Package 'randaes'

February 20, 2015

Title Random number generator based on AES cipher

Description The deterministic part of the Fortuna cryptographic pseudorandom number generator, described by Schneier & Ferguson ``Practical Cryptography"

Version 0.3

Author Thomas Lumley

Maintainer Thomas Lumley <tlumley@u.washington.edu>

License GPL-2

Depends

Suggests

Repository CRAN

Date/Publication 2012-01-13 08:06:00

NeedsCompilation yes

R topics documented:

`randaes-package` *Random number generator based on AES cipher.*

Description

This package implements the deterministic part of the Fortuna cryptographic PRNG described in "Practical Crytography" by Ferguson and Schneier. It does not implement the entropy accumulators needed for secure cryptographic use and is intended for statistical simulation.

Details

Package:	randaes
Type:	Package
Version:	0.1
Date:	2005-08-25
License:	GPL 2

After loading the package, `RNGkind("user")` will use the generator.

The generator encrypts a 128-bit counter using 256-bit AES, resetting the key after 2^{16} encryptions to minimise information leakage from lack of collisions. Each 128-bit result is used to provide two U[0,1] numbers, which thus have the full 53 bits of variability representable in double precision.

Because of the encryption step and the use of 64 bit rather than 32 bit results, this generator is 50-100% slower than the built-in Mersenne Twister. Ferguson and Schneier argue that any computation distinguishing the output of this generator from genuinely random numbers in less than 2^{113} operations implies an attack on the underlying cipher [or, much more likely, a bug in the implementation], so the generator should be particularly useful for rechecking the results of surprising simulations.

Author(s)

Thomas Lumley, using AES code by Christophe Devine

References

Ferguson N, Schneier B. (2003) Practical Cryptography. John Wiley \& Sons.

National Institute of Standard and Technology (2001). "Advanced Encryption Standard". Federal Information Processing Standard 197.

See Also

`Random.user`, `RNGkind`

Examples

```
RNGkind("user")
set.seed(42)
runif(10)

## generates random integers in blocks of 4
.C("fortuna_ints", as.integer(6), integer(6))
```

Index