# A Concise Introduction to Random Number Generation


Peter Hellekalek

Dept. of Mathematics, University of Salzburg

# *Overview of This Talk*

How to assess RNGs?

- criteria
- a checklist

A note on statistical testing

- strategies
- Maurer's Universal Test and related tests

Interesting RNGs

- AES
- HAVEG(E)

# *RNGs: The Goal*

What we want . . .
A device (hardware or software) whose output is random.

More precisely . . .
Want to generate bits (or numbers) that appear like being sampled from a uniform distribution on $\{0, 1\}$ (or $[0, 1[$), independently of each other.

# *RNGs: The Reality*

What we get...

Finite output streams that pass many tests of randomness.

Pseudorandom number generator (PRNG)

Deterministic algorithm whose output mimics finite random sequences.

Question

What are random sequences?

# *Randomness*

Quote

> *"A finite sequence is random if there is no short sequence that describes it fully, in some unambigous mathematical notation."*

... A. Kolmogoroff

Quote

> *"A string is random if it cannot be algorithmically compressed."*

... C. Calude

Remark

The basic idea of Kolmogoroff complexity:

Randomness = Incompressibility

# *RNGs: Practice*

Quote

> *"Monte Carlo results are misleading when correlations hidden in the random numbers and in the simulated system interfere constructively."*

... A. Compagner, Phys. Rev. E **52**(1995)

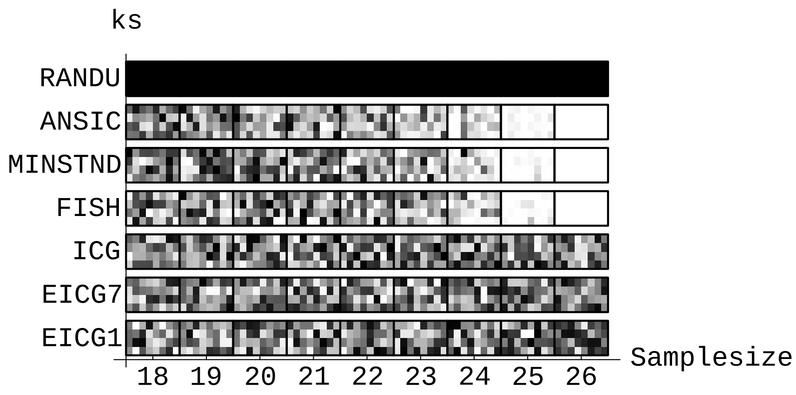
Quote

> *"Ironically, pseudorandom numbers often appear to be more random than random numbers obtained from physical sources."*

... A. Rukhin et al., NIST Special Publ. 800-22

# RNGs: An Illustration

With RNGs, there are no guarantees.




| | |
|---|---|
| True Value: | Irregular Pattern in Every Box |
| RNGs: | LCGs and (E)ICGs |
| Sample Size: | $2^{18}..2^{26}$ |

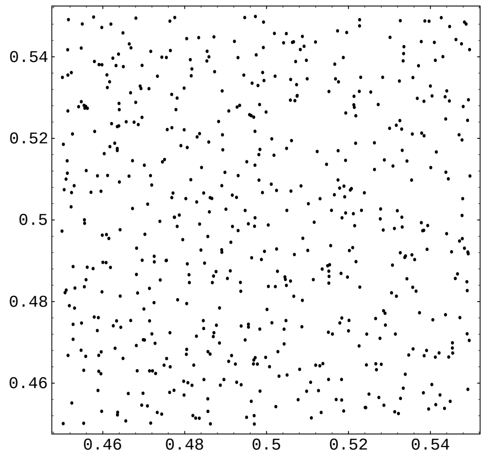
# *Phenomena*

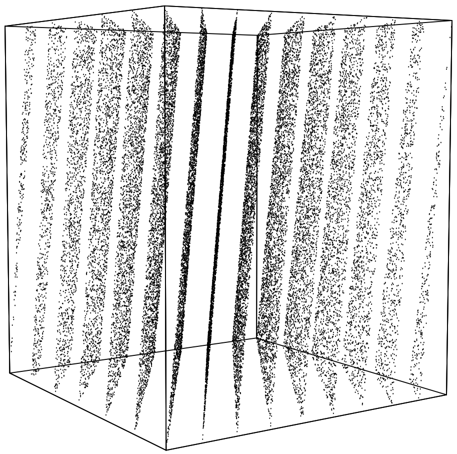
Setup

- RNG: LCG($2^{31}$, 65539, 0, 1), i.e. RANDU
- Dimension: $d = 2, 3$
- Sample size: $N = 2^{16}$
- Plot nonoverlapping pairs $(x_{2n}, x_{2n+1})$ and triples $(x_{3n}, x_{3n+1}, x_{3n+2})$, $0 \leq n < N$.

# *Phenomena: Increasing the Dimension*

We increase the dimension from $d = 2$ to $d = 3$:



Question

How to prevent such unpleasant surprises?

Answer

Theoretical correlation analysis and/or statistical testing.

# *Linear Congruential Generator (LCG)*

Parameters

- $m$ . . . modulus
- $a$ . . . multiplier
- $b$ . . . additive constant
- $y_0$ . . . initial value

Defining congruence

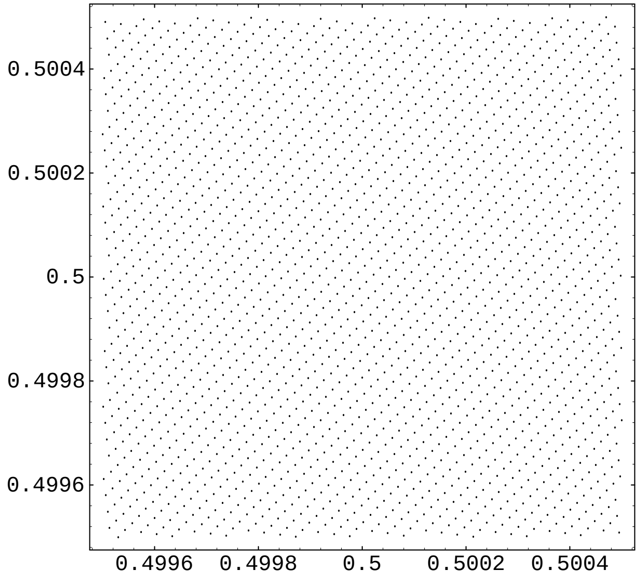
$$y_{n+1} \equiv a \cdot y_n + b \pmod{m}, \quad n \geq 0$$

. . . LCG($m, a, b, y_0$)

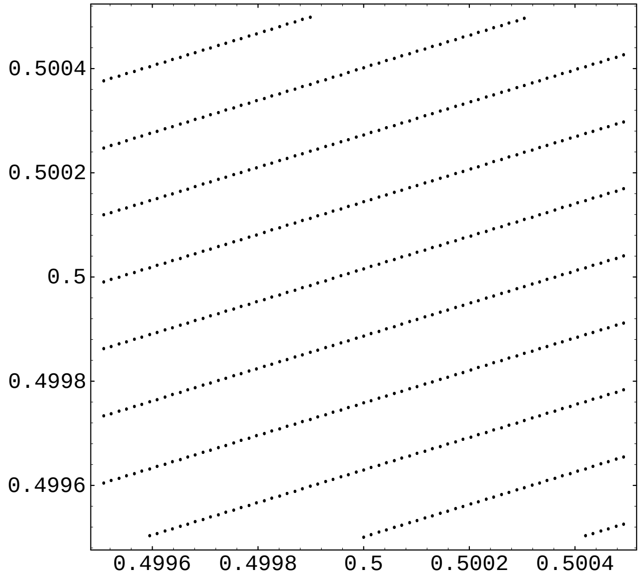
Output stream

$$x_n := \frac{y_n}{m} \in [0, 1[, \quad n = 0, 1, \ldots$$

# *LCGs: Two Examples*




LCG($2^{31} - 1$,630360016,0,1)




LCG($2^{32}$, 69069, 0,1)

# *Inversive Congruential Generator (ICG)*

Parameters

- $m$ ... modulus (usually a big prime)
- $a$ ... multiplier
- $b$ ... additive constant
- $y_0$ ... initial value

Defining congruence

$$y_{n+1} \equiv a \cdot \overline{y_n} + b \pmod{m}, \quad n \geq 0$$

($\overline{c} = c^{-1}$ for $c \neq 0$, $\quad \overline{c} = 0$ if $c = 0$.)

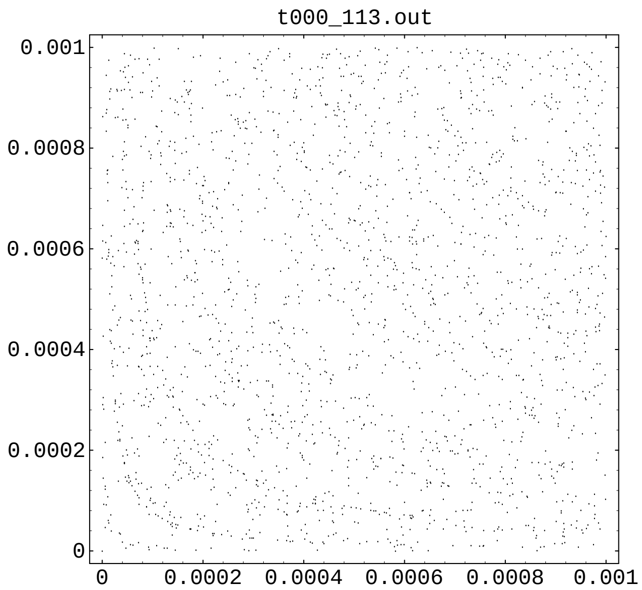
... ICG($m, a, b, y_0$)

Output stream

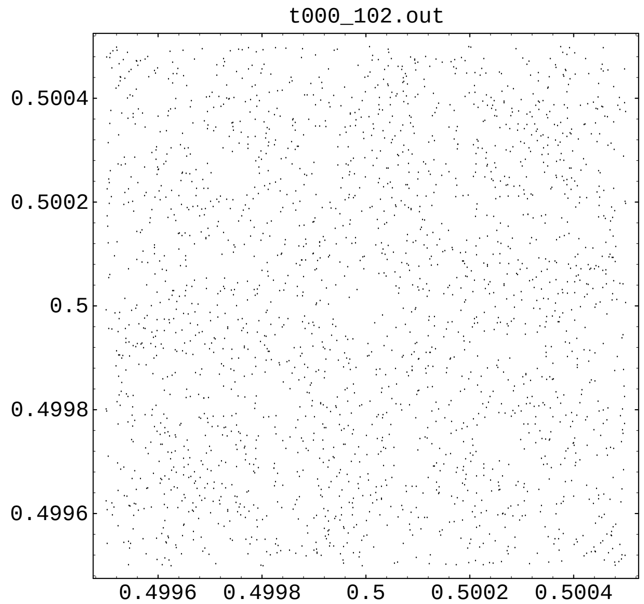
$$x_n := \frac{y_n}{m} \in [0, 1[, \quad n = 0, 1, \ldots$$

# ICG: Point Structure

ICG($2^{31} - 1$, 1288490188, 1, 1)




lower left corner




middle section

# *Pseudorandom Number Generators (PRNGs)*

PRNG: A tuple $G = (\mathcal{S}, \mathcal{I}, T, \mathcal{O}, g, s_0)$ , where

- $\mathcal{S}$ is the finite state space,
- $\mathcal{I}$ is the input space,
- $T : \mathcal{I} \times \mathcal{S} \to \mathcal{S}$ is the transition function,
- $\mathcal{O}$ is the finite output space,
- $g : \mathcal{S} \to \mathcal{O}$ is the output function,
- $s_0 \in \mathcal{S}$ is the seed.

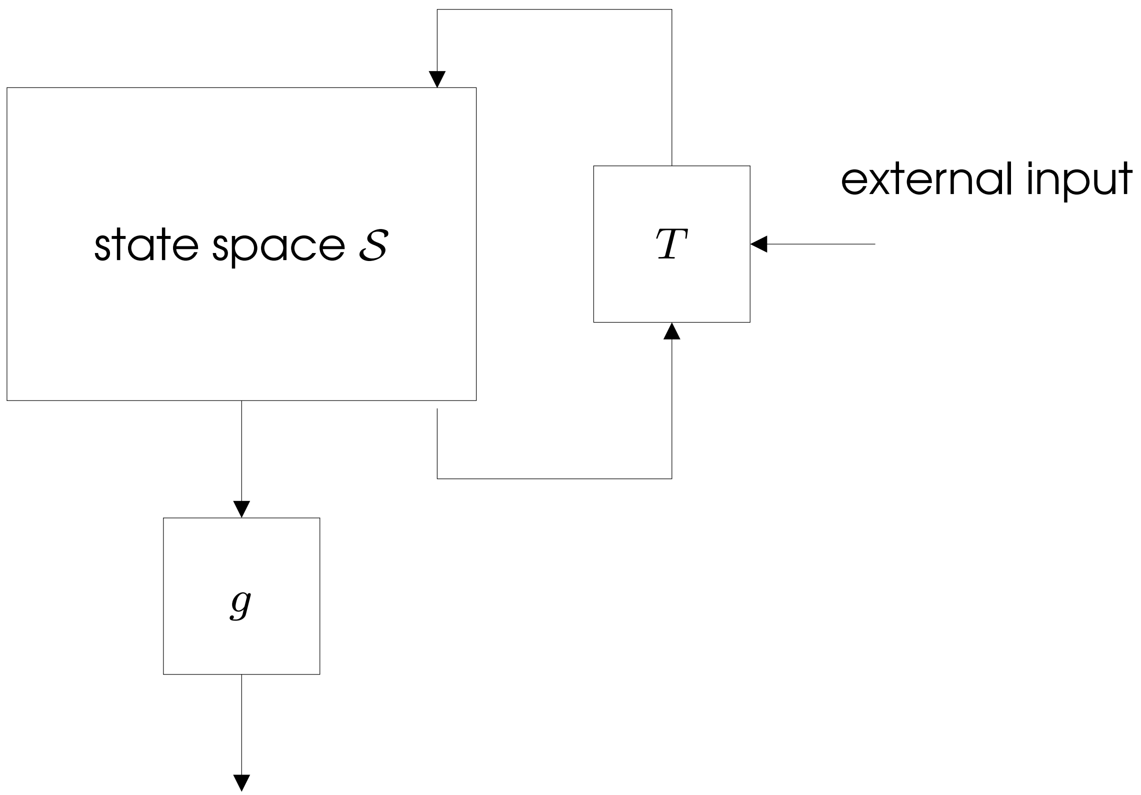
The next state $s_{n+1}$ is generated by

$$s_{n+1} = T(i_n, s_n), \qquad n \geq 0,$$

the output stream $(o_n)_{n \geq 0}$ is computed by

$$o_n = g(s_n), \qquad n \geq 0.$$

# PRNGs: The Structure




The Structure of a PRNG

# *Classification of RNGs*

Types of RNGs

| **Type of Application** | |
|---|---|
| Simulation<br>(Monte Carlo) | Cryptography<br>(stream ciphers) |

| **Type of Platform** | |
|---|---|
| Hardware<br>(“physical” randomness) | Software<br>(pseudo-randomness) |

Classes of PRNGs

| **PRNGs: Type of Algorithm** | |
|---|---|
| linear | nonlinear |

# *Which RNG?*

RNG vs. Application

| RNG \ Application | Simulation | Cryptography |
|---|---|---|
| Hardware | not recommended | task dependent |
| Software | recommended | task dependent |

PRNG vs. Application

| PRNG \ Application | Simulation | Cryptography |
|---|---|---|
| Linear | recommended (if chosen properly) | not recommended (insecure) |
| Nonlinear | task dependent (too small, too slow) | recommended (if chosen properly) |

# *Checklist: Theoretical Support*

| **A) Theoretical Support** | | |
|---|---|---|
| **Period Length** | Conditions | |
| | Algorithms for parameters | |
| **Structural Properties** | Intrinsic structures | |
| | Equidistribution properties | |
| | Predictability | |
| **Correlation Analysis** | For particular parameters | |
| | For particular initializations | |
| | For parts of the period | |
| | For subsequences | |
| | For combinations of RNGs | |

# *Checklist: Statistical Testing*

| | |
|---|---|
| **B) Statistical Testing** | |
| Variable sample size<br><br>Two- or higher level tests | |
| Performance with test batteries | |
| Serial test family | |
| Return times | |
| Other test quantities | |
| Transformation methods: sensitivity | |

# *Checklist: Practical Aspects*

| **C) Practical Aspects** | |
|---|---|
| Tables of parameters available? | |
| Portable implementations available? | |
| Parallelization techniques applicable? | |
| Large samples available? | |
| Efficiency? | |
| Cryptography: security aspects? | |

# RNGs: Simulation vs. Cryptography

| **Simulation** | **Cryptography** |
|---|---|
| **Theoretical Analysis** | |
| Period Length | |
| Known (in most cases) | Unknown (in most cases) |
| Structural Properties | |
| Intrinsic structures welcome | Intrinsic structures are to be avoided |
| **Statistical Testing** | |
| Extensive results | Lack of published test results |
| Batteries of tests | Under development (NIST) |

# RNGs: Simulation vs. Cryptography

| **Simulation** | **Cryptography** |
|---|---|
| **Practical Aspects** | |
| RNGs trimmed for efficiency | RNGs in many flavors |
| **Design Aspects** | |
| Prefer linear algorithms (efficiency!) | Require nonlinear algorithms (security!) |

# *RNGs: Simulation vs. Cryptography*

| **Simulation** | **Cryptography** |
|---|---|
| **RNG Testing** | |
| fair adversary:<br>RNG treated as a black box | freestyle:<br>all attacks allowed |
| test tries to find structures in the output stream | same goal |
| not interested in predictability | try to find the secret key |

# *NIST Test Suite (NTS)*

Comments

- Question I:
  What are the redundancies in this test suite?
  For example, NST contains various entropy estimators (Maurer's Universal Test, Approximate Entropy of Pincus and Singer, Serial Test). What is the relation between them?
- Question II
  Which NIST tests detect which kind of defect?
  The NTS has not been analyzed with respect to a defective RNG. Which tests will detect a given defect (and which tests will not)?

# *Testing Statistical Tests*

Question

How universal is Maurer's Universal Test?

Approach

- Construct bitstream $x_0, x_1, \ldots$
  induce correlations at distance $\kappa$:

$$\boxed{x_0},\ x_1,\ x_2,\ \ldots,\ x_{\kappa-1},\ \boxed{x_\kappa},\ x_{\kappa+1},\ \ldots$$

- Does the statistical test at hand detect this error?

Results

See our "Defective Source Analysis"

# *Defective Source Analysis*

Correlations

Choose order $\kappa$, $\kappa \geq 1$

Choose random bits

$$x_0, x_1, \ldots, x_{\kappa-1}$$

Choose bias $\lambda$

$$x_i = \begin{cases} x_{i-\kappa} & \text{with probability } \lambda \\ 1 - x_{i-\kappa} & \text{with probability } 1 - \lambda \end{cases}$$

Choose source probability distribution

$$\lambda = 0.5 \quad \ldots \text{i.i.d. uniform}$$

$$\lambda \neq 0.5 \quad \ldots \text{i.d. uniform}$$

# Defective Source Analysis

- Test input $x_0, x_1, \ldots, x_{m-1}$ ($m$ bits)
- Sample size $n > 1$
- Dimension $d \geq 1$
- Overlapping and non-overlapping $d$-tuples

$$\tilde{x}_i^d = (x_i, x_{i+1}, \ldots, x_{i+d-1})$$
$$\bar{x}_i^d = (x_{i\cdot d}, x_{i\cdot d+1}, \ldots, x_{i\cdot d+d-1})$$

- Frequency count

$$\begin{aligned} \mathbf{a} &= (a_1, \ldots, a_d) \in \{0,1\}^d \\ \tilde{\pi}_{\mathbf{a}}^d &= \frac{1}{n}\#\{0 \leq i < n : \tilde{x}_i = \mathbf{a}\} \\ (\bar{\pi}_{\mathbf{a}}^d &= \frac{1}{n}\#\{0 \leq i < n : \bar{x}_i = \mathbf{a}\}) \end{aligned}$$

# Defective Source Analysis

## Approximate Entropy

$$\hat{H}_f^d = -\sum_{\mathbf{a}\in\mathcal{A}^d} \tilde{\pi}_{\mathbf{a}}^d \log \tilde{\pi}_{\mathbf{a}}^d + \sum_{\mathbf{a}\in\mathcal{A}^{d-1}} \tilde{\pi}_{\mathbf{a}}^{d-1} \log \tilde{\pi}_{\mathbf{a}}^{d-1},$$

$$\hat{I}^d = 2n(1-\hat{H}_f^d) \xrightarrow{\mathrm{D}} \chi^2_{2^d-2^{d-1}}$$

. . . (Pincus and Singer, 1998)

## Universal Test

$$\hat{H}_r^d = \frac{1}{d\cdot n} \sum_{i=Q}^{Q+n-1} \log T(i)$$

$$\hat{N}^d = \frac{\hat{H}_r^d - E[\cdot]}{\sqrt{V[\cdot]}} \xrightarrow{\mathrm{D}} N[0,1]$$

. . . (Maurer, 1992)

($T(i)$: return time for $\bar{x}_i^d$)

# Defective Source Analysis

## Overlapping Serial Test

$$\hat{\chi}^d = n \sum_{\mathbf{a}\in\mathcal{A}^d} \frac{(\tilde{\pi}^d_{\mathbf{a}} - (1/2)^d)^2}{(1/2)^d} - n \sum_{\mathbf{a}\in\mathcal{A}^{d-1}} \frac{(\tilde{\pi}^{d-1}_{\mathbf{a}} - (1/2)^{d-1})^2}{(1/2)^{d-1}}$$

. . . (I.J. Good, 1953)

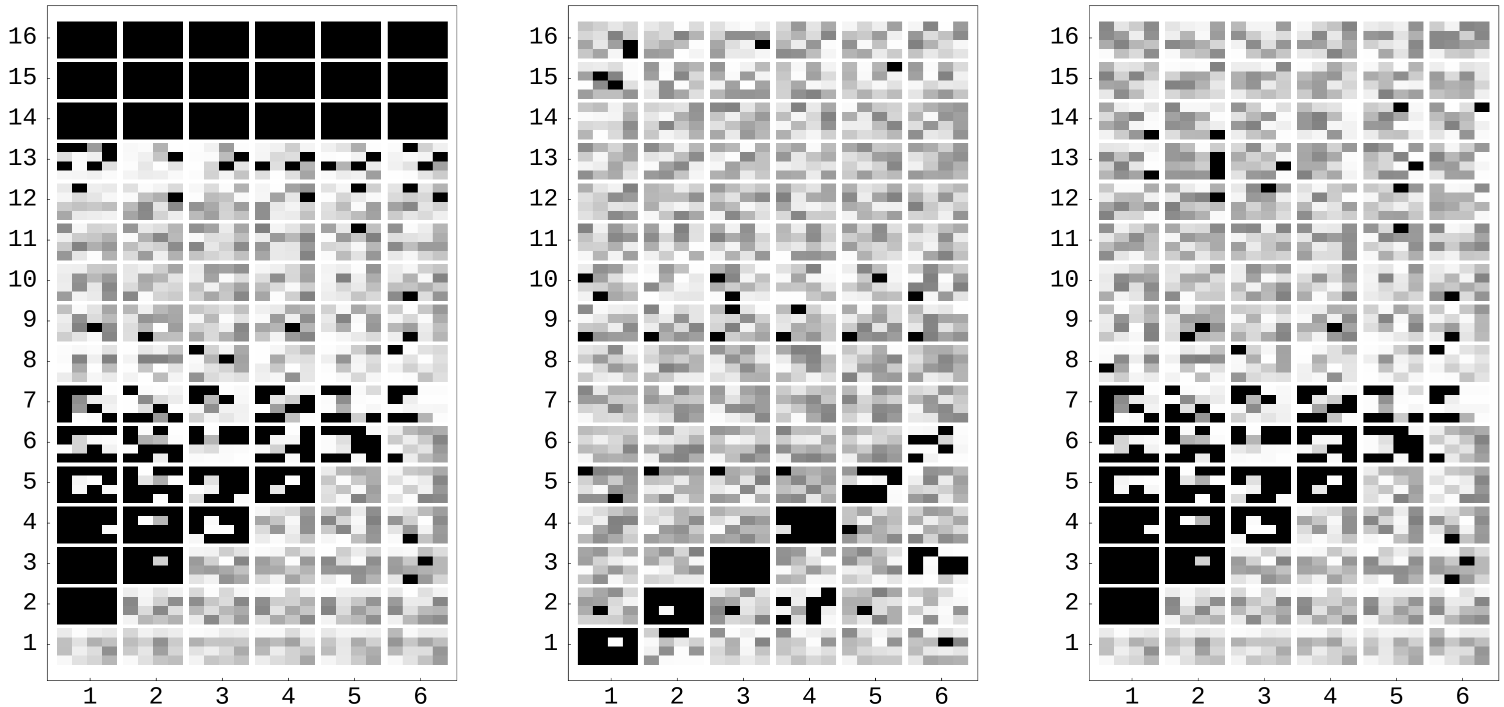
## Test Parameters

| | |
|---|---|
| Sample size | $n = 2^{16},\ 2^{18}$ bits |
| No. of repetitions | 16 indept. samples |
| Dimension | $d = 1..16$ |
| Order | $\kappa = 1..6$ |
| Bias $\lambda$ | $\lambda = 0.49$ |
| Entropy of source | $H \approx 0.999711$ |

# *Defective Source Analysis*

Results for $n = 2^{16}$ bits

Black dots denote $p$-values smaller than $0.01$.



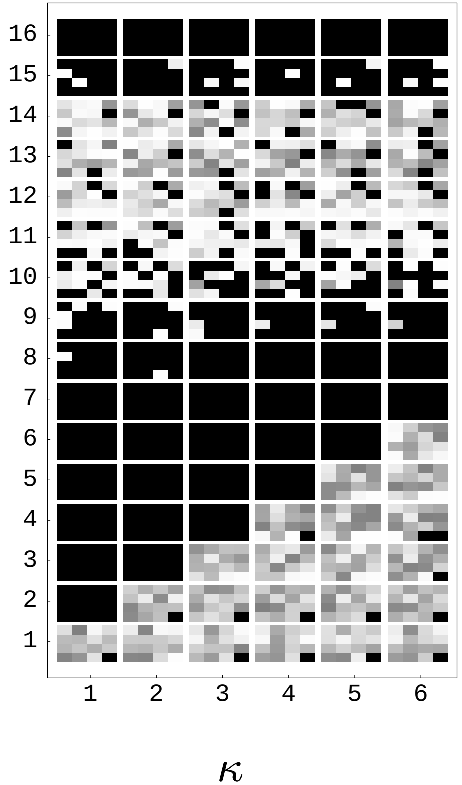

$\kappa$ $\kappa$ $\kappa$

$\hat{I}^d$, ApEn | $\hat{N}^d$, Universal Test | $\hat{\chi}^d$, Overl. Serial T.

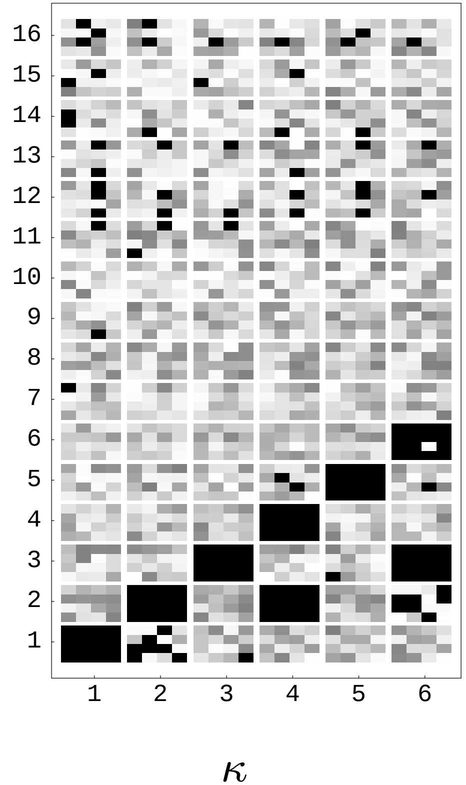
# Defective Source Analysis

Results for $n = 2^{18}$ bits

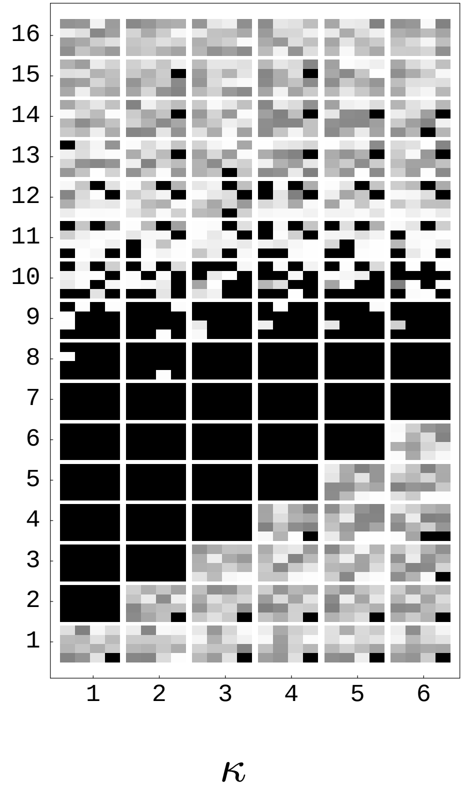
Black dots denote $p$-values smaller than $0.01$.




$\hat{I}^d$, ApEn




$\hat{N}^d$, Universal Test




$\hat{\chi}^d$, Overl. Serial T.

# *AES: Modes of Operation*

## Output Feedback Mode MODE (OFB)

| | | |
|---|---|---|
| choose | $k$ | ...key |
| choose | $z_0$ | ...initial value |
| compute | $\left(e_k^{(i)}(z_0)\right)_{i \geq 0}$ | ...output stream |
| | $e_k^{(i)} = \underbrace{e_k \circ \ldots \circ e_k}_{i \text{ times}}$ | |

## PRNG Mode

| | | |
|---|---|---|
| extract | $k$ | ...key |
| choose | $z_0$ | ...initial value |
| compute | $\left(e_k^{(i)}(z_0)\right)_{i \geq 0}$ | ...output stream |

## COUNTER MODE

| | | |
|---|---|---|
| choose | $k$ | ...key |
| compute | $x_0, x_1, \ldots$ (counter) | ...plaintext |
| compute | $(e_k(x_i))_{i \geq 0}$ | ...output stream |

# AES: Setups

Setup 1 – PRNG

| | |
|---|---|
| $k,\ z_0$ | …various cases (all-zero, random, …) |
| $\left(e_k^{(i)}(z_0)\right)_{i\geq 0}$ | …output stream |

Setup 2 – DIFF

| | |
|---|---|
| $k$ | …various cases (all-zero, random, …) |
| $(p_i)_{i\geq 0}$ | …highly patterned plaintext blocks |
| $(e_k(p_i))_{i\geq 0}$ | …output stream |

Setup 3 – PCOUNT

| | |
|---|---|
| $k$ | …various cases (all-zero, random, …) |
| $(p_i)_{i\geq 0}$ | …increasing counter |
| $(e_k(p_i))_{i\geq 0}$ | …output stream |

Setup 4 – KCOUNT

| | |
|---|---|
| $p_0$ | …plaintext block |
| $(k_i)_{i\geq 0}$ | …incrementing counter |
| $\left(e_{k_i}(p_0)\right)_{i\geq 0}$ | …output stream |

# *AES: Test I*

## Setup

- consider output bit stream $(y_i)_{i\geq 0}$ of AES
- cut out every 8th bit, i.e. take $y_0, y_8, \ldots$;
  this yields the bit stream

$$(x_i)_{i\geq 0} \qquad (x_i = y_{8i},\ i \geq 0)$$

- for each combination of dimension $d$ and sample size $n$, compute

$$\hat{\chi}_1^d(n),\ \hat{\chi}_2^d(n),\ \ldots,\ \hat{\chi}_{16}^d(n)$$

- Goodness-of-fit test (KS-Test)

## Test Parameters

| | |
|---|---|
| Sample size | $n = 2^{18}, 2^{19}, \ldots, 2^{28}$ bits |
| No. of repetitions | 16 indept. samples |
| Dimension | $d = 1, 2, 4, 8, 16$ |

# AES: Test II

## Setup

- consider output bit stream $(y_i)_{i \geq 0}$ of AES
- produce $d$-dimensional overlapping $d$-tuples

$$\tilde{y}_i^d = (y_i, y_{i+1}, \ldots, y_{i+d-1})$$

- (Dimension reduction) map each $d$-tuple to one of the three states -1, 0, 1
- for each combination of the dimension $d$ and the sample size $n$, compute

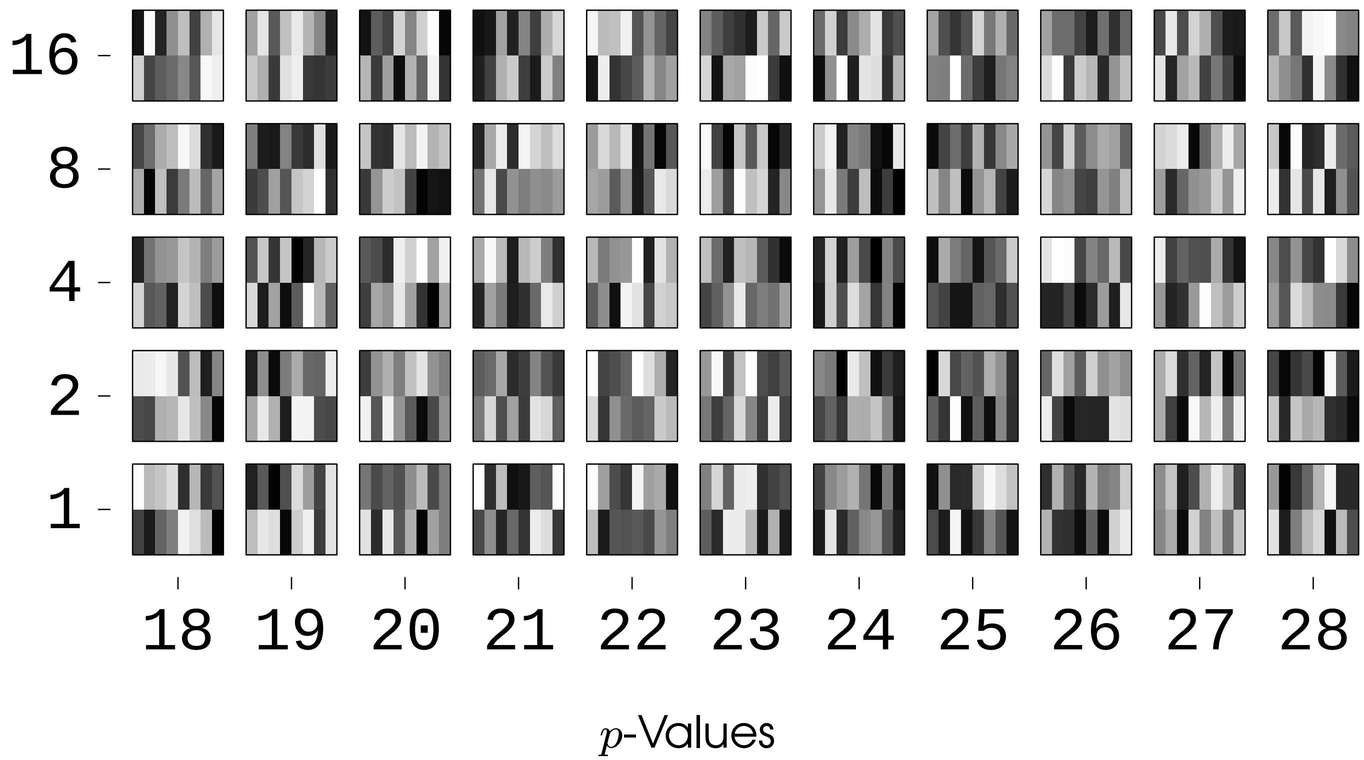
$$\hat{\chi}_1^d(n),\ \hat{\chi}_2^d(n),\ \ldots,\ \hat{\chi}_{16}^d(n)$$

- Goodness-of-fit test (KS-Test)

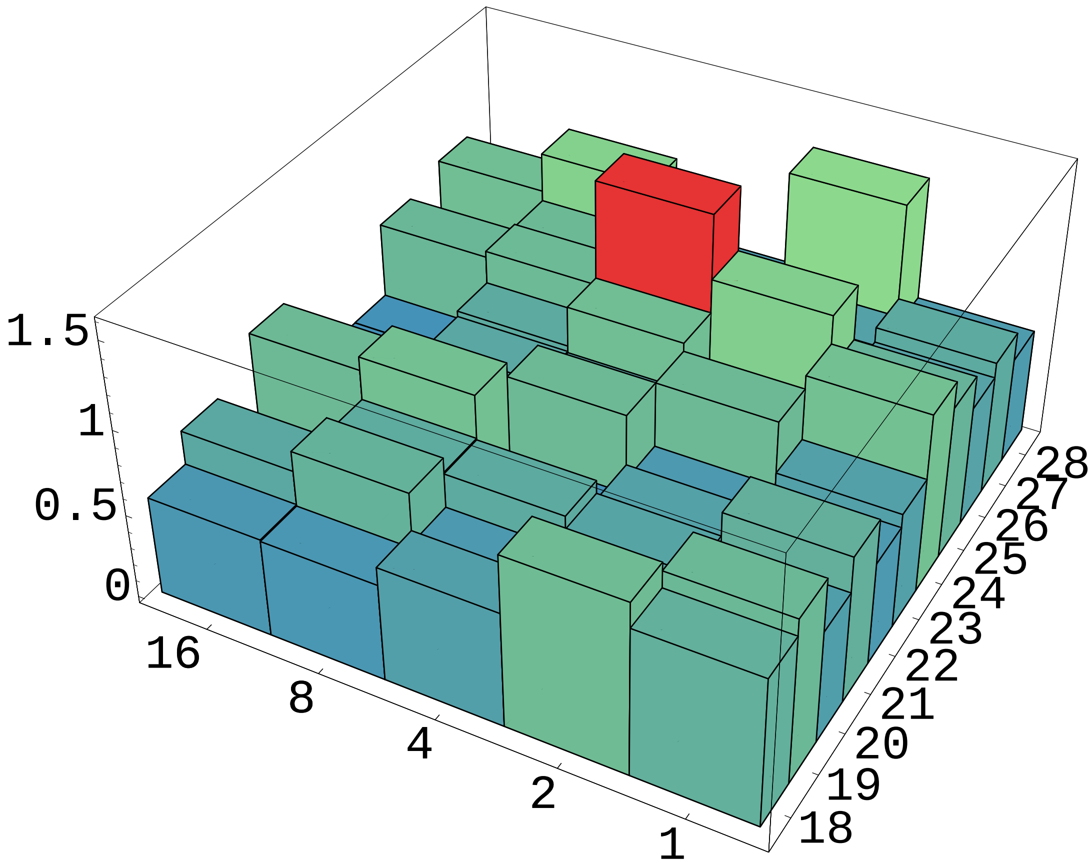
## Test Parameters

| | |
|---|---|
| Sample size | $n = 2^{22}, 2^{23}, \ldots, 2^{28}$ bits |
| No. of repetitions | 16 indept. samples |
| Dimension | $d = 32, 64, 128, 256$ |

# AES: Results of Test I



# AES: Results of Test I




KS-Test Values

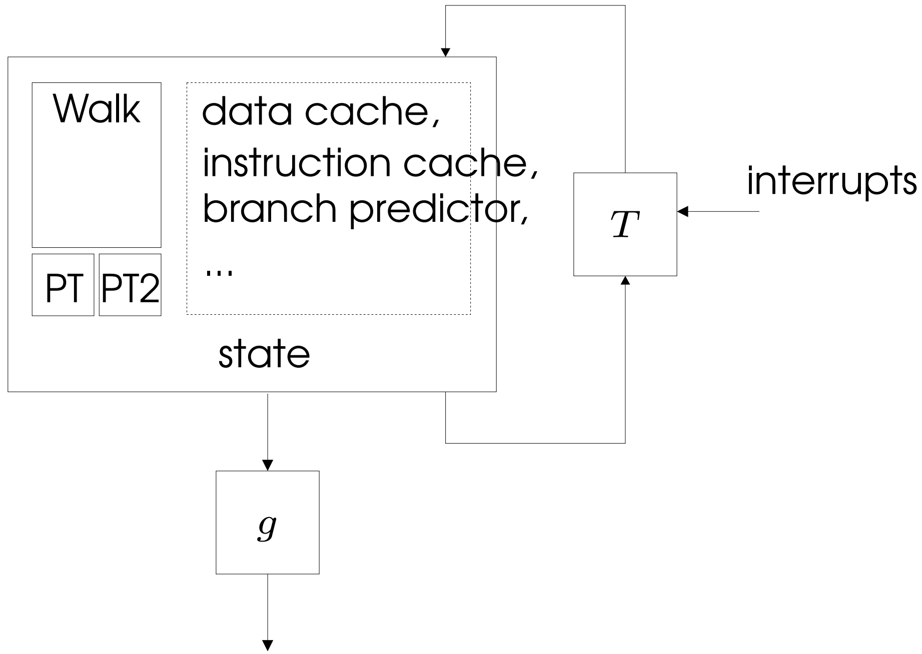
# *HAVEG and HAVEGE*

HArdware Volatile Entropy Gathering and Expansion
Sendrier and Seznec (INRIA, 2002)

- Uses processor interrupts to gather entropy
- HAVEG is a passive entropy harvester
- HAVEGE is active, acts on the processor

The idea behind
Each attempt to read the inner state of the processor alters it at the same time.
Complete state of HAVEGE cannot be observed without freezing the clock of the processor.

# Structure of HAVEGE




The General Structure of HAVEGE

# *RNGs: State of the Art*

Present Situation

Like in the race between cryptographers and cryptanalysts, presently the designers of RNGs are winning against the designers of statistical tests.

The intrinsic structures of modern RNGs, in particular of good cryptographic RNGs, are too complicated to be detected by current statistical tests.

Future developments

New ideas for testing are needed. This will take some time.