GnuPG Encrypt and sign

START HERE

`gpg --full-generate-key`
Create a key pair

`gpg --detach-sign <file>`
Sign a file

`gpg -k`
List public keys

`gpg --verify <file>.sig <file>`
Check a signature

`gpg -e -r <id> <file>`
Encrypt for someone

`gpg --armor --export <id>`
Share your public key

`gpg -d <file>.gpg`
Decrypt

`gpg --gen-revoke <id>`
Make a revocation cert

Plaintext your file → `encrypt: recipient public key` → **Ciphertext** .gpg / .asc → `decrypt: recipient secret key` → **Plaintext** restored

CONCEPTS

Others encrypt to your public key and verify your signatures; only your secret key decrypts and signs. Guard the secret key.

A key ID can be an email, a name part, a long key ID or a full fingerprint. Prefer the fingerprint when it must be unambiguous.

A key is a primary key (certify) plus subkeys (sign, encrypt, auth). Subkeys can expire or be replaced without losing your identity.

KEYS

`gpg --full-generate-key`
Create a key pair interactively

`gpg --quick-generate-key <email> default default 2y`
Create a key with defaults, 2y expiry

`gpg -k [<id>]`
List public keys

`gpg -K [<id>]`
List secret keys

`gpg --fingerprint <id>`
Show the fingerprint

`gpg --list-keys --keyid-format long`
Show long key IDs

`gpg --list-sigs <id>`
List signatures on a key

`gpg --edit-key <id>`
Edit a key in the key shell
Inside: adduid, addkey, expire, trust, sign, passwd, save

`gpg --delete-keys <id>`
Remove a public key
Delete the secret key first with --delete-secret-keys

`gpg --delete-secret-keys <id>`
Remove a secret key

`gpg --quick-set-expire <fpr> 1y`
Change the expiry date

`gpg --quick-add-uid <fpr> "<name> <email>"`
Add another identity

IMPORT AND EXPORT

`gpg --armor --export <id>`
Print public key as text

`gpg --export -o <file>.gpg <id>`
Export public key as binary

IMPORT AND EXPORT (cont.)

`gpg --import <file>`
Import a key file

`gpg --armor --export-secret-keys <id> > <file>.asc`
Export the secret key
Handle like a password; the file is protected only by its passphrase

`gpg --export-secret-subkeys <id>`
Export subkeys only

`gpg --show-keys <file>`
Inspect a key file without importing

`gpg --import-options show-only --import <file>`
Preview an import

ENCRYPT AND DECRYPT

`gpg -e -r <id> <file>`
Encrypt to a recipient

`gpg -ea -r <id> <file>`
Encrypt as ASCII text (.asc)

`gpg -c <file>`
Encrypt with a passphrase only

`gpg -d <file>.gpg`
Decrypt to stdout

`gpg -o <out> -d <file>.gpg`
Decrypt into a file

`gpg -e -r <id1> -r <id2> <file>`
Encrypt to several recipients

`gpg -e --hidden-recipient <id> <file>`
Hide the recipient key ID

`gpg --symmetric --cipher-algo AES256 <file>`
Choose the symmetric cipher

`gpg -e -r <id> --output - < <file>`
Encrypt stdin to stdout

`gpg --decrypt --list-only <file>.gpg`
Show who a file is for

SIGN AND VERIFY

`gpg --detach-sign <file>`
Write a separate .sig

`gpg -ab <file>`
Detached ASCII signature (.asc)

`gpg --verify <file>.sig <file>`
Verify a detached signature

`gpg --clearsign <file>`
Sign text, keep it readable

SIGN AND VERIFY (cont.)

`gpg -s <file>`
Sign and wrap the file

`gpg --verify <file>.asc`
Verify a signed or clearsigned file

`gpg -se -r <id> <file>`
Sign and encrypt in one step

`gpg -u <id> --detach-sign <file>`
Sign with a specific key

`gpg --sign-key <id>`
Certify someone's key
Do this only after checking the fingerprint

KEYSERVERS AND TRUST

`gpg --keyserver <server> --send-keys <fpr>`
Upload your public key

`gpg --keyserver <server> --recv-keys <fpr>`
Fetch a key by fingerprint

`gpg --search-keys <email>`
Search keyservers

`gpg --locate-keys <email>`
Find a key via WKD/DANE

`gpg --refresh-keys`
Update keys from keyservers

`gpg --edit-key <id> trust`
Set owner trust

`gpg --check-sigs <id>`
Verify signatures on a key

keys.openpgp.org only publishes identities after email confirmation; other servers may differ.

BACKUP AND REVOCATION

`gpg --gen-revoke <id>`
Create a revocation certificate
Store it offline now, before you ever need it

`gpg -o <file>.rev --gen-revoke <id>`
Write the certificate to a file

`gpg --import <file>.rev`
Revoke a key with the certificate

`gpg --export-ownertrust > <file>`
Back up trust settings

`gpg --import-ownertrust <file>`
Restore trust settings

`gpg --export-secret-keys --armor > <file>.asc`
Back up all secret keys

BACKUP AND REVOCATION (cont.)

`gpg --edit-key <id> revkey`
Revoke a subkey (key shell: revkey)

Keep an offline copy of the secret key and the revocation certificate; losing both means you cannot decrypt or retire the key.

AGENT AND CONFIG

`gpgconf --kill gpg-agent`
Stop the agent (restart on next use)
Fixes stale passphrase cache or a wedged agent

`gpgconf --launch gpg-agent`
Start the agent

`gpg-connect-agent reloadagent /bye`
Reload agent config

`echo RELOADAGENT | gpg-connect-agent`
Reload agent config (pipe form)

`export GPG_TTY=$(tty)`
Let pinentry find your terminal
Add to your shell rc; fixes 'inappropriate ioctl' errors

`gpgconf --list-dirs`
Show config and socket dirs

`gpg --version`
Show version and algorithms

~/.gnupg/gpg-agent.conf

```
default-cache-ttl 600
max-cache-ttl 7200
pinentry-program /usr/bin/pinentry-curses
```

RECIPES

Encrypt for a colleague

```
gpg --locate-keys <email>
gpg --fingerprint <email>
gpg -ea -r <email> -o <file>.asc <file>
```

Sign a release and verify it

```
gpg -ab <file>.tar.gz
gpg --verify <file>.tar.gz.asc <file>.tar.gz
```

Move a key to another machine

```
gpg --export-secret-keys --armor <id> > key.asc
gpg --export-ownertrust > trust.txt
# on the new machine
gpg --import key.asc && gpg --import-ownertrust trust.txt
```

Encrypted backup of a folder

```
tar -cz <dir> | gpg -c -o <dir>.tar.gz.gpg
gpg -d <dir>.tar.gz.gpg | tar -xz
```