# Случайные биты: теория и практика (online test, Computer Science Club, СПб)

Александр Шень, CNRS & University of Montpellier,
RaCAF ANR project

26 апреля 2020

# честная монета

- вероятность 1/2, бросания независимы
- что это значит
- «теория вероятностей доказывает»

# как проверить?

- нормальный цикл: гипотеза, эксперимент, согласуются или нет
- все последовательности из 100 битов имеют одинаковую вероятность $2^{-100}$
- как же тогда одни результаты опровергают гипотезу, а другие нет?
- примеры: ОТК для случайных битов, рандомизация заданий

# тестирование гипотез

- гипотеза = распределение на исходах
- тест = классификация исходов: пройден или нет
- significance level: вероятность (согласно гипотезе) не пройти тест

# ловушки

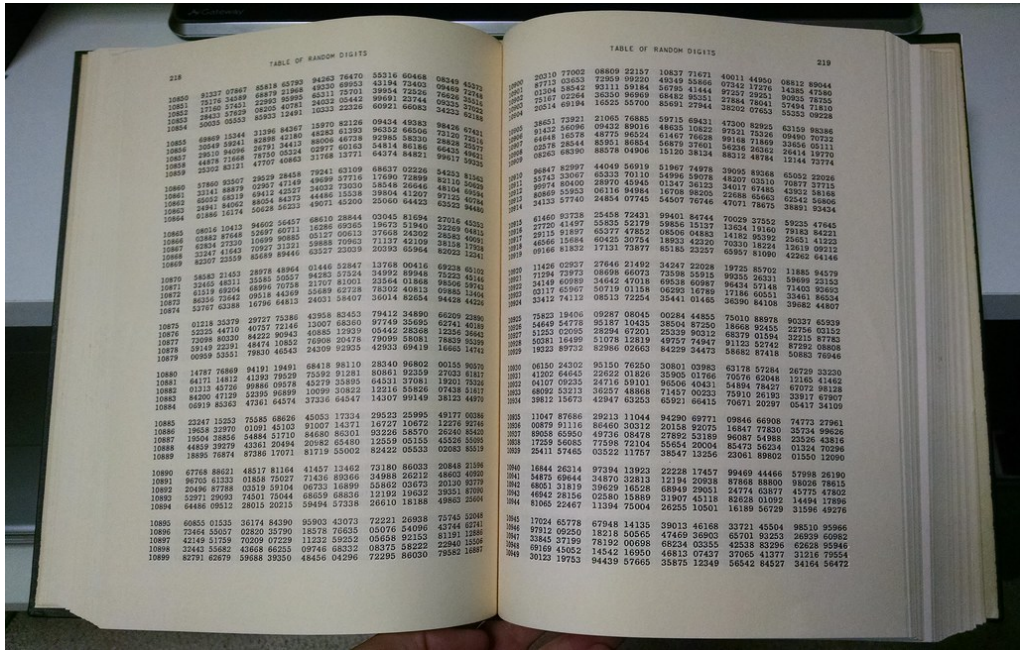
- тест после эксперимента
- Bonferroni correction

- отступление о статистике и интерпретация: covid
  - Чехия: смещённая выборка https://www.reuters.com/article/us-health-coronavirus-czech-tests/czechs-pack-test-centres-in-large-coronavirus-antibody
  - Калифорния: antibodies tests specificity https://www.sciencemag.org/news/2020/04/antibody-surveys-suggesting-vast-undercount-coronavirus https://www.evaluate.com/vantage/articles/analysis/spotlight/covid-19-antibody-tests-face-very-specific-problem
  - Нью-Йорк 88% https://www.independent.co.uk/news/world/americas/coronavirus-us-new-york-hospital-ventilator-death-rate.html

# для чего нужны случайные биты?

- лотереи, игры,...
- выборка в статистических исследованиях
- моделирование по методу Монте-Карло
- вообще моделирование
- вероятностные алгоритмы
  - быстрая сортировка
  - проверка простоты
  - оценки параметров
- криптография (one-time pad, secret sharing)

# книга случайности

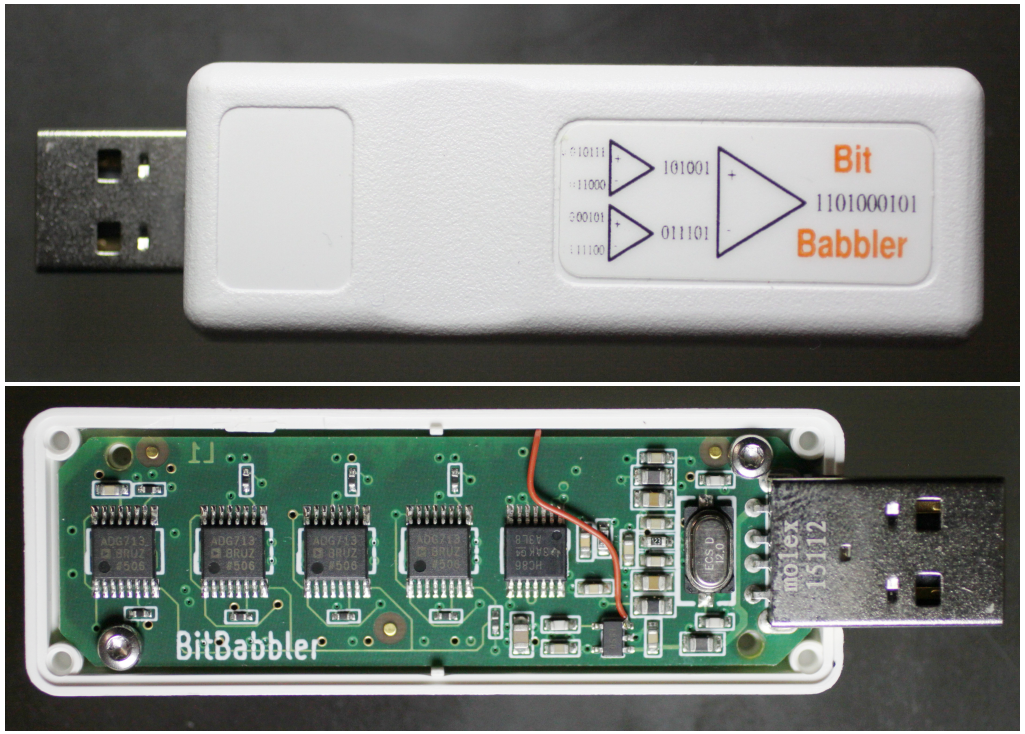


2 TABLE OF RANDOM DIGITS

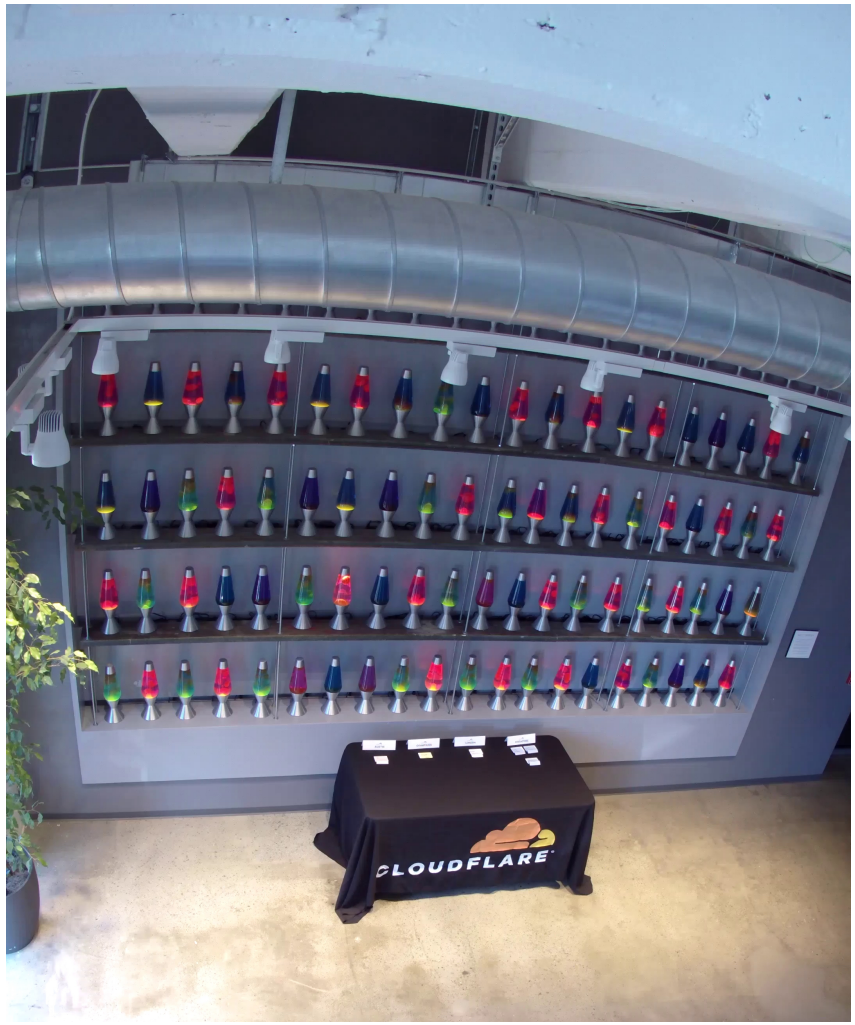
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 00050 | 09188 | 20097 | 32825 | 39527 | 04220 | 86304 | 83389 | 87374 | 64278 | 58044 |
| 00051 | 90045 | 85497 | 51981 | 50654 | 94938 | 81997 | 91870 | 76150 | 68476 | 64659 |
| 00052 | 73189 | 50207 | 47677 | 26269 | 62290 | 64464 | 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)

# bitbabbler (Австралия)

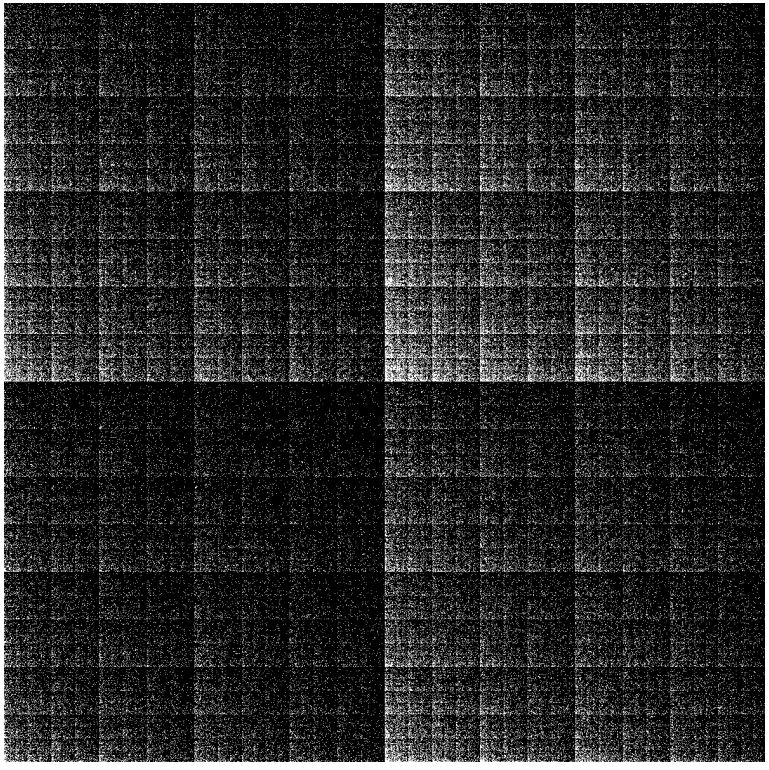


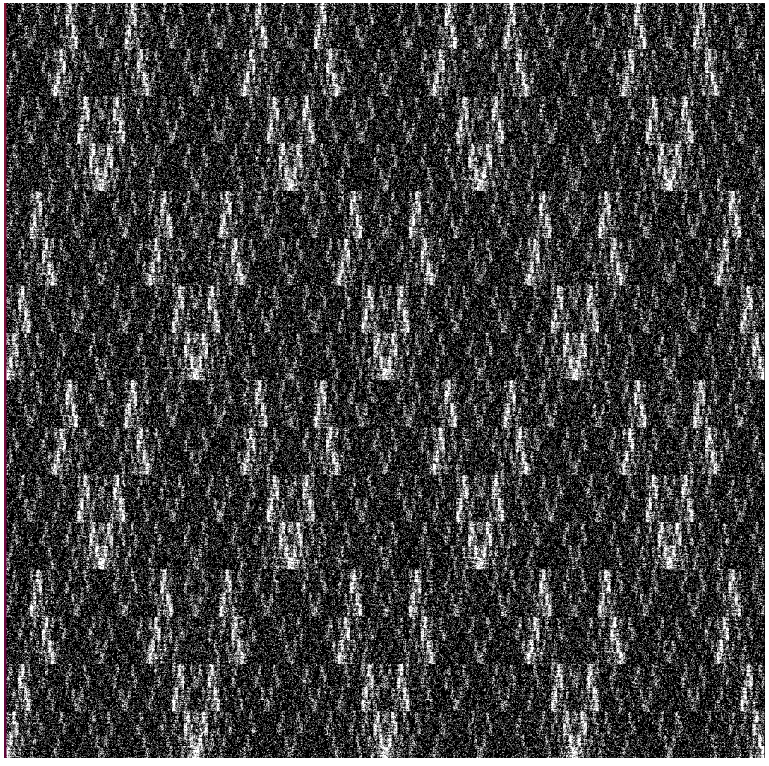
# Cloudflare



# тест: видимые отклонения

OneRNG [g1 g2]





# тест: $\chi^2$

```
$ sh g8
Entropy = 7.999997 bits per byte.

Chi square distribution for 102400000 samples is
484.39, and randomly would exceed this value less
than 0.01 percent of the times.

Arithmetic mean value of data bytes is 127.5025
(127.5 = random).
Monte Carlo value for Pi is 3.141912310 (error 0.01
percent).
Serial correlation coefficient is 0.000126 (totally
uncorrelated = 0.0).
```

# тест: dieharder

```
# Diehard 6x8 Binary Rank Test
This is the BINARY RANK TEST for 6x8 matrices. From each of six random
32-bit integers from the generator under test, a specified byte is
chosen, and the resulting six bytes form a 6x8 binary matrix whose rank
is determined. That rank can be from 0 to 6, but ranks 0,1,2,3 are rare;
their counts are pooled with those for rank 4. Ranks are found for
100,000 random matrices, and a chi-square test is performed on counts
for ranks 6,5 and <= 4.
As always, the test is repeated and a KS test applied to the resulting
p-values to verify that they are approximately uniform.
# |  |  |  |  |  |  |  |  |  |
# |  |  |  |  |  |  |  |  |  |
# |  |  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|  |  |  |  |  |  |  |  |
# |*|*|*|*|  |  |  |  |  |
test_name diehard_rank_6x8
p-value 0.00000000
Assessment FAILED
```

# тест: компрессор

- сжатие более чем на $n$ битов: вероятность меньше $2^{-n}$
- lossless decompression (сжатие не нужно)
- колмогоровская сложность: длина кратчайшей программы, порождающей $x$
- алгоритмическая теория информации: случайность $\Longleftrightarrow$ (сложность $\approx$ длина)
- ```
  $ ls -l tmp.dat
  1048576 Apr 25 14:29 tmp.dat
  $ bzip2 tmp.dat
  $ ls -l tmp.dat.bz2
  1040002 Apr 25 14:29 tmp.dat.bz2
  ```
  ```

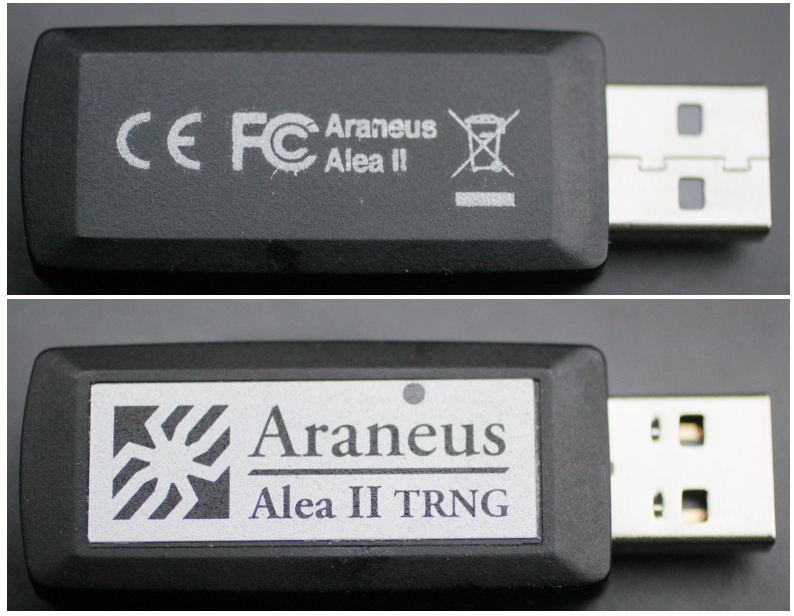
# псевдослучайность

- фон Нейман
- $x \longmapsto ax + b \bmod N$
- число $\pi$ нормально?
- «криптографические» (SHA etc.)
- Blum – Micali – Yao
- `gsl`, `dieharder`

# генераторы

- общее устройство: источник «шума»
- whitening / conditioning / randomness extraction
- доступ к исходному шуму?
- скорость выдачи битов
- цена
- доступ из программ / системы
- open source (hard/soft)

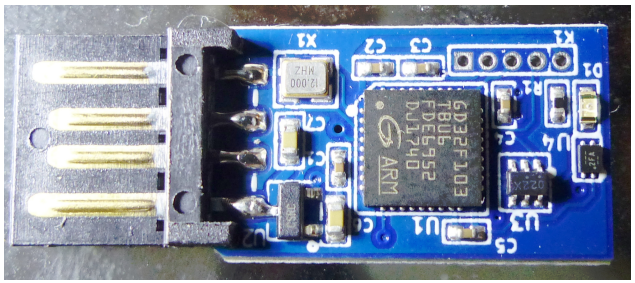
# Araneus




## $$$, стабилитрон, 100 kbits/s, raw=no, whitening=?

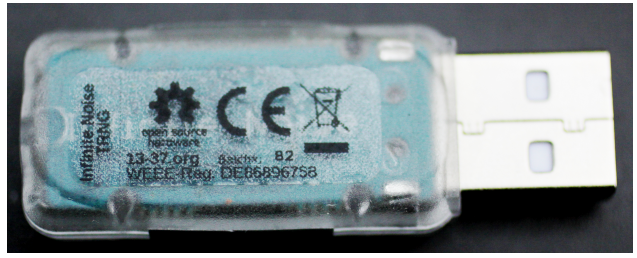
“The raw output bits from the A/D converter are then further processed by an embedded microprocessor to combine the entropy from multiple samples into each final output bit, resulting in a random bit stream that is practically free from bias and correlation”

# Gniibe



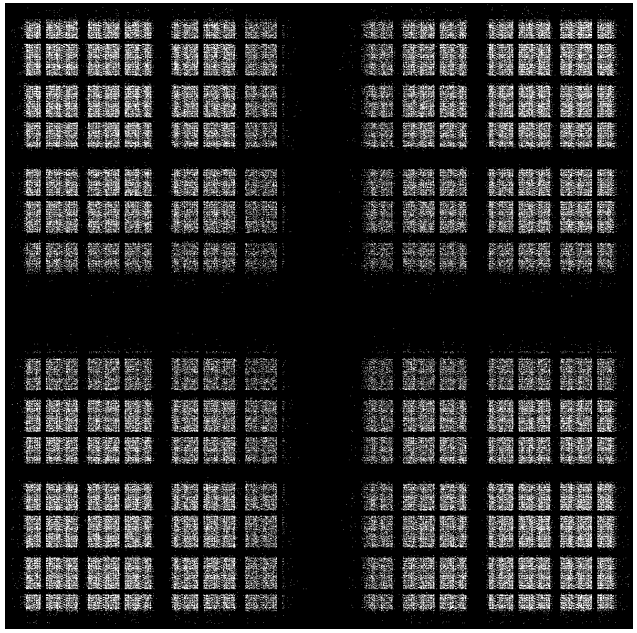

$$, шум питания и температура, 3 mbits/s, raw=yes, open source (GNU микропроцессор), whitening=CRC32 + SHA-256

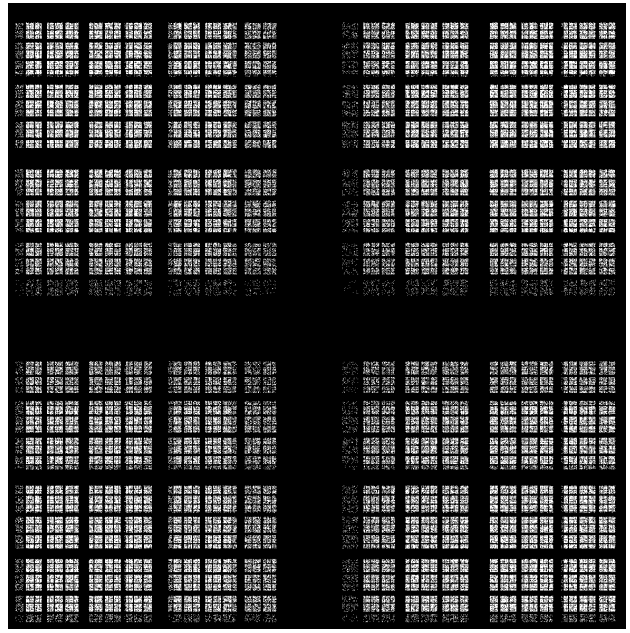
# Infinite Noise


$$, умножение уровня. $x \longmapsto 2x - 1$, 300 kbits/s, raw=yes, whitening=SHA3

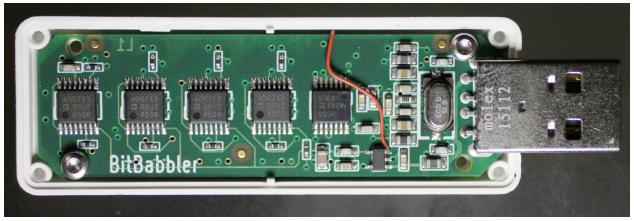
# умножение уровня

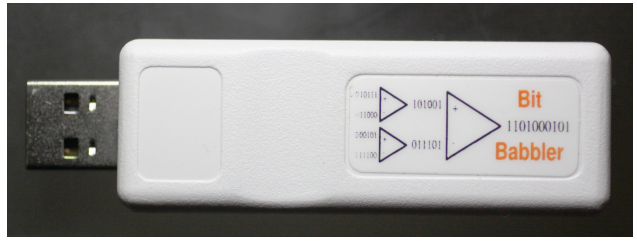




infinite noise и его модель

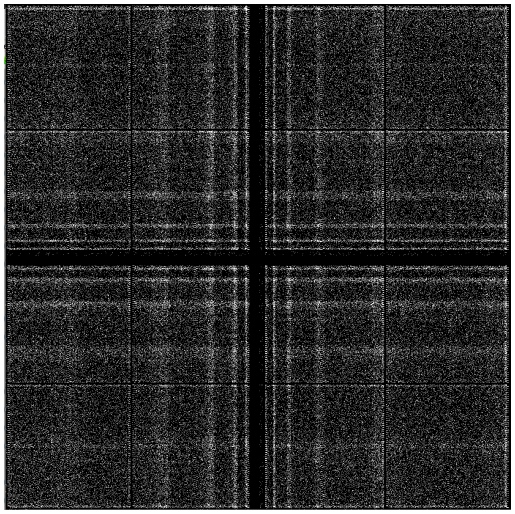
# Bitbabbler



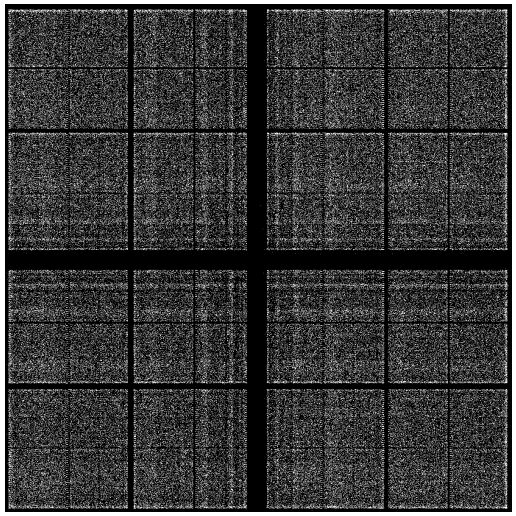




$$–$$$, electronic noise, $x \longmapsto 2x - 1$ digitization, 2.5 mbits/s default, 4 independent generators ($150 version), access to raw bits, variable discretization rate, whitening=XOR

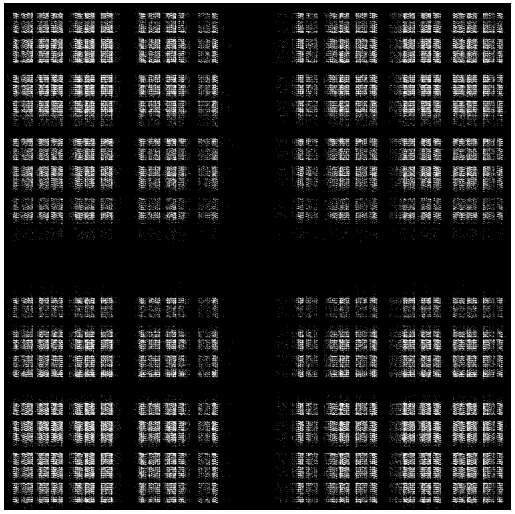
# Bitbabbler: умножение уровня



100 kHz



(default) 2.5 MHz

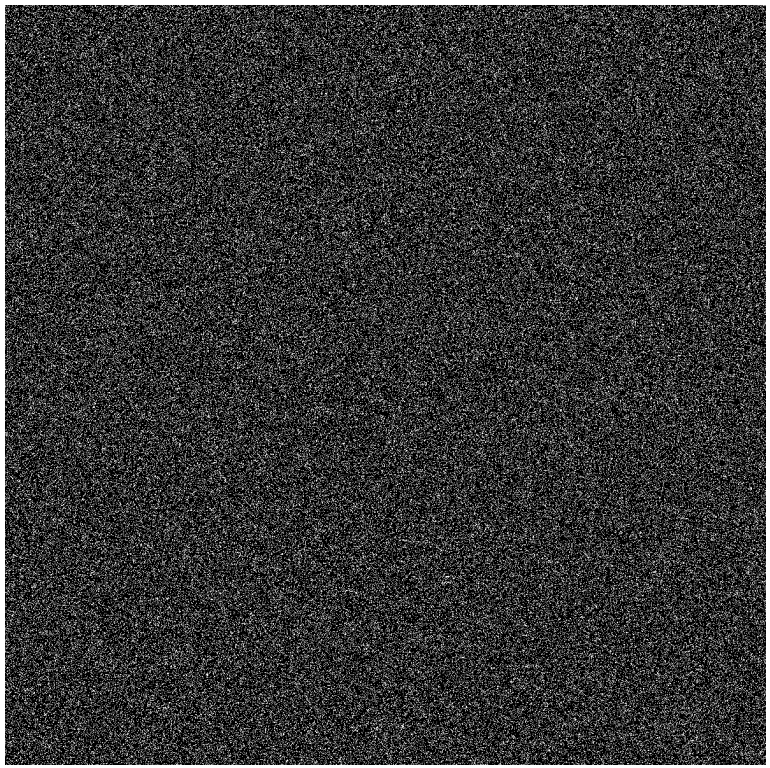


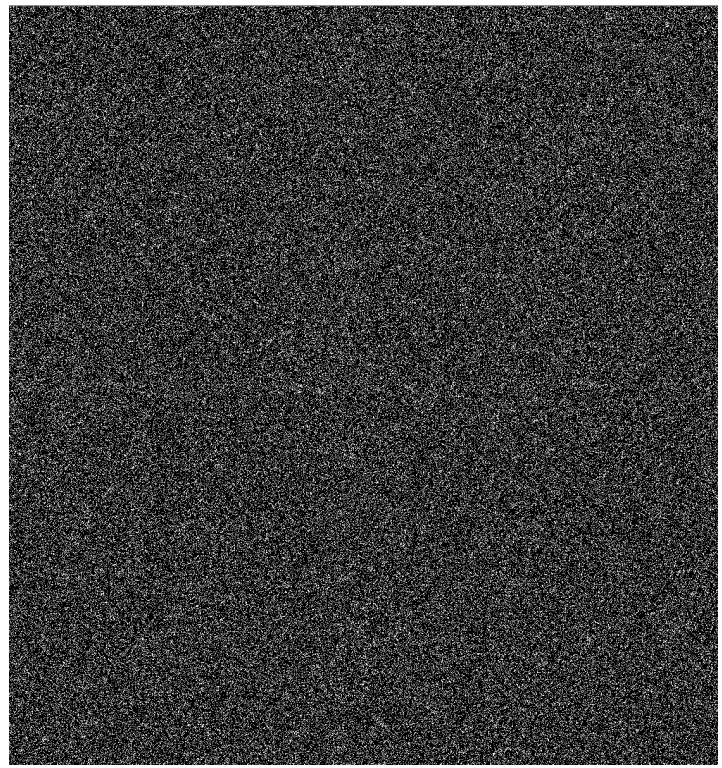
5 MHz

# whitening

- debiasing по фон Нейману
- xor (особенно независимых источников)
- линейный оператор («случайный»)
- криптографические конструкции
- теория: Santha–Vazirani, randomness extractors

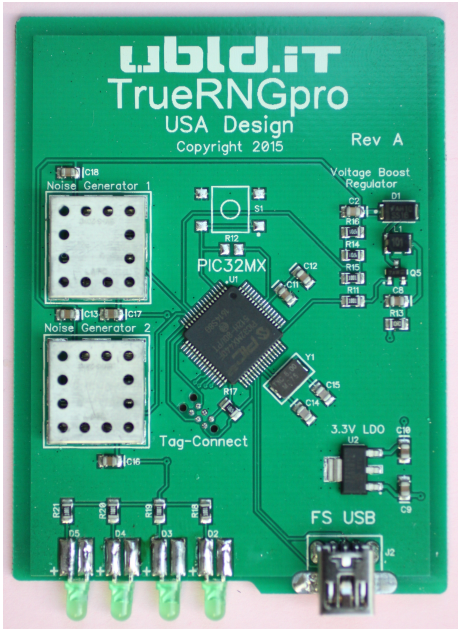
# Bitbabbler после 2 и 3 xor

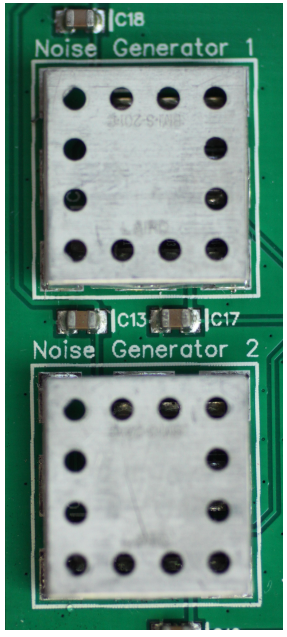




(5 мегагерц)

# TrueRNG



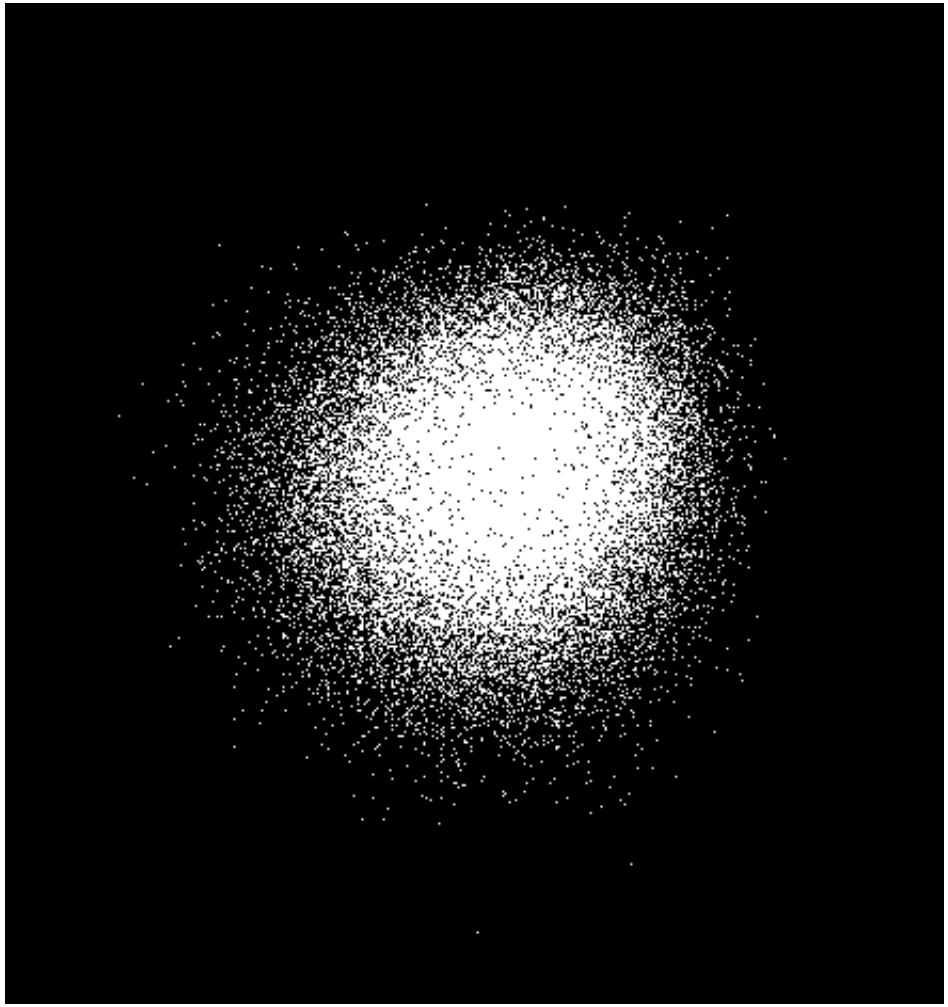





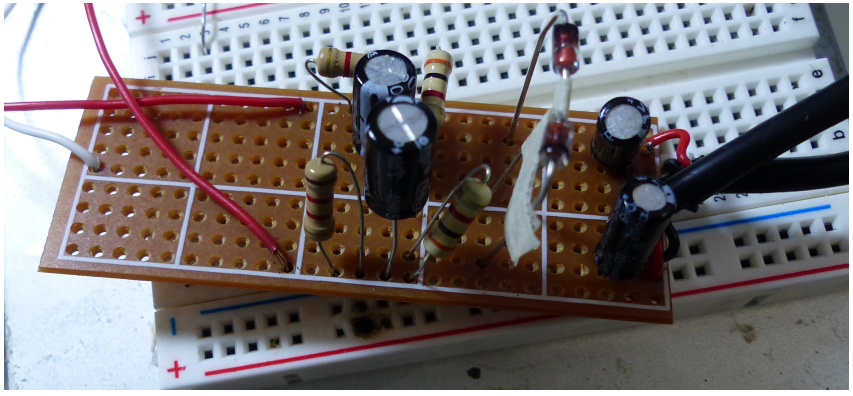
$$–$$$, стабилитрон + DAC,

3.2 mbits/s, 2 источника шума ($100), raw bits=yes, whitening=XOR/CRC

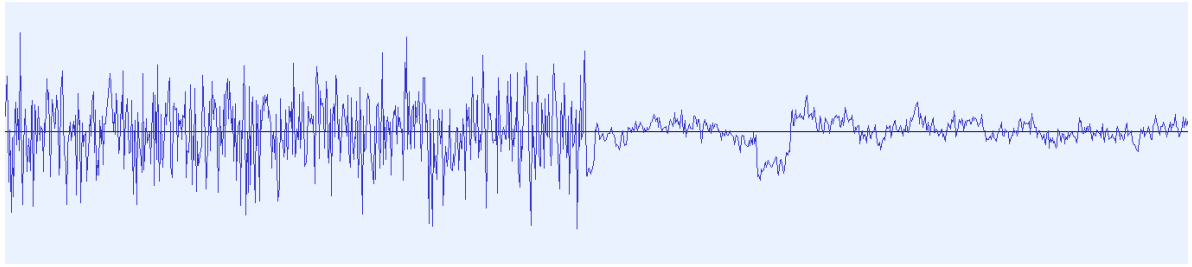
# TrueRNG raw noise

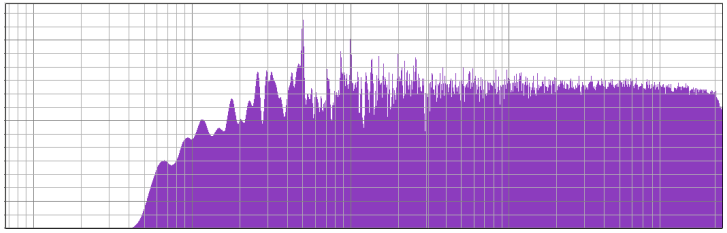


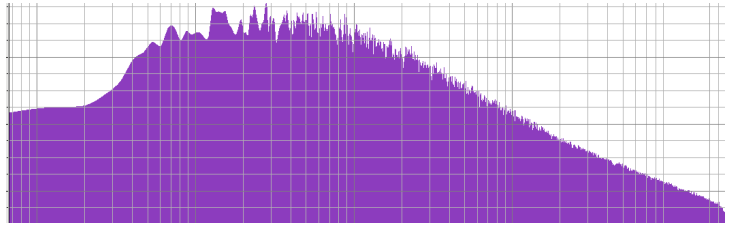
# «очумелые ручки»



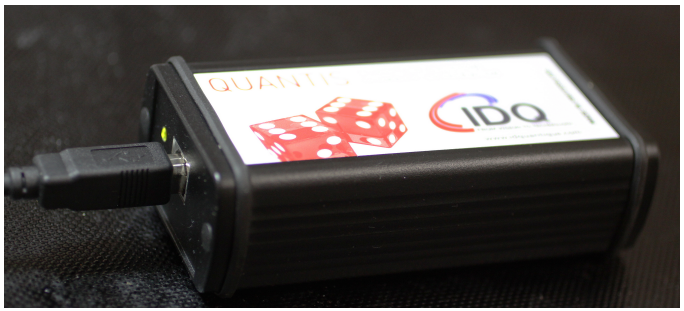
# разные стабилитроны







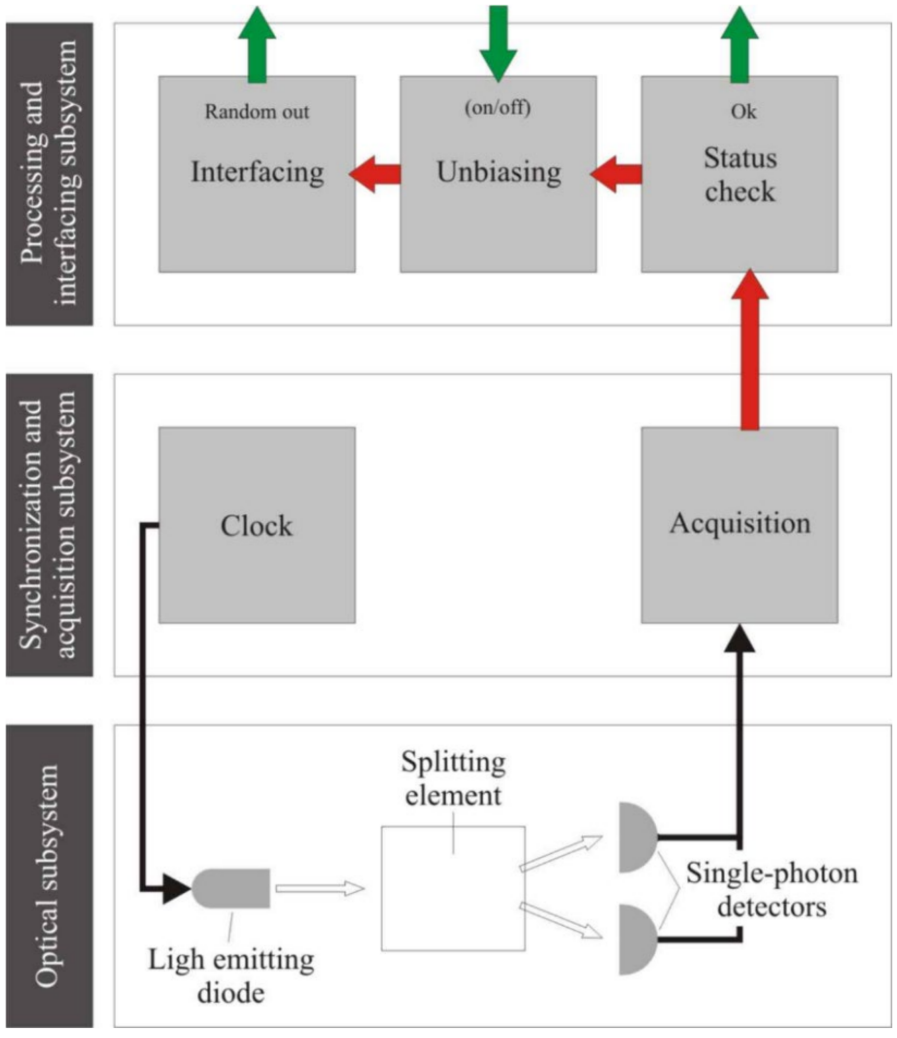
# ID Quantique





$$$–$$$$, отражение/пролёт фотонов, 4 mbits/s, raw=no, whitening=?

# ID Quantique: структура



# подробнее о тестах

- нулевая гипотеза $H_0$ = равномерное распределение на $n$-битовых строках
- тест: небольшое множество $n$-битовых строк
- тех, что не проходят тест
- должно быть указано до эксперимента

# история тестов

- начало описано у Кнута (vol.2, 1969)
- закон больших чисел ($\#0 \approx \#1$)
- нормальность по Борелю
- $\chi^2$ тесты для групп байтов
- Marsaglia `diehard` (1985–1995): до сих пор
- Brown `dieharder` (2005): улучшенная реализация
- NIST 800-22 (2000, 2010), `STS`
- Simard, l'Ecuyer `TestU01` (2007)

## примеры

- несжимаемость (компрессоры)
- предельные теоремы теории вероятностей
- $p$-values: пусть есть произвольная функция $S\colon \mathbb{B}^n \to \mathbb{R}$
- для каждого $x \in \mathbb{B}^n$ определим $p$-value для $x$ $p_S(x) = \Pr[S(r) \geqslant S(x)]$ для случайного $r \in \mathbb{B}^n$
- $p_S(x) < \varepsilon$ с вероятностью не больше $\varepsilon$
- если каждое значение $S$ очень маловероятно, $p_S(x)$ имеет почти равномерное распределение на $[0, 1]$
- и можно применить Колмогорова–Смирнова
- вторичные тесты (Knuth, diehard)

# недоразумения: IDquantique

However, if one were to be given a number, it is simply impossible to verify whether it was produced by a random number generator or not. It is hence absolutely essential to consider sequences of numbers in order to study the randomness of the output of such a generator.

It is quite straightforward to define whether a sequence of infinite length is random or not. This sequence is random if the quantity of information it contains – in the sense of Shannon's information theory – is also infinite.

In other words, it must not be possible for a computer program, whose length is finite, to produce this sequence. Interestingly, an infinite random sequence contains all possible finite sequences.

(white paper)

- отождествляют случайность с невычислимостью
- (а последнее предложение ложно)

# случайность в алгоритмической теории

- Martin-Löf: случайные бесконечные последовательности
- тест: убывающие множества последовательностей $U_1 \supset U_2 \supset \ldots$, мера $U_i$ не больше $2^{-i}$
- не пройти тест: принадлежать всем $U_i$
- случайные = проходят все тесты = проходят универсальный тест
- = начальные отрезки несжимаемы

# недоразумения: NIST 800-22-1a

- “type I error probability of failing the test assuming the null hypothesis $H_0$” (ok)
- “Type II error probability is $\langle \ldots \rangle$ $P\big(\text{accept } H_0 | H_0 \text{ is false}\big)$” (1-4)
- но “$H_0$ is false” — не распределение вероятностей
- “Unlike $\alpha$ [the probability of Type I error], $\beta$ is not a fixed value. $\langle \ldots \rangle$ The calculation of Type II error $\beta$ is more difficult than the calculation of $\alpha$ because of the many possible types of non-randomness”
- “If a *P-value* for a test is determined to be equal to 1, then the sequence appears to have perfect randomness” (1-4)
- “For a *P-value* $\geqslant 0.001$, a sequence would be considered to be random with a confidence of 99.9%. For a *P-value* $< 0.001$, a sequence would be considered to be non-random with a confidence of 99.9%” (1-4)
- два некорректных теста (удалены в следующей версии)

## недоразумения: `diehard[er]`

- неизбежно: PASS ничего не гарантирует
- FAIL? должно быть редкое событие
- вычисления вероятностей основаны на эвристических предположениях (и на некоторых явно ошибочных)
- dieharder: “At this point I think there is rock solid evidence that this test [one of the `diehard` tests] is completely useless in every sense of the word. It is broken, and it is so broken that there is no point in trying to fix it. The problem is that the transformation above is not linear, and doesn’t work. Don’t use it.”

# ошибки в программах

- локальные и глобальные переменные (dieharder)
- неверное вычисление статистики Колмогорова–Смирнова (dieharder)
- несоответствие теста и описания (NIST)
- NIST: In practice, many reasons can be given to explain why a data set has failed a statistical test. The following is a list of possible explanations. The list was compiled based upon NIST statistical testing efforts.
  1. An incorrectly programmed statistical test.
  2. An underdeveloped (immature) statistical test.
  3. An improper implementation of a random number generator.
  4. Improperly written codes to harness test input data.
  5. Poor mathematical routines for computing *P-values*.
  6. Incorrect choices for input parameters.

# недоразумения: энтропия как субстанция

- производится генераторами и накапливается в резервуарах
  “The central mathematical concept underlying this [NIST] Recommendation is entropy. Entropy is defined relative to one’s knowledge of an experiment’s output prior to observation, and reflects the uncertainty associated with predicting its value – the larger the amount of entropy, the greater the uncertainty in predicting the value of an observation”
- “Each bit of a bitstring with full entropy has a uniform distribution and is independent of every other bit of that bitstring. Simplistically, this means that a bitstring has full entropy if every bit of the bitstring has one bit of entropy; the amount of entropy in the bitstring is equal to its length’ (NIST)

# шенноновская теория информации

- энтропия распределения (но не битовой строки)
- $m$ исходов с вероятностями $p_i$:

$$H = \sum p_i \log \frac{1}{p_i}$$

- «среднее удивление»
- средняя длина оптимального кода (на символ)
- не больше $\log m$, достигается при равных $p_i$
- minentropy $\leqslant k$: все $p_i \geqslant 2^{-k}$ (оценка снизу)

# выделение случайности (whitening)

- физический генератор $m$ битов
- детерминированное преобразование $F\colon \mathbb{B}^m \longrightarrow \mathbb{B}^n$ $(n < m)$
- (неравномерное неизвестное) выходное распределение датчика $P$ на $\mathbb{B}^m$ (случайная величина $\xi \in \mathbb{B}^m$)
- (близкое к равномерному?) распределение $Q$ для $F(\xi)$
- близость $d(Q_1, Q_2) = \max_{A \subset \mathbb{B}^n} |Q_1(A) - Q_2(A)|$

# выделение случайности: желания и реальность

- физически правдоподобные предположения о $P$
- детерминированное преобразование $F$
- теорема: если $P$ удовлетворяет предположениям, то $Q = F(P)$ близко к равномерному
- предположение о $P$: велика минэнтропия (тогда и энтропия)
- NIST стандарт: рекомендует такой подход и конкретные $F$
- безо всяких на то оснований

# отрицательный результат: Santha–Vazirani

- SV-источник: $N$ битов $\xi_1 \dots \xi_N$, каждый следующий бит при известных предыдущих имеет вероятность в $(1/3, 2/3)$.
- whitening $F\colon \mathbb{B}^N \longrightarrow \mathbb{B}$
- теорема: для любого $F$ существует SV-источник $\xi$, для которого $F(\xi) = 0$ имеет вероятность $\leqslant 1/3$ или $\geqslant 2/3$.
- «ничего лучше не придумаешь, чем взять первый бит»
- обобщение: «ничего лучше не придумаешь, чем взять первые $k$ битов»

# randomness extractor

- функция $F$ с двумя входами: длинным $X$ и коротким $R$; при этом $F(X,R)$ близко к равномерному, если
  - длинное $X$ имеет достаточную min-энтропию
  - короткое $R$ равномерно распределено
  - $X$ и $R$ независимы
- существование доказано, и даже явные конструкции есть
- замечательно, но откуда взять $R$?
- аналогичный подход используется IDQuantique, но математически некорректно

# псевдослучайные генераторы

- подход «мне повезёт» (ну и что, что тест не проходит, в конкретном приложении это может не повредить)
- теоретически обоснованный подход (Yao, Blum–Micali)
- $G$: короткое $n$-битовое зерно (seed) $\longmapsto$ длинная $N$-битовая строка ($N \gg n$)
- $G$ быстро вычислимо
- никакой быстро вычислимый тест $T \subset \{0,1\}^N$ не отличает от случайного:

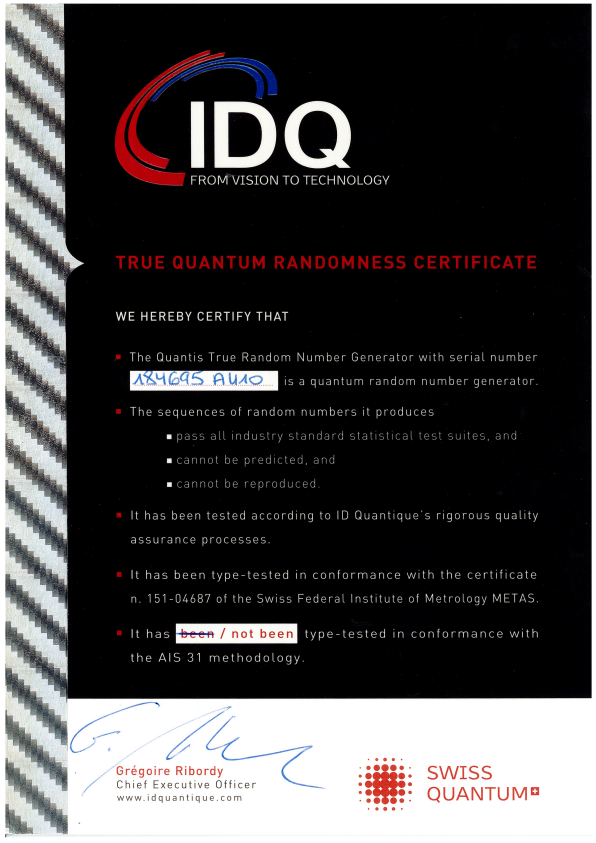
$$\Pr_{x \in \{0,1\}^n}[G(x) \in T] \approx \Pr_{y \in \{0,1\}^N}[y \in T]$$

- быстро вычислимый $\approx$ схемы полиномиального размера
- существуют вместе с односторонним функциями (Hastad, Impagliazzo, Luby, Levin) и только если $\mathrm{P} \neq \mathrm{NP}$

## надёжные тесты

- режем поток случайных битов от датчика на $N$ строк $x_1, \ldots, x_N$ длины $M$ и ещё $N$ строк $y_1, \ldots, y_N$ той же длины;
- берём строку $z$ той же длины $M$ (от другого датчика, но не важно)
- применяем тест ent к $x_1, \ldots, x_N$ и $y_1 \oplus z, \ldots, y_N \oplus z$
- если выяснилось, что результат теста ($\chi_2$) в первой группе всегда хуже чем во второй, отвергаем
- вероятность отвергнуть хороший датчик $1/C_{2N}^{N}$, то есть примерно $2^{-2N}/2N$
- ID quantuque, $N = 20$, $M = 200$ мегабайтов, для $y_i \oplus z$ 235..282, для $x_i$ 527..697

# paranoid mode on



IDQ
FROM VISION TO TECHNOLOGY

TRUE QUANTUM RANDOMNESS CERTIFICATE

WE HEREBY CERTIFY THAT

- The Quantis True Random Number Generator with serial number 184695 AU10 is a quantum random number generator.
- The sequences of random numbers it produces
  - pass all industry standard statistical test suites, and
  - cannot be predicted, and
  - cannot be reproduced.
- It has been tested according to ID Quantique's rigorous quality assurance processes.
- It has been type-tested in conformance with the certificate n. 151-04687 of the Swiss Federal Institute of Metrology METAS.
- It has ~~been~~ / not been type-tested in conformance with the AIS 31 methodology.

Grégoire Ribordy
Chief Executive Officer
www.idquantique.com

SWISS QUANTUM

но не проходит надёжный тест на основе ent (без дополнительного умножения на случайную матрицу)

# security through obscurity

- NIST рекомендует (и даже требует) криптографического отбеливания
- “approved hash function”
- но ничего при них не доказано
- и даже обычные криптографические свойства хеширования не помогли бы

# так говорит NIST

*Hash_DRBG's [the random generator based on hash functions] security depends on the underlying hash function's behavior when processing a series of sequential input blocks. If the hash function is replaced by a random oracle, Hash_DRBG is secure. It is difficult to relate the properties of the hash function required by Hash_DRBG with common properties, such as collision resistance, pre-image resistance, or pseudorandomness.*

## опасности

- испорченная программа в микропроцессоре
- отказ физического датчика
- возможно, замаскированный whitening
- hash function как подстава от авторов стандарта
- близкое, но не очень, распределение
- last but not least: нелепые ошибки (пример: AMD Zen FF)

# что может помочь

- `xor` незваисимых устройств
- самодельный генератор
- open source hardware/software
- несколько недорогих разумных генераторов разных производителей
- отказ от obscure cryptography