ENTROPY, RANDOM NUMBERS AND KEYS: WHAT'S GOOD ENOUGH?

John Leiseboer
QuintessenceLabs

RSA CONFERENCE EUROPE 2013

Session ID: ADS-R08

Session Classification: Advanced

In the news

Security Analysis of Pseudo-Random Number Generators with Input:
/dev/random is not Robust

Yevgeniy Dodis[1], David Pointcheval[2], Sylvain Ruhault[3], Damien Vergnaud

Stealthy Dopant-Level Hardware Trojans *

T. Becker[1], Francesco Regazzon
and Wayne P. Burles

On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng

Dan Shumow
Niels Ferguson
Microsoft

MIT Research: Encryption Less Secure Than We Thought

Posted by **Soulskill** on Wednesday August 14, 2013 @02:50PM
from the but-still-pretty-darn-secure dept.

A group of researchers from MIT and the University of Ireland has presented a paper (PDF) showing t
graphic security is wrong. As a result, certain encryption-breaking methods
explains, is that information-theoretic analyses of secure systems have ger
on so-called Shannon entropy, named after the founder of information theo

Fatal crypto flaw in some government-certified smartcards makes forgery a snap

With government certifications this broken, the NSA may not need backdoors.

by **Dan Goodin** - Sept 17 2013, 1:25am +1000

Deliberately flawed? RSA Security tells customers to drop NSA-related encryption algorithm

Published time: September 20, 2013 10:31
Edited time: September 20, 2013 13:15

Get short URL

Introduction

- Relationships
  - Entropy
  - Random numbers
  - Cryptographic keys
- Random bit generator construction
- Entropy sources
- Measuring and testing entropy and randomness
- Putting it all together

Cryptographic strength

- Number of bits of security
  - Algorithm
  - Key size
- Security strength relies on
  - Robust algorithms
  - Correct implementation
  - No feasible mathematical attack or advances
  - Brute force attack infeasible

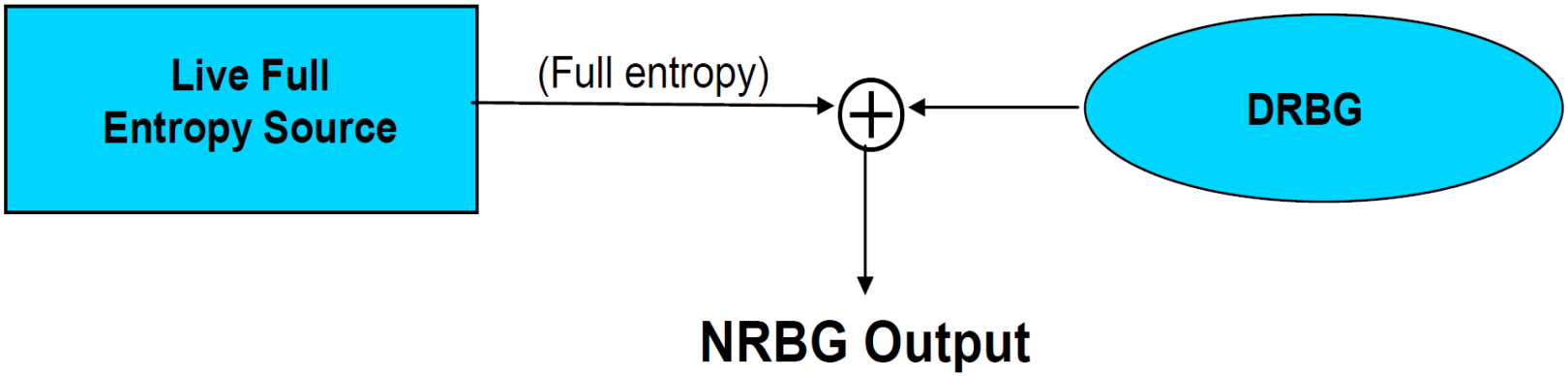
Random use cases

- RNG seed
- Key generator seed
- IV
- Nonce
- Random challenge
- Authentication
- DSA signing
- One-time pad cipher
- Zero knowledge proof
- E-voting
- Random beacon
  - Transaction protection
  - Contract signing
  - PII protection
  - Cloud entropy

NIST's RBG standards

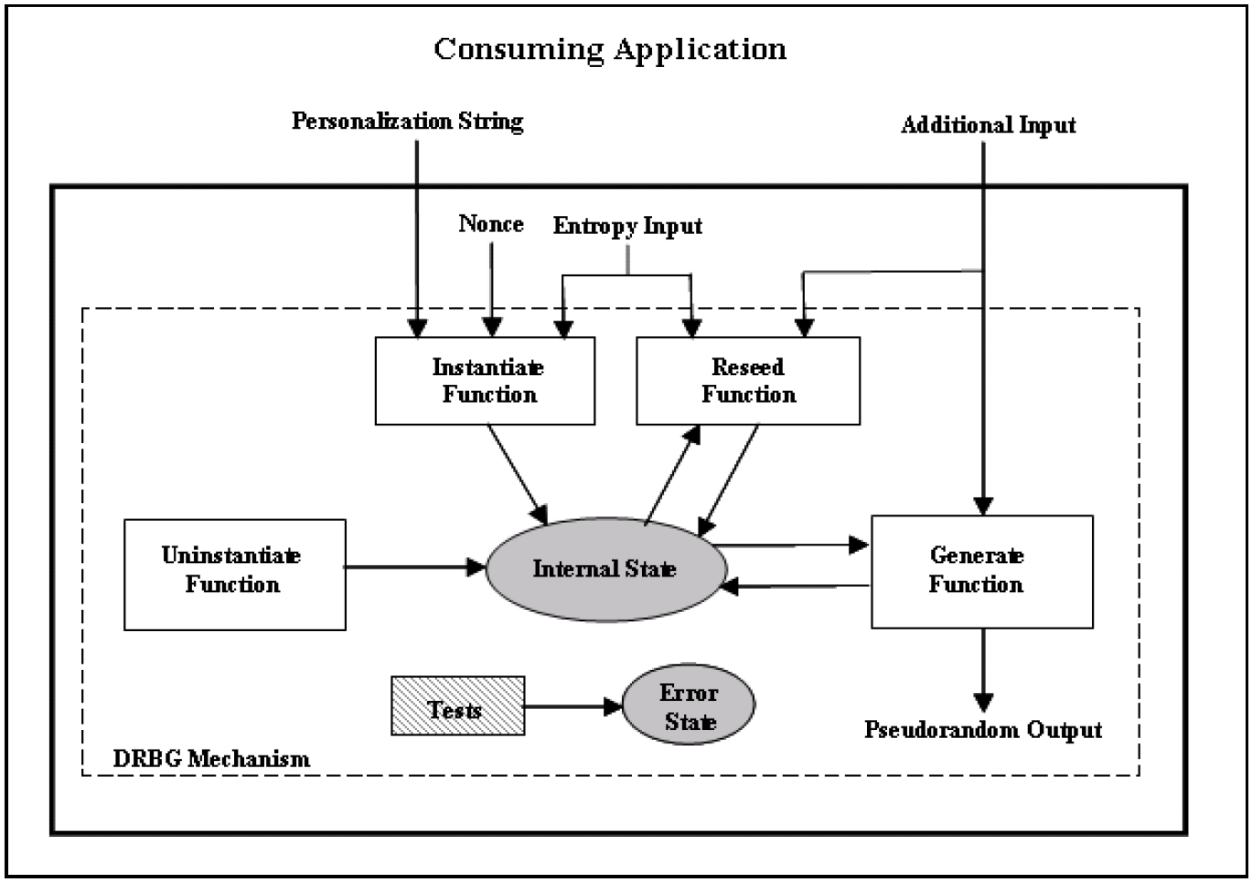
- NIST SP 800-90A: DRBG
- NIST SP 800-90B: Entropy source
- NIST SP 800-90C: RBG = Entropy source + DRBG

- Example:



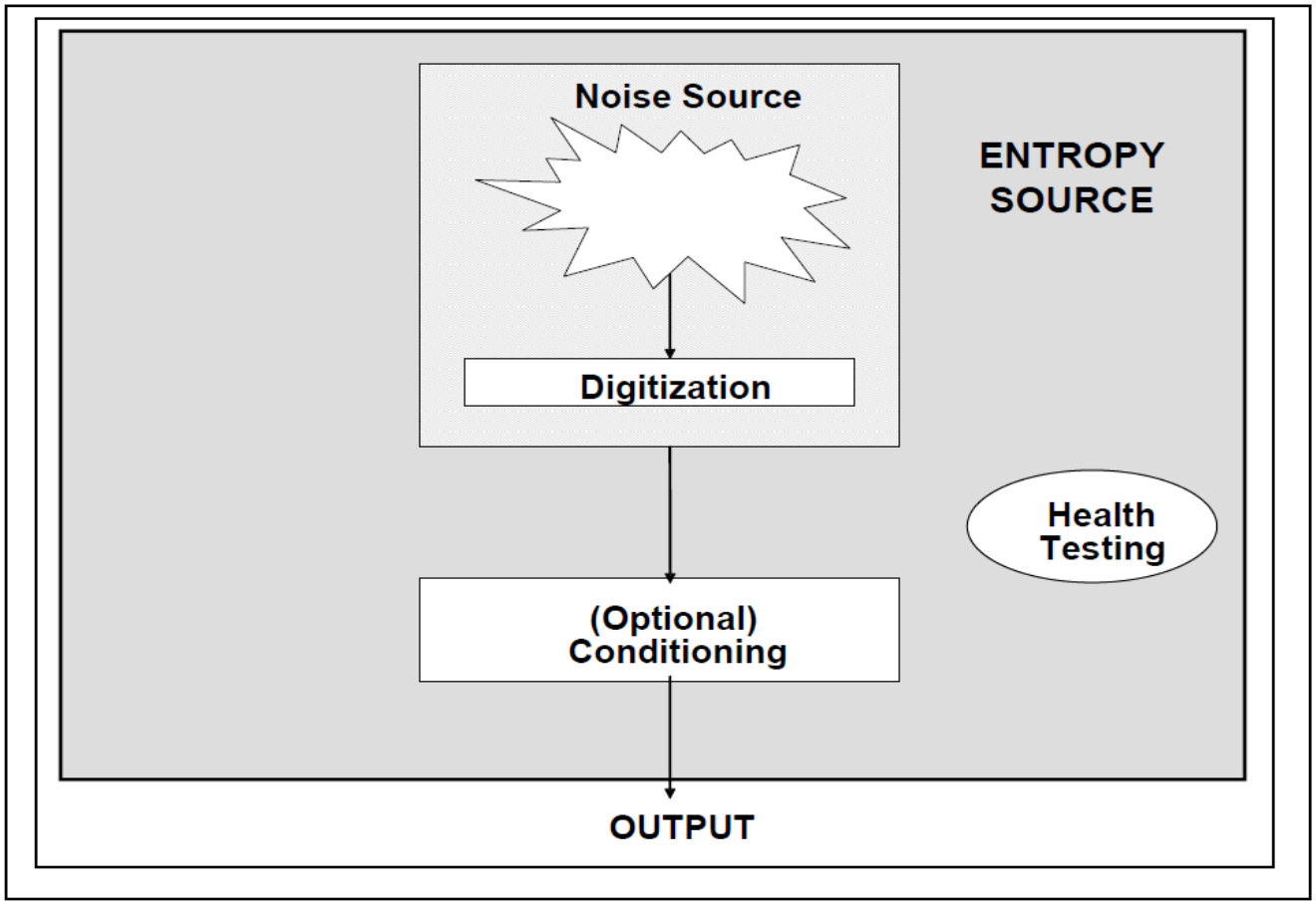
NIST SP 800-90A

DRBG Functional Model



NIST SP 800-90B

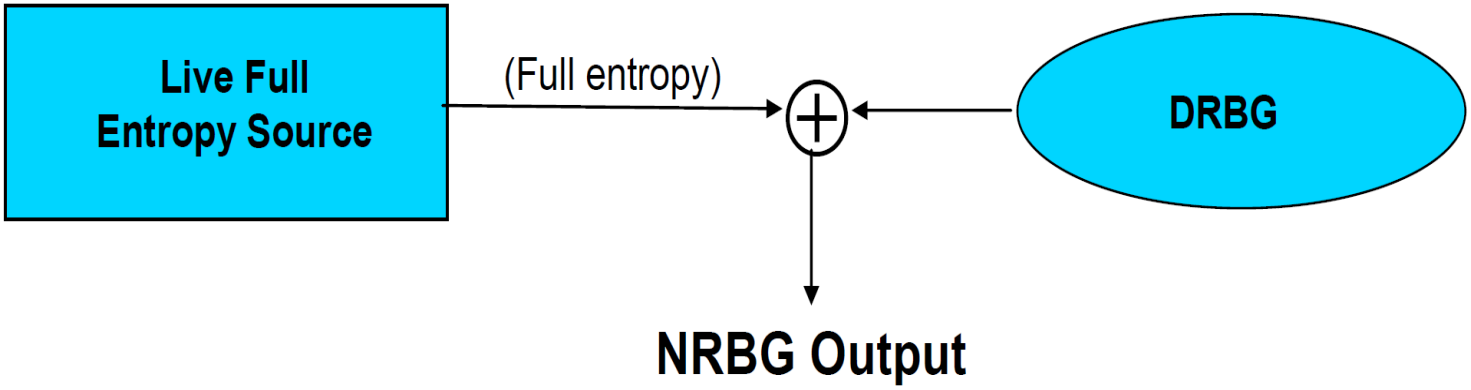
Entropy Source Functional Model



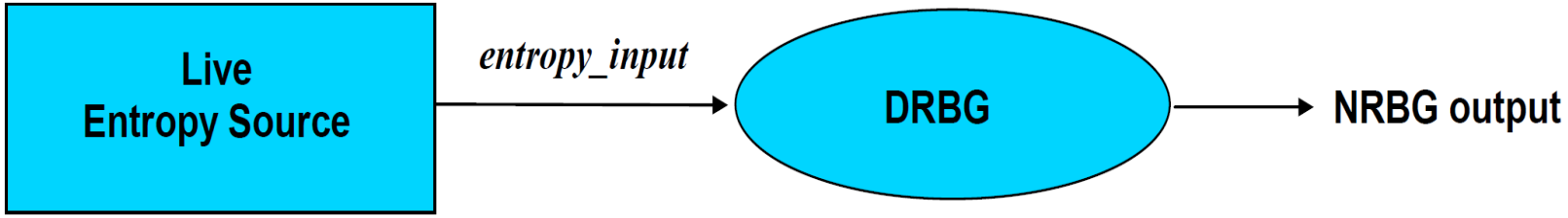
NIST SP 800-90C

RBG Construction

Enhanced NRBG – XOR Construction

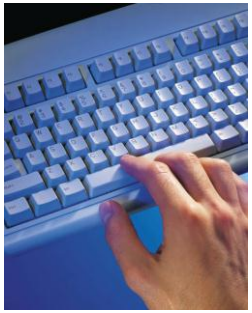


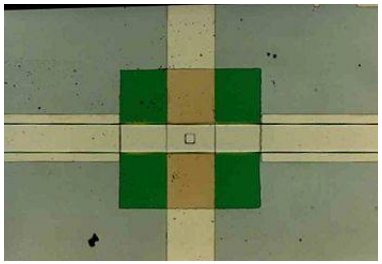
Enhanced NRBG – Oversampling Construction



Entropy sources

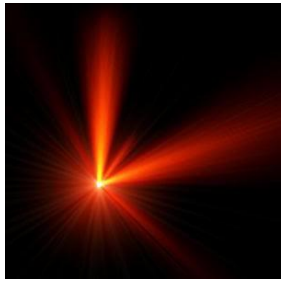









[redacted] Estimating entropy

- 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, …
- 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 9, ….
- Independent and identical distribution (IID)

```
Non-i.i.d. Min-Entropy Estimation Test Results:
Collision test                      4.310757 bits per 8-bit symbol
Partial collection test             2.467009 bits per 8-bit symbol
Markov test                         5.692803 bits per 6-bit symbol (remapped due to
test limit)
Compression test                    3.401208 bits per 8-bit symbol
Frequency test                      6.699898 bits per 8-bit symbol

Sanity Check Results:
Compression test                    passed - (840168 bits, 840800 bits, 840336 bits,
839440 bits, 840688 bits, 839776 bits, 840312 bits, 839688 bits, 839800 bits, 840016
bits)
Collision test                      passed - (0 13-symbol values with a count of 3 or
more, 0 colliding 13-symbol values in total)

Min-entropy estimate is 2.467009 bits per 8-bit symbol, based on a 95% confidence
interval.
```

Random number tests

- NIST STS
  - http://csrc.nist.gov/groups/ST/toolkit/rng/index.html
  - *"…no set of statistical tests can absolutely certify a generator as appropriate…"*
- Dieharder
  - http://www.phy.duke.edu/~rgb/General/dieharder.php
  - *"dieharder is a tool designed to permit one to push a weak generator to unambiguous failure …"*

Dieharder RNGs and tests

```
#=============================================================================#
#    Id Test Name           | Id Test Name           | Id Test Name           #
#=============================================================================#
|   000 borosh13            |001 cmrg                |002 coveyou             |
|   003 fishman18           |004 fishman20           |005 fishman2x           |
|   006 gfsr4               |007 knuthran            |008 knuthran2           |
|   009 knuthran2002        |010 lecuyer21           |011 minstd              |
|   012 mrg                 |013 mt19937             |014 mt19937_1999        |
|   015 mt19937_1998        |016 r250                |017 ran0                |
|   018 ran1                |019 ran2                |020 ran3                |
|   021 rand                |022 rand48              |023 random128-bsd       |
|   024 random128-glibc2    |025 random128-libc5     |026 random256-bsd       |
|   027 random256-glibc2    |028 random256-libc5     |029 random32-bsd        |
|   030 random32-glibc2     |031 random32-libc5      |032 random64-bsd        |
|   033 random64-glibc2     |034 random64-libc5      |035 random8-bsd         |
|   036 random8-glibc2      |037 random8-libc5       |038 random-bsd          |
|   039 random-glibc2       |040 random-libc5        |041 randu               |
|   042 ranf                |043 ranlux              |044 ranlux389           |
|   045 ranlxd1             |046 ranlxd2             |047 ranlxs0             |
|   048 ranlxs1             |049 ranlxs2             |050 ranmar              |
|   051 slatec              |052 taus                |053 taus2               |
|   054 taus113             |055 transputer          |056 tt800               |
|   057 uni                 |058 uni32               |059 vax                 |
|   060 waterman14          |061 zuf                 |                        |
#=============================================================================#
|   200 stdin_input_raw     |201 file_input_raw      |202 file_input          |
|   203 ca                  |204 uvag                |205 AES_OFB             |
|   206 Threefish_OFB       |207 XOR (supergenerator)|208 kiss                |
|   209 superkiss           |                        |                        |
#=============================================================================#
|   400 R_wichmann_hill     |401 R_marsaglia_multic. |402 R_super_duper       |
|   403 R_mersenne_twister  |404 R_knuth_taocp       |405 R_knuth_taocp2      |
#=============================================================================#
|   500 /dev/random         |501 /dev/urandom        |                        |
#=============================================================================#
```

Dieharder RNGs and tests

```
#==============================================================================#
#    Id Test Name          | Id Test Name          | Id Test Name           #
#==============================================================================#
|   000 boro                                                                   |
|   003 fish                                                                   |
|   006 gfsr                                                                   |
|   009 knut                                                                   |
|   012 mrg                                                                    |
|   015 mt19                                                                   |
|   018 ran1                                                                   |
|   021 rand                                                                   |
|   024 rand                                                                   |
|   027 rand                                                                   |
|   030 rand                                                                   |
|   033 rand                                                                   |
|   036 rand                                                                   |
|   039 rand                                                                   |
|   042 ranf                                                                   |
|   045 ranl                                                                   |
|   048 ranl                                                                   |
|   051 slat                                                                   |
|   054 taus                                                                   |
|   057 uni                                                                    |
|   060 wate                                                                   |
#===========                                                                   |
|   200 stdi                                                                   |
|   203 ca                                                                     |
|   206 Thre                                                                   |
|   209 supe                                                                   |
#===========                                                                   |
|   400 R_wi                                                                   |
|   403 R_me                                                                   |
#===========                                                                   |
|   500 /dev/random         |501 /dev/urandom       |                          |
#==============================================================================#
```

```
#=========================================================================
#                 Diehard "Birthdays" test (modified).
# Each test determines the number of matching intervals from 512
# "birthdays" (by default) drawn on a 24-bit "year" (by
# default).  This is repeated 100 times (by default) and the
# results cumulated in a histogram.  Repeated intervals should be
# distributed in a Poisson distribution if the underlying generator
# is random enough, and a a chisq and p-value for the test are
# evaluated relative to this null hypothesis.
#
# It is recommended that you run this at or near the original
# 100 test samples per p-value with -t 100.
#
# Two additional parameters have been added. In diehard, nms=512
# but this CAN be varied and all Marsaglia's formulae still work.  It
# can be reset to different values with -x nmsvalue.
# Similarly, nbits "should" 24, but we can really make it anything
# we want that's less than or equal to rmax_bits = 32.  It can be
# reset to a new value with -y nbits.  Both default to diehard's
# values if no -x or -y options are used.
#=========================================================================
```

Dieharder RNGs and tests

```
#==========================================================================#
#    Id Test Name          | Id Test Name          | Id Test Name          #
#==========================================================================#
|   000 boro                                                               |
|   003 fish                                                               |
|   006 gfsr                                                               |
|   009 knut                                                               |
|   012 mrg                                                                |
|   015 mt19                                                               |
|   018 ran1                                                               |
|   021 rand                                                               |
|   024 rand                                                               |
|   027 rand                                                               |
|   030 rand                                                               |
|   033 rand                                                               |
|   036 rand                                                               |
|   039 rand                                                               |
|   042 ranf                                                               |
|   045 ranl                                                               |
|   048 ranl                                                               |
|   051 slat                                                               |
|   054 taus                                                               |
|   057 uni                                                                |
|   060 wate                                                               |
#==========                                                                #
|   200 stdi                                                               |
|   203 ca                                                                 |
|   206 Thre                                                               |
|   209 supe                                                               |
#==========                                                                #
|   400 R_wi                                                               |
|   403 R_me                                                               |
#==========                                                                #
|   500 /dev/random       |501 /dev/urandom       |                        |
#==========================================================================#
```

```
#==========================================================================
#          Diehard Overlapping 5-Permutations Test.
# This is the OPERM5 test.  It looks at a sequence of one mill-
# ion 32-bit random integers.  Each set of five consecutive
# integers can be in one of 120 states, for the 5! possible or-
# derings of five numbers.  Thus the 5th, 6th, 7th,...numbers
# each provide a state. As many thousands of state transitions
# are observed,  cumulative counts are made of the number of
# occurences of each state.  Then the quadratic form in the
# weak inverse of the 120x120 covariance matrix yields a test
# equivalent to the likelihood ratio test that the 120 cell
# counts came from the specified (asymptotically) normal dis-
# tribution with the specified 120x120 covariance matrix (with
# rank 99).  This version uses 1,000,000 integers, twice.
#
# Note that Dieharder runs the test 100 times, not twice, by
# default.
#==========================================================================
```

Dieharder RNGs and tests

```
#==========================================================================#
#    Id Test Name          | Id Test Name          | Id Test Name          #
#==========================================================================#
|   000 boro                |                       |                       |
|   003 fish                |                       |                       |
|   006 gfsr                |                       |                       |
|   009 knut                |                       |                       |
|   012 mrg                 |                       |                       |
|   015 mt19                |                       |                       |
|   018 ran1                |                       |                       |
|   021 rand                |                       |                       |
|   024 rand                |                       |                       |
|   027 rand                |                       |                       |
|   030 rand                |                       |                       |
|   033 rand                |                       |                       |
|   036 rand                |                       |                       |
|   039 rand                |                       |                       |
|   042 ranf                |                       |                       |
|   045 ranl                |                       |                       |
|   048 ranl                |                       |                       |
|   051 slat                |                       |                       |
|   054 taus                |                       |                       |
|   057 uni                 |                       |                       |
|   060 wate                |                       |                       |
#===========                |                       |                       |
|   200 stdi                |                       |                       |
|   203 ca                  |                       |                       |
|   206 Thre                |                       |                       |
|   209 supe                |                       |                       |
#===========                |                       |                       |
|   400 R_wi                |                       |                       |
|   403 R_me                |                       |                       |
#===========                |                       |                       |
|   500 /dev/random         |501 /dev/urandom       |                       |
#==========================================================================#
```

```
#=================================================================
#               Diehard 32x32 Binary Rank Test
# This is the BINARY RANK TEST for 32x32 matrices. A random 32x
# 32 binary matrix is formed, each row a 32-bit random integer.
# The rank is determined. That rank can be from 0 to 32, ranks
# less than 29 are rare, and their counts are pooled with those
# for rank 29.  Ranks are found for 40,000 such random matrices
# and a chisquare test is performed on counts for ranks  32,31,
# 30 and <=29.
#
# As always, the test is repeated and a KS test applied to the
# resulting p-values to verify that they are approximately uniform.
#=================================================================
```

Dieharder RNGs and tests

```
#=============================================================================#
#    Id Test Name          | Id Test Name          | Id Test Name           #
#==========
|   000 boro
|   003 fish
|   006 gfsr
|   009 knut
|   012 mrg
|   015 mt19
|   018 ran1
|   021 rand
|   024 rand
|   027 rand
|   030 rand
|   033 rand
|   036 rand
|   039 rand
|   042 ranf
|   045 ranl
|   048 ranl
|   051 slat
|   054 taus
|   057 uni
|   060 wate
#==========
|   200 stdi
|   203 ca
|   206 Thre
|   209 supe
#==========
|   400 R_wi
|   403 R_me
#==========
|   500 /dev
#==========
```

```
#=============================================================================#
#                    Diehard Bitstream Test.
# The file under test is viewed as a stream of bits. Call them
# b1,b2,... .  Consider an alphabet with two "letters", 0 and 1
# and think of the stream of bits as a succession of 20-letter
# "words", overlapping.  Thus the first word is b1b2...b20, the
# second is b2b3...b21, and so on.  The bitstream test counts
# the number of missing 20-letter (20-bit) words in a string of
# 2^21 overlapping 20-letter words.  There are 2^20 possible 20
# letter words.  For a truly random string of 2^21+19 bits, the
# number of missing words j should be (very close to) normally
# distributed with mean 141,909 and sigma 428.  Thus
# (j-141909)/428 should be a standard normal variate (z score)
# that leads to a uniform [0,1) p value.  The test is repeated
# twenty times.
#
#                            NOTE WELL!
#
# The test is repeated 100 times by default in dieharder, but the
# size of the sample is fixed (tsamples cannot/should not be
# varied from the default).  The sigma of this test REQUIRES the
# use of overlapping samples, and overlapping samples are not
# independent.  If one uses the non-overlapping version of this
# test, sigma = 290 is used instead, smaller because now there
# are 2^21 INDEPENDENT samples.
#=============================================================================#
```

Dieharder test results

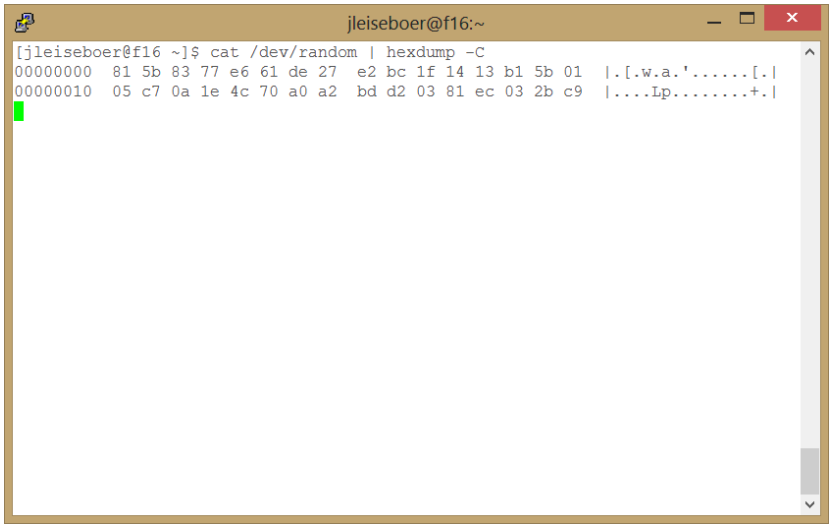
```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
 file_input_raw|                        /dev/urandom|  3.14e+06  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
   diehard_birthdays|   0|       100|     100|0.74056677|  PASSED
      diehard_operm5|   0|   1000000|     100|0.06667553|  PASSED
  diehard_rank_32x32|   0|     40000|     100|0.02200848|  PASSED
    diehard_rank_6x8|   0|    100000|     100|0.09107953|  PASSED
   diehard_bitstream|   0|   2097152|     100|0.31603737|  PASSED
        diehard_opso|   0|   2097152|     100|0.34660453|  PASSED
        diehard_oqso|   0|   2097152|     100|0.37564518|  PASSED
         diehard_dna|   0|   2097152|     100|0.36650750|  PASSED
diehard_count_1s_str|   0|    256000|     100|0.48806562|  PASSED
diehard_count_1s_byt|   0|    256000|     100|0.55880694|  PASSED
 diehard_parking_lot|   0|     12000|     100|0.83939978|  PASSED
    diehard_2dsphere|   2|      8000|     100|0.04126659|  PASSED
    diehard_3dsphere|   3|      4000|     100|0.06311990|  PASSED
     diehard_squeeze|   0|    100000|     100|0.50805779|  PASSED
        diehard_sums|   0|       100|     100|0.08429191|  PASSED
```

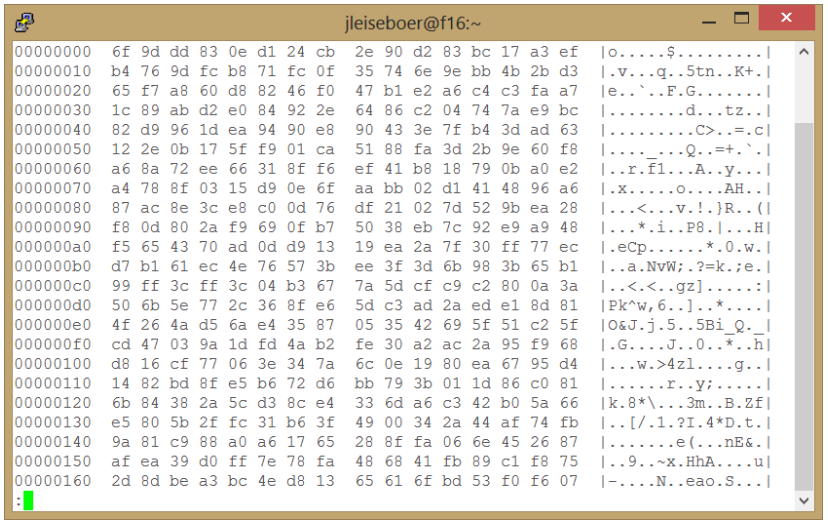
Practical issues

- Entropy sources
- Hardware and software implementations
- Standards compliance
- Performance requirements
- Security requirements
- User experience – application developer and end-user
- Trust

/dev/random and /dev/urandom

- Try this:
  - $ cat /dev/random
  - $ cat /dev/urandom





- Implementations differ
  - Blocking vs. non-blocking
  - Entropy quality

Hardware RNG's

- Microcontrollers
- Intel RDRAND and RDSEED
- HSM's
- RNG appliances, cards and components
- DIY
- Entropy vs. random numbers
  - Deterministic output (DRNG/PRNG)
  - Non-deterministic (NRBG/TRNG)
  - "Full entropy" vs. cryptographic strength

Application interfaces

- PKCS#11: C_GenerateRandom()
- Java: SecureRandom()
- Microsoft CAPI and CNG: CryptGenKey()
- OpenSSL: RAND_bytes()
- Others

- OASIS KMIP – client/server network protocol

Conclusion

- Relationships
  - Entropy
  - Random numbers
  - Cryptographic keys
- Random bit generator construction
- Entropy sources
- Measuring and testing entropy and randomness
- Putting it all together

Learning points

- Entropy
  - Identify sources of entropy and assess min entropy
- RBG construction
  - RBG contains required and approved components
- Seeding and re-seeding PRNGs
  - PRNG is seeded appropriately before use
  - PRNG is re-seeded as required to meet security requirements
- Correctly using the API
  - Use the correct functions in the correct order

Thank you!

John Leiseboer

QuintessenceLabs

jl@quintessencelabs.com

www.quintessencelabs.com

#RSAC

RSA CONFERENCE EUROPE 2013