Package: gpg (via r-universe)

October 31, 2024

Type Package

Title GNU Privacy Guard for R

Version 1.3.0

Description Bindings to GnuPG for working with OpenGPG (RFC4880) cryptographic methods. Includes utilities for public key encryption, creating and verifying digital signatures, and managing your local keyring. Some functionality depends on the version of GnuPG that is installed on the system. On Windows this package can be used together with 'GPG4Win' which provides a GUI for managing keys and entering passphrases.

License MIT + file LICENSE

SystemRequirements GPGME: libgpgme-dev (deb), gpgme-devel (rpm) gpgme (brew). On Linux 'haveged' is recommended for generating entropy when using the GPG key generator.

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Imports curl, askpass

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/jeroen/gpg

BugReports https://github.com/jeroen/gpg/issues

Encoding UTF-8

Repository https://jeroen.r-universe.dev

RemoteUrl https://github.com/jeroen/gpg

RemoteRef HEAD

RemoteSha 974c0cc91284454f249941e55c8aaef880886bb9

Contents

`gpg_encrypt` *Encryption*

Description

Encrypt or decrypt a message using the public key from the `receiver`. Optionally the message can be signed using the private key of the sender.

Usage

```
gpg_encrypt(data, receiver, signer = NULL)

gpg_decrypt(data, verify = TRUE, as_text = TRUE)
```

Arguments

`data`	path or raw vector with data to encrypt / decrypt
`receiver`	key id(s) or fingerprint(s) for recepient(s)
`signer`	(optional) key id(s) or fingerprint(s) for the sender(s) to sign the message
`verify`	automatically checks that all signatures (if any) can be verified and raises an error otherwise
`as_text`	convert output to text. Set to FALSE if you expect binary data.

See Also

Other gpg: `gpg_keygen()`, `gpg_keys`, `gpg_sign()`

gpg_keygen *GPG key generation*

Description

Generates a new standard private-public keypair. This function is mostly for testing purposes. Use the gpg `--gen-key` command line utility to generate an official GPG key with custom fields and options.

Usage

```
gpg_keygen(name, email, passphrase = NULL)
```

Arguments

`name`	value for the `Name-Real` field
`email`	value for the `Name-Email` field
`passphrase`	(optional) protect with a passphrase

References

GPG manual section on Unattended key generation.

See Also

Other gpg: `gpg_encrypt()`, `gpg_keys`, `gpg_sign()`

gpg_keys *GPG keyring management*

Description

Signing or encrypting with GPG require that the keys are stored in your personal keyring. Use gpg_version to see which keyring (home dir) you are using. Also see gpg_keygen for generating a new key.

Usage

```
gpg_import(file)

gpg_recv(id, search = NULL, keyserver = NULL)

gpg_send(id, keyserver = NULL)

gpg_delete(id, secret = FALSE)
```

```
gpg_export(id, secret = FALSE)

gpg_list_keys(search = "", secret = FALSE)

gpg_list_signatures(id)
```

Arguments

file	path to the key file or raw vector with key data
id	unique ID of the pubkey to import (starts with 0x). Alternatively you can specify a search string.
search	string with name or email address to match the key info.
keyserver	address of http keyserver. Default behavior is to try several commonly used servers (MIT, Ubuntu, GnuPG, Surfnet)
secret	set to TRUE to list/export/delete private (secret) keys

See Also

Other gpg: gpg_encrypt(), gpg_keygen(), gpg_sign()

Examples

```
## Not run:
# Submit key to a specific key server.
gpg_send("87CC261267801A17", "https://keys.openpgp.org")
# Submit key to many key servers.
gpg_send("87CC261267801A17")

## End(Not run)
```

---

gpg_restart *Manage the GPG engine*

Description

Use gpg_restart() to find the gpg program and home directory (which contains configuration and keychains). Usually the default should be fine and you do not need to run this function manually.

Usage

```
gpg_restart(home = NULL, path = NULL, debug = "none", silent = FALSE)

gpg_version(silent = FALSE)

gpg_info()

gpg_options()
```

Arguments

home	path to your GPG configuration directory (including keyrings)
path	location of gpg or gpg2 or gpgconf or (on windows) gpgme-w32spawn.exe
debug	debugging level, integer between 1 and 9
silent	suppress output of gpg --version

Details

Use gpg_info() to get your current engine settings. The gpg_version() function simply calls gpg --version to see some verbose output about the gpg executable.

gpg_options reads options in the GnuPG configuration file, which is stored by default in ~/.gnupg/gpg.conf. Note that changing options might affect other software using GnuPG.

Examples

```
gpg_version()
gpg_info()
```

---

gpg_sign *PGP Signatures*

Description

Utilities to create and verify PGP signatures.

Usage

```
gpg_verify(signature, data = NULL, error = TRUE)

gpg_sign(data, signer = NULL, mode = c("detach", "normal", "clear"))
```

Arguments

signature	path or raw vector for the gpg signature (contains the PGP SIGNATURE block)
data	path or raw vector with data to sign or verify. In gpg_verify this should be NULL if signature is not detached (i.e. clear or normal signature)
error	raise an error if verification fails because you do not have the signer public key in your keyring.
signer	(optional) vector with key ID's to use for signing. If NULL, GPG tries the user default private key.
mode	use normal to create a full OpenPGP message containing both data and signature or clear append the signature to the clear-text data (for email messages). Default detach only returns the signature itself.

See Also

Other gpg: gpg_encrypt(), gpg_keygen(), gpg_keys

Examples

```
## Not run:
# This requires you have the Debian master key in your keyring
msg <- tempfile()
sig <- tempfile()
download.file("http://http.us.debian.org/debian/dists/stable/Release", msg)
download.file("http://http.us.debian.org/debian/dists/stable/Release.gpg", sig)
gpg_verify(sig, msg, error = FALSE)

## End(Not run)
```

---

pinentry *Password Entry*

Description

Function to prompt the user for a password to read a protected private key.

Usage

```
pinentry(prompt = "Enter your GPG passphrase:")
```

Arguments

prompt	the string printed when prompting the user for input.

Details

If available, this function calls the GnuPG `pinentry` program. However this only works in a terminal. Therefore the IDE can provide a custom password entry widget by setting the `askpass` option. If no such option is specified we default to `readline`.

Index