципы PGP	6
3	PGP в Ubuntu	8
4	Выводы	21
	Список литературы	22

Список иллюстраций

3.1	Общая схема шифрования и дешифрования	9
3.2	Установка GPG	10
3.3	Проверка правильности установки	10
3.4	Создание пары ключей	13
3.5	Состав ключа	13
3.6	Просмотр ключа и его отпечатка	15
3.7	Экспорт ключа	16
3.8	Передача ключа	16
3.9	Импорт ключа	17
3.10	Создание, экспорт, передача, импорт ключа Иры	17
3.11	Создание секретного файла	17
3.12	Файл secret_file.txt	18
3.13	Создание подписи файла, шифрование файла	19
3.14	Файл цифровой подписи	19
3.15	Зашифрованный файл с текстом	19
3.16	Дешифровка и проверка подписи	20
3.17	Просмотр дешифрованного файла	20

Список таблиц

3.1 Основные опции команды pgr	11
--	----

1 Введение

В XXI веке одной из основных проблем является качественная защита пользовательских данных. С ростом популярности Интернета возросла потребность и в защите информации, ведь никто не хочет, чтобы злоумышленники получили доступ к его финансам, интернет-перепискам и различным аккаунтам.

В этом контексте криптография стала ключевым инструментом для обеспечения конфиденциальности персональных данных в современном обществе. Стремясь обеспечить безопасность электронной переписки, американский программист Филипп Циммерман разработал пакет программ для обмена сообщениями по электронной почте, известный как PGP (Pretty Good Privacy).

PGP была выпущена компанией Phil's Pretty Good Software в 1991 году и является криптографической системой с высокой степенью секретности для ОС Linux, MacOS, Windows. Программа позволяет пользователям обмениваться файлами или сообщениями с использованием функций секретности, установлением подлинности, и высокой степенью удобства. Секретность означает, что сообщение будет прочитано только тем, кому оно адресовано. Установление подлинности гарантирует, что сообщение, полученное от какого-либо человека, было послано именно им. Преимуществом PGP является отсутствие необходимости в использовании секретных каналов связи, что делает PGP простым в использовании. Это связано с тем, что PGP базируется на мощной новой технологии, которая называется шифрованием с "открытым ключом" и выполняет функции быстрее, чем большинство реализаций этого алгоритма.

2 Основные принципы PGP

Напомним суть криптографии с открытым ключом: система работает на основе двух ключей: открытого и закрытого. Открытый ключ является доступным и позволяет зашифровать сообщение отправителя, закрытый же ключ имеется только у получателя для расшифровки письма. Это обеспечивает конфиденциальность без использования специальных каналов связи [1].

Как было сказано во вступлении, помимо защиты, PGP дает возможность аутентифицировать информацию. Оказывается, что закрытый ключ может также использоваться как «подпись» посылаемой информации; полная аналогия с подписями, которые часто ставят на бумажных документах.

Для обеспечения защиты и аутентификации могут использоваться оба процесса, как кодирование, так и подпись. Они могут быть объединены для обеспечения и секретности, и установления подлинности: сначала подписывается сообщение вашим собственным закрытым ключом, а потом шифруется уже подписанное сообщение открытым ключом получателя. Получатель делает наоборот: расшифровывает сообщение с помощью собственного закрытого ключа, а затем проверяет подпись с помощью вашего открытого ключа. Эти шаги выполняются автоматически с помощью программного обеспечения получателя.

Открытый ключ хранится в виде «сертификата ключа», который включает в себя идентификатор пользователя-владельца (обычно это имя пользователя) и сам ключ с датой его создания. Закрытый, помимо этого, защищен отдельным паролем.

Оба ключа хранятся в файле, известном как кольцо ключей – «keyring», в кото-

ром также хранятся различные сертификаты ключей. Обычно есть кольцо для открытых ключей и кольцо для закрытых.

Ключи имеют внутренний идентификатор ключа, который состоит из 64 последних бит открытого ключа. При отображении информации о ключе на самом деле показываются последние 32 бита для краткости. Эти идентификаторы ключа используются PGP, например, для определения ключа при декодировании сообщения.

При подписывании документа PGP формирует 128 бит, которые представляют документ – «дайджест сообщения». Эта подпись является своего рода контрольной суммой, или CRC, которая позволяет обнаружить изменения в документе. В отличие от обычных CRC или контрольных сумм, никто не может заново создать эту подпись чтобы узаконить любые изменения исходного документа. Подпись создается при помощи закрытого ключа отправителя и тот, кто хочет внести изменения, не имеет к нему доступа.

3 PGP в Ubuntu

Чтобы показать на примере как работает PGP будем использовать GPG – GNU Privacy Guard [2]. Это свободная программа для шифрования информации и создания электронных цифровых подписей. Разработана как альтернатива PGP и выпущена под свободной лицензией GNU General Public License. Этот инструмент обеспечивает цифровое шифрование и службы подписи используя стандарт OpenPGP, расширяя его возможности.

Принцип работы GnuPG состоит в следующем: когда пользователь шифрует сообщение, то программа сначала сжимает текст, что сокращает время на отправку сообщения и увеличивает надежность шифрования. Большинство приемов криптоанализа (взлома зашифрованных сообщений) основаны на исследовании «рисунков», присущих текстовым файлам, что помогает взломать ключ. Сжатие ликвидирует эти «рисунки» и таким образом повышает надежность зашифрованного сообщения. Затем GnuPG генерирует сессионный ключ, который представляет собой случайное число, созданное за счет движений вашей мышки и нажатий на клавиши клавиатуры. Как только данные будут зашифрованы, сессионный ключ зашифровывается с помощью публичного ключа получателя сообщения, который отправляется к получателю вместе с зашифрованным текстом. Расшифровка происходит в обратной последовательности. Программа PGP получателя сообщения использует закрытый ключ получателя для извлечения временного сессионного ключа, с помощью которого программа затем дешифрует зашифрованный текст (рис. 3.1).

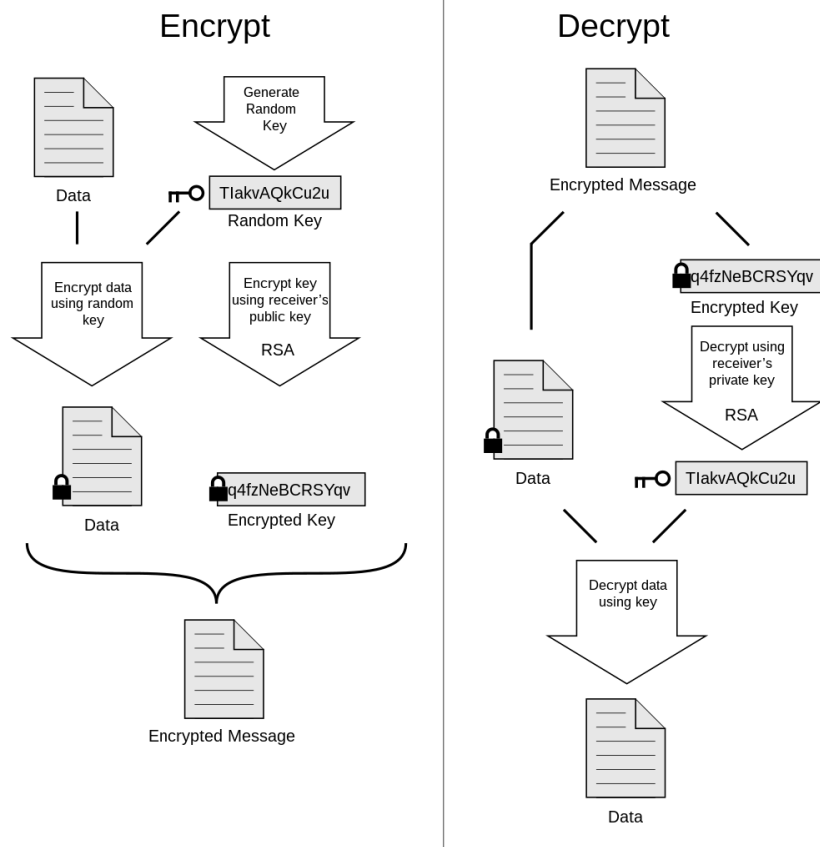


Рис. 3.1: Общая схема шифрования и дешифрования

Для начала установим GPG с помощью команды: `sudo apt install pgpgpg` (рис. 3.2). Проверим успешность установки командой: `gpg --version`, видим версию 2.4.4 (рис. 3.3).

```

aamishina@server:~$ sudo apt update
Суц:1 http://ports.ubuntu.com/ubuntu-ports noble InRelease
Суц:2 http://ports.ubuntu.com/ubuntu-ports noble-updates InRelease
Суц:3 http://ports.ubuntu.com/ubuntu-ports noble-backports InRelease
Суц:4 http://ports.ubuntu.com/ubuntu-ports noble-security InRelease
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Все пакеты имеют последние версии.
aamishina@server:~$ sudo apt install pgpgpg
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие НОВЫЕ пакеты будут установлены:
  pgpgpg
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Необходимо скачать 16,5 kB архивов.
После данной операции объем занятого дискового пространства возрастет на 93,2 kB.
Пол:1 http://ports.ubuntu.com/ubuntu-ports noble/universe arm64 pgpgpg arm64 0.13-12 [16,5 kB]
Получено 16,5 kB за 0с (89,4 kB/s)
Выбор ранее не выбранного пакета pgpgpg.
(Чтение базы данных ... на данный момент установлено 166524 файла и каталога.)
Подготовка к распаковке .../pgpgpg_0.13-12_arm64.deb ...
Распаковывается pgpgpg (0.13-12) ...
Настраивается пакет pgpgpg (0.13-12) ...
update-alternatives: используется /usr/bin/pgpgpg для предоставления /usr/bin/pg
r (pgp) в автоматическом режиме
Обрабатываются триггеры для man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

```

Рис. 3.2: Установка GPG

```

aamishina@server:~$ gpg --version
gpg (GnuPG) 2.4.4
libgcrypt 1.10.3
Copyright (C) 2024 g10 Code GmbH
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: /home/aamishina/.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
aamishina@server:~$

```

Рис. 3.3: Проверка правильности установки

Для начала следует рассказать об основных возможностях данной команды. В таблице 3.1 приведу основные опции `pgp`, необходимые для работы:

Таблица 3.1: Основные опции команды `gpg`

Опция	Значение
<code>--export</code>	Экспортировать публичный ключ в файл, который потом можно отправить.
<code>--import</code>	Импортировать публичный ключ.
<code>--armor -a</code>	Создаёт ASCII (символьный) вывод. При шифровании GPG по умолчанию создаёт бинарный вывод. При использовании этой опции GPG кодирует информацию кодировкой Radix-64 (Разновидность Base64). Этот текстовой вывод можно, например, отправить в мессенджере или по электронной почте, а также вывести на экран.
<code>--encrypt -e</code>	Зашифровать сообщение.
<code>--recipient -r</code>	Указать ключ, который будет использоваться для шифрования. Можно использовать информацию идентификатор пользователя (имя, почта), идентификатор ключа, отпечаток ключа.
<code>--decrypt -d</code>	Расшифровать сообщение.
<code>--sign -s</code>	Подписать сообщение. Подпись при этом будет располагаться отдельно от самого сообщения.
<code>--clear-sign</code>	Подписать сообщение. Подпись при этом сохраняется вместе с сообщением.
<code>--local-user -u</code>	Указать ключ, который будет использоваться для подписи. Схож с опцией <code>--recipient</code> , но это не одно и то же.
<code>--verify</code>	Проверить подпись.
<code>--list-keys -k</code>	Вывести список публичных ключей.

Опция	Значение
<code>--list-secret-keys</code>	Вывести список приватных ключей.
<code>-K</code>	
<code>--edit-key</code>	Редактировать ключ.
<code>--expert</code>	Режим эксперта.

Теперь смоделируем следующую ситуацию: допустим, у нас есть два студента Настя и Ира. У Насти есть файл, информация в котором очень важна, поэтому перед его отправкой необходимо выполнить шифрование и создание подписи. Задача Иры – дешифровать файл после получения, а также понять, действительно ли файл отправила Настя. Будем считать, что Настя работает в каталоге `first`, а Ира в каталоге `second`.

Всё начинается с генерации ключей. Для генерирования пары ключей (открытый и закрытый) можно использовать команду: `gpg --gen-key`. Однако, вместо `--gen-key` мы выберем опцию `--full-generate-key`, которая также сгенерирует пару ключей, но позволит настроить наш выбор (если запустить GPG ещё и с аргументом `--expert`, то выбор типа ключа будет намного шире). Создаем пару ключей (рис. 3.4), выбираем тип RSA, выделяем 2048 бит под ключ (меньше не рекомендуется использовать, так как это небезопасно), ставим срок действия ключа – 14 дней. При желании срок действия ключа можно продлить. Далее указываем имя и адрес электронной почты для пользователя Настя, придумываем пароль. На этом этапе ключ генерируется и добавляется в связку ключей. В связке ключей может находиться множество ключей. Также на этом этапе создаётся сертификат отзыва — файл, с помощью которого созданный ключ можно отозвать (признать недействительным).

```

aamishina@server:~/first$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
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
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (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) 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) 14
Key expires at Вc 12 мая 2024 19:41:47 UTC
Is this correct? (y/N) y

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

Real name: Nastya
Email address: nastya@gmail.com
Comment:
You selected this USER-ID:
    "Nastya <nastya@gmail.com>"

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

```

Рис. 3.4: Создание пары ключей

Что же из себя представляет ключ (рис. 3.5):

```

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.
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.
gpg: revocation certificate stored as '/home/aamishina/.gnupg/openpgp-revocs.d/E2E27AC8DC84DC38CA9183EC0648F9A2650556F2.rev'
public and secret key created and signed.

pub   rsa2048 2024-04-28 [SC] [expires: 2024-05-12]
       E2E27AC8DC84DC38CA9183EC0648F9A2650556F2
uid     Nastya <nastya@gmail.com>
sub    rsa2048 2024-04-28 [E] [expires: 2024-05-12]

aamishina@server:~/first$

```

Рис. 3.5: Состав ключа

rsa — Алгоритм шифрования RSA.

2048 — Длина ключа.

2024-04-28 — Дата создания ключа.

E2E27A...556F2 — Отпечаток ключа. Его следует сверять при импортировании чужого публичного ключа — у обеих сторон он должен быть одинаков.

uid — Идентификатор (User-ID).

pub и sub — Типы ключа:

- pub — Публичный ключ.
- sub — Публичный подключ.
- sec — Секретный ключ.
- ssb — Секретный подключ.

[SC] и [E] — Предназначение каждого ключа. Когда вы создаёте ключ, вы получаете аж 4 криптоключа: для шифрования, расшифровки, подписи и проверки подписи:

- S — Подпись (Signing).
- C — Подпись ключа (Certification). Об этом пойдёт речь чуть позже.
- E — Шифрование (Encryption).
- A — Авторизация (Authentication). Может использоваться, например, в SSH.

С помощью команды `gpg -list-keys` просматриваем все публичные ключи в системе, командой `gpg -fingerprint` смотрим на отпечаток нашего ключа (рис. 3.6). Отпечаток может выполнять функцию идентификатора ключа — то есть вместо указания имени пользователя можно использовать отпечаток. Вторая функция применения отпечатка — это верификация публичного ключа. В качестве примера приводится следующая история: Настя передала Ире клочок бумаги, на котором записан отпечаток (ну или позвонила по телефону и продиктовала отпечаток). Затем Настя переслала Ире свой публичный ключ. Но так как ключ пришёл из Интернета — то непонятно, от кого он именно? Не был ли этот ключ подменён по пути? После импорта ключа, можно просмотреть его отпечаток.

Поскольку у Иры есть отпечаток, который она получила из доверенного источника (Настя продиктовала его своим голосом или лично передала записку с отпечатком), то Ира теперь может сравнить эти два отпечатка — если отпечатки идентичные, значит публичный ключ действительно отправлен Настей и значит ему можно доверять.

```
aamishina@server:~/first$ 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
gpg: next trustdb check due at 2024-05-12
/home/aamishina/.gnupg/pubring.kbx
-----
pub   rsa2048 2024-04-28 [SC] [expires: 2024-05-12]
      E2E27AC8DC84DC38CA9183EC0648F9A2650556F2
uid           [ultimate] Nastya <nastya@gmail.com>
sub   rsa2048 2024-04-28 [E] [expires: 2024-05-12]

aamishina@server:~/first$ gpg --fingerprint
/home/aamishina/.gnupg/pubring.kbx
-----
pub   rsa2048 2024-04-28 [SC] [expires: 2024-05-12]
      E2E2 7AC8 DC84 DC38 CA91 83EC 0648 F9A2 6505 56F2
uid           [ultimate] Nastya <nastya@gmail.com>
sub   rsa2048 2024-04-28 [E] [expires: 2024-05-12]

aamishina@server:~/first$
```

Рис. 3.6: Просмотр ключа и его отпечатка

Затем экспортируем публичный ключ Насти в файл `public.key`, опция `--armor` означает, что выводимые данные должны быть не в бинарном формате, а в ASCII (то есть текстовом, пригодном для копирования-вставки в сообщение мессенджера или электронной почты) (рис. 3.7).

```

aamishina@server:~/first$ gpg --export --armor Nastya > public.key
aamishina@server:~/first$ cat public.key
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBGYupokBCADj0jifPsXGzwQCw+YUDjLOdvrD8MSXe0jj0rc/hlWXZJP52fgo
DCoZsUBS3wY+xgm/xhYBsHr1eZD0ehG19zAkE4vPUIU4vT2EGhhe4piHe/GHA7+Y
xNLw09uKb05LxyAoIZzUmgSgF6e6KSIQIasltf2sF0bNQYx3sm/V695QxH5m0bVS
STkVctblR6Pv0dQex/iwTLQUjyLnXjq2Xdwb9FNqU+1raGuPOB63BEDkuyjQPhrX
5yPC+Im2MDEc/urb3k+sXZ8eu6S9ki4zvHJkGenowDJNHIP1Gnc3Qp7urKQ0/5w6
0sW0uB+uoB9qgEM0DE2mAwIofv2L9FcoRNmHABEBAAG0GU5hc3R5SA8bmFzdHlh
QGdtYWlsLmNvbT6JAVcEEwEKAEEWIQTi4nrI3ITcOMqRg+wGSPmiZQVW8gUCZi6m
iQIbAwUJABJ1AAULCQgHAGIiAgYVCgkICwIEFgIDAQIEBwIXgAAKCRAGSPmiZQVW
8h2SB/40Ixp95dNgk9Ut9zRefkePxZgZ9rc1FRsuitz10smbvd01gsnTnoFM3amG
EmhmeTOASRFz3zgZ8HCpfRzm6VKsxi+48PzW/snhCJXzkgnJD9T3kyTL5IGfnKF
j6JuTpLusS34rQak7n0V+d8/rQMqD8J9MNTqo7x0T7SevZhdgNv8lNiPoxpryHlA
05hNVc/EndXEkcucqTcWfU+zIe9zFDXcmFgmI49gDaELJAJXV/ZhjUOtYBdi5tKWj
VEf24ocBtGAt8uXD5vuqylQQDPMm2Fla0mdPXLHRLG2yvDKTxgcbqGMftFnEsWhv
tM/RcJvCoC8Y3vdEMRGBWSFCpiWkuQENBGYupokBCACy8faezTEjRzY0Lm0JBCEu
0IUIZ22vfPDy4v5VuYL4TdYLPtXvjHzlNd09I848GWgzAjJmNEZe+KYrXVjuS40
ufNxnNiZHu0XEkaW8KwYXCA1l2pwHgu9eZl9f5wF00VC5Q4LAQVkhRbD4pRDptTj
Ux0F2Vkr5ZlpKPtubDvZiZoiWehrl8RaQahuzEMKMcr3j2p6PtWBL/BbKC/HPaBJ
/ahX/Bgl2/nVgubFvS3C2d2kx3RMBBDQLXSkM2GSw+UTtYp/koGxdYzVN0PyYw0x
hk957/j1Z1LvXKxRWc0lypmPZkfX0f9ELl4ru0u1LAz7XhpSdBg1nr2rWK5AWLzT
ABEBAAGJATwEGAeKACYWIQTi4nrI3ITcOMqRg+wGSPmiZQVW8gUCZi6miQIbDAUJ
ABJ1AAAKCRAGSPmiZQVW8oAwCADjXwix7UrbKpHRdWVEk3oDTjLQ+Tu20XXJbqlw
yBDUZtQ/bnZ9bX05pihWmngdg3LeNDzS9Qv7qUpsb2xspgzbv/TYXqvD52CIgpTu
+Whua6+B9K9XSgMg0LH7SnnL7eBbKE7aTVrT+xhkIMIS5INZr2BN503y0F0YQwQ5
gxQKZREjQ5CILFAi0un6FtIMfQZeGssYdNHSw5V9rTvfiHqhP3qUK6Z0qGo0gPVp
a3lIBitV9+CrNnbic2AS2MicwqxAthYPMg0M7r6cNTstn04QQ1/R9gY9o0V65dqh
jGY+nNsNaQyOnEx0o8azXSxYKqWvCZoieyDJNMiMzNGRqdHu
=5Lzw
-----END PGP PUBLIC KEY BLOCK-----
aamishina@server:~/first$

```

Рис. 3.7: Экспорт ключа

Далее имитируем передачу публичного ключа, а именно копируем его в папку, к которой имеет доступ получатель - Ира (рис. 3.8).

```

aamishina@server:~/first$ scp ~/first/public.key ~/second/
aamishina@server:~/first$

```

Рис. 3.8: Передача ключа

Импортируем публичный ключ с помощью команды `gpg --import` (рис. 3.9). Таким образом, ключ добавляется в кольцо ключей, чтобы мы могли шифровать сообщения или проверять подписанные сообщения (кольцо ключей - локальное хранилище ключей GPG на компьютере, которое содержит собственные ключи пользователя, а также ключи других пользователей, которые были импортированы).


```

aamishina@server:~/second$ gpg --import public.key
gpg: key 0648F9A2650556F2: "Nastya <nastya@gmail.com>" not changed
gpg: Total number processed: 1
gpg:                unchanged: 1

```

Рис. 3.9: Импорт ключа

Продолжаем аналогичные действия для пользователя Ира (рис. 3.10).

```

aamishina@server:~/second$ gpg --export --armor Ira > public2.key
aamishina@server:~/second$ scp /home/aamishina/second/public2.key /home/aamishina/first/
aamishina@server:~/second$ cd ~/first
aamishina@server:~/first$ gpg --import public2.key
gpg: key D3442FE98307C2F4: "Ira <ira@gmail.com>" not changed
gpg: Total number processed: 1
gpg:                unchanged: 1
aamishina@server:~/first$ gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 2 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2024-05-12
/home/aamishina/.gnupg/pubring.kbx
-----
pub   rsa2048 2024-04-28 [SC] [expires: 2024-05-12]
      E2E27AC8DC84DC38CA9183EC0648F9A2650556F2
uid           [ultimate] Nastya <nastya@gmail.com>
sub   rsa2048 2024-04-28 [E] [expires: 2024-05-12]

pub   rsa2048 2024-04-28 [SC] [expires: 2024-05-12]
      89D914869AA20B28F8E912DED3442FE98307C2F4
uid           [ultimate] Ira <ira@gmail.com>
sub   rsa2048 2024-04-28 [E] [expires: 2024-05-12]

aamishina@server:~/first$

```

Рис. 3.10: Создание, экспорт, передача, импорт ключа Иры

На данном этапе мы уже можем воссоздать модель шифрования, передачи и дешифровки файла (рис. 3.11), (рис. 3.12).

```

aamishina@server:~/first$ touch secret_file.txt
aamishina@server:~/first$ vi secret_file.txt
aamishina@server:~/first$

```

Рис. 3.11: Создание секретного файла



Рис. 3.12: Файл secret_file.txt

В реальной жизни публичные программы и другие файлы намного чаще распространяются в виде пары: исходный файл программы + отдельно подпись, так как большинству пользователей не нужна электронная подпись, и они не умеют извлекать файлы из .gpg формата. С одной стороны, их позиция ясна, так как они лично не знают тех, кто подписал файл и тех, кто удостоверяет подлинность публичного ключа. С другой стороны, такая подпись является гарантом того, что файл действительно подписан автором программы, а не мошенником.

Мы будем пересылать файл подписи и зашифрованный файл с текстом (рис. 3.13). Для начала создадим файл подписи: в PGP есть опция `-b` (или более длинное написание `--detach-sign`), которая применяется следующим образом: `gpg -u Nastya -b secret_file.txt`. Будет создан дополнительный файл `.sig` (рис. 3.14), в нем нет ничего, кроме цифровой подписи.

Затем шифруем файл с помощью команды `gpg -e -u [Sender_uid] -r [Receiver_uid] [Filename]` (рис. 3.15), где опция `-u` или же `--local-user` указывает на секретный ключ отправителя, а опция `-r` или же `--recipient` позволяет указать публичный ключ получателя, для которого будет выполняться шифрование файла. Это может быть полезно, если вы хотите убедиться, что только указанный получатель сможет расшифровать сообщение.

```

aamishina@server:~/first$ gpg -u Nastya -b secret_file.txt
aamishina@server:~/first$ ls -l
total 20
-rw-rw-r-- 1 aamishina aamishina 0 anp 28 20:12 'big secret'
-rw-rw-r-- 1 aamishina aamishina 1745 anp 28 20:10 public2.key
-rw-rw-r-- 1 aamishina aamishina 1753 anp 28 19:45 public.key
-rw-rw-r-- 1 aamishina aamishina 1753 anp 28 19:46 second
-rw-rw-r-- 1 aamishina aamishina 11 anp 28 20:13 secret_file.txt
-rw-rw-r-- 1 aamishina aamishina 328 anp 28 20:28 secret_file.txt.sig
aamishina@server:~/first$ cat secret_file.txt.sig
+E
/!00z080.Z00H000V0f.0gnastya@gmail.com
H000V0C00%00d000*000s:{00
D_0-0000,{.#j00I:000-000-00?000V>vU%$0F000000H0B0GU0+0Lz0(gg0000000000b0P00000D0000H0DZ0
>Z0y00y;F0000xz:4aamishina@server:~/first$
aamishina@server:~/first$
aamishina@server:~/first$ gpg -e -u Nastya -r Ira secret_file.txt
aamishina@server:~/first$ ls -l
total 24
-rw-rw-r-- 1 aamishina aamishina 0 anp 28 20:12 'big secret'
-rw-rw-r-- 1 aamishina aamishina 1745 anp 28 20:10 public2.key
-rw-rw-r-- 1 aamishina aamishina 1753 anp 28 19:45 public.key
-rw-rw-r-- 1 aamishina aamishina 1753 anp 28 19:46 second
-rw-rw-r-- 1 aamishina aamishina 11 anp 28 20:13 secret_file.txt
-rw-rw-r-- 1 aamishina aamishina 364 anp 28 20:30 secret_file.txt.gpg
-rw-rw-r-- 1 aamishina aamishina 328 anp 28 20:28 secret_file.txt.sig
aamishina@server:~/first$

```

Рис. 3.13: Создание подписи файла, шифрование файла



Рис. 3.14: Файл цифровой подписи

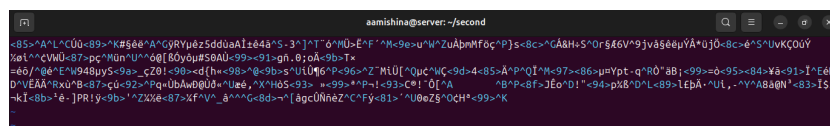


Рис. 3.15: Зашифрованный файл с текстом

“Пересылаем файлы”, копируя их в каталог second (рис. 3.16). Для начала дешифруем файл с текстом командой `gpg -d -o decrypted.txt secret_file.txt.gpg` (с помощью опции `-o` расшифрованный текст не выведется в терминал, а запишется в файл `decrypted.txt`), теперь можно посмотреть отправленный текст (рис. 3.17). Также проверяем цифровую подпись. Так как она отделена от файла, то после опции `-verify` нужно указать два аргумента: вначале идёт файл с подписью, а затем подписанные данные. Видим, что файл был отправлен Настей.

```

aamishina@server:~/first$ scp secret_file.txt.sig /home/aamishina/second/
aamishina@server:~/first$ scp secret_file.txt.gpg /home/aamishina/second/
aamishina@server:~/first$ cd ~/second
aamishina@server:~/second$ ls
public2.key public.key secret_file.txt.gpg secret_file.txt.sig
aamishina@server:~/second$ gpg -d -o decrypted.txt secret_file.txt.gpg
gpg: encrypted with rsa2048 key, ID DAFB890B23A7EAEB, created 2024-04-28
      "Ira <ira@gmail.com>"
aamishina@server:~/second$ gpg --verify secret_file.txt.sig decrypted.txt
gpg: [don't know]: invalid packet (ctb=0a)
gpg: Signature made Bc 28 anp 2024 20:28:23 UTC
gpg:          using RSA key E2E27AC8DC84DC38CA9183EC0648F9A2650556F2
gpg:          issuer "nastya@gmail.com"
gpg: Good signature from "Nastya <nastya@gmail.com>" [ultimate]

```

Рис. 3.16: Дешифровка и проверка подписи

```

aamishina@server:~/second$ cat decrypted.txt
big secret
aamishina@server:~/second$ █

```

Рис. 3.17: Просмотр дешифрованного файла

На этом заканчивается моделирование ситуации, файл был передан и дешифрован успешно.

4 Выводы

Система PGP представляет собой мощный инструмент криптографической защиты данных. Он широко используется для защиты электронной почты, файлов и других цифровых данных, делая PGP одним из наиболее эффективных способов для обеспечения безопасности в сетевых коммуникациях. Открытость и свободный доступ этой системы к криптографическим средствам способствуют распространению принципов безопасности в цифровом мире.

Список литературы

1. Левин М. PGP: Кодирование и шифрование информации с открытым ключом. Москва: Бук-Пресс, 2006. 166 с.
2. Evanty L. Tutorial Encrypt and Decrypt with PGP on Ubuntu 20.04. <https://blog.neuronvm.com/encrypt-and-decrypt-with-pgp-on-ubuntu-20-04/#introduction-to-pgp>, 2023.