Package 'argon2'

October 30, 2021

Type Package

Title Secure Password Hashing

Version 0.4-0

Description Utilities for secure password hashing via the argon2 algorithm. It is a relatively new hashing algorithm and is believed to be very secure. The 'argon2' implementation included in the package is the reference implementation. The package also includes some utilities that should be useful for digest authentication, including a wrapper of 'blake2b'. For similar R packages, see sodium and 'bcrypt'. See <https://en.wikipedia.org/wiki/Argon2> or <https://eprint.iacr.org/2015/430.pdf> for more information.

License BSD 2-clause License + file LICENSE

Copyright See inst/COPYRIGHTS for files in src/argon2.

Depends R (>= 3.0.0)

NeedsCompilation yes

ByteCompile yes

URL https://github.com/wrathematics/argon2

BugReports https://github.com/wrathematics/argon2/issues

Maintainer Drew Schmidt <wrathematics@gmail.com>

RoxygenNote 7.1.2

Author Drew Schmidt [aut, cre]

Repository CRAN

Date/Publication 2021-10-30 19:30:02 UTC

R topics documented:

argon2-package *argon2*

Description

Utilities for secure password hashing via the argon2 algorithm. It is a relatively new hashing algorithm and is believed to be very secure. The argon2 implementation included in the package is the reference implementation. The package also includes some utilities that should be useful for digest authentication, including a wrapper of blake2b. For similar R packages, see sodium and bcrypt.

Author(s)

Drew Schmidt

References

Project URL: `https://github.com/wrathematics/argon2`

blake2b *blake2b*

Description

A 512-bit blake2b hash implementation.

Usage

```
blake2b(x, key = NULL)
```

Arguments

`x`	Input to be hashed. Can be a single string or a raw vector.
`key`	An optional key. Should be `NULL` (for no key), a single string, or a raw vector.

Value

The hash of the string as a raw vector.

References

Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z. and Winnerlein, C., 2013, June. BLAKE2: simpler, smaller, fast as MD5. In International Conference on Applied Cryptography and Network Security (pp. 119-135). Springer Berlin Heidelberg.

Examples

```
## Not run:
library(argon2)
blake2b("some string")
blake2b("another")

## End(Not run)
```

---

gen_nonce *Generate a nonce*

Description

Generates a random raw (unsigned char*) vector.

Usage

```
gen_nonce(length = 64)
```

Arguments

`length`	The number of elements to return.

Value

A random raw vector.

---

hashing *Password Hashing*

Description

Basic password hashing. Use `pw_hash()` to hash and `pw_check()` to compare a possible password with the hashed password.

Usage

```
pw_hash(pass, variant = "i", iterations = 16, memory = 8, nthreads = 2)

pw_check(hash, pass)
```

Arguments

`pass`	The (plaintext) password.
`variant`	Choice of algorithm; currently the only supported choices are "i" and "d".
`iterations`	A time cost. Recommended to be at least 10. Can be any integer from 1 to 2^31 - 1.
`memory`	A memory cost, given in MiB. Recommended to be at least 8. Can be any integer from 1 to 2^21 - 1 (but don't be ridiculous).
`nthreads`	Number of threads. This affects the speed of hashing, so more is better.
`hash`	The hashed password; this is the output of `pw_hash()`.

Details

The default options for `iterations` and `memory` should be sufficient for most purposes. You are encouraged to read the official documentation before modifying these values, which can be found here https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf.

On the other hand, `nthreads` is safe to change to fit your available resources, and you are encouraged to do so.

This uses the argon2 (i or d variety) hash algorithm. See references for details and implementation source code (also bundled with this package).

Our binding uses a 512 bit salt with data generated from MT.

Value

`pw_hash()` returns a hash to be used as an input to `pw_check()`.

`pw_check()` returns `TRUE` or `FALSE`, whether or not the plaintext password matches its hash.

References

Biryukov, A., Dinu, D. and Khovratovich, D., 2015. Fast and Tradeoff-Resilient Memory-Hard Functions for Cryptocurrencies and Password Hashing. IACR Cryptology ePrint Archive, 2015, p.430.

Reference implementation https://github.com/P-H-C/phc-winner-argon2

Examples

```
## Not run:
library(argon2)

pass <- "myPassw0rd!"
hash <- pw_hash(pass)
hash # store this

pw_check(hash, pass)
pw_check(hash, "password")
pw_check(hash, "1234")

## End(Not run)
```

`raw_as_char` *raw_as_char*

Description

Convert the literal bytes of a raw (unsigned char*) to a string representation. This is different from R's `rawToChar()`. See examples for details.

Usage

```
raw_as_char(raw, upper = TRUE, spaces = FALSE)
```

Arguments

`raw`	A raw vector.
`upper`	Should hex digits A-F be given in uppercase?
`spaces`	Should the str use spaces?

Value

A character string.

Examples

```
## Not run:
library(argon2)
str <- "some text"
raw <- charToRaw(str)
raw

rawToChar(raw)
raw_as_char(raw)

## End(Not run)
```

Index