Package 'rcdf'

October 12, 2025

Type Package

Title A Comprehensive Toolkit for Working with Encrypted Parquet Files

Version 0.1.1

Description Utilities for reading, writing, and managing RCDF files, including encryption and decryption support. It offers a flexible interface for handling data stored in encrypted Parquet format, along with metadata extraction, key management, and secure operations using Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) encryption.

Author Bhas Abdulsamad [aut, cre, cph] (ORCID: <https://orcid.org/0009-0002-5891-8124>)

Maintainer Bhas Abdulsamad <aeabdulsamad@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

Imports arrow, duckdb, haven, openxlsx, fs, zip, glue, utils (>= 4.0.0), openssl (>= 2.1.1), dplyr (>= 1.1.0), stringr (>= 1.4.0), jsonlite (>= 1.8.0), DBI (>= 1.1.0), RSQLite (>= 2.2.0), uuid (>= 0.1.2), methods

Suggests dbplyr (>= 2.4.0), rlang (>= 1.0.2), testthat (>= 3.0.0), cli, devtools, knitr, rmarkdown, mockery, tibble, withr, gt (>= 0.10.0)

Config/testthat/edition 3

RoxygenNote 7.2.3

BugReports https://github.com/yng-me/rcdf/issues

VignetteBuilder knitr

Depends R (>= 4.1.0)

URL https://yng-me.github.io/rcdf/

NeedsCompilation no

Repository CRAN

Date/Publication 2025-10-12 13:50:02 UTC

Contents

`add_metadata` *Add metadata attributes to a data frame*

Description

Adds variable labels and value labels to a data frame based on a metadata dictionary. This is particularly useful for preparing datasets for use with packages like `haven` or for exporting to formats like SPSS or Stata.

Usage

```
add_metadata(data, metadata, ..., set_data_types = FALSE)
```

Arguments

`data`	A data frame containing the raw dataset.
`metadata`	A data frame that serves as a metadata dictionary. It must contain at least the columns: `variable_name`, `label`, and `type`. Optionally, it may include a `valueset` column for categorical variables, which should be a list column with data frames containing `value` and `label` columns.
`...`	Additional arguments (currently unused).
`set_data_types`	Logical; if `TRUE`, attempts to coerce column data types to match those implied by the metadata. (Note: currently not fully implemented.)

Details

The function first checks the structure of the `metadata` using an internal helper. Then, for each variable listed in `metadata`, it: - Adds a label using the `label` attribute - Converts values to labelled vectors using `haven::labelled()` if a `valueset` is provided

If value labels are present, the function tries to align data types between the data and the valueset (e.g., converting character codes to integers if necessary).

Value

A 'tibble' with the same data as `data`, but with added attributes: - Variable labels (via the `label` attribute) - Value labels (as a `haven::labelled` class, if applicable)

Examples

```
data <- data.frame(
  sex = c(1, 2, 1),
  age = c(23, 45, 34)
)

metadata <- data.frame(
  variable_name = c("sex", "age"),
  label = c("Gender", "Age in years"),
  type = c("categorical", "numeric"),
  valueset = I(list(
    data.frame(value = c(1, 2), label = c("Male", "Female")),
    NULL
  ))
)

labelled_data <- add_metadata(data, metadata)
str(labelled_data)
```

---

as_rcdf *Convert to* `rcdf` *class*

Description

Converts an existing list or compatible object into an object of class `rcdf`.

Usage

```
as_rcdf(data)
```

Arguments

`data`	A list or object to be converted to class `rcdf`.

Value

The input object with class set to `rcdf`.

Examples

```
my_list <- list(a = 1, b = 2)
rcdf_obj <- as_rcdf(my_list)
class(rcdf_obj)
```

`generate_pw` *Generate a random password*

Description

This function generates a random password of a specified length. It includes alphanumeric characters by default and can optionally include special characters.

Usage

```
generate_pw(length = 16, special_chr = TRUE)
```

Arguments

`length`	Integer. The length of the password to generate. Default is `16`.
`special_chr`	Logical. Whether to include special characters (e.g., '¡, '@', '#', etc.) in the password. Default is `TRUE`.

Value

A character string representing the generated password.

Examples

```
generate_pw()
generate_pw(32)
generate_pw(12, special_chr = FALSE)
```

generate_rsa_keys *Generate RSA key pair and save to files*

Description

This function generates an RSA key pair (public and private) and saves them to specified files.

Usage

```
generate_rsa_keys(path, ..., password = NULL, which = "public", prefix = NULL)
```

Arguments

path	A character string specifying the directory path where the key files in .pem format should be saved.
...	Additional arguments passed to the openssl::rsa_keygen() function, such as key size.
password	A character string specifying the password for the private key. If NULL, the private key will not be encrypted.
which	A character string specifying which key to return. Can be either "public" or "private". Default is "public".
prefix	A character string used as a prefix for the key file names. Defaults to NULL, which will result in no prefix.

Value

A character string representing the file path of the generated key (either public or private, based on the which argument).

Examples

```
# Generate both public and private RSA keys and save them to the temp directory
path_to <- tempdir()
generate_rsa_keys(path = path_to, password = "securepassword")
```

get_data_dictionary *Extract data dictionary from RCDF object*

Description

This function retrieves the data dictionary embedded in the RCDF object

Usage

```
get_data_dictionary(data)
```

Arguments

data	Object of class rcdf.

Value

A data frame that serves as a metadata dictionary. It must contain at least the columns: variable_name, label, and type. Optionally, it may include a valueset column for categorical variables, which should be a list column with data frames containing value and label columns.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
data_dictionary <- get_data_dictionary(rcdf_data)
names(data_dictionary)
```

---

get_rcdf_metadata *Extract metadata from an RCDF file*

Description

Retrieves a specific metadata value from a .rcdf file.

Usage

```
get_rcdf_metadata(path, key)
```

Arguments

path	Character string. The file path to the .rcdf file.
key	Character string. The metadata key to extract from the file.

Value

The value associated with the specified metadata key, or NULL if the key does not exist.

Examples

```
## Not run:
# Assuming "example.rcdf" is a valid RCDF file in the working directory:
get_rcdf_metadata("example.rcdf", "creation_date")

## End(Not run)
```

rcdf_list *Create an empty* rcdf *object*

Description

Initializes and returns an empty rcdf object. This is a convenient constructor for creating a new rcdf-class list structure.

Usage

```
rcdf_list(...)
```

Arguments

...	Optional elements to include in the list. These will be passed to the internal list constructor and included in the resulting rcdf object.

Value

A list object of class rcdf.

Examples

```
rcdf <- rcdf_list()
class(rcdf)
```

read_env *Read environment variables from a file*

Description

Reads a .env file containing environment variables in the format KEY=VALUE, and returns them as a named list. Lines starting with # are considered comments and ignored.

Usage

```
read_env(path)
```

Arguments

path	A string specifying the path to the .env file. If not provided, defaults to .env in the current working directory.

Value

A named list of environment variables. Each element is a key-value pair extracted from the file. If no variables are found, NULL is returned.

Examples

```
## Not run:
# Assuming an `.env` file with the following content:
# DB_HOST=localhost
# DB_USER=root
# DB_PASS="secret"

env_vars <- read_env(".env")
print(env_vars)
# Should output something like:
# $DB_HOST
# [1] "localhost"

# If no path is given, it defaults to `.env` in the current directory.
env_vars <- read_env()

## End(Not run)
```

`read_parquet` *Read Parquet file with optional decryption*

Description

This function reads a Parquet file, optionally decrypting it using the provided decryption key. If no decryption key is provided, it reads the file normally without decryption. It supports reading Parquet files as Arrow tables or regular data frames, depending on the `as_arrow_table` argument.

Usage

```
read_parquet(
  path,
  ...,
  decryption_key = NULL,
  as_arrow_table = TRUE,
  metadata = NULL
)
```

Arguments

`path`	The file path to the Parquet file.
`...`	Additional arguments passed to `arrow::open_dataset()` when no decryption key is provided.
`decryption_key`	A list containing `aes_key` and `aes_iv`. If provided, the Parquet file will be decrypted using these keys. Default is 'NULL'.
`as_arrow_table`	Logical. If `TRUE`, the function will return the result as an Arrow table. If `FALSE`, a regular data frame will be returned. Default is `TRUE`.
`metadata`	Optional metadata (e.g., a data dictionary) to be applied to the resulting data.

Value

An Arrow table or a data frame, depending on the value of `as_arrow_table`.

Examples

```
# Using sample Parquet files from `mtcars` dataset
dir <- system.file("extdata", package = "rcdf")

# Without decryption
df <- read_parquet(file.path(dir, "mtcars.parquet"))
df

# With decryption
decryption_key <- list(
  aes_key = "5bddd0ea4ab48ed5e33b1406180d68158aa255cf3f368bdd4744abc1a7909ead",
  aes_iv = "7D3EF463F4CCD81B11B6EC3230327B2D"
)

df_with_encryption <- read_parquet(
  file.path(dir, "mtcars-encrypted.parquet"),
  decryption_key = decryption_key
 )
df_with_encryption
```

---

read_rcdf *Read and decrypt RCDF data*

Description

This function reads an RCDF (Reusable Data Container Format) archive, decrypts its contents using the specified decryption key, and loads it into R as an RCDF object. The data files within the archive (usually Parquet files) are decrypted and, if provided, metadata (such as data dictionary and value sets) are applied to the data.

Usage

```
read_rcdf(
  path,
  decryption_key,
  ...,
  password = NULL,
  metadata = list(),
  ignore_duplicates = TRUE,
  recursive = FALSE,
  return_meta = FALSE
)
```

Arguments

path	A string specifying the path to the RCDF archive (zip file). If a directory is provided, all .rcdf files within that directory will be processed.
decryption_key	The key used to decrypt the RCDF contents. This can be an RSA or AES key, depending on how the RCDF was encrypted.
...	Additional parameters passed to other functions, if needed.
password	A password used for RSA decryption (optional).
metadata	An optional list of metadata object containing data dictionaries, value sets, and primary key constraints for data integrity measure (a data.frame or tibble that includes at least two columns: file and pk_field_name. This metadata is applied to the data if provided.
ignore_duplicates	A logical flag. If TRUE, a warning is issued when duplicates are found, based on the primary key/s defined during creation of RCDF file. If FALSE, the function stops with an error.
recursive	Logical. If TRUE and path is a directory, the function will search recursively for .rcdf files.
return_meta	Logical. If TRUE, the metadata will be returned as an attribute of the RCDF object.

Value

An RCDF object, which is a list of Parquet files (one for each record) along with attached metadata.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
rcdf_data

# Using encrypted/password protected private key
rcdf_path_pw <- file.path(dir, 'mtcars-pw.rcdf')
private_key_pw <- file.path(dir, 'sample-private-key-pw.pem')
pw <- '1234'

rcdf_data_with_pw <- read_rcdf(
  path = rcdf_path_pw,
  decryption_key = private_key_pw,
  password = pw
)

rcdf_data_with_pw
```

`write_parquet` *Write Parquet file with optional encryption*

Description

This function writes a dataset to a Parquet file. If an encryption key is provided, the data will be encrypted before writing. Otherwise, the function writes the data as a regular Parquet file without encryption.

Usage

```
write_parquet(data, path, ..., encryption_key = NULL)
```

Arguments

`data`	A data frame or tibble to write to a Parquet file.
`path`	The file path where the Parquet file will be written.
`...`	Additional arguments passed to `arrow::write_parquet()` if no encryption key is provided.
`encryption_key`	A list containing `aes_key` and `aes_iv`. If provided, the data will be encrypted using AES before writing to Parquet.

Value

None. The function writes the data to a Parquet file at the specified `path`.

Examples

```
data <- mtcars
key <- "5bddd0ea4ab48ed5e33b1406180d68158aa255cf3f368bdd4744abc1a7909ead"
iv <- "7D3EF463F4CCD81B11B6EC3230327B2D"

temp_dir <- tempdir()

rcdf::write_parquet(
  data = data,
  path = file.path(temp_dir, "mtcars.parquet"),
  encryption_key = list(aes_key = key, aes_iv = iv)
)

unlink(file.path(temp_dir, "mtcars.parquet"), force = TRUE)
```

write_rcdf *Write data to RCDF format*

Description

This function writes data to an RCDF (Reusable Data Container Format) archive. It encrypts the data using AES, generates metadata, and then creates a zip archive containing both the encrypted Parquet files and metadata. The function supports the inclusion of metadata such as system information and encryption keys.

Usage

```
write_rcdf(
  data,
  path,
  pub_key,
  ...,
  metadata = list(),
  ignore_duplicates = TRUE
)
```

Arguments

data	A list of data frames or tables to be written to RCDF format. Each element of the list represents a record.
path	The path where the RCDF file will be written. The file will be saved with a `.rcdf` extension if not already specified.
pub_key	The public RSA key used to encrypt the AES encryption keys.
...	Additional arguments passed to helper functions if needed.
metadata	A list of metadata to be included in the RCDF file.
ignore_duplicates	A `logical` flag. If `TRUE`, a warning is issued when duplicates are found. If `FALSE`, the function stops with an error.

Value

NULL. The function writes the data to a `.rcdf` file at the specified path.

Examples

```
# Example usage of writing an RCDF file

rcdf_data <- rcdf_list()
rcdf_data$mtcars <- mtcars

dir <- system.file("extdata", package = "rcdf")
```

```
temp_dir <- tempdir()

write_rcdf(
  data = rcdf_data,
  path = file.path(temp_dir, "mtcars.rcdf"),
  pub_key = file.path(dir, 'sample-public-key.pem')
)

write_rcdf(
  data = rcdf_data,
  path = file.path(temp_dir, "mtcars-pw.rcdf"),
  pub_key = file.path(dir, 'sample-public-key-pw.pem')
)

unlink(file.path(temp_dir, "mtcars.rcdf"), force = TRUE)
unlink(file.path(temp_dir, "mtcars-pw.rcdf"), force = TRUE)
```

`write_rcdf_as` *Write RCDF data to multiple formats*

Description

Exports RCDF-formatted data to one or more supported open data formats. The function automatically dispatches to the appropriate writer function based on the `formats` provided.

Usage

```
write_rcdf_as(data, path, formats, ...)
```

Arguments

`data`	A named list or RCDF object. Each element should be a table or tibble-like object (typically a `dbplyr` or `dplyr` table).
`path`	The target directory where output files should be saved.
`formats`	A character vector of file formats to export to. Supported formats include: `"csv"`, `"tsv"`, `"json"`, `"parquet"`, `"xlsx"`, `"dta"`, `"sav"`, and `"sqlite"`.
`...`	Additional arguments passed to the respective writer functions.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_csv write_rcdf_tsv write_rcdf_json write_rcdf_xlsx write_rcdf_dta write_rcdf_sav write_rcdf_sqlite

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_as(data = rcdf_data, path = temp_dir, formats = c("csv", "xlsx"))

unlink(temp_dir, force = TRUE)
```

`write_rcdf_csv` *Write RCDF data to CSV files*

Description

Writes each table in the RCDF object as a separate `.csv` file.

Usage

```
write_rcdf_csv(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The base output directory.
`...`	Additional arguments passed to `write.csv()`.
`parent_dir`	Optional subdirectory under `path` to group CSV files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_csv(data = rcdf_data, path = temp_dir)
```

```
unlink(temp_dir, force = TRUE)
```

`write_rcdf_dta` *Write RCDF data to Stata* `.dta` *files*

Description

Writes each table in the RCDF object to a `.dta` file for use in Stata.

Usage

```
write_rcdf_dta(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for files.
`...`	Additional arguments passed to `foreign::write.dta()`.
`parent_dir`	Optional subdirectory under `path` to group Stata files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_dta(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_json` *Write RCDF data to JSON files*

Description

Writes each table in the RCDF object as a separate `.json` file.

Usage

```
write_rcdf_json(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The output directory for files.
`...`	Additional arguments passed to `jsonlite::write_json()`.
`parent_dir`	Optional subdirectory under `path` to group JSON files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_json(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_parquet` *Write RCDF data to Parquet files*

Description

This function writes an RCDF object (a list of data frames) to multiple Parquet files. Each data frame in the list is written to its corresponding Parquet file in the specified path.

Usage

```
write_rcdf_parquet(
  data,
  path,
  ...,
  parent_dir = NULL,
  primary_key = NULL,
  ignore_duplicates = TRUE
)
```

Arguments

`data`	A list where each element is a data frame or tibble that will be written to a Parquet file.
`path`	The directory path where the Parquet files will be written.
`...`	Additional arguments passed to `rcdf::write_parquet()` while writing each Parquet file.
`parent_dir`	An optional parent directory to be included in the path where the files will be written.
`primary_key`	A `data.frame` or `tibble` that includes at least two columns: `file` and `pk_field_name`.
`ignore_duplicates`	A `logical` flag. If `TRUE`, a warning is issued when duplicates are found. If `FALSE`, the function stops with an error.

Value

A character vector of file paths to the written Parquet files.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_parquet(data = rcdf_data, path = temp_dir)
```

```
unlink(temp_dir, force = TRUE)
```

`write_rcdf_sav` *Write RCDF data to SPSS* `.sav` *files*

Description

Writes each table in the RCDF object to a `.sav` file using the haven package for compatibility with SPSS.

Usage

```
write_rcdf_sav(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for files.
`...`	Additional arguments passed to `haven::write_sav()`.
`parent_dir`	Optional subdirectory under `path` to group SPSS files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_sav(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_sqlite` *Write RCDF data to a SQLite database*

Description

Writes all tables in the RCDF object to a single SQLite database file.

Usage

```
write_rcdf_sqlite(data, path, db_name = "cbms_data", ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for the database file.
`db_name`	Name of the SQLite database file (without extension).
`...`	Additional arguments passed to `DBI::dbWriteTable()`.
`parent_dir`	Optional subdirectory under ‘path‘ to store the SQLite file.

Value

Invisibly returns `NULL`. A `.db` file is written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_sqlite(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_tsv` *Write RCDF data to TSV files*

Description

Writes each table in the RCDF object as a separate tab-separated `.txt` file.

Usage

```
write_rcdf_tsv(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The base output directory.
`...`	Additional arguments passed to `write.table()`.
`parent_dir`	Optional subdirectory under `path` to group TSV files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_tsv(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_xlsx` *Write RCDF data to Excel files*

Description

Writes each table in the RCDF object as a separate `.xlsx` file using the `openxlsx` package.

Usage

```
write_rcdf_xlsx(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The output directory.
`...`	Additional arguments passed to `openxlsx::write.xlsx()`.
`parent_dir`	Optional subdirectory under `path` to group Excel files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key)
temp_dir <- tempdir()

write_rcdf_xlsx(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

Index