

Instantly share code, notes, and snippets.



[MorganGeek](#) / [gpg cheat sheet.md](#)

Last active 5 months ago

★ Star

7

🍴 Fork

2

↔ Code

🔗 Revisions 23

★ Stars 7

🍴 Forks 2

Embed ▼

<script src="https://gi:



Download ZIP

GPG Cheat Sheet

🔗 [gpg cheat sheet.md](#)

Raw

Basics

generate key in batch mode using a custom profile

```
gpg --gen-key --batch gpgspecs
```

create a file with your fingerprint info and display the related information. A fingerprint is used as a robust key identifier

```
gpg --fingerprint
```

Best practices

check you have at least a OpenPGPv4 key (v3 is not considered as robust)

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep version
```

check you have at least a DSA-2 or (preferably) RSA key with a length of 4K (or more).

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep -A2 '^:public key packet:$' | grep algo
```

check the output : RSA corresponds to algo 1, DSA to algo 17, ECDSA corresponds to algo 19, ECC to algo 18

in case you have RSA or DSA, check you have at least a key with a length of at least 4K (RSA) or at least 1K (DSA-2).
ECDSA and ECC have different kind of keys

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep -A2 'public key'|  
grep 'pkey\[0\]:'
```

auto signatures should not use MD5. check that with the following command.

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep -A 2 signature|  
grep 'digest algo'
```

auto signatures should not use SHA-1. check that with the following command.

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep -A 2 signature|  
grep 'digest algo 2, '
```

If any of previous commands results contains 'digest algo 1' or 'digest algo 2', you should regenerate your key after adding
cert-digest-algo SHA512 in ~/.gnupg/gpg.conf :

```
echo "cert-digest-algo SHA512" >> ~/.gnupg/gpg.conf
```

you can regenerate an existing key, by simply updating the expiry date

```
gpg --edit-key '<fingerprint>'
gpg> expire
gpg> 2y
...
gpg> save
```

check if your preferences for hashing algorithm include a member of SHA-2 family before SHA-1 and MD5.

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep 'pref-hash-algos'
```

if you see one of numbers '3', '2' ou '1' preceeding '11', '10', '9' or '8', you have weak preferences. Fix them

```
echo "default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP
Uncompressed" >> ~/.gnupg/gpg.conf
```

then fix your key :

```
gpg --edit-key '<fingerprint>'
gpg> setpref
...
gpg> save
```

check you key expiry date, it should be max 2 years in the future from now

```
gpg --export-options export-minimal --export '<fingerprint>' | gpg --list-packets | grep 'key expires
after'
```

or

```
gpg --list-keys '<fingerprint>'
```

you can fix that it it's not the case

```
gpg --edit-key '<fingerprint>'
gpg> expire
gpg> 2y
...
gpg> save
```

List your keys

list your public keys, printing also fingerprint because default ids are too short and not secure

```
gpg --list-keys --with-fingerprint
```

list your private keys, with fingerprint since default ids are too short and not secure

```
gpg --list-secret-keys --with-fingerprint
```

avoid passing --with-fingerprint each time, by changing your default settings :

```
echo "with-fingerprint" >> ~/.gnupg/gpg.conf
```

Export your keys

export a public key

```
gpg -ao user@system-pgp-pub.key --export '<fingerprint>'
```

export a private key

```
gpg -ao user@system-gpg-prv.key --export-secret-keys '<fingerprint>'
```

Revocation and deletion

generate a revocation key (in case you forget your pass or your key is compromised)

```
gpg --output revoke.asc --gen-revoke '<fingerprint>'
```

delete private keys

```
gpg --delete-secret-keys '<fingerprint>'
```

delete public keys

```
gpg --delete-keys '<fingerprint>'
```

Import

check the fingerprint of a key before you import it

```
gpg --with-fingerprint <keyfile>
```

Import it (either it be private or public) `gpg --import <path to key>`

Encrypt files

create an archive of your secret files

```
tar czf mysecrets.tar.gz folder_with_secrets
tar -ztvf mysecrets.tar.gz
gpg --encrypt --recipient <uid> mysecrets.tar.gz // or gpg --encrypt --recipient <fingerprint> mysecrets.t
```

decrypt file the archive

```
gpg --output restoredsecrets.tar.gz --decrypt mysecrets.tar.gz.gpg
tar -ztvf restoredsecrets.tar.gz
```

 **gpgspecs**

Raw

```
1 Key-Type: RSA
2 Key-Length: 4096
3 Subkey-Type: RSA
4 Subkey-Length: 4096
5 Name-Real: <Firstname Lastname>
6 Name-Comment: <user@system>
7 Name-Email: <user@emailprovider.com>
8 #Passphrase: <specify the passphrase or be prompted for entering it later>
9 Expire-Date: 2y
10 # note : it's better to set an expiry date less or equal to 2 years (2y), otherwise people may be reticent to trust your key
```



MorganGeek commented on Jun 9, 2017 • edited ▼

Author

Owner

...

Infos :

<https://www.gnupg.org/documentation/manuals/gnupg/Unattended-GPG-key-generation.html>

<https://superuser.com/questions/1003403/how-to-use-gpg-gen-key-in-a-script>
<https://2buntu.com/articles/1503/pgp-and-ssh-keys-generate-export-backup-and-restore/>
<https://riseup.net/fr/security/message-security/openpgp/best-practices>
<https://github.com/ioerror/duraconf/blob/master/configs/gnupg/gpg.conf>

Sign up for free to join this conversation on **GitHub**. Already have an account? [Sign in to comment](#)