

Package ‘homomorpheR’

January 23, 2019

Type Package

Title Homomorphic Computations in R

Version 0.2-2

Date 2019-01-22

VignetteBuilder knitr

URL <http://github.com/bnaras/homomorpheR>

BugReports <http://github.com/bnaras/homomorpheR/issues>

Suggests knitr, rmarkdown, survival

Imports R6, gmp, sodium

Description Homomorphic computations in R for privacy-preserving applications. Currently only the Paillier Scheme is implemented.

License MIT + file LICENSE

RoxygenNote 5.0.0

NeedsCompilation no

Author Balasubramanian Narasimhan [aut, cre]

Maintainer Balasubramanian Narasimhan <naras@stat.Stanford.EDU>

Repository CRAN

Date/Publication 2019-01-23 08:30:10 UTC

R topics documented:

homomorpheR	2
PaillierKeyPair	2
PaillierPrivateKey	3
PaillierPublicKey	4
random.bigz	5

Index	6
--------------	----------

homomorpheR	<i>homomorpheR: Homomorphic computations in R</i>
-------------	---

Description

homomorpheR is a start at a rudimentary package for homomorphic computations in R. The goal is to collect homomorphic encryption schemes in this package for privacy-preserving distributed computations; for example, applications of the sort implemented in package `distcomp`.

Details

At the moment, only one scheme is implemented, the Paillier scheme. The current implementation makes no pretense at efficiency and also uses direct translations of other implementations, particularly the one in Javascript.

For a quick overview of the features, read the [homomorpheR vignette](#) by running `vignette("homomorpheR")`.

References

https://en.wikipedia.org/wiki/Homomorphic_encryption

<https://mhe.github.io/jspaillier/>

Examples

```
keys <- PaillierKeyPair$new(1024) # Generate new key pair
encryptAndDecrypt <- function(x) keys$getPrivateKey()$decrypt(keys$pubkey$encrypt(x))
a <- gmp::as.bigz(1273849)
identical(a + 10L, encryptAndDecrypt(a+10L))
x <- lapply(1:100, function(x) random.bigz(nBits = 512))
edx <- lapply(x, encryptAndDecrypt)
identical(x, edx)
```

PaillierKeyPair	<i>Construct a Paillier public and private key pair given a fixed number of bits</i>
-----------------	--

Description

Construct a Paillier public and private key pair given a fixed number of bits

Usage

```
PaillierKeyPair
```

Format

An [R6Class](#) generator object

Fields

pubkey the Paillier public key

Methods

PaillierKeyPair\$new(modulusBits) Create a new private key with specified number of modulus bits

PaillierKeyPair\$getPrivateKey() Return the private key

See Also

PaillierPublicKey and PaillierPrivateKey

Examples

```
keys <- PaillierKeyPair$new(1024)
keys$pubkey
keys$getPrivateKey()
```

PaillierPrivateKey *Construct a Paillier private key with the given secret and a public key*

Description

Construct a Paillier private key with the given secret and a public key

Usage

```
PaillierPrivateKey
```

Format

An [R6Class](#) generator object

Fields

pubkey the Paillier public key

Methods

PaillierPrivateKey\$new(lambda, pubkey) Create a new private key with given secret lambda and the public key

PaillierPrivateKey\$getLambda() Return the secret lambda

PaillierPrivateKey\$decrypt(c) Decrypt a message. The value c should be an encrypted value

See Also

PaillierPublicKey which goes hand-in-hand with this object

PaillierPublicKey *Construct a Paillier public key with the given modulus.*

Description

Construct a Paillier public key with the given modulus.

Usage

PaillierPublicKey

Format

An [R6Class](#) generator object

Fields

bits the number of bits in the modulus

n the modulus

nSquared the square of the modulus

nPlusOne one more than the modulus

Methods

PaillierPublicKey\$new(bits, n) Create a new public key with given bits and modulus n. It also precomputes a few values for more efficient computations

PaillierPublicKey\$encrypt(m) Encrypt a message. The value m should be less than the modulus, not checked

PaillierPublicKey\$add(a, b) Return the sum of two encrypted messages a and b

PaillierPublicKey\$mult(a, b) Return the product of two encrypted messages a and b

See Also

PaillierPrivateKey which goes hand-in-hand with this object

<code>random.bigz</code>	<i>Return a random big number using the cryptographically secure random number generator from in the sodium package.</i>
--------------------------	--

Description

Return a random big number using the cryptographically secure random number generator from in the sodium package.

Usage

```
random.bigz(nBits)
```

Arguments

`nBits`, the number of bits, which must be a multiple of 8, is not checked for efficiency.

Index

*Topic **datasets**

PaillierKeyPair, [2](#)

PaillierPrivateKey, [3](#)

PaillierPublicKey, [4](#)

homomorpheR, [2](#), [2](#)

homomorpheR-package (homomorpheR), [2](#)

PaillierKeyPair, [2](#)

PaillierPrivateKey, [3](#)

PaillierPublicKey, [4](#)

R6Class, [2–4](#)

random.bigz, [5](#)