

Package ‘encryptr’

July 22, 2025

Type Package

Title Easily Encrypt and Decrypt Data Frame/Tibble Columns or Files
using RSA Public/Private Keys

Version 0.1.4

Maintainer Ewen Harrison <ewen.harrison@ed.ac.uk>

Description It is important to ensure that sensitive data is protected.
This straightforward package is aimed at the end-user.
Strong RSA encryption using a public/private key pair is used to encrypt data frame or tibble columns.
A public key can be shared to allow others to encrypt data to be sent to you.
This is particularly aimed a healthcare settings so patient data can be pseudonymised.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

BugReports <https://github.com/SurgicalInformatics/encryptr/issues>

URL <https://github.com/SurgicalInformatics/encryptr>

Imports dplyr, openssl, purrr, readr, rlang

RoxygenNote 7.2.3

Suggests testthat, withr

NeedsCompilation no

Author Cameron Fairfield [aut],
Riinu Ots [aut],
Stephen Knight [aut],
Tom Drake [aut],
Ewen Harrison [aut, cre]

Repository CRAN

Date/Publication 2025-05-13 09:40:02 UTC

Contents

encryptr-package	2
decrypt	2
decrypt_file	3
decrypt_vec	4
encrypt	5
encrypt_file	6
encrypt_vec	7
genkeys	8
gp	9
Index	10

encryptr-package	<i>encryptr: Encrypt and decrypt data frame or tibble columns using the strong RSA public/private keys.</i>
------------------	---

Description

Use openssl to encrypt and decrypt data frame or tibble columns.

encryptr **key generation**

[genkeys](#),

encryptr **encrypt/decrypt**

[encrypt](#), [decrypt](#)

decrypt	<i>Decrypt a data frame or tibble column using an RSA public/private key</i>
---------	--

Description

Decrypt a data frame or tibble column using an RSA public/private key

Usage

```
decrypt(  
  .data,  
  ...,  
  private_key_path = "id_rsa",  
  lookup_object = NULL,  
  lookup_path = NULL  
)
```

Arguments

.data A data frame or tibble.

... The unquoted names of columns to decrypt.

private_key_path Character. A quoted path to an RSA private key created using [genkeys](#).

lookup_object An unquote name of a lookup object in the current environment created using `link{encrypt}`.

lookup_path Character. A quoted path to an RSA private key created using [encrypt](#).

Value

The original dataframe or tibble with the specified columns decrypted.

Examples

```
#' This will run:
# genkeys()
# gp_encrypt = gp %>%
#   select(-c(name, address1, address2, address3)) %>%
#   encrypt(postcode, telephone)
# gp_encrypt %>%
#   decrypt(postcode, telephone)

## Not run:
# For CRAN and testing:
library(dplyr)
temp_dir = tempdir()
genkeys(file.path(temp_dir, "id_rsa")) # temp directory for testing only
gp_encrypt = gp %>%
  select(-c(name, address1, address2, address3)) %>%
  encrypt(postcode, telephone, public_key_path = file.path(temp_dir, "id_rsa.pub"))
gp_encrypt %>%
  decrypt(postcode, telephone, private_key_path = file.path(temp_dir, "id_rsa"))

## End(Not run)
```

decrypt_file	<i>Decrypt a file</i>
--------------	-----------------------

Description

See [encrypt_file](#) for details.

Usage

```
decrypt_file(.path, file_name = NULL, private_key_path = "id_rsa")
```

Arguments

.path Quoted path to file to encrypt.
 file_name Optional new name for unencrypted file.
 private_key_path
 Quoted path to private key, created with [genkeys](#).

Value

The decrypted file is saved with optional file name.

Examples

```
#' # For CRAN and testing:
## Not run:
# Run only once in decrypt_file example
temp_dir = tempdir() # temp directory for testing only
genkeys(file.path(temp_dir, "id_rsa"))
write.csv(gp, file.path(temp_dir, "gp.csv"))
encrypt_file(file.path(temp_dir, "gp.csv"), public_key_path = file.path(temp_dir, "id_rsa.pub"))

## End(Not run)

## Not run:
# For CRAN and testing:
temp_dir = tempdir() # temp directory for testing only
genkeys(file.path(temp_dir, "id_rsa4"))
write.csv(gp, file.path(temp_dir, "gp.csv"))
encrypt_file(file.path(temp_dir, "gp.csv"), public_key_path = file.path(temp_dir, "id_rsa4.pub"))
decrypt_file(file.path(temp_dir, "gp.csv.encrypt.bin"),
  private_key_path = file.path(temp_dir, "id_rsa4"),
  file_name = "file.path(temp_dir, gp2.csv)")

## End(Not run)
```

 decrypt_vec

Decrypt ciphertext using an RSA public/private key

Description

Not usually called directly. Password for private key required.

Usage

```
decrypt_vec(.data, private_key_path = "id_rsa")
```

Arguments

.data A vector of ciphertexts created using [encrypt](#).
 private_key_path
 Character. A quoted path to an RSA private key created using [genkeys](#).

Value

A character vector.

Examples

```
## Not run:
hospital_number = c("1010761111", "2010761212")
genkeys(file.path(tempdir(), "id_rsa") # temp directory for testing only
hospital_number_encrypted = encrypt_char(hospital_number)
decrypt_vec(hospital_number_encrypted)

## End(Not run)
```

encrypt	<i>Encrypt a data frame or tibble column using an RSA public/private key</i>
---------	--

Description

Encrypt a data frame or tibble column using an RSA public/private key

Usage

```
encrypt(
  .data,
  ...,
  public_key_path = "id_rsa.pub",
  lookup = FALSE,
  lookup_name = "lookup",
  write_lookup = TRUE
)
```

Arguments

.data	A data frame or tibble.
...	The unquoted names of columns to encrypt.
public_key_path	Character. A quoted path to an RSA public key created using genkeys .
lookup	Logical. Whether to substitute the encrypted columns for key-column of integers.
lookup_name	Character. A quoted name to give lookup table and file.
write_lookup	Logical. Write a lookup table as a .csv file.

Value

The original dataframe or tibble with the specified columns encrypted.

Examples

```
# This will run:
# genkeys()
# gp_encrypt = gp %>%
#   select(-c(name, address1, address2, address3)) %>%
#   encrypt(postcode, telephone)

# For CRAN and testing:
library(dplyr)
temp_dir = tempdir()
genkeys(file.path(temp_dir, "id_rsa2")) # temp directory for testing only
gp_encrypt = gp %>%
  select(-c(name, address1, address2, address3)) %>%
  encrypt(postcode, telephone, public_key_path = file.path(temp_dir, "id_rsa2.pub"))
```

encrypt_file	<i>Encrypt a file</i>
--------------	-----------------------

Description

Encryption and decryption with asymmetric keys is computationally expensive. This is how [encrypt](#) works, in order to allow each piece of data in a data frame to be decrypted without compromise of the whole data frame. This works on the presumption that each cell contains less than 245 bytes of data.

Usage

```
encrypt_file(.path, crypt_file_name = NULL, public_key_path = "id_rsa.pub")
```

Arguments

.path	Quoted path to file to encrypt.
crypt_file_name	Optional new name to give encrypted file. Must end with ".encrypt.bin".
public_key_path	Quoted path to public key, created with genkeys .

Details

File encryption requires a different approach as files are often larger in size. This function encrypts a file using a symmetric "session" key and the AES-256 cipher. This key is itself then encrypted using a public key generated using [genkeys](#). In OpenSSL this combination is referred to as an envelope.

Value

The encrypted file is saved.

Examples

```
# This will run:
# Create example file to encrypt
# write.csv(gp, "gp.csv")
# genkeys()
# encrypt_file("gp.csv")

# For CRAN and testing:
## Not run:
# Run only once in decrypt_file example
temp_dir = tempdir() # temp directory for testing only
genkeys(file.path(temp_dir, "id_rsa"))
write.csv(gp, file.path(temp_dir, "gp.csv"))
encrypt_file(file.path(temp_dir, "gp.csv"), public_key_path = file.path(temp_dir, "id_rsa.pub"))

## End(Not run)
```

encrypt_vec

Encrypt a character vector using an RSA public/private key

Description

Not usually called directly.

Usage

```
encrypt_vec(.data, public_key_path = "id_rsa.pub")
```

Arguments

`.data` A vector, which if not a character vector is coerced to one.

`public_key_path` Character. A quoted path to an RSA public key created using [genkeys](#).

Value

A vector of ciphertexts.

Examples

```
## Not run:
hospital_number = c("1010761111", "2010761212")
encrypt_vec(hospital_number)

## End(Not run)
```

`genkeys`*Create and write RSA private and public keys*

Description

The first step for the `encryptr` workflow is to create a pair of encryption keys. This uses the `openssl` package. The public key is used to encrypt information and can be shared. The private key allows decryption of the encrypted information. It requires a password to be set. This password cannot be recovered if lost. If the file is lost or overwritten, any data encrypted with the public key cannot be decrypted.

Usage

```
genkeys(  
  private_key_name = "id_rsa",  
  public_key_name = paste0(private_key_name, ".pub")  
)
```

Arguments

<code>private_key_name</code>	Character string. Do not change default unless good reason.
<code>public_key_name</code>	Character string. Do not change default unless good reason.

Value

Two files containing the public key and encrypted private key are written to the working directory.

See Also

`encrypt` `decrypt`

Examples

```
# Function can be used as this:  
# genkeys()  
  
# For CRAN purposes and testing  
temp_dir = tempdir()  
genkeys(file.path(temp_dir, "id_rsa3"))
```

gp*General Practitioner (family doctor) practices in Scotland 2018*

Description

From NHS Digital Organisational Services Downloaded February 2019

Usage

`data(gp)`

Format

A data frame with 1212 rows and 12 variables

Index

* **data**

gp, [9](#)

decrypt, [2](#), [2](#)

decrypt_file, [3](#)

decrypt_vec, [4](#)

encrypt, [2–4](#), [5](#), [6](#)

encrypt_file, [3](#), [6](#)

encrypt_vec, [7](#)

encryptr-package, [2](#)

genkeys, [2–7](#), [8](#)

gp, [9](#)