Package 'safer'

August 29, 2026

Type Package

Title Encrypt and Decrypt Strings, R Objects and Files

Version 0.2.3

Description A consistent interface to encrypt and decrypt strings, R objects and files using symmetric and asymmetric key encryption.

Imports sodium (>= 0.4), base64enc (>= 0.1.3), base64url (>= 1.4), assertthat (>= 0.1)

Suggests openssl (>= 1.2.0), testthat, knitr, rmarkdown

Depends R (>= 3.3.0),

License GPL-3

URL https://github.com/talegari/safer

BugReports https://github.com/talegari/safer/issues

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Komala Sheshachala Srikanth [aut, cre]

Maintainer Komala Sheshachala Srikanth <sri.teach@gmail.com>

Repository CRAN

Date/Publication 2026-08-29 15:20:02 UTC

Contents

safer-package 2
decrypt_file 2
decrypt_object 4
decrypt_string 5
encrypt_file 6
encrypt_object 7
encrypt_string 8
keypair 9

read_keypair 10
retrieve_object 11
save_object 12

Index **14**

`safer-package` *safer*

Description

A consistent interface to encrypt/decrypt strings, R objects, files. Alternatives for base R functions 'serialize/unserialize', 'save/load' are provided.

The following functions are provided:

encrypt_string/decrypt_string: `encrypt_string` encrypts a string as a string and `decrypt_string` decrypts the encrypted string(encrypted using `encrypt_string`)

encrypt_object/decrypt object: `encrypt_object` encrypts a R object as a raw object or a string and `decrypt_object` decrypts a raw object or a string(encrypted by `encrypt_object`)

encrypt_file/decrypt_file: `encrypt_file` encrypts file into another binary or ascii file. `decrypt_file`) decrypts a file (encrypted by `encrypt_file`)

save_object/retrieve_object: `save_object` encrypts a R object to raw or text connection or a file. `retrieve_object` decrypts a raw or a text connection or a file (encrypted by `save_object`.)

Author(s)

Maintainer: Komala Sheshachala Srikanth <sri.teach@gmail.com>

See Also

Useful links:

- https://github.com/talegari/safer
- Report bugs at https://github.com/talegari/safer/issues

`decrypt_file` *Decrypt a file*

Description

`encrypt_file`) encrypts a file as a binary or a ascii file. `decrypt_file`) decrypts a text or a binary file (encrypted by `encrypt_file`)

Usage

```
decrypt_file(infile, key = "pass", pkey = NULL, ascii = FALSE, outfile)
```

Arguments

infile	file to be decrypted
key	For symmetric decryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric decryption, both 'key' (private key of the decrypter) and 'pkey' (public key of the encrypter) should be raw objects.
pkey	See 'key'
ascii	TRUE if the outfile is to be decrypted as a ascii file. Default is FALSE
outfile	Non-existant file where the decrypted output is to be written

Value

An invisible TRUE

Examples

```
# symmetric case:
write.table(iris, "iris.csv")
all(
  encrypt_file("iris.csv", outfile = "iris_encrypted.bin")
  , file.exists("iris_encrypted.bin")
  , decrypt_file("iris_encrypted.bin", outfile = "iris_2.csv")
  , file.exists("iris_2.csv")
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
  , unlink("iris_encrypted.bin") == 0
)

write.table(iris, "iris.csv")
all(
  encrypt_file("iris.csv", outfile = "iris_encrypted.txt", ascii = TRUE)
  , file.exists("iris_encrypted.txt")
  , decrypt_file("iris_encrypted.txt", outfile = "iris_2.csv", ascii = TRUE)
  , file.exists("iris_2.csv")
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
  , unlink("iris_encrypted.txt") == 0
)

# asymmetric case:
alice <- keypair()
bob   <- keypair()
write.table(iris, "iris.csv")
all(
 encrypt_file("iris.csv", alice$private_key, bob$public_key, outfile = "iris_encrypted.bin")
  , file.exists("iris_encrypted.bin")
 , decrypt_file("iris_encrypted.bin", bob$private_key, alice$public_key, outfile = "iris_2.csv")
  , file.exists("iris_2.csv")
```

```
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
  , unlink("iris_encrypted.bin") == 0
)
```

---

decrypt_object *Decrypt a object*

Description

encrypt_object encrypts a R object as a raw object or a string and decrypt_object decrypts a raw object or a string(encrypted by encrypt_object)

Usage

```
decrypt_object(object, key = "pass", pkey = NULL)
```

Arguments

object	Object to be decrypted
key	For symmetric decryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric decryption, both 'key' (private key of the decrypter) and 'pkey' (public key of the encrypter) should be raw objects.
pkey	See 'key'

Value

A raw object if ascii is FALSE. A string if ascii is TRUE.

Examples

```
# symmetric case:
temp <- encrypt_object(1:3)
all(
  is.raw(temp)
  , decrypt_object(temp) == 1:3)

temp <- encrypt_object(iris, ascii = TRUE)
all(
  is.character(temp)
  , decrypt_object(temp) == iris
  , identical(decrypt_object(temp), iris))
rm(temp)

# asymmetric case:
alice <- keypair()
```

```
bob   <- keypair()
temp  <- encrypt_object(1:10, alice$private_key, bob$public_key)
temp2 <- decrypt_object(temp, bob$private_key, alice$public_key)
identical(1:10, temp2)
```

---

decrypt_string *Decrypt a string or a raw vector*

Description

encrypt_string encrypts a string as a string or a raw vector and decrypt_string decrypts the encrypted string or a raw vector (encrypted using encrypt_string)

Usage

```
decrypt_string(string, key = "pass", pkey = NULL, url = FALSE)
```

Arguments

string	A string(character vector of length 1) without embedded NULL to be encrypted. or a raw vector.
key	For symmetric decryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric decryption, both 'key' (private key of the decrypter) and 'pkey' (public key of the encrypter) should be raw objects.
pkey	See 'key'
url	(flag, default: FALSE) Whether the encoded string was generated url friendly.

Value

decrypted string

Examples

```
# symmetric case:
temp <- encrypt_string("hello, how are you", key = "secret")
all(
  is.character(temp)
  , decrypt_string(temp, "secret") == "hello, how are you"
  , class(try(decrypt_string(temp, "nopass"), silent = TRUE)) == "try-error"
  )

# string encoded as raw
res <- encrypt_string("tatvamasi", ascii = FALSE)
res

isTRUE(identical(decrypt_string(res), "tatvamasi"))
```

```
# asymmetric case:
alice <- keypair()
bob   <- keypair()
temp  <- encrypt_string("hello asymmetric", alice$private_key, bob$public_key)
temp2 <- decrypt_string(temp, bob$private_key, alice$public_key)
identical("hello asymmetric", temp2)
```

`encrypt_file` *Encrypt a file*

Description

`encrypt_file`) encrypts a file as a binary or a ascii file. `decrypt_file`) decrypts a text or a binary file (encrypted by `encrypt_file`)

Usage

```
encrypt_file(infile, key = "pass", pkey = NULL, ascii = FALSE, outfile)
```

Arguments

`infile`	file to be encrypted
`key`	For symmetric encryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric encryption, both 'key' (private key of the encrypter) and 'pkey' (public key of the decrypter) should be raw objects.
`pkey`	See 'key'
`ascii`	`TRUE` if the outfile is to be encrypted as a ascii file. Default is `FALSE`
`outfile`	Non-existant file where the encrypted output is to be written

Value

An invisible TRUE

Examples

```
# symmetric case:
write.table(iris, "iris.csv")
all(
  encrypt_file("iris.csv", outfile = "iris_encrypted.bin")
  , file.exists("iris_encrypted.bin")
  , decrypt_file("iris_encrypted.bin", outfile = "iris_2.csv")
  , file.exists("iris_2.csv")
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
```

```
  , unlink("iris_encrypted.bin") == 0
)

write.table(iris, "iris.csv")
all(
  encrypt_file("iris.csv", outfile = "iris_encrypted.txt", ascii = TRUE)
  , file.exists("iris_encrypted.txt")
  , decrypt_file("iris_encrypted.txt", outfile = "iris_2.csv", ascii = TRUE)
  , file.exists("iris_2.csv")
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
  , unlink("iris_encrypted.txt") == 0
)

# asymmetric case:
alice <- keypair()
bob   <- keypair()
write.table(iris, "iris.csv")
all(
 encrypt_file("iris.csv", alice$private_key, bob$public_key, outfile = "iris_encrypted.bin")
  , file.exists("iris_encrypted.bin")
 , decrypt_file("iris_encrypted.bin", bob$private_key, alice$public_key, outfile = "iris_2.csv")
  , file.exists("iris_2.csv")
  , tools::md5sum("iris_2.csv") == tools::md5sum("iris.csv")
  , unlink("iris.csv") == 0
  , unlink("iris_2.csv") == 0
  , unlink("iris_encrypted.bin") == 0
)
```

`encrypt_object` *Encrypt a object*

Description

`encrypt_object` encrypts a object as a raw object or a string and `decrypt_object` decrypts a raw object or a string(encrypted by `encrypt_object`)

Usage

```
encrypt_object(object, key = "pass", pkey = NULL, ascii = FALSE)
```

Arguments

`object`	Object to be encrypted
`key`	For symmetric encryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric encryption, both 'key' (private key of the encrypter) and 'pkey' (public key of the decrypter) should be raw objects.

pkey	See 'key'
ascii	TRUE if the object is to be encrypted as a string. Default is FALSE

Value

A raw object if ascii is FALSE. A string if ascii is TRUE.

Examples

```
# symmetric case:
temp <- encrypt_object(1:3)
all(
  is.raw(temp)
  , decrypt_object(temp) == 1:3)

temp <- encrypt_object(iris, ascii = TRUE)
all(
  is.character(temp)
  , decrypt_object(temp) == iris
  , identical(decrypt_object(temp), iris))
rm(temp)

# asymmetric case:
alice <- keypair()
bob   <- keypair()
temp  <- encrypt_object(1:10, alice$private_key, bob$public_key)
temp2 <- decrypt_object(temp, bob$private_key, alice$public_key)
identical(1:10, temp2)
```

---

encrypt_string *Encrypt a string*

Description

encrypt_string encrypts a string as a string or a raw vector and decrypt_string decrypts the encrypted string or a raw vector (encrypted using encrypt_string)

Usage

```
encrypt_string(string, key = "pass", pkey = NULL, ascii = TRUE, url = FALSE)
```

Arguments

string	A string(character vector of length 1) without embedded NULL to be encrypted or a raw vector.
key	For symmetric encryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric encryption, both 'key' (private key of the encrypter) and 'pkey' (public key of the decrypter) should be raw objects.

pkey	See 'key'
ascii	(flag) When TRUE (default), the output is a string after base64 encoding. Else, the output is a raw vector.
url	(flag, default: FALSE) Whether the encoded string has to be url friendly.

Value

An encrypted string or a raw vector.

Examples

```
# symmetric case:
temp <- encrypt_string("hello, how are you", key = "secret")
all(
  is.character(temp)
  , decrypt_string(temp, "secret") == "hello, how are you"
  , class(try(decrypt_string(temp, "nopass"), silent = TRUE)) == "try-error"
  )

# string encoded as raw
res <- encrypt_string("tatvamasi", ascii = FALSE)
res

isTRUE(identical(decrypt_string(res), "tatvamasi"))

# asymmetric case:
alice <- keypair()
bob   <- keypair()
temp  <- encrypt_string("hello asymmetric", alice$private_key, bob$public_key)
temp2 <- decrypt_string(temp, bob$private_key, alice$public_key)
identical("hello asymmetric", temp2)
```

---

keypair *Generate a public key and private key pair*

Description

Using sodium's 'keygen' and 'pubkey' based on curve25519

Usage

```
keypair(seed = NULL)
```

Arguments

seed	A raw object. If NULL, a randon seed will be chosen.

Value

A list with:

- public_key: A raw object
- private_key: A raw object
- seed: A raw object

Examples

```
temp <- keypair()
str(temp)
```

`read_keypair` *Read an existing public and private Ed25519 key pair*

Description

Reads Ed25519 keys from files using the openssl package. The public key is automatically derived from the private key file.

Usage

```
read_keypair(private_key_path)
```

Arguments

`private_key_path`
: Path to the private key file.

Value

A list with:

- public_key: A raw object
- private_key: A raw object
- seed: NULL

Examples

```
# Generate a keypair and save to disk using openssl
key = openssl::ed25519_keygen()
tmp = tempfile()
openssl::write_pem(key, tmp)

# Read it back using read_keypair
keys = read_keypair(tmp)
str(keys)
```

```
# Validate that keys are same
identical(keys$private_key, key$data)
identical(keys$public_key, key$pubkey$data)
```

`retrieve_object` *Retrieve an object from a connection(or a file)*

Description

`save_object` encrypts a R object to raw or text connection or a file. `retrieve_object` decrypts a raw or a text connection or a file (encrypted by `save_object`). Note that `retrieve_object` returns the object.

Usage

```
retrieve_object(conn, key = "pass", pkey = NULL, ascii = FALSE)
```

Arguments

`conn`	A connection or a file where the decrypted content is written. If `ascii` is TRUE, an decrypted text is written to the connection. Else, when `ascii` is FALSE(default), a raw object is written to the connection
`key`	For symmetric decryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric decryption, both 'key' (private key of the decrypter) and 'pkey' (public key of the encrypter) should be raw objects.
`pkey`	See 'key'
`ascii`	TRUE, if the encrypted output is a string(written to the text connection). FALSE, if the encrypted output is a raw object(written to the raw connection)

Value

An invisible TRUE

Examples

```
# symmetric case:
all(
  save_object(iris, conn = "iris_safer.bin")
  , identical(retrieve_object(conn = "iris_safer.bin"), iris)
  , unlink("iris_safer.bin") == 0
)

all(
  save_object(iris, conn = "iris_safer_2.txt", ascii = TRUE)
  , identical(retrieve_object(conn = "iris_safer_2.txt", ascii = TRUE), iris)
  , unlink("iris_safer_2.txt") == 0
)
```

```
# asymmetric case:
alice <- keypair()
bob   <- keypair()
all(
  save_object(iris, alice$private_key, bob$public_key, conn = "iris_safer.bin")
 , identical(retrieve_object(conn = "iris_safer.bin", bob$private_key, alice$public_key), iris)
  , unlink("iris_safer.bin") == 0
)
```

`save_object` *Save an object to a connection(or a file)*

Description

`save_object` encrypts a R object to raw or text connection or a file. `retrieve_object` decrypts a raw or a text connection or a file (encrypted by `save_object`). Note that `retrieve_object` returns the object.

Usage

```
save_object(object, key = "pass", pkey = NULL, ascii = FALSE, conn)
```

Arguments

`object`	A R object to be encrypted
`key`	For symmetric encryption, 'pkey' should be NULL (default) and 'key' can be either a string (Default is 'pass') or a raw object. For asymmetric encryption, both 'key' (private key of the encrypter) and 'pkey' (public key of the decrypter) should be raw objects.
`pkey`	See 'key'
`ascii`	TRUE, if the encrypted output is a string(written to the text connection). FALSE, if the encrypted output is a raw object(written to the raw connection)
`conn`	A connection or a file where the encrypted content is written. If `ascii` is TRUE, an encrypted text is written to the connection. Else, when `ascii` is FALSE(default), a raw object is written to the connection

Value

An invisible TRUE

Examples

```
# symmetric case:
all(
  save_object(iris, conn = "iris_safer.bin")
  , identical(retrieve_object(conn = "iris_safer.bin"), iris)
  , unlink("iris_safer.bin") == 0
)

all(
  save_object(iris, conn = "iris_safer_2.txt", ascii = TRUE)
  , identical(retrieve_object(conn = "iris_safer_2.txt", ascii = TRUE), iris)
  , unlink("iris_safer_2.txt") == 0
)

# asymmetric case:
alice <- keypair()
bob   <- keypair()
all(
  save_object(iris, alice$private_key, bob$public_key, conn = "iris_safer.bin")
 , identical(retrieve_object(conn = "iris_safer.bin", bob$private_key, alice$public_key), iris)
  , unlink("iris_safer.bin") == 0
)
```

Index

decrypt_file, 2
decrypt_object, 4
decrypt_string, 5

encrypt_file, 6
encrypt_object, 7
encrypt_string, 8

keypair, 9

read_keypair, 10
retrieve_object, 11

safer *(*safer-package*)*, 2
safer-package, 2
save_object, 12