Using the GNU Privacy Guard

Version 1.9.14
22 December 2004

Werner Koch (wk@gnupg.org)

Short Contents

Table of Contents

Introduction

This manual documents how to use the GNU Privacy Guard system as well as the administration and the architecture.

1 Invoking GPG

`gpg` is the OpenPGP part of GnuPG. The version included in this package is not as matured as the standard versions (1.2.x or 1.4.x) and thus we strongly suggest to keep on using the one of the standard versions. Both versions may be installed side by side and should coexists without problems. To help for that, the `gpg` from this package gets installed under the name `gpg2`. If you really want to use this `gpg2` command you should name the configuration file '`gpg.conf-1.9`' to keep it separate from the one used with the standard `gpg`.

Documentation for the old standard `gpg` is available in a man page and at See Info file '`gpg`', node '`Top`'.

2 Invoking GPGSM

`gpgsm` is a tool similar to `gpg` to provide digital encryption and signing servicesd on X.509 certificates and the CMS protocoll. It is mainly used as a backend for S/MIME mail processing. `gpgsm` includes a full features certificate management and complies with all rules defined for the German Sphinx project.

See [Option Index], page 61, for an index to `GPGSM`'s commands and options.

2.1 Commands

Commands are not distinguished from options execpt for the fact that only one one command is allowed.

2.1.1 Commands not specific to the function

`--version`
: Print the program version and licensing information. Not that you can abbreviate this command.

`--help, -h`
: Print a usage message summarizing the most usefule command-line options. Not that you can abbreviate this command.

`--dump-options`
: Print a list of all available options and commands. Not that you can abbreviate this command.

2.1.2 Commands to select the type of operation

`--encrypt`
: Perform an encryption.

`--decrypt`
: Perform a decryption; the type of input is automatically detmerined. It may either be in binary form or PEM encoded; automatic determination of base-64 encoding is not done.

`--sign` Create a digital signature. The key used is either the fist one found in the keybox or thise set with the -u option

`--verify` Check a signature file for validity. Depending on the arguments a detached signatrue may also be checked.

`--server` Run in server mode and wait for commands on the `stdin`.

`--call-dirmngr` *command* [*args*]
: Behave as a Dirmngr client issuing the request *command* with the optional list of *args*. The output of the Dirmngr is printed stdout. Please note that file names given as arguments should have an absulte file name (i.e. commencing with `/`

because they are passed verbatim to the Dirmngr and the working directory of the Dirmngr might not be the same as the one of this client. Currently it is not possible to pass data via stdin to the Dirmngr. *command* should not contain spaces.

This is command is required for certain maintaining tasks of the dirmngr where a dirmngr must be able to call back to `gpgsm`. See the Dirmngr manual for details.

`--call-protect-tool` *arguments*
: Certain maintenance operations are done by an external program call `gpg-protect-tool`; this is usually not installed in a directory listed in the PATH variable. This command provides a simple wrapper to access this tool. *arguments* are passed verbatim to this command; use '`--help`' to get a list of supported operations.

2.1.3 How to manage the certificate and keys

`--gen-key`
: Generate a new key and a certificate request.

`--list-keys`
`-k`
: List all available certificates stored in the local key database. Note that the displayed data might be reformatted for better human readability and illegal characters are replaced by safe substitutes.

`--list-secret-keys`
`-K`
: List all available certificates for which a corresponding a secret key is available.

`--list-external-keys` *pattern*
: List certificates matching *pattern* using an external server. This utilizes the `dirmngr` service.

`--dump-keys`
: List all available certificates stored in the local key database using a format useful mainly for debugging.

`--dump-secret-keys`
: List all available certificates for which a corresponding a secret key is available using a format useful mainly for debugging.

`--dump-external-keys` *pattern*
: List certificates matching *pattern* using an external server. This utilizes the `dirmngr` service. It uses a format useful mainly for debugging.

`--keydb-clear-some-cert-flags`
: This is a debugging aid to reset certain flags in the key database which are used to cache certain certificate stati. It is especially useful if a bad CRL or a weird running OCSP reponder did accidently revoke certificate. There is no security issue with this command because `gpgsm` always make sure that the validity of a certificate is checked right before it is used.

`--delete-keys` *pattern*
: Delete the keys matching *pattern*.

`--export` [*pattern*]
: Export all certificates stored in the Keybox or those specified by the optional *pattern*. When using along with the `--armor` option a few informational lines are prepended before each block.

`--export-secret-key-p12` *key-id*
: Export the private key and the certificate identified by *key-id* in a PKCS#12 format. When using along with the `--armor` option a few informational lines are prepended to the output. Note, that the PKCS#12 format is higly insecure and this command is only provided if there is no other way to exchange the private key.

`--learn-card`
: Read information about the private keys from the smartcard and import the certificates from there. This command utilizes the GPG-AGENT and in turn the SCDAEMON.

`--passwd` *user_id*
: Change the passphrase of the private key belonging to the certificate specified as *user_id*. Note, that changing the passphrase/PIN of a smartcard is not yet supported.

2.2 Option Summary

`GPGSM` comes features a bunch ofoptions to control the exact behaviour and to change the default configuration.

2.2.1 How to change the configuration

These options are used to change the configuraton and are usually found in the option file.

`--options` *file*
: Reads configuration from *file* instead of from the default per-user configuration file. The default configuration file is named '`gpgsm.conf`' and expected in the '`.gnupg`' directory directly below the home directory of the user.

`--homedir` *dir*
: Set the name of the home directory to *dir*. If his option is not used, the home directory defaults to '`~/.gnupg`'. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable `GNUPGHOME` or (on W32 systems) by means on the Registry entry *HKCU\Software\GNU\GnuPG:HomeDir*.

`-v`

`--verbose`
: Outputs additional information while running. You can increase the verbosity by giving several verbose commands to `gpgsm`, such as '`-vv`'.

`--policy-file` *filename*
: Change the default name of the policy file to *filename*.

`--agent-program` *file*
: Specify an agent program to be used for secret key operations. The default value is the '`/usr/local/bin/gpg-agent`'. This is only used as a fallback when the envrionment variable `GPG_AGENT_INFO` is not set or a running agent can't be connected.

`--dirmngr-program` *file*
: Specify a dirmngr program to be used for CRL checks. The default value is '`/usr/sbin/dirmngr`'. This is only used as a fallback when the environment variable `DIRMNGR_INFO` is not set or a running dirmngr can't be connected.

`--prefer-system-dirmngr`
: If a system wide `dirmngr` is running in daemon mode, first try to connect to this one. Fallback to a pipe based server if this does not work.

`--no-secmem-warning`
: Don't print a warning when the so called `"secure memory"` can't be used.

`--log-file` *file*
: When running in server mode, append all logging output to *file*.

2.2.2 Certificate related options

`--enable-policy-checks`
`--disable-policy-checks`
: By default policy checks are enabled. These options may be used to change it.

`--enable-crl-checks`
`--disable-crl-checks`
: By default the CRL checks are enabled and the DirMngr is used to check for revoked certificates. The disable option is most useful with an off-line network connection to suppress this check.

`--force-crl-refresh`
: Tell the dirmngr to reload the CRL for each request. For better performance, the dirmngr will actually optimize this by suppressing the loading for short time intervalls (e.g. 30 minutes). This option is useful to make sure that a fresh CRL is available for certificates hold in the keybox. The suggested way of doing this is by using it along with the option '`--with-validation`' for a ke listing command. This option should not be used in a configuration file.

`--enable-ocsp`
`--disable-ocsp`
: Be default OCSP checks are disabled. The enable opton may be used to enable OCSP checks via Dirmngr. If CRL checks are also enabled, CRLs will be used as a fallback if for some reason an OCSP request won't succeed. Note, that you have to allow OCSP requests in Dirmngr's configuration too (option '`--allow-ocsp`' and configure dirmngr properly. If you don't do so you will get the error code '`Not supported`'.

2.2.3 Input and Output

`--armor`
`-a` Create PEM encoded output. Default is binary output.

`--base64` Create Base-64 encoded output; i.e. PEM without the header lines.

`--assume-armor`
Assume the input data is PEM encoded. Default is to autodetect the encoding but this is may fail.

`--assume-base64`
Assume the input data is plain base-64 encoded.

`--assume-binary`
Assume the input data is binary encoded.

`--local-user` *user_id*
`-u` *user_id*
Set the user(s) to be used for signing. The default is the first secret key found in the database.

`--with-key-data`
Displays extra information with the `--list-keys` commands. Especially a line tagged `grp` is printed which tells you the keygrip of a key. This string is for example used as the file name of the secret key.

`--with-validation`
When doing a key listing, do a full validation check for each key and print the result. This is usually a slow operation because it requires a CRL lookup and other operations.

When used along with –import, a validation of the certificate to import is done and only imported if it succeeds the test. Note that this does not affect an already available cwertificate in the DB. This option is therefore useful to simply verify a certificate.

`--with-md5-fingerprint`
For standard key listings, also print the MD5 fingerprint of the certificate.

2.2.4 How to change how the CMS is created.

`--include-certs` *n*
Using *n* of -2 includes all certificate except for the root cert, -1 includes all certs, 0 does not include any certs, 1 includes only the signers cert (this is the default) and all other positive values include up to *n* certificates starting with the signer cert.

2.2.5 Doing things one usually don't want to do.

`--faked-system-time` *epoch*
This option is only useful for testing; it sets the system time back or forth to *epoch* which is the number of seconds elapsed since the year 1970.

`--with-ephemeral-keys`
: Include ephemeral flagged keys in the output of key listings.

`--debug-level` *level*
: Select the debug level for investigating problems. *level* may be one of:

  `none` no debugging at all.

  `basic` some basic debug messages

  `advanced` more verbose debug messages

  `expert` even more detailed messages

  `guru` all of the debug messages you can get

  How these messages are mapped to the actual debugging flags is not specified and may change with newer releaes of this program. They are however carefully selected to best aid in debugging.

`--debug` *flags*
: This option is only useful for debugging and the behaviour may change at any time without notice; using `--debug-levels` is the preferred method to select the debug verbosity. FLAGS are bit encoded and may be given in usual C-Syntax. The currently defined bits are:

  `0 (1)` X.509 or OpenPGP protocol related data

  `1 (2)` values of big number integers

  `2 (4)` low level crypto operations

  `5 (32)` memory allocation

  `6 (64)` caching

  `7 (128)` show memory statistics.

  `9 (512)` write hashed data to files named `dbgmd-000*`

  `10 (1024)` trace Assuan protocol

  Note, that all flags set using this option may get overriden by `--debug-level`.

`--debug-all`
: Same as `--debug=0xffffffff`

`--debug-allow-core-dump`
: Usually `gpgsm` tries to avoid dumping core by well written code and by disabling core dumps for security reasons. However, bugs are pretty durable beasts and to squash them it is sometimes useful to have a core dump. This option enables core dumps unless the Bad Thing happened before the option parsing.

`--debug-no-chain-validation`
: This is actually not a debugging option but only useful as such. It lets `gpgsm` bypass all certificate chain validation checks.

`--debug-ignore-expiration`
: This is actually not a debugging option but only useful as such. It lets `gpgsm` ignore all notAfter dates, this is used by the regresssion tests.

`--fixed-passphrase` *string*
: Supply the passphrase *string* to the gpg-protect-tool. This option is only useful for the regression tests included with this package and may be revised or removed at any time without notice.

All the long options may also be given in the configuration file after stripping off the two leading dashes.

2.3 Examples

```
$ gpgsm -er goo@bar.net <plaintext >ciphertext
```

2.4 Unattended Usage

`gpgsm` is often used as a backend engine by other software. To help with this a machine interface has been defined to have an unambiguous way to do this. This is most likely used with the `--server` command but may also be used in the standard operation mode by using the `--status-fd` option.

2.5 Automated signature checking

It is very important to understand the semantics used with signature verification. Checking a signature is not as simple as it may sound and so the ooperation si a bit complicated. In mosted cases it is required to look at several status lines. Here is a table of all cases a signed message may have:

The signature is valid
: This does mean that the signature has been successfully verified, the certificates are all sane. However there are two subcases with important information: One of the certificates may have expired or a signature of a message itself as expired. It is a sound practise to consider such a signature still as valid but additional information should be displayed. Depending on the subcase `gpgsm` will issue these status codes:

  signature valid and nothing did expire
  : `GOODSIG`, `VALIDSIG`, `TRUST_FULLY`

  signature valid but at least one certificate has expired
  : `EXPKEYSIG`, `VALIDSIG`, `TRUST_FULLY`

  signature valid but expired
  : `EXPSIG`, `VALIDSIG`, `TRUST_FULLY` Note, that this case is currently not implemented.

The signature is invalid
: This means that the signature verification failed (this is an indication of af a transfer error, a programm error or tampering with the message). `gpgsm` issues one of these status codes sequences:

`BADSIG`

`GOODSIG, VALIDSIG TRUST_NEVER`

Error verifying a signature
: For some reason the signature could not be verified, i.e. it can't be decided whether the signature is valid or invalid. A common reason for this is a missing certificate.

2.6 The Protocol the Server Mode Uses.

Description of the protocol used to access `GPGSM`. `GPGSM` does implement the Assuan protocol and in addition provides a regular command line interface which exhibits a full client to this protocol (but uses internal linking). To start `gpgsm` as a server the command line the option `--server` must be used. Additional options are provided to select the communication method (i.e. the name of the socket).

We assume that the connection has already been established; see the Assuan manual for details.

2.6.1 Encrypting a Message

Before encrytion can be done the recipient must be set using the command:

`RECIPIENT` *userID*

Set the recipient for the encryption. *userID* should be the internal representation of the key; the server may accept any other way of specification. If this is a valid and trusted recipient the server does respond with OK, otherwise the return is an ERR with the reason why the recipient can't be used, the encryption will then not be done for this recipient. If the policy is not to encrypt at all if not all recipients are valid, the client has to take care of this. All `RECIPIENT` commands are cumulative until a `RESET` or an successful `ENCRYPT` command.

`INPUT FD=`*n* `[--armor|--base64|--binary]`

Set the file descriptor for the message to be encrypted to *n*. Obviously the pipe must be open at that point, the server establishes its own end. If the server returns an error the client should consider this session failed.

The `--armor` option may be used to advice the server that the input data is in PEM format, `--base64` advices that a raw base-64 encoding is used, `--binary` advices of raw binary input (BER). If none of these options is used, the server tries to figure out the used encoding, but this may not always be correct.

`OUTPUT FD=`*n* `[--armor|--base64]`

Set the file descriptor to be used for the output (i.e. the encrypted message). Obviously the pipe must be open at that point, the server establishes its own end. If the server returns an error he client should consider this session failed.

The option armor encodes the output in PEM format, the `--base64` option applies just a base 64 encoding. No option creates binary output (BER).

The actual encryption is done using the command

```
ENCRYPT
```

It takes the plaintext from the `INPUT` command, writes to the ciphertext to the file descriptor set with the `OUTPUT` command, take the recipients from all the recipients set so far. If this command fails the clients should try to delete all output currently done or otherwise mark it as invalid. `GPGSM` does ensure that there won't be any security problem with leftover data on the output in this case.

This command should in general not fail, as all necessary checks have been done while setting the recipients. The input and output pipes are closed.

2.6.2 Decrypting a message

Input and output FDs are set the same way as in encryption, but `INPUT` refers to the ciphertext and output to the plaintext. There is no need to set recipients. `GPGSM` automatically strips any S/MIME headers from the input, so it is valid to pass an entire MIME part to the INPUT pipe.

The encryption is done by using the command

```
DECRYPT
```

It performs the decrypt operation after doing some check on the internal state. (e.g. that all needed data has been set). Because it utilizes the GPG-Agent for the session key decryption, there is no need to ask the client for a protecting passphrase - GpgAgent takes care of this by requesting this from the user.

2.6.3 Signing a Message

Signing is usually done with these commands:

```
INPUT FD=n [--armor|--base64|--binary]
```

This tells `GPGSM` to read the data to sign from file descriptor *n*.

```
OUTPUT FD=m [--armor|--base64]
```

Write the output to file descriptor *m*. If a detached signature is requested, only the signature is written.

```
SIGN [--detached]
```

Sign the data set with the INPUT command and write it to the sink set by OUTPUT. With `--detached`, a detached signature is created (surprise).

The key used for signining is the default one or the one specified in the configuration file. To get finer control over the keys, it is possible to use the command

```
SIGNER userID
```

to the signer's key. *userID* should be the internal representation of the key; the server may accept any other way of specification. If this is a valid and trusted recipient the server does respond with OK, otherwise the return is an ERR with the reason why the key can't be used, the signature will then not be created using this key. If the policy is not to sign at all if not all keys are valid, the client has to take care of this. All `SIGNER` commands are cumulative until a `RESET` is done. Note that a `SIGN` does not reset this list of signers which is in contrats to the `RECIPIENT` command.

2.6.4 Verifying a Message

To verify a mesage the command:

```
VERIFY
```

is used. It does a verify operation on the message send to the input FD. The result is written out using status lines. If an output FD was given, the signed text will be written to that. If the signature is a detached one, the server will inquire about the signed material and the client must provide it.

2.6.5 Generating a Key

This is used to generate a new keypair, store the secret part in the PSE and the public key in the key database. We will probably add optional commands to allow the client to select whether a hardware token is used to store the key. Configuration options to `GPGSM` can be used to restrict the use of this command.

```
GENKEY
```

`GPGSM` checks whether this command is allowed and then does an INQUIRY to get the key parameters, the client should then send the key parameters in the native format:

```
S: INQUIRE KEY_PARAM native
C: D foo:fgfgfg
C: D bar
C: END
```

Please note that the server may send Status info lines while reading the data lines from the client. After this the key generation takes place and the server eventually does send an ERR or OK response. Status lines may be issued as a progress indicator.

2.6.6 List available keys

To list the keys in the internal database or using an external key provider, the command:

```
LISTKEYS  pattern
```

is used. To allow multiple patterns (which are ORed during the search) quoting is required: Spaces are to be translated into `"+"` or into `"%20"`; in turn this requires that the usual escape quoting rules are done.

```
LISTSECRETKEYS pattern
```

Lists only the keys where a secret key is available.

The list commands commands are affected by the option

```
OPTION list-mode=mode
```

where mode may be:

`0` Use default (which is usually the same as 1).

`1` List only the internal keys.

`2` List only the external keys.

`3` List internal and external keys.

Note that options are valid for the entire session.

2.6.7 Export certificates

To export certificate from the internal key database the command:

```
EXPORT pattern
```

is used. To allow multiple patterns (which are ORed) quoting is required: Spaces are to be translated into "+" or into "%20"; in turn this requires that the usual escape quoting rules are done.

The format of the output depends on what was set with the OUTPUT command. When using PEM encoding a few informational lines are prepended.

2.6.8 Import certificates

To import certificates into the internal key database, the command

```
IMPORT
```

is used. The data is expected on the file descriptor set with the `INPUT` command. Certain checks are performend on the certificate. Note that the code will also handle PKCS\#12 files and import private keys; a helper program is used for that.

2.6.9 Delete certificates

To delete certificate the command

```
DELKEYS pattern
```

is used. To allow multiple patterns (which are ORed) quoting is required: Spaces are to be translated into "+" or into "%20"; in turn this requires that the usual escape quoting rules are done.

The certificates must be specified unambiguously otherwise an error is returned.

3 Invoking GPG-AGENT

`gpg-agent` is a daemon to manage secret (private) keys independelty from any protocol. It is used as a backend for `gpg` and `gpgsm` as well as for a couple of other utilities.

The usual way to run the agent is from the `~/.xsession` file:

```
eval `gpg-agent --daemon`
```

If you don't use an X server, you can also put this into your regular startup file `~/.profile` or `.bash_profile`. It is best not to run multiple instance of the `gpg-agent`, so you should make sure that only is running: `gpg-agent` uses an environment variable to inform clients about the communication parameters. You can write the content of this environment variable to a file so that you can test for a running agent. This short script may do the job:

```
if test -f $HOME/.gpg-agent-info && \
   kill -0 `cut -d: -f 2 $HOME/.gpg-agent-info` 2>/dev/null; then
     GPG_AGENT_INFO=`cat $HOME/.gpg-agent-info`
     export GPG_AGENT_INFO
else
     eval `gpg-agent --daemon`
     echo $GPG_AGENT_INFO >$HOME/.gpg-agent-info
fi
```

You should always add the following lines to your `.bashrc` or whatever initialization file is used for all shell invocations:

```
GPG_TTY=`tty`
export GPG_TTY
```

It is important that this environment variable always reflects the output of the `tty` command. For W32 systems this option is not required.

Please make sure that a proper pinentry program has been installed under the default filename (which is system dependant) or use the option `pinentry-pgm` to specify the full name of that program. It is often useful to install a symbolic link from the actual used pinentry (e.g. '`/usr/bin/pinentry-gtk`') to the expected one (e.g. '`/usr/bin/pinentry`').

See [Option Index], page 61, for an index to `GPG-AGENT`'s commands and options.

3.1 Commands

Commands are not distinguished from options execpt for the fact that only one one command is allowed.

`--version`
: Print the program version and licensing information. Not that you can abbreviate this command.

`--help, -h`
: Print a usage message summarizing the most usefule command-line options. Not that you can abbreviate this command.

`--dump-options`
: Print a list of all available options and commands. Not that you can abbreviate this command.

`--server` Run in server mode and wait for commands on the `stdin`. The default mode is to create a socket and listen for commands there.

`--daemon` Run the program in the background. This option is required to prevent it from being accidently running in the background. A common way to do this is:

$ eval ‘gpg-agent –daemon‘

3.2 Option Summary

`--options` *file*
Reads configuration from *file* instead of from the default per-user configuration file. The default configuration file is named ‘`gpg-agent.conf`’ and expected in the ‘`.gnupg`’ directory directly below the home directory of the user.

`--homedir` *dir*
Set the name of the home directory to *dir*. If his option is not used, the home directory defaults to ‘`~/.gnupg`’. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable `GNUPGHOME` or (on W32 systems) by means on the Registry entry *HKCU\Software\GNU\GnuPG:HomeDir*.

`-v`

`--verbose`
Outputs additional information while running. You can increase the verbosity by giving several verbose commands to GPGSM, such as ‘`-vv`’.

`-q`

`--quiet` Try to be as quiet as possible.

`--batch` Don’t invoke a pinentry or do any other thing requiring human interaction.

`--faked-system-time` *epoch*
This option is only useful for testing; it sets the system time back or forth to *epoch* which is the number of seconds elapsed since the year 1970.

`--debug-level` *level*
Select the debug level for investigating problems. *level* may be one of:

`none` no debugging at all.

`basic` some basic debug messages

`advanced` more verbose debug messages

`expert` even more detailed messages

`guru` all of the debug messages you can get

How these messages are mapped to the actual debugging flags is not specified and may change with newer releaes of this program. They are however carefully selected to best aid in debugging.

`--debug` *flags*
: This option is only useful for debugging and the behaviour may change at any time without notice. FLAGS are bit encoded and may be given in usual C-Syntax. The currently defined bits are:

  `0 (1)` X.509 or OpenPGP protocol related data

  `1 (2)` values of big number integers

  `2 (4)` low level crypto operations

  `5 (32)` memory allocation

  `6 (64)` caching

  `7 (128)` show memory statistics.

  `9 (512)` write hashed data to files named `dbgmd-000*`

  `10 (1024)` trace Assuan protocol

  `12 (4096)` bypass all certificate validation

`--debug-all`
: Same as `--debug=0xffffffff`

`--debug-wait` *n*
: When running in server mode, wait *n* seconds before entering the actual processing loop and print the pid. This gives time to attach a debugger.

`--no-detach`
: Don't detach the process from the console. This is manly usefule for debugging.

`-s`
`--sh`
`-c`
`--csh`
: Format the info output in daemon mode for use with the standard Bourne shell respective the C-shell . The default ist to guess it based on the environment variable `SHELL` which is in almost all cases sufficient.

`--no-grab`
: Tell the pinentryo not to grab the keyboard and mouse. This option should in general not be used to avaoid X-sniffing attacks.

`--log-file` *file*
: Append all logging output to *file*. This is very helpful in seeing what the agent actually does.

`--disable-pth`
: Don't allow multiple connections. This option is in general not very useful.

`--allow-mark-trusted`
: Allow clients to mark keys as trusted, i.e. put them into the `trustlist.txt` file. This is by default not allowed to make it harder for users to inadvertly accept Root-CA keys.

`--ignore-cache-for-signing`
: This option will let `gpg-agent` bypass the passphrase cache for all signing operation. Note that there is also a per-session option to control this behaviour but this command line option takes precedence.

`--default-cache-ttl` *n*
: Set the time a cache entry is valid to *n* seconds. The default are 600 seconds.

`--max-cache-ttl` *n*
: Set the maximum time a cache entry is valid to *n* seconds. After this time a cache entry will get expired even if it has been accessed recently. The default are 2 hours (7200 seconds).

`--pinentry-program` *filename*
: Use program *filename* as the PIN entry. The default is installation dependend and can be shown with the `--version` command.

`--scdaemon-program` *filename*
: Use program *filename* as the Smartcard daemon. The default is installation dependend and can be shown with the `--version` command.

`--use-standard-socket`
`--no-use-standard-socket`
: By enabling this option `gpg-agent` will listen on the socket named '`S.gpg-agent`', located in the home directory, and not create a random socket below a temporary directory. Tools connecting to `gpg-agent` should first try to connect to the socket given in environment variable *GPG_AGENT_INFO* and the fall back to this socket. This option may not be used if the home directory is mounted as a remote file system.

  Note, that as of now, W32 systems default to this option.

`--display` *string*
`--ttyname` *string*
`--ttytype` *string*
`--lc-type` *string*
`--lc-messages` *string*
: These options are used with the server mode to pass localization information.

`--keep-tty`
`--keep-display`
: Ignore requests to change change the current TTY respective the X window system's `DISPLAY` variable. This is useful to lock the pinentry to pop up at the TTY or display you started the agent.

All the long options may also be given in the configuration file after stripping off the two leading dashes.

3.3 Use of some signals.

A running `gpg-agent` may be controlled by signals, i.e. using the `kill` command to send a signal to the process.

Here is a list of supported signals:

`SIGHUP` This signals flushes all chached passphrases and when the program was started with a configuration file, the configuration file is read again. Only certain options are honored: `quiet`, `verbose`, `debug`, `debug-all`, `no-grab`, `pinentry-program`, `default-cache-ttl` and `ignore-cache-for-signing`. `scdaemon-program` is also supported but due to the current implementation, which calls the scdaemon only once, it is not of much use.

`SIGTERM` Shuts down the process but waits until all current requests are fulfilled. If the process has received 3 of these signals and requests are still pending, a shutdown is forced.

`SIGINT` Shuts down the process immediately.

`SIGUSR1`
`SIGUSR2` These signals are used for internal purposes.

3.4 Examples

```
$ eval `gpg-agent --daemon`
```

3.5 Agent's Assuan Protocol

The `gpg-agent` should be started by the login shell and set an environment variable to tell clients about the socket to be used. Clients should deny to access an agent with a socket name which does not match its own configuration. An application may choose to start an instance of the gpgagent if it does not figure that any has been started; it should not do this if a gpgagent is running but not usable. Because `gpg-agent` can only be used in background mode, no special command line option is required to activate the use of the protocol.

To identify a key we use a thing called keygrip which is the SHA-1 hash of an canoncical encoded S-Expression of the the public key as used in Libgcrypt. For the purpose of this interface the keygrip is given as a hex string. The advantage of using this and not the hash of a certificate is that it will be possible to use the same keypair for different protocols, thereby saving space on the token used to keep the secret keys.

3.5.1 Decrypting a session key

The client asks the server to decrypt a session key. The encrypted session key should have all information needed to select the appropriate secret key or to delegate it to a smartcard.

```
SETKEY <keyGrip>
```

Tell the server about the key to be used for decryption. If this is not used, `gpg-agent` may try to figure out the key by trying to decrypt the message with each key available.

```
PKDECRYPT
```

The agent checks whether this command is allowed and then does an INQUIRY to get the ciphertext the client should then send the cipher text.

```
S: INQUIRE CIPHERTEXT
C: D (xxxxxx
C: D xxxx)
C: END
```

Please note that the server may send status info lines while reading the data lines from the client. The data send is a SPKI like S-Exp with this structure:

```
(enc-val
  (<algo>
    (<param_name1> <mpi>)
...
    (<param_namen> <mpi>)))
```

Where algo is a string with the name of the algorithm; see the libgcrypt documentation for a list of valid algorithms. The number and names of the parameters depend on the algorithm. The agent does return an error if there is an inconsistency.

If the decryption was successful the decrypted data is returned by means of "D" lines.

Here is an example session:

```
C: PKDECRYPT
S: INQUIRE CIPHERTEXT
C: D (enc-val elg (a 349324324)
C: D    (b 3F444677CA)))
C: END
S: # session key follows
S: D 1234567890ABCDEF0
S: OK descryption successful
```

3.5.2 Signing a Hash

The client ask the agent to sign a given hash value. A default key will be chosen if no key has been set. To set a key a client first uses:

```
SIGKEY <keyGrip>
```

This can be used multiple times to create multiple signature, the list of keys is reset with the next PKSIGN command or a RESET. The server test whether the key is a valid key to sign something and responds with okay.

```
SETHASH <hexstring>
```

The client can use this command to tell the server about the data (which usually is a hash) to be signed.

The actual signing is done using

```
PKSIGN <options>
```

Options are not yet defined, but my later be used to choosen among different algorithms (e.g. pkcs 1.5)

The agent does then some checks, asks for the passphrase and if SETHASH has not been used asks the client for the data to sign:

```
S: INQUIRE HASHVAL
C: D ABCDEF012345678901234
C: END
```

As a result the server returns the signature as an SPKI like S-Exp in "D" lines:

```
(sig-val
  (<algo>
    (<param_name1> <mpi>)
...
    (<param_namen> <mpi>)))
```

The operation is affected by the option

```
OPTION use-cache-for-signing=0|1
```

The default of `1` uses the cache. Setting this option to `0` will lead `gpg-agent` to ignore the passphrase cache. Note, that there is also a global command line option for `gpg-agent` to globally disable the caching.

Here is an example session:

```
C: SIGKEY <keyGrip>
S: OK key available
C: SIGKEY <keyGrip>
S: OK key available
C: PKSIGN
S: # I did ask the user whether he really wants to sign
S: # I did ask the user for the passphrase
S: INQUIRE HASHVAL
C: D ABCDEF012345678901234
C: END
S: # signature follows
S: D (sig-val rsa (s 45435453654612121212))
S: OK
```

3.5.3 Generating a Key

This is used to create a new keypair and store the secret key inside the active PSE -w which is in most cases a Soft-PSE. An not yet defined option allows to choose the storage location. To get the secret key out of the PSE, a special export tool has to be used.

```
GENKEY
```

Invokes the key generation process and the server will then inquire on the generation parameters, like:

```
S: INQUIRE KEYPARM
C: D (genkey (rsa (nbits  1024)))
C: END
```

The format of the key parameters which depends on the algorithm is of the form:

```
(genkey
  (algo
    (parameter_name_1 ....)
```

```
      ....
    (parameter_name_n ....)))
```

If everything succeeds, the server returns the *public key* in a SPKI like S-Expression like this:

```
  (public-key
    (rsa
(n <mpi>)
(e <mpi>)))
```

Here is an example session:

```
C: GENKEY
S: INQUIRE KEYPARM
C: D (genkey (rsa (nbits  1024)))
C: END
S: D (public-key
S: D   (rsa (n 326487324683264) (e 10001)))
S  OK key created
```

3.5.4 Importing a Secret Key

This operation is not yet supportted by GpgAgent. Specialized tools are to be used for this.

There is no actual need because we can expect that secret keys created by a 3rd party are stored on a smartcard. If we have generated the key ourself, we do not need to import it.

3.5.5 Export a Secret Key

Not implemented.

Should be done by an extra tool.

3.5.6 Importing a Root Certificate

Actually we do not import a Root Cert but provide a way to validate any piece of data by storing its Hash along with a description and an identifier in the PSE. Here is the interface desription:

```
ISTRUSTED <fingerprint>
```

Check whether the OpenPGP primary key or the X.509 certificate with the given fingerprint is an ultimately trusted key or a trusted Root CA certificate. The fingerprint should be given as a hexstring (without any blanks or colons or whatever in between) and may be left padded with 00 in case of an MD5 fingerprint. GPGAgent will answer with:

```
OK
```

The key is in the table of trusted keys.

```
ERR 304 (Not Trusted)
```

The key is not in this table.

Gpg needs the entire list of trusted keys to maintain the web of trust; the following command is therefore quite helpful:

```
LISTTRUSTED
```

GpgAgent returns a list of trusted keys line by line:

```
S: D 000000001234454556565656677878AF2F1ECCFF P
S: D 3403875634856348564356456348564386576457A P
S: D FEDC6532453745367FD834743574957437574357435D S
S: OK
```

The first item on a line is the hexified fingerprint where MD5 ingerprints are `00` padded to the left and the second item is a flag to indicate the type of key (so that gpg is able to only take care of PGP keys). P = OpenPGP, S = S/MIME. A client should ignore the rest of the line, so that we can extend the format in the future.

Finally a client should be able to mark a key as trusted:

```
MARKTRUSTED fingerprint "P"|"S"
```

The server will then pop up a window to ask the user whether she really trusts this key. For this it will probably ask for a text to be displayed like this:

```
S: INQUIRE TRUSTDESC
C: D Do you trust the key with the fingerprint @FPR@
C: D bla fasel blurb.
C: END
S: OK
```

Known sequences with the pattern `@foo@` are replaced according to this table:

`@FPR16@` Format the fingerprint according to gpg rules for a v3 keys.

`@FPR20@` Format the fingerprint according to gpg rules for a v4 keys.

`@FPR@` Choose an appropriate format to format the fingerprint.

`@@` Replaced by a single `@`

3.5.7 Ask for a passphrase

This function is usually used to ask for a passphrase to be used for conventional encryption, but may also be used by programs which need special handling of passphrases. This command uses a syntax which helps clients to use the agent with minimum effort.

```
GET_PASSPHRASE cache_id [error_message prompt description]
```

cache_id is expected to be a hex string used for caching a passphrase. Use a `X` to bypass the cache. With no other arguments the agent returns a cached passphrase or an error.

error_message is either a single `X` for no error message or a string to be shown as an error message like (e.g. "invalid passphrase"). Blanks must be percent escaped or replaced by `+`'.

prompt is either a single `X` for a default prompt or the text to be shown as the prompt. Blanks must be percent escaped or replaced by `+`.

description is a text shown above the entry field. Blanks must be percent escaped or replaced by `+`.

The agent either returns with an error or with a OK followed by the hex encoded passphrase. Note that the length of the strings is implicitly limited by the maximum length of a command.

`CLEAR_PASSPHRASE` *`cache_id`*

may be used to invalidate the cache entry for a passphrase. The function returns with OK even when there is no cached passphrase.

3.5.8 Ask for confirmation

This command may be used to ask for a simple confirmation by presenting a text and 2 bottonts: Okay and Cancel.

`GET_CONFIRMATION` *`description`*

*description*is displayed along with a Okay and Cancel button. Blanks must be percent escaped or replaced by `+`. A `X` may be used to display confirmation dialog with a default text.

The agent either returns with an error or with a OK. Note, that the length of *description* is implicitly limited by the maximum length of a command.

3.5.9 Check whether a key is available

This can be used to see whether a secret key is available. It does not return any information on whether the key is somehow protected.

`HAVEKEY` *`keygrip`*

The Agent answers either with OK or `No_Secret_Key` (208). The caller may want to check for other error codes as well.

3.5.10 Register a smartcard

`LEARN [--send]`

This command is used to register a smartcard. With the –send option given the certificates are send back.

3.5.11 Change a Passphrase

`PASSWD` *`keygrip`*

This command is used to interactively change the passphrase of the key indentified by the hex string *keygrip*.

4 Invoking the SCDAEMON

The `scdaemon` is a daemon to manage smartcards. It is usually invoked by gpg-agent and in general not used directly.

See [Option Index], page 61, for an index to GPG-AGENTS's commands and options.

4.1 Commands

Commands are not distinguished from options execpt for the fact that only one one command is allowed.

`--version`
: Print the program version and licensing information. Not that you can abbreviate this command.

`--help, -h`
: Print a usage message summarizing the most usefule command-line options. Not that you can abbreviate this command.

`--dump-options`
: Print a list of all available options and commands. Not that you can abbreviate this command.

`--server` Run in server mode and wait for commands on the `stdin`. This is default mode is to create a socket and listen for commands there.

`--daemon` Run the program in the background. This option is required to prevent it from being accidently running in the background.

`--print-atr`
: This is mainly a debugging command, used to print the ATR (Answer-To-Reset) of a card and exit immediately.

4.2 Option Summary

`--options` *file*
: Reads configuration from *file* instead of from the default per-user configuration file. The default configuration file is named '`scdaemon.conf`' and expected in the '`.gnupg`' directory directly below the home directory of the user.

`--homedir` *dir*
: Set the name of the home directory to *dir*. If his option is not used, the home directory defaults to '`~/.gnupg`'. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable `GNUPGHOME` or (on W32 systems) by means on the Registry entry *HKCU\Software\GNU\GnuPG:HomeDir*.

`-v`

`--verbose`
: Outputs additional information while running. You can increase the verbosity by giving several verbose commands to `gpgsm`, such as '`-vv`'.

`--debug-level` *level*
: Select the debug level for investigating problems. *level* may be one of:

  `none` no debugging at all.

  `basic` some basic debug messages

  `advanced` more verbose debug messages

  `expert` even more detailed messages

  `guru` all of the debug messages you can get

  How these messages are mapped to the actual debugging flags is not specified and may change with newer releaes of this program. They are however carefully selected to best aid in debugging.

`--debug` *flags*
: This option is only useful for debugging and the behaviour may change at any time without notice. FLAGS are bit encoded and may be given in usual C-Syntax. The currently defined bits are:

  `0 (1)` X.509 or OpenPGP protocol related data

  `1 (2)` values of big number integers

  `2 (4)` low level crypto operations

  `5 (32)` memory allocation

  `6 (64)` caching

  `7 (128)` show memory statistics.

  `9 (512)` write hashed data to files named `dbgmd-000*`

  `10 (1024)` trace Assuan protocol

  `12 (4096)` bypass all certificate validation

`--debug-all`
: Same as `--debug=0xffffffff`

`--debug-wait` *n*
: When running in server mode, wait *n* seconds before entering the actual processing loop and print the pid. This gives time to attach a debugger.

`--debug-sc` *n*
: Set the debug level of the OpenSC library to *n*.

`--no-detach`
: Don't detach the process from the console. This is manly usefule for debugging.

`--log-file` *file*
: Append all logging output to *file*. This is very helpful in seeing what the agent actually does.

`--reader-port` *number*
: When the program has been build without OpenSC support, this option must be used to specify the port of the card terminal. A value of 0 refers to the first serial device; add 32768 to access USB devices. The default is 32768 (first USB device).

`--ctapi-driver` *library*
: Use *library* to access the smartcard reader. The current default is `libtowitoko.so`. Note that the use of this interface is deprecated; it may be removed in future releases.

`--allow-admin`
`--deny-admin`
: This enables the use of Admin class commands for card applications where this is supported. Currently we support it for the OpenPGP card. Deny is the default. This commands is useful to inhibit accidental access to admin class command which could ultimately lock the card through worng PIN numbers.

`--disable-application` *name*
: This option disables the use of the card application named *name*. This is mainly useful for debugging or if a application with lower priority should be used by default.

All the long options may also be given in the configuration file after stripping off the two leading dashes.

4.3 Description of card applications

`scdaemon` supports the card applications as described below.

4.3.1 The OpenPGP card application "openpgp"

This application is currently only used by `gpg` but may in future also be useful with `gpgsm`.

The specification for such a card is available at `http://g10code.com/docs/openpgp-card-1.0.pdf`.

4.3.2 The Telesec NetKey card "nks"

This is the main application of the Telesec cards as available in Germany. It is a superset of the German DINSIG card. The card is used by `gpgsm`.

4.3.3 The DINSIG card application "dinsig"

This is an application as described in the German draft standard *DIN V 66291-1*. It is intended to be used by cards supporteing the German signature law and its bylaws (SigG and SigV).

4.3.4 The PKCS#15 card application "p15"

This is common fraqmework for smart card applications; support is only available if compiled with support for the OpenSC library. It is used by `gpgsm`.

4.4 Examples

```
$ scdaemon --server -v
```

4.5 Scdaemon's Assuan Protocol

The SC-Daemon should be started by the system to provide access to external tokens. Using Smartcards on a multi-user system does not make much sense expcet for system services, but in this case no regular user accounts are hosted on the machine.

A client connects to the SC-Daemon by connecting to the socket named '`/var/run/scdaemon/socket`', configuration information is read from */etc/scdaemon.conf*

Each connection acts as one session, SC-Daemon takes care of syncronizing access to a token between sessions.

4.5.1 Return the serial number

This command should be used to check for the presence of a card. It is special in that it can be used to reset the card. Most other commands will return an error when a card change has been detected and the use of this function is therefore required.

Background: We want to keep the client clear of handling card changes between operations; i.e. the client can assume that all operations are done on the same card unless he call this function.

```
SERIALNO
```

Return the serial number of the card using a status reponse like:

```
S SERIALNO D27600000000000000000000 0
```

The trailing 0 should be ignored for now, it is reserved for a future extension. The serial number is the hex encoded value identified by the `0x5A` tag in the GDO file (FIX=0x2F02).

4.5.2 Read all useful information from the card

```
LEARN [--force]
```

Learn all useful information of the currently inserted card. When used without the force options, the command might do an INQUIRE like this:

```
INQUIRE KNOWNCARDP <hexstring_with_serialNumber> <timestamp>
```

The client should just send an `END` if the processing should go on or a `CANCEL` to force the function to terminate with a cancel error message. The response of this command is a list of status lines formatted as this:

```
S KEYPAIRINFO hexstring_with_keygrip hexstring_with_id
```

If there is no certificate yet stored on the card a single "X" is returned in *hexstring_with_keygrip*.

4.5.3 Return a certificate

```
READCERT hexified_certid
```

This function is used to read a certificate identified by *hexified_certid* from the card.

4.5.4 Return a public key

```
READKEY hexified_certid
```

Return the public key for the given cert or key ID as an standard S-Expression.

4.5.5 Signing data with a Smartcard

To sign some data the caller should use the command

```
SETDATA hexstring
```

to tell `scdaemon` about the data to be signed. The data must be given in hex notation. The actual signing is done using the command

```
PKSIGN keyid
```

where *keyid* is the hexified ID of the key to be used. The key id may have been retrieved using the command `LEARN`.

4.5.6 Decrypting data with a Smartcard

To decrypt some data the caller should use the command

```
SETDATA hexstring
```

to tell `scdaemon` about the data to be decrypted. The data must be given in hex notation. The actual decryption is then done using the command

```
PKDECRYPT keyid
```

where *keyid* is the hexified ID of the key to be used.

4.5.7 Read an attribute's value.

TO BE WRITTEN.

4.5.8 Update an attribute's value.

TO BE WRITTEN.

4.5.9 Generate a new key on-card.

TO BE WRITTEN.

4.5.10 Return random bytes generate on-card.

TO BE WRITTEN.

4.5.11 Change PINs.

TO BE WRITTEN.

4.5.12 Perform a VERIFY operation.

TO BE WRITTEN.

5 Helper Tools

GnuPG comes with a couple of smaller tools:

5.1 Read logs from a socket

Most of the main utilities are able to write there log files to a Unix Domain socket if configured that way. `watchgnupg` is a simple listener for such a socket. It ameliorates the output with a time stamp and makes sure that long lines are not interspersed with log output from other utilities.

`watchgnupg` is commonly invoked as

'`watchgnupg --force ~/.gnupg/S.log`'

This starts it on the current terminal for listening on the socket '`~/.gnupg/S.log`'.

`watchgnupg` understands these options:

`--force` Delete an already existing socket file.

`--verbose`
Enable extra informational output.

`--version`
print version of the program and exit

`--help` Display a brief help page and exit

5.2 Create .gnupg home directories.

If GnuPG is installed on a system with existing user accounts, it is sometimes required to populate the GnuPG home directory with existing files. Especially a '`trustlist.txt`' and a keybox with some initial certificates are often desired. This scripts help to do this by copying all files from '`/etc/skel/.gnupg`' to the home directories of the accounts given on the command line. It takes care not to overwrite existing GnuPG home directories.

`addgnupghome` is invoked by root as:

'`addgnupghome account1 account2 ... accountn`'

5.3 Modify .gnupg home directories.

The `gpgconf` is a utility to automatically and reasonable safely query and modify configuration files in the '`.gnupg`' home directory. It is designed not to be invoked manually by the user, but automatically by graphical user interfaces (GUI).[1]

`gpgconf` provides access to the configuration of one or more components of the GnuPG system. These components correspond more or less to the programs that exist in the

[1] Please note that currently no locking is done, so concurrent access should be avoided. There are some precautions to avoid corruption with concurrent usage, but results may be inconsistent and some changes may get lost. The stateless design makes it difficult to provide more guarantees.

GnuPG framework, like GnuPG, GPGSM, DirMngr, etc. But this is not a strict one-to-one relationship. Not all configuration options are available through `gpgconf`. `gpgconf` provides a generic and abstract method to access the most important configuration options that can feasibly be controlled via such a mechanism.

`gpgconf` can be used to gather and change the options available in each component, and can also provide their default values. `gpgconf` will give detailed type information that can be used to restrict the user's input without making an attempt to commit the changes.

`gpgconf` provides the backend of a configuration editor. The configuration editor would usually be a graphical user interface program, that allows to display the current options, their default values, and allows the user to make changes to the options. These changes can then be made active with `gpgconf` again. Such a program that uses `gpgconf` in this way will be called GUI throughout this section.

5.3.1 Invoking gpgconf

One of the following commands must be given:

`--list-components`
: List all components. This is the default command used if none is specified.

`--list-options` *component*
: List all options of the component *component*.

`--change-options` *component*
: Change the options of the component *component*.

The following options may be used:

`-v`
`--verbose`
: Outputs additional information while running. Specifically, this extends numerical field values by human-readable descriptions.

`-r`
`--runtime`
: Only used together with `--change-options`. If one of the modified options can be changed in a running daemon process, signal the running daemon to ask it to reparse its configuration file after changing.

  This means that the changes will take effect at run-time, as far as this is possible. Otherwise, they will take effect at the next start of the respective backend programs.

5.3.2 Format conventions

Some lines in the output of `gpgconf` contain a list of colon-separated fields. The following conventions apply:

- The GUI program is required to strip off trailing newline and/or carriage return characters from the output.

- `gpgconf` will never leave out fields. If a certain version provides a certain field, this field will always be present in all `gpgconf` versions from that time on.
- Future versions of `gpgconf` might append fields to the list. New fields will always be separated from the previously last field by a colon separator. The GUI should be prepared to parse the last field it knows about up until a colon or end of line.
- Not all fields are defined under all conditions. You are required to ignore the content of undefined fields.

There are several standard types for the content of a field:

verbatim
: Some fields contain strings that are not escaped in any way. Such fields are described to be used *verbatim*. These fields will never contain a colon character (for obvious reasons). No de-escaping or other formatting is required to use the field content. This is for easy parsing of the output, when it is known that the content can never contain any special characters.

percent-escaped
: Some fields contain strings that are described to be *percent-escaped*. Such strings need to be de-escaped before their content can be presented to the user. A percent-escaped string is de-escaped by replacing all occurences of `%XY` by the byte that has the hexadecimal value `XY`. `X` and `Y` are from the set `0-9a-f`.

localised
: Some fields contain strings that are described to be *localised*. Such strings are translated to the active language and formatted in the active character set.

unsigned number
: Some fields contain an *unsigned number*. This number will always fit into a 32-bit unsigned integer variable. The number may be followed by a space, followed by a human readable description of that value (if the verbose option is used). You should ignore everything in the field that follows the number.

signed number
: Some fields contain a *signed number*. This number will always fit into a 32-bit signed integer variable. The number may be followed by a space, followed by a human readable description of that value (if the verbose option is used). You should ignore everything in the field that follows the number.

option
: Some fields contain an *option* argument. The format of an option argument depends on the type of the option and on some flags:

  no argument
  : The simplest case is that the option does not take an argument at all (*type* `0`). Then the option argument is an unsigned number that specifies how often the option occurs. If the `list` flag is not set, then the only valid number is `1`. Options that do not take an argument never have the `default` or `optional arg` flag set.

  number
  : If the option takes a number argument (*alt-type* is `2` or `3`), and it can only occur once (`list` flag is not set), then the option argument is either empty (only allowed if the argument is optional), or it is a number. A number is a string that begins with an optional minus

character, followed by one or more digits. The number must fit into an integer variable (unsigned or signed, depending on *alt-type*).

number list
: If the option takes a number argument and it can occur more than once, then the option argument is either empty, or it is a comma-separated list of numbers as described above.

string
: If the option takes a string argument (*alt-type* is 1), and it can only occur once (`list` flag is not set) then the option argument is either empty (only allowed if the argument is optional), or it starts with a double quote character (`"`) followed by a percent-escaped string that is the argument value. Note that there is only a leading double quote character, no trailing one. The double quote character is only needed to be able to differentiate between no value and the empty string as value.

string list
: If the option takes a number argument and it can occur more than once, then the option argument is either empty, or it is a comma-separated list of string arguments as described above.

The active language and character set are currently determined from the locale environment of the `gpgconf` program.

5.3.3 Listing components

The command `--list-components` will list all components that can be configured with `gpgconf`. Usually, one component will correspond to one GnuPG-related program and contain the options of that programs configuration file that can be modified using `gpgconf`. However, this is not necessarily the case. A component might also be a group of selected options from several programs, or contain entirely virtual options that have a special effect rather than changing exactly one option in one configuration file.

A component is a set of configuration options that semantically belong together. Furthermore, several changes to a component can be made in an atomic way with a single operation. The GUI could for example provide a menu with one entry for each component, or a window with one tabulator sheet per component.

The command argument `--list-components` lists all available components, one per line. The format of each line is:

`name:description`

name
: This field contains a name tag of the component. The name tag is used to specify the component in all communication with `gpgconf`. The name tag is to be used *verbatim*. It is thus not in any escaped format.

description
: The *string* in this field contains a human-readable description of the component. It can be displayed to the user of the GUI for informational purposes. It is *percent-escaped* and *localized*.

Example:

```
$ gpgconf --list-components
gpg:GPG for OpenPGP
gpg-agent:GPG Agent
scdaemon:Smartcard Daemon
gpgsm:GPG for S/MIME
dirmngr:Directory Manager
```

5.3.4 Listing options

Every component contains one or more options. Options may be gathered into option groups to allow the GUI to give visual hints to the user about which options are related.

The command argument `--list-options` *component* lists all options (and the groups they belong to) in the component *component*, one per line. *component* must be the string in the field *name* in the output of the `--list-components` command.

There is one line for each option and each group. First come all options that are not in any group. Then comes a line describing a group. Then come all options that belong into each group. Then comes the next group and so on. There does not need to be any group (and in this case the output will stop after the last non-grouped option).

The format of each line is:

name`:`*flags*`:`*level*`:`*description*`:`*type*`:`*alt-type*`:`*argname*`:`*default*`:`*argdef*`:`*value*

name This field contains a name tag for the group or option. The name tag is used to specify the group or option in all communication with `gpgconf`. The name tag is to be used *verbatim*. It is thus not in any escaped format.

flags The flags field contains an *unsigned number*. Its value is the OR-wise combination of the following flag values:

`group (1)` If this flag is set, this is a line describing a group and not an option.

The following flag values are only defined for options (that is, if the `group` flag is not used).

`optional arg (2)`
If this flag is set, the argument is optional. This is never set for *type* `0` (none) options.

`list (4)` If this flag is set, the option can be given multiple times.

`runtime (8)`
If this flag is set, the option can be changed at runtime.

`default (16)`
If this flag is set, a default value is available.

`default desc (32)`
If this flag is set, a (runtime) default is available. This and the `default` flag are mutually exclusive.

`no arg desc (64)`
If this flag is set, and the `optional arg` flag is set, then the option has a special meaning if no argument is given.

level This field is defined for options and for groups. It contains an *unsigned number* that specifies the expert level under which this group or option should be displayed. The following expert levels are defined for options (they have analogous meaning for groups):

`basic (0)` This option should always be offered to the user.

`advanced (1)`
This option may be offered to advanced users.

`expert (2)`
This option should only be offered to expert users.

`invisible (3)`
This option should normally never be displayed, not even to expert users.

`internal (4)`
This option is for internal use only. Ignore it.

The level of a group will always be the lowest level of all options it contains.

description
This field is defined for options and groups. The *string* in this field contains a human-readable description of the option or group. It can be displayed to the user of the GUI for informational purposes. It is *percent-escaped* and *localized*.

type This field is only defined for options. It contains an *unsigned number* that specifies the type of the option's argument, if any. The following types are defined:

Basic types:

`none (0)` No argument allowed.

`string (1)`
An *unformatted string*.

`int32 (2)` A *signed number*.

`uint32 (3)`
An *unsigned number*.

Complex types:

`pathname (32)`
A *string* that describes the pathname of a file. The file does not necessarily need to exist.

`ldap server (33)`
A *string* that describes an LDAP server in the format:

`hostname``:`*`port`*`:`*`username`*`:`*`password`*`:`*`base_dn`*

More types will be added in the future. Please see the *alt-type* field for information on how to cope with unknown types.

alt-type This field is identical to *type*, except that only the types `0` to `31` are allowed. The GUI is expected to present the user the option in the format specified by *type*. But if the argument type *type* is not supported by the GUI, it can still display the option in the more generic basic type *alt-type*. The GUI must support all the defined basic types to be able to display all options. More basic types may be added in future versions. If the GUI encounters a basic type it doesn't support, it should report an error and abort the operation.

argname This field is only defined for options with an argument type *type* that is not `0`. In this case it may contain a *percent-escaped* and *localised string* that gives a short name for the argument. The field may also be empty, though, in which case a short name is not known.

default This field is defined only for options. Its format is that of an *option argument* (See Section 5.3.2 [Format conventions], page 34, for details). If the default value is empty, then no default is known. Otherwise, the value specifies the default value for this option. Note that this field is also meaningful if the option itself does not take a real argument.

argdef This field is defined only for options for which the `optional arg` flag is set. If the `no arg desc` flag is not set, its format is that of an *option argument* (See Section 5.3.2 [Format conventions], page 34, for details). If the default value is empty, then no default is known. Otherwise, the value specifies the default value for this option. If the `no arg desc` flag is set, the field is either empty or contains a description of the effect of this option if no argument is given. Note that this field is also meaningful if the option itself does not take a real argument.

value This field is defined only for options. Its format is that of an *option argument*. If it is empty, then the option is not explicitely set in the current configuration, and the default applies (if any). Otherwise, it contains the current value of the option. Note that this field is also meaningful if the option itself does not take a real argument.

5.3.5 Changing options

The command `--change-options` *component* will attempt to change the options of the component *component* to the specified values. *component* must be the string in the field *name* in the output of the `--list-components` command. You have to provide the options that shall be changed in the following format on standard input:

name`:`*flags*`:`*new-value*

name This is the name of the option to change. *name* must be the string in the field *name* in the output of the `--list-options` command.

flags The flags field contains an *unsigned number*. Its value is the OR-wise combination of the following flag values:

`default (16)`
: If this flag is set, the option is deleted and the default value is used instead (if applicable).

new-value The new value for the option. This field is only defined if the `default` flag is not set. The format is that of an *option argument*. If it is empty (or the field is omitted), the default argument is used (only allowed if the argument is optional for this option). Otherwise, the option will be set to the specified value.

Examples:

To set the force option, which is of basic type `none (0)`:

```
$ echo 'force:0:1' | gpgconf --change-options dirmngr
```

To delete the force option:

```
$ echo 'force:16:' | gpgconf --change-options dirmngr
```

The `--runtime` option can influence when the changes take effect.

5.4 Generate an X.509 certificate request

This is a simple tool to interactivly generate a certificate request whicl will be printed to stdout.

`gpgsm-gencert.sh` is invoked as:

'`gpgsm-cencert.sh`'

5.5 Put a passphrase into the cache.

The `gpg-preset-passphrase` is a utility to seed the internal cache of a running `gpg-agent` with passphrases. It is mainly useful for unattended machines, where the usual `pinentry` tool may not be used and the passphrases for the to be used keys are given at machine startup.

Passphrases set with this utility don't expire unless the '`--forget`' option is used to explicitly clear them from the cache — or `gpg-agent` is either restarted or reloaded (by sending a SIGHUP to it). It is necessary to allow this passphrase presetting by starting `gpg-agent` with the '`--allow-preset-passphrase`'.

5.5.1 List of all commands and options.

`gpg-preset-passphrase` is invoked this way:

```
gpg-preset-passphrase [options] [command] keygrip
```

keygrip is a 40 character string of hexadecimal characters identifying the key for which the passphrase should be set or cleared. This keygrip is listed along with the key when running the command: `gpgsm --dump-secret-keys`. One of the following command options must be given:

`--preset` Preset a passphrase. This is what you usually will use. `gpg-preset-passphrase` will then read the passphrase from `stdin`.

`--forget` Flush the passphrase for the given keygrip from the cache.

The following additional options may be used:

`-v`
`--verbose`
: Output additional information while running.

`-P` *`string`*
`--passphrase` *`string`*
: Instead of reading the passphrase from `stdin`, use the supplied *string* as passphrase. Note that this makes the passphrase visible for other users.

6 Notes pertaining to certain OSes.

GnuPG has been developed on GNU/Linux systems and is know to work on almost all Free OSes. All modern POSIX systems should be supproted right now, however there are probably a lot of smaller glitches we need to fix first. The major problem areas are:

- For logging to sockets and other internal operations the `fopencookie` function (`funopen` under *BSD) is used. This is a very convient function which makes it possible to create outputs in a structures and easy maintainable way. The drawback however is that most proprietary OSes don't support this function. At g10 Code we have looked into several ways on how to overcome this limitation but no sufficiently easy and maintainable way has been found. Porting *glibc* to a general POSIX system is of course an option and would make writing portable software much easier; this it has not yet been done and the system administrator wouldneed to cope with the GNU specific admin things in addition to the generic ones of his system.

  We have now settled to use explicit stdio wrappers with a functionality similar to funopen. Although the code for this has already been written (*libestream*), we have not yet changed GnuPG to use it.

  This means that on systems not supporting either `funopen` or `fopencookie`, logging to a socket won't work, prompts are not formatted as pretty as theyshould be and `gpgsm`'s `LISTKEYS` Assuan command does not work.
- We are planning to use file descriptor passing for interprocess communication. This will allow us save a lot of resources and improve performance of certain operations a lot. Systems not supporting this won't gain these benefits but we try to keep them working the satndard way as it is done today.
- We require more or less full POSIX compatibility. This has been arround for 15 years now and thus we don't believe it makes sense to support non POSIX systems anymore. Well, we of course the usual workarounds for near POSIX systems well be applied.

  There is one exception of this rule: Systems based the Microsoft Windows API (called here *W32*) will be supported to some extend.

6.1 Microsoft Windows Notes

The port to Microsoft Windows based OSes is pretty new and has some limitations we might remove over time. Note, that we have not yet done any security audit and you should not use any valuable private key. In particular, **using it on a box with more than one user, might lead to a key compromise**.

Current limitations are:

- The `LISTKEYS` Assuan command of `gpgsm` is not supported. Using the command line options '`--list-keys`' or '`--list-secret-keys`' does however work.
- No support for CRL checks. By default the option '`--disable-crl-checks`' has been turned on and the log will show an appropriate warning message. The reason for this is that the separate CRL checking daemin (`dirmngr`) has not been ported to W32.
- `gpgconf` does not create backup files, so in case of trouble your configuration file might get lost.

- `watchgnupg` is not available. Logging to sockets is not possible.
- The periodical smartcard status checking done by `scdaemon` is not yet supported.
- Detached running of the gpg-agent is not directly supported. It needs to be started in a console and left alone then.

7 How to solve problems

Everyone knows that software often does not do what it should do and thus there is a need to track down problems. We call this debugging in a reminiscent to the moth jamming a relay in a Mark II box back in 1947.

Most of the probelsm a merely configuration and user problems but nevertheless there are the most annoying ones and reposnible for may gray hairs. We try to give some guidelines here on how to identify and solve the problem at hand.

7.1 Debugging Tools

The GnuPG distribution comes with a couple of tools, useful to help find and solving problems.

7.1.1 Scrutinizing a keybox file

A keybox is a file fomat used to store public keys along with meta information and indices. The commonly used one is the file '`pubring.kbx`' in the '`.gnupg`' directory. It contains all X.509 certificates as well as OpenPGP keys[1] .

When called the standard way, e.g.:

'`kbxutil ~/.gnupg/pubring.kbx`'

it lists all records (called BLOBS) with there meta-information in a human readable format.

To see statistics on the keybox in question, run it using

'`kbxutil --stats ~/.gnupg/pubring.kbx`'

and you get an output like:

```
Total number of blobs:       99
               header:        1
                empty:        0
              openpgp:        0
                 x509:       98
         non flagged:       81
      secret flagged:        0
   ephemeral flagged:       17
```

In this example you see that the keybox does not have any OpenPGP keys but contains 98 X.509 cerificates and a total of 17 keys or certificates are flagges as ephemeral, meaning that they are only temporary stored (cached) in the keybox and won't get listed using the usual commands provided by `gpgsm` or `gpg`. 81 certifcates are stored in a standard way and directly available from `gpgsm`.

[1] Well, OpenPGP keys are not implemented, `gpg` still used the keyring file '`pubring.gpg`'

7.2 Commonly Seen Problems

- Error code '`Not supported`' from Dirmngr

  Most likely the option '`enable-ocsp`' is active for gpgsm but Dirmngr's OCSP feature has not been enabled using '`allow-ocsp`' in '`dirmngr.conf`'.

- The Curses based Pinentry does not work

  The far most common reason for this is that the environment variable `GPG_TTY` has not been set correctly. Make sure that it has been set to a real tty devce and not just to '`/dev/tty`'; i.e. '`GPG_TTY=tty`' is plainly wrong; what you want is '`GPG_TTY=`tty``' — note the back ticks. Also make sure that this environment variable gets exported, that is you should follow up the setting with an '`export GPG_TTY`' (assuming a Bourne style shell). Even for GUI based Pinentries; you should have set `GPG_TTY`. See the section on installing the `gpg-agent` on how to do it.

Appendix A Description of the Assuan protocol.

The architecture of the modular GnuPG system is based on a couple of highly specialized modules which make up a network of client server communication. A common framework for intermodule communication is therefore needed and should be implemented in a library.

A.0.1 Goals

- Common framework for module communication
- Easy debugging
- Easy module testing
- Extendible
- Optional authentication and encryption facility
- Usable to access external hardware

A.0.2 Design criteria

- Client Server with back channel
- Use a mainly text based protocol
- Escape certain control characters
- Allow indefinite data length
- Request confidentiality for parts of the communication
- Dummy module should allow direct linking of client and server.
- Inline data or descriptor passing for bulk data
- No protection against DoS needed
- Subliminal channels are not an issue

A.0.3 Implementation

The implementation is line based with a maximum line size of 1000 octects. The default IPC mechanism are Unix Domain Sockets.

On a connect request the server responds either with an okay or an error status. For authentication check the server may send an Inquiry Response prior to the first Okay, it may also issue Status messages. The server must check that the client is allowed to connect, this is done by requesting the credentials for the peer and comparing them to those of the server. This avoids attacks based on wrong socket permissions.

It may choose to delay the first response in case of an error. The server never closes the connection - however the lower protocol may do so after some time of inactivity or when the connection is in an error state.

All textual messages are assumed to be in UTF-8 unless otherwise noted.

A.0.4 Server responses

`OK [<arbitary debugging information>]`
: Request was successful.

`ERR` *`errorcode`* `[<human readable error description>]`
: Request could not be fulfilled. The error codes are mostly application specific except for a few common ones.

`S` *`keyword`* `<status information depending on keyword>`
: Informational output by the server, still processing the request.

`# <string>`
: Comment line issued only for debugging purposes. Totally ignored.

`D <raw data>`
: Raw data returned to client. There must be exactly one space after the 'D'. The values for '%', CR and LF must be percent escaped; this is encoded as %25, %0D and %0A. Only uppercase letters should be used in the hexadecimal representation. Other characters may be percent escaped for easier debugging. All these Data lines are considered one data stream up to the OK or ERR response. Status and Inquiry Responses may be mixed with the Data lines.

`INQUIRE` *`keyword`*`> <parameters>`
: Server needs further information from the client. The client should answer with a command which is allowed after an inquiry. Note that the server does not confirm that client command but either continues processing or ends processing with an error status. Not all commands are allowed.

A client should only check the first letter of each line and then skip over to the next token (except for data lines where the raw data starts exactly after 2 bytes). Lines larger than 1000 bytes should be treated as a communication error. (The rationale for having a line length limit is to allow for easier multiplexing of multiple channels).

A.0.5 Client requests

The server waits for client requests after he sent an Okay or Error. The client should not issue a request in other cases with the exception of the CANCEL command.

`command` `<parameters>`

command is a one word string without preceding white space. Parameters are command specific, CR, LF and the percent signs should be percent escaped as described above. To send a backslash as the last character it should also be percent escaped. Percent escaping is allowed anywhere in the parameters but not in the command. The line ends with a CR, LF or just a LF.

Not yet implemented feature: If there is a need for a parameter list longer than the line length limit (1000 characters including command and CR, LF), the last character of the line (right before the CR/LF or LF) must be a non-escape encoded backslash. The following line is then expected to be a continuation of the line with the backslash replaced by a blank and the line ending removed.

```
D <raw data>
```

Raw data to the server. There must be exactly one space after the 'D'. The values for '%', CR and LF must be percent escaped; this is encoded as %25, %0D and %0A. Only uppercase letters should be used in the hexadecimal representation. Other characters may be percent escaped for easier debugging. All these Data lines are considered one data stream up to the OKAY or ERROR response. Status and Inquiry Responses may be mixed with the Data lines.

```
END
```

Lines beginning with a `#` or empty lines are ignored. This is useful to comment test scripts.

Although the commands are application specific, some of them are used by all protocols and partly directly supported by the Assuan library:

`CANCEL` his is the one special command which aborts the current request. it can be sent at any time and the server will stop its operation right before it would send the next response line (of any type).

`BYE` Close the connect, the server will reply with an `OK`.

`AUTH` Not yet specified as we don't implement it in the first phase. See my mail to gpa-dev on 2001-10-25 about the rationale for measurements against local attacks.

`RESET` Reset the connection but not any existing authentication. The server should release all resources associated with the connection.

`END` Used by a client to mark the end of raw data. The server may send END to indicate a partial end of data.

A.0.6 Error Codes

Here we keep a list of error codes used in any Assuan based protocol. The format is the string `ERR`, white space, the error number, white space, a textual description of the error.

```
100 Unknown Command
101 Not Implemented
301 certificate has been revoked [DirMngr]
302 no CRL known for this certificate [DirMngr]
303 CRL is too old and a new one could not be retrieved [DirMngr]
```

Appendix B GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

B.0.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

B.0.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions

for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you

indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program 'Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Contributors to GnuPG

The GnuPG project would like to thank its many contributors. Without them the project would not have been nearly as successful as it has been. Any omissions in this list are accidental. Feel free to contact the maintainer if you have been left out or some of your contributions are not listed.

David Shaw, Matthew Skala, Michael Roth, Niklas Hernaeus, Nils Ellmenreich, Rmi Guyomarch, Stefan Bellon, Timo Schulz and Werner Koch wrote the code. Birger Langkjer, Daniel Resare, Dokianakis Theofanis, Edmund GRIMLEY EVANS, Gal Quri, Gregory Steuck, Nagy Ferenc Lszl, Ivo Timmermans, Jacobo Tarri'o Barreiro, Janusz Aleksander Urbanowicz, Jedi Lin, Jouni Hiltunen, Laurentiu Buzdugan, Magda Procha'zkova', Michael Anckaert, Michal Majer, Marco d'Itri, Nilgun Belma Buguner, Pedro Morais, Tedi Heriyanto, Thiago Jung Bauermann, Rafael Caetano dos Santos, Toomas Soome, Urko Lusa, Walter Koch, Yosiaki IIDA did the official translations. Mike Ashley wrote and maintains the GNU Privacy Handbook. David Scribner is the current FAQ editor. Lorenzo Cappelletti maintains the web site.

The new modularized architecture of gnupg 1.9 as well as the X.509/CMS part has been developed as part of the gypten project. Direct contributors to this project are: Bernhard Herzog, who did extensive testing and tracked down a lot of bugs. Bernhard Reiter, who made sure that we met the specifications and the deadlines. He did extensive testing and came up with a lot of suggestions. Jan-Oliver Wagner made sure that we met the specifications and the deadlines. He also did extensive testing and came up with a lot of suggestions. Karl-Heinz Zimmer and Marc Mutz had to struggle with all the bugs and misconceptions while working on KDE integration. Marcus Brinkman extended GPGME, cleaned up the Assuan code and fixed bugs all over the place. Moritz Schulte took over Libgcrypt maintenance and developed it into a stable an useful library. Steffen Hansen had a hard time to write the dirmngr due to underspecified interfaces. Thomas Koester did extensive testing and tracked down a lot of bugs. Werner Koch designed the system and wrote most of the code.

The following people helped greatly by suggesting improvements, testing, fixing bugs, providing resources and doing other important tasks: Adam Mitchell, Albert Chin, Alec Habig, Allan Clark, Anand Kumria, Andreas Haumer, Anthony Mulcahy, Ariel T Glenn, Bob Mathews, Bodo Moeller, Brendan O'Dea, Brenno de Winter, Brian M. Carlson, Brian Moore, Brian Warner, Bryan Fullerton, Caskey L. Dickson, Cees van de Griend, Charles Levert, Chip Salzenberg, Chris Adams, Christian Biere, Christian Kurz, Christian von Roques, Christopher Oliver, Christian Recktenwald, Dan Winship, Daniel Eisenbud, Daniel Koening, Dave Dykstra, David C Niemi, David Champion, David Ellement, David Hallinan, David Hollenberg, David Mathog, David R. Bergstein, Detlef Lannert, Dimitri, Dirk Lattermann, Dirk Meyer, Disastry, Douglas Calvert, Ed Boraas, Edmund GRIMLEY EVANS, Edwin Woudt, Enzo Michelangeli, Ernst Molitor, Fabio Coatti, Felix von Leitner, fish stiqz, Florian Weimer, Francesco Potorti, Frank Donahoe, Frank Heckenbach, Frank Stajano, Frank Tobin, Gabriel Rosenkoetter, Gal Quri, Gene Carter, Geoff Keating, Georg Schwarz, Giampaolo Tomassoni, Gilbert Fernandes, Greg Louis, Greg Troxel, Gregory Steuck, Gregery Barton, Harald Denker, Holger Baust, Hendrik Buschkamp, Holger Schurig, Holger Smolinski, Holger Trapp, Hugh Daniel, Huy Le, Ian McKellar, Ivo Timmermans, Jan Krueger, Jan Niehusmann, Janusz A. Urbanowicz, James Troup, Jean-loup Gailly, Jeff

Long, Jeffery Von Ronne, Jens Bachem, Jeroen C. van Gelderen, J Horacio MG, J. Michael Ashley, Jim Bauer, Jim Small, Joachim Backes, Joe Rhett, John A. Martin, Johnny Teveen, Jrg Schilling, Jos Backus, Joseph Walton, Juan F. Codagnone, Jun Kuriyama, Kahil D. Jallad, Karl Fogel, Karsten Thygesen, Katsuhiro Kondou, Kazu Yamamoto, Keith Clayton, Kevin Ryde, Klaus Singvogel, Kurt Garloff, Lars Kellogg-Stedman, L. Sassaman, M Taylor, Marcel Waldvogel, Marco d'Itri, Marco Parrone, Marcus Brinkmann, Mark Adler, Mark Elbrecht, Mark Pettit, Markus Friedl, Martin Kahlert, Martin Hamilton, Martin Schulte, Matt Kraai, Matthew Skala, Matthew Wilcox, Matthias Urlichs, Max Valianskiy, Michael Engels, Michael Fischer v. Mollard, Michael Roth, Michael Sobolev, Michael Tokarev, Nicolas Graner, Mike McEwan, Neal H Walfield, Nelson H. F. Beebe, NIIBE Yutaka, Niklas Hernaeus, Nimrod Zimerman, N J Doye, Oliver Haakert, Oskari Jskelinen, Pascal Scheffers, Paul D. Smith, Per Cederqvist, Phil Blundell, Philippe Laliberte, Peter Fales, Peter Gutmann, Peter Marschall, Peter Valchev, Piotr Krukowiecki, QingLong, Ralph Gillen, Rat, Reinhard Wobst, Rmi Guyomarch, Reuben Sumner, Richard Outerbridge, Robert Joop, Roddy Strachan, Roger Sondermann, Roland Rosenfeld, Roman Pavlik, Ross Golder, Ryan Malayter, Sam Roberts, Sami Tolvanen, Sean MacLennan, Sebastian Klemke, Serge Munhoven, SL Baur, Stefan Bellon, Dr.Stefan.Dalibor, Stefan Karrmann, Stefan Keller, Steffen Ullrich, Steffen Zahn, Steven Bakker, Steven Murdoch, Susanne Schultz, Ted Cabeen, Thiago Jung Bauermann, Thijmen Klok, Thomas Roessler, Tim Mooney, Timo Schulz, Todd Vierling, TOGAWA Satoshi, Tom Spindler, Tom Zerucha, Tomas Fasth, Tommi Komulainen, Thomas Klausner, Tomasz Kozlowski, Thomas Mikkelsen, Ulf Mller, Urko Lusa, Vincent P. Broman, Volker Quetschke, W Lewis, Walter Hofmann, Walter Koch, Wayne Chapeskie, Wim Vandeputte, Winona Brown, Yosiaki IIDA, Yoshihiro Kajiki and Gerlinde Klaes.

This software has been made possible by the previous work of Chris Wedgwood, Jeanloup Gailly, Jon Callas, Mark Adler, Martin Hellmann Paul Kendall, Philip R. Zimmermann, Peter Gutmann, Philip A. Nelson, Taher ElGamal, Torbjorn Granlund, Whitfield Diffie, some unknown NSA mathematicians and all the folks who have worked hard to create complete and free operating systems.

And finally we'd like to thank everyone who uses these tools, submits bug reports and generally reminds us why we're doing this work in the first place.

Glossary

'ARL' The *Authority Revocation List* is technical identical to a CRL but used for CAs and not for end user certificates.

'CRL' The *Certificate Revocation List* is a list containing certificates revoked by the issuer.

'Keygrip' This term is used by GnuPG to describe a 20 byte hash value used to identify a certain key without referencing to a concrete protocol. It is used internally to access a private key. Usually it is shown and entered as a 40 character hexadecimal formatted string.

'OCSP' The *Online Certificate Status Protocol* is used as an alternative to a CRL. It is described in RFC 2560.

Option Index

O

P

Q

S

T

U

V

W

Index