Random number generator: testing and whitening

Andrei Romashchenko & Alexander Shen

ESCAPE / LIRMM

invitation to an internship (2020)

What is a random sequence of digits?

- `0000000000000000000000000000000000000000`

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101

- `0000000000000000000000000000000000000000`
- `0101010101010101010101010101010101010101`
- `1101111011100001101101010110001111101010`

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001

Can you guess which sequence here is truly random?

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001
  (btw, the last sequence is a binary expansion of π)

Can you guess which sequence here is truly random?

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001
  (btw, the last sequence is a binary expansion of π)

Can you guess which sequence here is truly random?

(Yes, to prepare one of these sequences we tossed a real coin 40 times.)

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001
  (btw, the last sequence is a binary expansion of π)

Can you guess which sequence here is truly random?
(Yes, to prepare one of these sequences we tossed a real coin 40 times.)

In some sense, all 40-bit sequences are equally random
(they all have the same probability 2^{-40} in a fair coin model)

- 0000000000000000000000000000000000000000
- 0101010101010101010101010101010101010101
- 1101111011100001101101010110001111101010
- 1100100100001111110110101010001000100001
  (btw, the last sequence is a binary expansion of π)

Can you guess which sequence here is truly random?
(Yes, to prepare one of these sequences we tossed a real coin 40 times.)

In some sense, all 40-bit sequences are equally random
(they all have the same probability 2^{-40} in a fair coin model),
but some look more random than others.

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

Possible answer:

random = no regularities

non-random = apparent or hidden regularities

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

Possible answer:

random = no regularities

non-random = apparent or hidden regularities = a short description

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

Possible answer:
random = no regularities
non-random = apparent or hidden regularities = a short description

In practice:
various statistical tests (sequences with no regularities pass these tests)

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

Possible answer:
random = no regularities
non-random = apparent or hidden regularities = a short description

In practice:
various statistical tests (sequences with no regularities pass these tests)

In theory: the notion of Kolmogorov complexity,
$C(x_1 \ldots x_n) =$ length of the shortest program that prints $x_1 \ldots x_n$

In some sense, all n-bit sequences are equally random
(they all have the same probability 2^{-n} in a fair coin model).

What is the difference between truly random and non-random sequences?

Possible answer:
random = no regularities
non-random = apparent or hidden regularities = a short description

In practice:
various statistical tests (sequences with no regularities pass these tests)

In theory: the notion of Kolmogorov complexity,
$C(x_1 \dots x_n) =$ length of the shortest program that prints $x_1 \dots x_n$

$x_1 \dots x_n$ is random means $C(x_1 \dots x_n) \approx n$
$x_1 \dots x_n$ is non-random means $C(x_1 \dots x_n) \ll n$

traditional **practical** usage of randomness

traditional **practical** usage of randomness

- cryptography (random secret keys etc.)

traditional **practical** usage of randomness

- cryptography (random secret keys etc.)
- randomized algorithms (Monte-Carlo methods etc.)

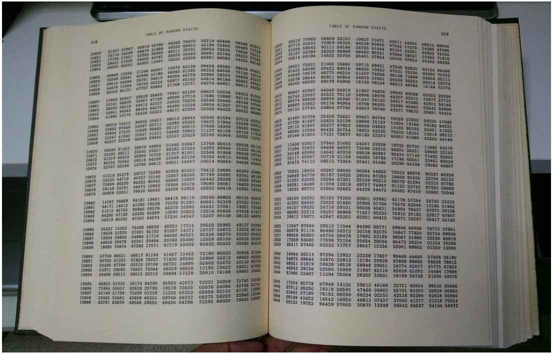
traditional **practical** usage of randomness

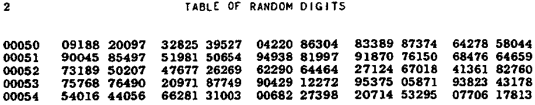
- cryptography (random secret keys etc.)
- randomized algorithms (Monte-Carlo methods etc.)
- random combinatorial structures (e.g., random error correcting codes)

traditional **practical** usage of randomness

- cryptography (random secret keys etc.)
- randomized algorithms (Monte-Carlo methods etc.)
- random combinatorial structures (e.g., random error correcting codes)

Big practical question: where do we get truly random bits?



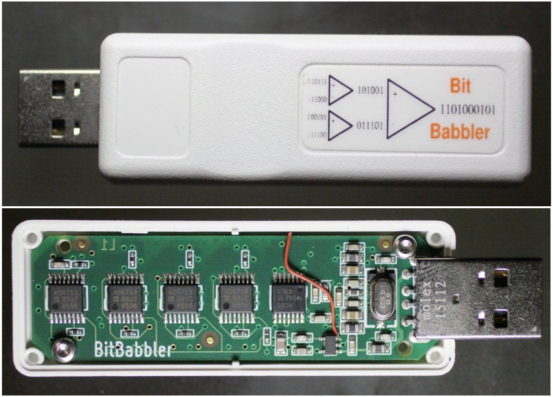


2 TABLE OF RANDOM DIGITS

00050	09188	20097	32825	39527	04220	86304	83389	87374	64278	58044
00051	90045	85497	51981	50654	94938	81997	91870	76150	68476	64659
00052	73189	50207	47677	26269	62290	64464	27124	67018	41361	82760
00053	75768	76490	20971	87749	90429	12272	95375	05871	93823	43178
00054	54016	44056	66281	31003	00682	27398	20714	53295	07706	17813

Rand Corporation, *A Million Random Digits with 100,000 Normal Deviates* (1955)

[random digits kindly generated for us in 1955]



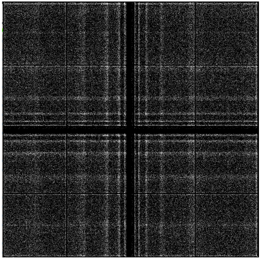
[random digits from a noise in an electric circuit]

[random bits from quantum phenomena]

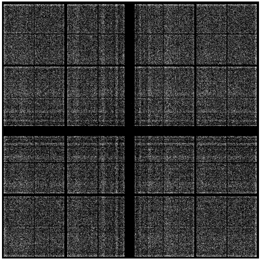
Do these gadgets give perfectly random bits?

Do these gadgets give perfectly random bits? Not really.

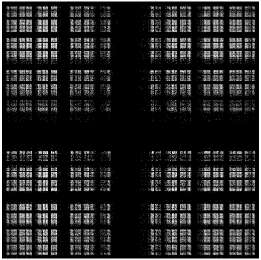
Do these gadgets give perfectly random bits? Not really. Example:



100 kHz

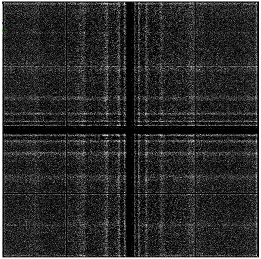


default rate 2.5 MHz

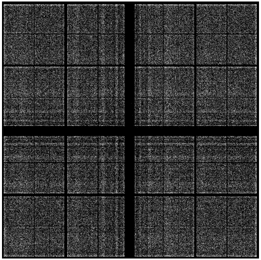


5 MHz

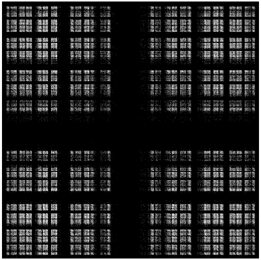
Do these gadgets give perfectly random bits? Not really. Example:



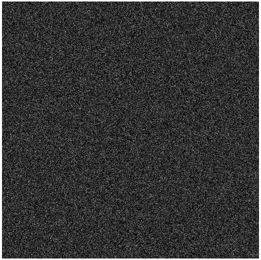
100 kHz



default rate 2.5 MHz



5 MHz



successful whitening: XOR of 3 (apparently biased) data flows looks pretty random

First practical problem:

Randomly produced bits are usually not truly random

First practical problem:

Randomly produced bits are usually not truly random,
they fail standard randomness tests.

First practical problem:

Randomly produced bits are usually not truly random,
they fail standard randomness tests.

Second practical problem:

You cannot trust blindly the standard implementations of randomness tests

First practical problem:

Randomly produced bits are usually not truly random,
they fail standard randomness tests.

Second practical problem:

You cannot trust blindly the standard implementations of randomness tests
(mathematically unsound tests, errors in the code).

The challenges:

The challenges:

- cleanup/enhance existing randomness tests

The challenges:

- cleanup/enhance existing randomness tests
- experiments with physical random generators

The challenges:

- cleanup/enhance existing randomness tests
- experiments with physical random generators
- try new approaches to whitening

The challenges:

- cleanup/enhance existing randomness tests
- experiments with physical random generators
- try new approaches to whitening
- apply (certainly non-perfect) pseudo-random generators to produce useful random objects (e.g., error correcting codes)

Internship proposal:

Generation of random bits is a classical problem known in the context of pseudo-random generators and also in connection with of truly random physical processes (there exist electronic devices that produce random bits using an unpredictable physical noise or intrinsically nondeterministic quantum phenomena). However, the quality of physical generators of random bits remains badly founded and poorly tested. The first objective of this project is an experimental study of the validity and quality of several physical random numbers generators.

When we talk about the quality of random or pseudo-random generators, we have to use randomness tests. The second objective of the project is an inventory and revision of statistical tests for random and pseudo-random generators. We suggest to improve the quality of statistical tests and develop new techniques of “whitening” that improves the quality of non-ideal sources of random bits. Another axis of the project is a conversion of various probabilistic proofs into unconventional randomness tests.

Prerequisites: Basic knowledge of probability and statistics, and solid programming skills. The main tools in the project are pretty standard: C / gcc / Linux. The project requires not only writing your own code but also reading and maintaining the code that already exists.