Previous: The quick key manipulation interface, Up: Unattended Usage of GPG [Contents][Index]

4.5.4 Unattended key generation

The command `--generate-key` may be used along with the option `--batch` for unattended key generation. This is the most flexible way of generating keys, but it is also the most complex one. Consider using the quick key manipulation interface described in the previous subsection “The quick key manipulation interface”.

The parameters for the key are either read from stdin or given as a file on the command line. The format of the parameter file is as follows: Text only, line length is limited to about 1000 characters. UTF-8 encoding must be used to specify non-ASCII characters. Empty lines are ignored. Leading and trailing white space is ignored. A hash sign as the first non white space character indicates a comment line. Control statements are indicated by a leading percent sign, their arguments are separated by white space from the keyword. Parameters are specified by a keyword, followed by a colon; arguments are separated by white space. The first parameter must be ‘`Key-Type`’ but control statements may be placed anywhere. The order of the parameters does not matter except for ‘`Key-Type`’. The parameters are only used for the generated keyblock (primary and subkeys); parameters from previous sets are not used. Some syntax checks may be performed. Key commences when either the end of the parameter file is reached, the next ‘`Key-Type`’ parameter is encountered, or the control statement ‘`%commit`’ is encountered.

Control statements:

%echo *text*

> Print *text* as diagnostic.

%dry-run

> Suppress actual key generation (useful for syntax checking).

%commit

> Perform the key generation. Note that an implicit commit is done at the next Key-Type parameter.

%pubring *filename*

> Do not write the key to the default or commandline given keyring but to *filename*. This must be given before the first commit to take place, duplicate specification of the same filename is ignored, the last filename before a commit is used. The filename is used until a new filename is used (at commit points) and all keys are written to that file. If a new filename is given, this file is created (and overwrites an existing one).
>
> See the previous subsection “Ephemeral home directories” for a more robust way to contain side-effects.

%secring *filename*

> This option is a no-op for GnuPG 2.1 and later.
>
> See the previous subsection “Ephemeral home directories”.

%ask-passphrase
%no-ask-passphrase

> This option is a no-op since GnuPG version 2.1.

%no-protection

> Using this option allows the creation of keys without any passphrase protection. This option is mainly intended for regression tests.

%transient-key

> If given the keys are created using a faster and a somewhat less secure random number generator. This option may be used for keys which are only used for a short time and do not require full cryptographic strength. It takes only effect if used together with the control statement ‘`%no-protection`’.

General Parameters:

Key-Type: *algo*

> Starts a new parameter block by giving the type of the primary key. The algorithm must be capable of signing. This is a required parameter. *algo* may either be an OpenPGP algorithm number or a string with the algorithm name. The special value ‘`default`’ may be used for *algo* to create the default key type; in this case a ‘`Key-Usage`’ shall not be given and ‘`default`’ also be used for ‘`Subkey-Type`’.

Key-Length: *nbits*

> The requested length of the generated key in bits. The default is returned by running the command ‘`gpg --gpgconf-list`’. For ECC keys this parameter is ignored.

Key-Curve: *curve*

> The requested elliptic curve of the generated key. This is a required parameter for ECC keys. It is ignored for non-ECC keys.

Key-Grip: *hexstring*

This is optional and used to generate a CSR or certificate for an already existing key. Key-Length will be ignored when given.

Key-Usage: *usage-list*

Space or comma delimited list of key usages. Allowed values are '`encrypt`', '`sign`', and '`auth`'. This is used to generate the key flags. Please make sure that the algorithm is capable of this usage. Note that OpenPGP requires that all primary keys are capable of certification, so no matter what usage is given here, the '`cert`' flag will be on. If no '`Key-Usage`' is specified and the '`Key-Type`' is not '`default`', all allowed usages for that particular algorithm are used; if it is not given but '`default`' is used the usage will be '`sign`'.

Subkey-Type: *algo*

This generates a secondary key (subkey). Currently only one subkey can be handled. See also '`Key-Type`' above.

Subkey-Length: *nbits*

Length of the secondary key (subkey) in bits. The default is returned by running the command '`gpg --gpgconf-list`'.

Subkey-Curve: *curve*

Key curve for a subkey; similar to '`Key-Curve`'.

Subkey-Usage: *usage-list*

Key usage lists for a subkey; similar to '`Key-Usage`'.

Passphrase: *string*

If you want to specify a passphrase for the secret key, enter it here. Default is to use the Pinentry dialog to ask for a passphrase.

Name-Real: *name*
Name-Comment: *comment*
Name-Email: *email*

The three parts of a user name. Remember to use UTF-8 encoding here. If you don't give any of them, no user ID is created.

Expire-Date: *iso-date*|(*number*[d|w|m|y])

Set the expiration date for the key (and the subkey). It may either be entered in ISO date format (e.g. "20000815T145012") or as number of days, weeks, month or years after the creation date. The special notation "seconds=N" is also allowed to specify a number of seconds since creation. Without a letter days are assumed. Note that there is no check done on the overflow of the type used by OpenPGP for timestamps. Thus you better make sure that the given value make sense. Although OpenPGP works with time intervals, GnuPG uses an absolute value internally and thus the last year we can represent is 2105.

Creation-Date: *iso-date*

Set the creation date of the key as stored in the key information and which is also part of the fingerprint calculation. Either a date like "1986-04-26" or a full timestamp like "19860426T042640" may be used. The time is considered to be UTC. The special notation "seconds=N" may be used to directly specify a the number of seconds since Epoch (Unix time). If it is not given the current time is used.

Preferences: *string*

Set the cipher, hash, and compression preference values for this key. This expects the same type of string as the sub-command '`setpref`' in the `--edit-key` menu.

Revoker: *algo*:*fpr* [sensitive]

Add a designated revoker to the generated key. Algo is the public key algorithm of the designated revoker (i.e. RSA=1, DSA=17, etc.) *fpr* is the fingerprint of the designated revoker. The optional '`sensitive`' flag marks the designated revoker as sensitive information. Only v4 keys may be designated revokers.

Keyserver: *string*

This is an optional parameter that specifies the preferred keyserver URL for the key.

Handle: *string*

This is an optional parameter only used with the status lines KEY_CREATED and KEY_NOT_CREATED. *string* may be up to 100 characters and should not contain spaces. It is useful for batch key generation to associate a key parameter block with a status line.

Here is an example on how to create a key in an ephemeral home directory:

```
$ export GNUPGHOME="$(mktemp -d)"
$ cat >foo <<EOF
     %echo Generating a basic OpenPGP key
     Key-Type: DSA
     Key-Length: 1024
     Subkey-Type: ELG-E
     Subkey-Length: 1024
     Name-Real: Joe Tester
```

```
     Name-Comment: with stupid passphrase
     Name-Email: joe@foo.bar
     Expire-Date: 0
     Passphrase: abc
     # Do a commit here, so that we can later print "done" :-)
     %commit
     %echo done
EOF
$ gpg --batch --generate-key foo
 [...]
$ gpg --list-secret-keys
/tmp/tmp.0NQxB74PEf/pubring.kbx
------------------------------
sec   dsa1024 2016-12-16 [SCA]
      768E895903FC1C44045C8CB95EEBDB71E9E849D0
uid           [ultimate] Joe Tester (with stupid passphrase) <joe@foo.bar>
ssb   elg1024 2016-12-16 [E]
```

If you want to create a key with the default algorithms you would use these parameters:

```
     %echo Generating a default key
     Key-Type: default
     Subkey-Type: default
     Name-Real: Joe Tester
     Name-Comment: with stupid passphrase
     Name-Email: joe@foo.bar
     Expire-Date: 0
     Passphrase: abc
     # Do a commit here, so that we can later print "done" :-)
     %commit
     %echo done
```

Previous: The quick key manipulation interface, Up: Unattended Usage of GPG [Contents][Index]