Package ‘EQUALencrypt’

August 26, 2025

Title Encryption and Decryption of Files and Data for Researchers Without Coding Skills

Version 0.1.0

Date 2025-08-20

Author Kurinchi Gurusamy [aut, cre]

Maintainer Kurinchi Gurusamy <k.gurusamy@ucl.ac.uk>

Depends openssl, stringr, uuid, zip

Description Support functions for R-based ``EQUALencrypt - Encrypt and decrypt whole files" and ``EQUALencrypt - Encrypt and decrypt columns of data" shiny applications which allow researchers without coding skills or expertise in encryption algorithms to share data after encryption. Gurusamy,K (2025)<doi:10.5281/zenodo.16743676> and Gurusamy,K (2025)<doi:10.5281/zenodo.16744058>.

License GPL (>= 3)

Encoding UTF-8

URL https://sites.google.com/view/equal-group/home

NeedsCompilation no

Repository CRAN

Date/Publication 2025-08-26 13:50:02 UTC

Contents

`EQUAL_decrypt_data` *Decrypt data*

Description

"Decrypts data using the private key generated by `EQUAL_encrypt_generate_keys()` function and **openssl**. This reverses the process followed in `EQUAL_encrypt_data()` function."

Usage

```
EQUAL_decrypt_data(encrypted_data, private_key_folder, key_name)
```

Arguments

`encrypted_data`	Encrypted data that must be decrypted
`private_key_folder`	Location of the private key
`key_name`	Name of the private key

Value

decrypted data

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`EQUAL_encrypt_generate_keys()` `openssl::aes_cbc_decrypt()` `openssl::rsa_decrypt()`

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_data(data = data,
                                     public_key_folder = public_key_folder,
                                     key_name = "encryption_key.txt")
# Decrypt data ####
decrypted_data <- EQUAL_decrypt_data(encrypted_data = encrypted_data,
                                     private_key_folder = private_key_folder,
                                     key_name = "encryption_key.txt")
```

`EQUAL_decrypt_file` *Decrypt a file*

Description

"Decrypts a file using the private key generated by `EQUAL_encrypt_generate_keys()` function and **openssl**. This reverses the process followed in `EQUAL_encrypt_file()` function."

Usage

```
EQUAL_decrypt_file(encrypted_data, private_key_folder, key_name,
data_storage_folder)
```

Arguments

`encrypted_data`	Encrypted data that must be decrypted
`private_key_folder`	Location of the private key

`key_name`	Name of the private key
`data_storage_folder`	Location to store the decrypted file temporarily

Value

0 (the decrypted file is saved in the temporary directory)

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`EQUAL_encrypt_generate_keys()` `openssl::aes_cbc_decrypt()` `openssl::rsa_decrypt()`

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"),
row.names = FALSE, na = "")
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_file(file_name = paste0(tempdir(), "/test.csv"),
public_key_folder = public_key_folder, key_name = "encryption_key.txt")
data_storage_folder <- paste0(test_folder, "/data_storage_folder")
dir.create(data_storage_folder)
# Results ####
results <- EQUAL_decrypt_file(encrypted_data = encrypted_data,
                              private_key_folder = private_key_folder,
                              key = "encryption_key.txt",
                              data_storage_folder = data_storage_folder)
```

`EQUAL_encrypt_data` *Encrypt data*

Description

"Encrypts data using the public key generated by `EQUAL_encrypt_generate_keys()` function and **openssl**. This encrypts the file using symmetric AES256 algorithm and encrypts the AES key using the asymmetric RSA algorithm (4096 bits) and includes padding according to PKCS #1 v2.0 specifications."

Usage

```
EQUAL_encrypt_data(data, public_key_folder, key_name)
```

Arguments

`data`	Data that must be encrypted
`public_key_folder`	Location of the public key
`key_name`	Name of the public key

Value

`iv`	initialisation vector for AES key
`session`	RSA encrypted AES key
`data`	AES encrypted data

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

EQUAL_encrypt_generate_keys() openssl::aes_cbc_encrypt() openssl::rsa_encrypt()

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_data(data = data,
                                     public_key_folder = public_key_folder,
                                     key_name = "encryption_key.txt")
```

EQUAL_encrypt_file *Encrypt a file*

Description

"Encrypts a file using the public key generated by EQUAL_encrypt_generate_keys() function and **openssl**. This encrypts the file using symmetric AES256 algorithm and encrypts the AES key using the asymmetric RSA algorithm (4096 bits) and includes padding according to PKCS #1 v2.0 specifications."

Usage

```
EQUAL_encrypt_file(file_name, public_key_folder, key_name)
```

Arguments

`file_name`	Name of the file that must be encrypted
`public_key_folder`	Location of the public key
`key_name`	Name of the public key

Value

`iv`	initialisation vector for AES key
`session`	RSA encrypted AES key
`data`	AES encrypted data

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

`https://sites.google.com/view/equal-group/home`

See Also

`EQUAL_encrypt_generate_keys()` `openssl::aes_cbc_encrypt()` `openssl::rsa_encrypt()`

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
```

```
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Results ####
results <- EQUAL_encrypt_file(file_name = paste0(tempdir(), "/test.csv"),
public_key_folder = public_key_folder, key_name = "encryption_key.txt")
```

EQUAL_encrypt_generate_keys

Generate the encryption keys

Description

"Generates the public and private encryption keys using **openssl**. This uses the asymmetric RSA algorithm 4096 bits for generating the keys. These keys are used for encrypting and decrypting data and files and for inserting and verifying digital signatures."

Usage

```
EQUAL_encrypt_generate_keys(public_key_folder, private_key_folder, key_name)
```

Arguments

`public_key_folder`	Location to store the public key
`private_key_folder`	Location to store the private key
"	
`key_name`	Name of the key (a single name for both public and private keys)
"	

Value

`private_key`	private key generated by the algorithm
`public_key`	public key generated by the algorithm

Note

This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`openssl::rsa_keygen()`

Examples

```
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
results <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
```

EQUAL_insert_signature_data

Insert digital signature for data

Description

"Insert digital signature for data using the private key generated by `EQUAL_encrypt_generate_keys()` function and **openssl**. This uses the SHA384 algorithm for the hash function."

Usage

```
EQUAL_insert_signature_data(data, private_key_folder, key_name)
```

Arguments

`data`	Data for which signature must be inserted
`private_key_folder`	Location of the private key
`key_name`	Name of the private key

Value

"

`path_to_signed_file`	path to the signed data which is stored in a file

"

`signature`	signature

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

EQUAL_encrypt_generate_keys() openssl::signature_create()

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_data(data = data,
                                     public_key_folder = public_key_folder,
                                     key_name = "encryption_key.txt")
# Insert signature ####
signature <- EQUAL_insert_signature_data(data = encrypted_data,
private_key_folder = private_key_folder,
key_name = "encryption_key.txt")
```

EQUAL_insert_signature_file
Insert digital signature for a file

Description

"Insert digital signature for a file using the private key generated by `EQUAL_encrypt_generate_keys()` function and **openssl**. This uses the SHA384 algorithm for the hash function."

Usage

```
EQUAL_insert_signature_file(file_name, private_key_folder, key_name)
```

Arguments

`file_name`	Name of the file for which signature must be inserted
`private_key_folder`	Location of the private key
`key_name`	Name of the private key

Value

signature

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

`https://sites.google.com/view/equal-group/home`

See Also

`EQUAL_encrypt_generate_keys()` `openssl::signature_create()` `openssl::hashing()`

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_file(file_name = paste0(tempdir(), "/test.csv"),
                                     public_key_folder = public_key_folder,
                                     key_name = "encryption_key.txt")
data_storage_folder <- paste0(test_folder, "/data_storage_folder")
dir.create(data_storage_folder)
saveRDS(encrypted_data, paste0(data_storage_folder, "/encrypted_file.RDS"))
# Insert signature ####
results <- EQUAL_insert_signature_file(file_name = paste0(data_storage_folder,
"/encrypted_file.RDS"), private_key_folder = private_key_folder,
                            key_name = "encryption_key.txt")
```

EQUAL_perform_data_decryption

Wrapper function for data decryption

Description

"A wrapper function which takes the user input obtained via the Rshiny app, decrypts the encrypted data file using the `EQUAL_decrypt_data()` function after verifying the digital signature on the encrypted file using the `EQUAL_verify_signature()` function."

Usage

```
EQUAL_perform_data_decryption(rv)
```

Arguments

`rv`	A list supplied by EQUAL-STATS application based on user input

Value

`html_message`	message to the user which includes whether the decryption was successfully performed
`decrypted_file_name`	path to the decrypted file

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`EQUAL_decrypt_data()` `EQUAL_verify_signature()`

Examples

```
library(openssl)
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE, na = "")
# Simulate the rv variable ####
rv <- {list(
  file_upload_encrypt = cbind.data.frame(datapath = paste0(tempdir(), "/test.csv")),
  level_1 = "v000002",
  level_2 = "",
  level_3 = "v000003",
  level_4 = "",
  level_5 = "v000001",
  level_6 = "",
  level_7 = ""
)}
```

```
# Encrypt data ####
encrypted_data <- EQUAL_perform_data_encryption(rv, server_address = tempdir())
# Simulate what happens before user input for decryption ####
unzipped_files_folder <- paste0(tempfile(), "/unzipped_files")
dir.create(unzipped_files_folder, recursive = TRUE)
zip::unzip(encrypted_data$encrypted_file_name, exdir = unzipped_files_folder)
zip::unzip(paste0(unzipped_files_folder, "/publicly_shareable.zip"),
exdir = unzipped_files_folder)
zip::unzip(paste0(unzipped_files_folder, "/not_publicly_shareable.zip"),
exdir = unzipped_files_folder)
# Simulated rv list for decryption
rv <- {list(
  file_upload_decrypt = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "/level_7_main_content.zip")),
  public_keys_upload = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "/level_7_public_keys.zip")),
  private_keys_upload = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "/level_7_private_keys.zip"))
)}
results <- EQUAL_perform_data_decryption(rv)
```

EQUAL_perform_data_encryption

Wrapper function for data encryption

Description

"A wrapper function which takes the user input obtained via the Rshiny app, generates muliple sets of private and public encryption keys corresponding to the levels of access using the `EQUAL_encrypt_generate_keys()` function, encrypts different columns using encryption keys corresponding to the level of access using the `EQUAL_encrypt_data()` function, and inserts digital signature on the encrypted data using the `EQUAL_insert_signature_data()` function."

Usage

```
EQUAL_perform_data_encryption(rv, server_address = tempdir())
```

Arguments

`rv`	A list supplied by EQUAL-STATS application based on user input
`server_address`	default address is `tempdir()`. If a different address is provided, a local copy of the file uploaded for encryption is retained.

Value

"

`html_message`	message to the user which includes whether the encryption was successfully performed

"

`encrypted_file_name`	path to the encrypted file

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`EQUAL_encrypt_data()` `EQUAL_insert_signature_file()`

Examples

```
library(openssl)
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Simulate the rv variable ####
rv <- {list(
  file_upload_encrypt = cbind.data.frame(datapath = paste0(tempdir(),
  "/test.csv")),
  level_1 = "v000002",
  level_2 = "",
  level_3 = "v000003",
  level_4 = "",
  level_5 = "v000001",
  level_6 = "",
  level_7 = ""
)}
# Encrypt data ####
encrypted_data <- EQUAL_perform_data_encryption(rv, server_address = tempdir())
```

EQUAL_perform_file_decryption *Wrapper function for file decryption*

Description

"A wrapper function which takes the user input obtained via the Rshiny app, decrypts a file using the `EQUAL_decrypt_file()` function after verifying the digital signature on the encrypted file using the `EQUAL_verify_signature()` function."

Usage

```
EQUAL_perform_file_decryption(rv)
```

Arguments

`rv`	A list supplied by EQUAL-STATS application based on user input

Value

"

`html_message`	message to the user which includes whether the decryption was successfully performed

"

`decrypted_file_path`	path to the decrypted file

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`EQUAL_decrypt_file()` `EQUAL_verify_signature()`

Examples

```
library(openssl)
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Simulated rv list ####
rv <- {list(
  file_upload_encrypt = cbind.data.frame(datapath = paste0(tempdir(),
  "/test.csv"))
)}
# Perform file encryption ####
encryption_results <- EQUAL_perform_file_encryption(rv,
server_address = tempdir())
# Simulate what happens prior to user input for decryption ####
# The encrypted files are unzipped and the individual files are shared
unzipped_files_folder <- paste0(tempfile(), "/unzipped_files")
dir.create(unzipped_files_folder, recursive = TRUE)
zip::unzip(encryption_results$encrypted_file_path,
exdir = unzipped_files_folder)
zip::unzip(paste0(unzipped_files_folder, "/publicly_shareable.zip"),
exdir = unzipped_files_folder)
zip::unzip(paste0(unzipped_files_folder, "/not_publicly_shareable.zip"),
exdir = unzipped_files_folder)
# Simulated rv list for decryption
rv <- {list(
  file_upload_decrypt = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "encrypted_file.RDS")),
  signature_upload = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "signature.RDS")),
  public_keys_upload = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "public_encryption_key.txt")),
  private_keys_upload = cbind.data.frame(datapath =
  paste0(unzipped_files_folder, "private_encryption_key.txt"))
)}
results <- EQUAL_perform_file_decryption(rv)
```

EQUAL_perform_file_encryption

Wrapper function for file encryption

Description

"A wrapper function which takes the user input obtained via the Rshiny app, generates a set of private and public encryption keys using the `EQUAL_encrypt_generate_keys()` function, encrypts a file using the EQUAL_encrypt_file() function, and inserts digital signature on the encrypted file using the EQUAL_insert_signature_file() function."

Usage

```
EQUAL_perform_file_encryption(rv, server_address = tempdir())
```

Arguments

rv	A list supplied by EQUAL-STATS application based on user input
server_address	default address is tempdir(). If a different address is provided, a local copy of the file uploaded for encryption is retained.

Value

"

html_message	message to the user which includes whether the encryption was successfully performed

"

encrypted_file_path	path to the encrypted file

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

EQUAL_encrypt_file() EQUAL_insert_signature_file()

Examples

```
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
```

```
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Simulated rv list ####
rv <- {list(
  file_upload_encrypt = cbind.data.frame(datapath =
  paste0(tempdir(), "/test.csv"))
  )}
# Perform the test ####
results <- EQUAL_perform_file_encryption(rv, server_address = tempdir())
```

EQUAL_verify_signature

Verify signature on a file

Description

"Verifies the digital signature on a file using the public key generated by `EQUAL_encrypt_generate_keys()` function, the signature created using `EQUAL_insert_signature_file()` function, and **openssl**."

Usage

```
EQUAL_verify_signature(file_name, signature, key_name, public_key_folder)
```

Arguments

`file_name`	Name of the file for which signature must be verified
"	
`signature`	Signature created during the `EQUAL_insert_signature_file()` function
"	
`key_name`	Name of the public key
`public_key_folder`	Location of the public key

Value

logical indicating whether the signature is verified

Note

"This is part of a suite of functions required to allow encrypting and decrypting whole files and encrypting and decrypting columns of data programs to run. This is unlikely to be used as a stand alone function."

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

EQUAL_encrypt_generate_keys() openssl::signature_verify()

Examples

```
library(openssl)
# Encryption keys ####
test_folder <- tempfile(pattern = "folder_")
public_key_folder <- paste0(test_folder, "/public_key_folder")
private_key_folder <- paste0(test_folder, "/private_key_folder")
dir.create(test_folder)
dir.create(public_key_folder)
dir.create(private_key_folder)
encryption_keys <- EQUAL_encrypt_generate_keys(
  public_key_folder = public_key_folder,
  private_key_folder = private_key_folder,
  key_name = "encryption_key.txt")
# Data ####
data <- lapply(1:3, function(x) {
  mean = sample(1:100, 1, replace = FALSE)
  sd = sample(1:100, 1, replace = FALSE)
  rnorm(100, mean = mean, sd = sd)
})
data <- do.call(cbind.data.frame, data)
colnames(data) <- paste0("v", formatC(1:3, width = 6, flag = "0"))
test_file <- write.csv(data, paste0(tempdir(), "/test.csv"), row.names = FALSE,
na = "")
# Encrypt data ####
encrypted_data <- EQUAL_encrypt_file(file_name = paste0(tempdir(), "/test.csv"),
                                     public_key_folder = public_key_folder,
                                     key_name = "encryption_key.txt")
data_storage_folder <- paste0(test_folder, "/data_storage_folder")
dir.create(data_storage_folder)
saveRDS(encrypted_data, paste0(data_storage_folder, "/encrypted_file.RDS"))
# Insert signature ####
signature <- EQUAL_insert_signature_file(
file_name = paste0(data_storage_folder, "/encrypted_file.RDS"),
private_key_folder = private_key_folder,
key_name = "encryption_key.txt")
# Verify signature ####
results <- EQUAL_verify_signature(
file_name = paste0(data_storage_folder, "/encrypted_file.RDS"),
signature = signature, key_name = "encryption_key.txt",
public_key_folder = public_key_folder)
```

Index