Доклад на тему: Система PGP

Дисциплина: Основы информационной безопасности

Мишина Анастасия Алексеевна НПИбд-02-22

Содержание

Список иллюстраций

Список таблиц

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).

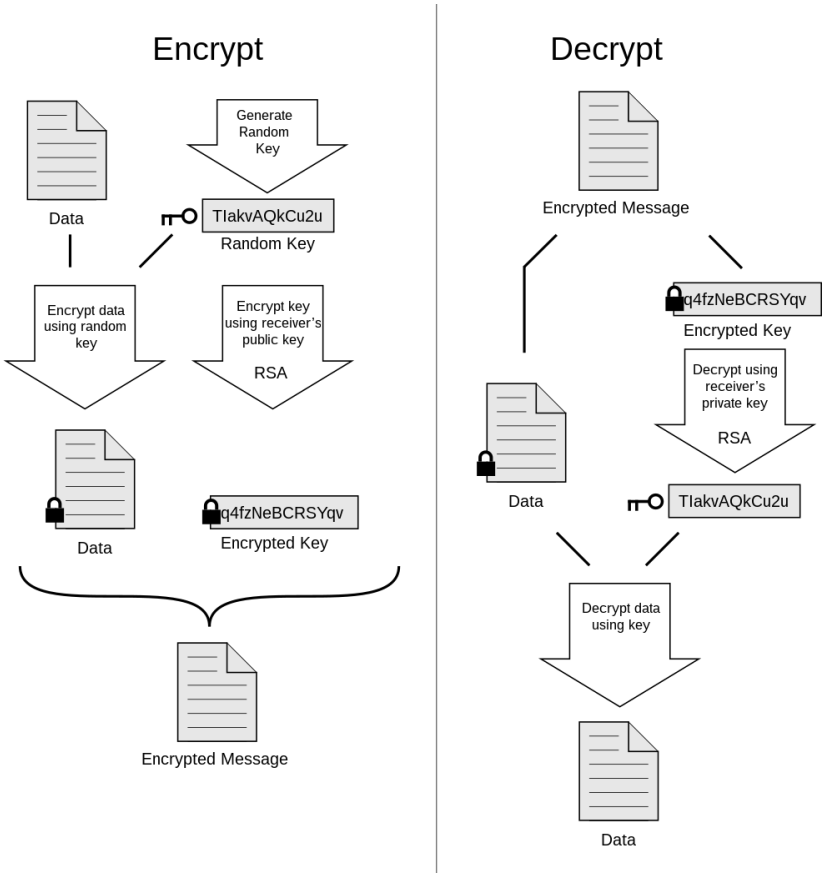

Рис. 3.1: Общая схема шифрования и дешифрования

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).

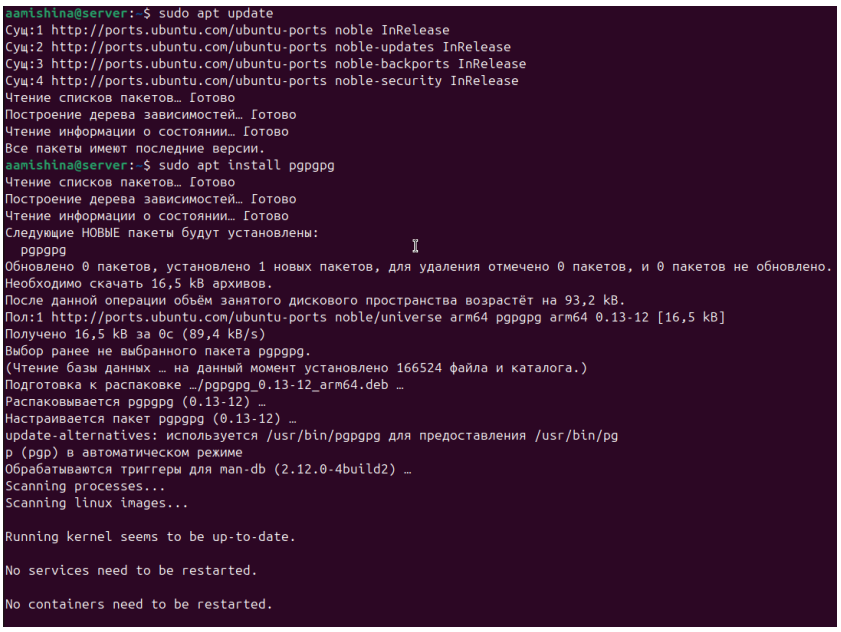

Рис. 3.2: Установка GPG

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 pgp

Опция	Значение
`--export`	Экспортировать публичный ключ в файл, который потом можно отправить.
`--import`	Импортировать публичный ключ.
`--armor -a`	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.
`--encrypt -e`	Зашифровать сообщение.
`--recipient -r`	Указать ключ, который будет использоваться для шифрования. Можно использовать информацию идентификатор пользователя (имя, почта), идентификатор ключа, отпечаток ключа.
`--decrypt -d`	Расшифровать сообщение.
`--sign -s`	Подписать сообщение. Подпись при этом будет располагаться отдельно от самого сообщения.
`--clear-sign` `--clearsign`	Подписать сообщение. Подпись при этом сохраняется вместе с сообщением.
`--local-user -u`	Указать ключ, который будет использоваться для подписи. Схож с опцией –recipient, но это не одно и то же.
`--verify`	Проверить подпись.
`--list-keys -k`	Вывести список публичных ключей.

Опция	Значение
`--list-secret-keys`	Вывести список приватных ключей.
`-K`	
`--edit-key`	Редактировать ключ.
`--expert`	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 находится множество ключей. Также на этом этапе создаётся сертификат отзыва — файл, с помощью которого созданный ключ можно отозвать (признать недействительным).

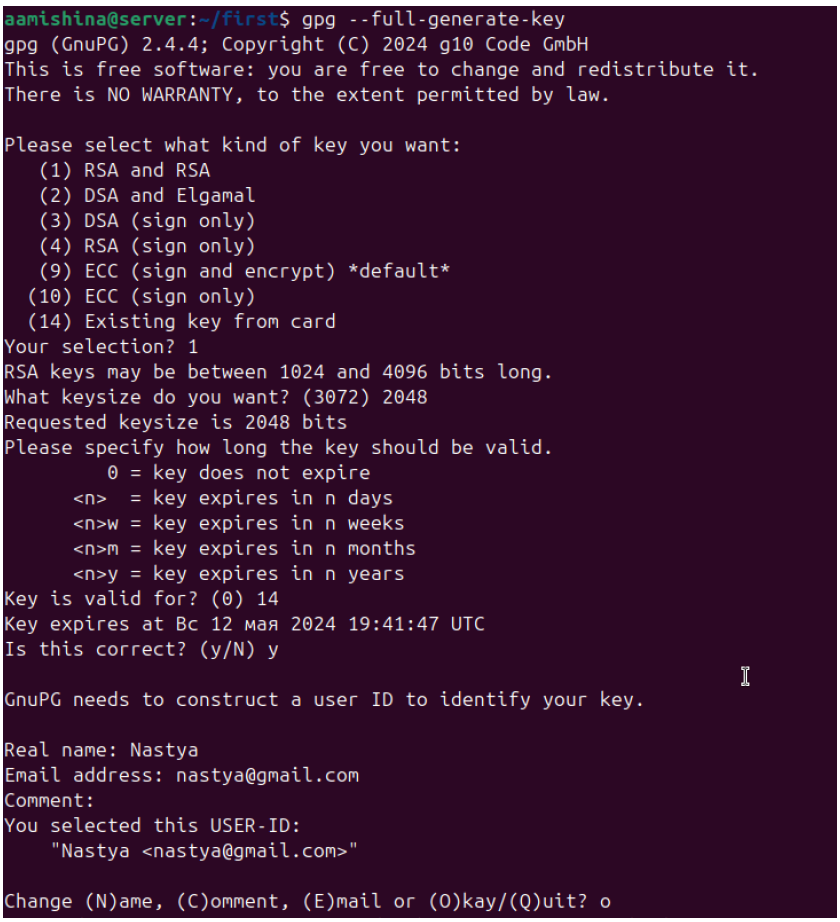

Рис. 3.4: Создание пары ключей

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

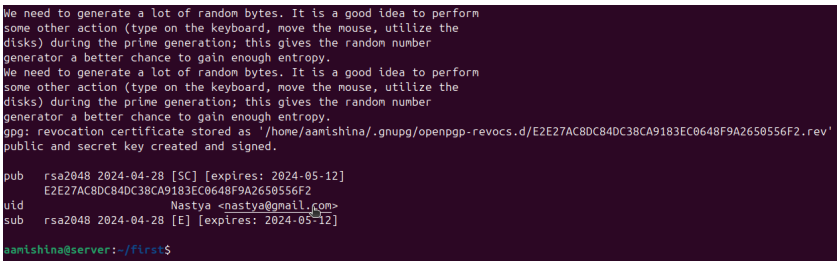

Рис. 3.5: Состав ключа

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 обоих 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 получил 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.

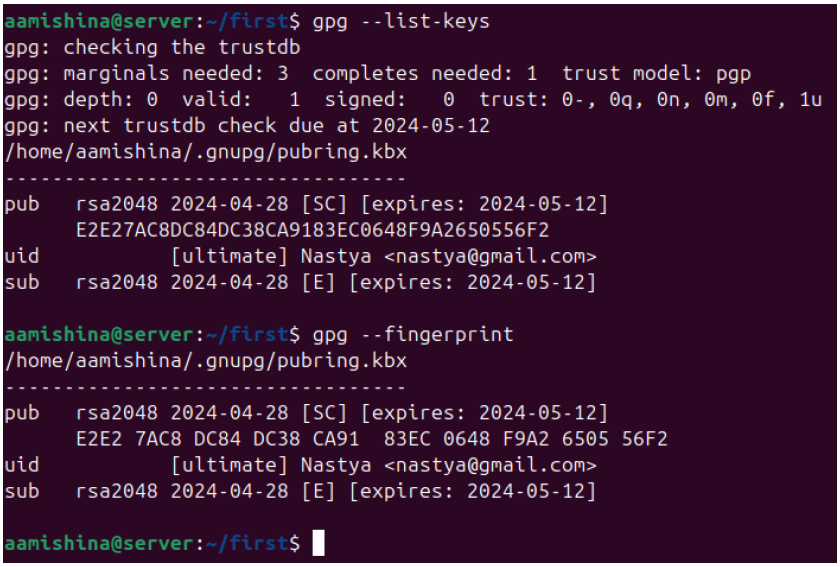

Рис. 3.6: Просмотр ключа и его отпечатка

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).

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).

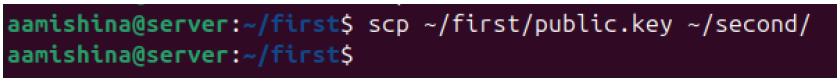

Рис. 3.8: Передача ключа

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

Проделываем 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

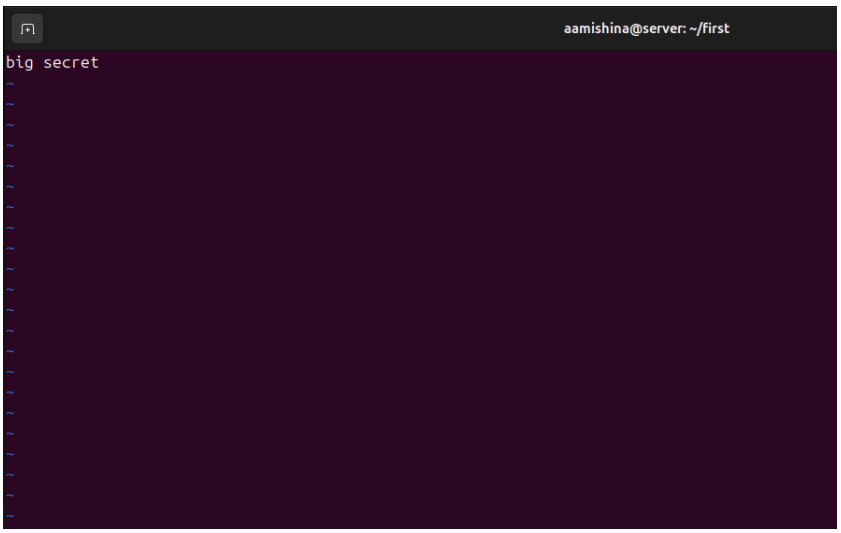

Рис. 3.12: Файл secret_file.txt

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.

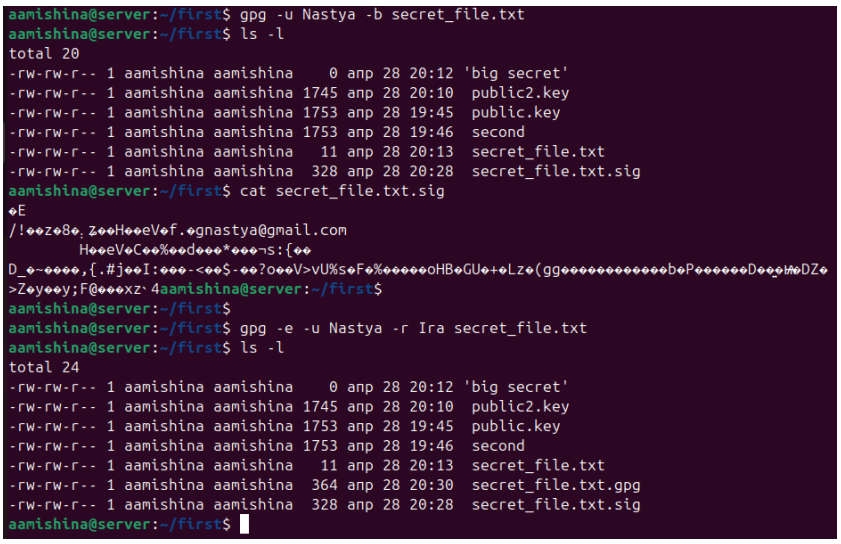

Рис. 3.13: Создание подписи файла, шифрование файла

Рис. 3.14: Файл цифровой подписи

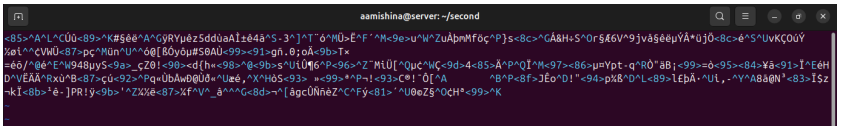

Рис. 3.15: Зашифрованный файл с текстом

“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.

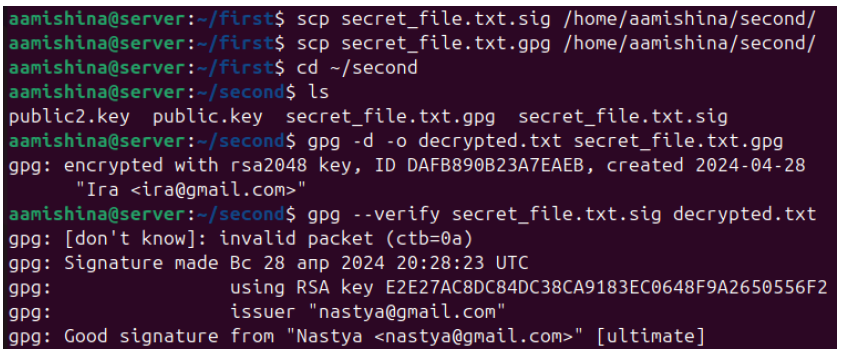

Рис. 3.16: Дешифровка и проверка подписи

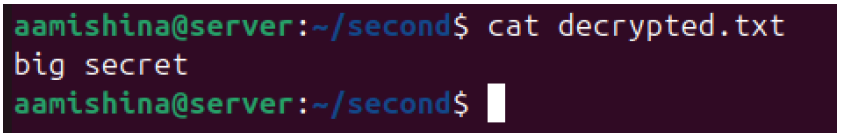

Рис. 3.17: Просмотр дешифрованного файла

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.