Generating Random and Pseudorandom Numbers

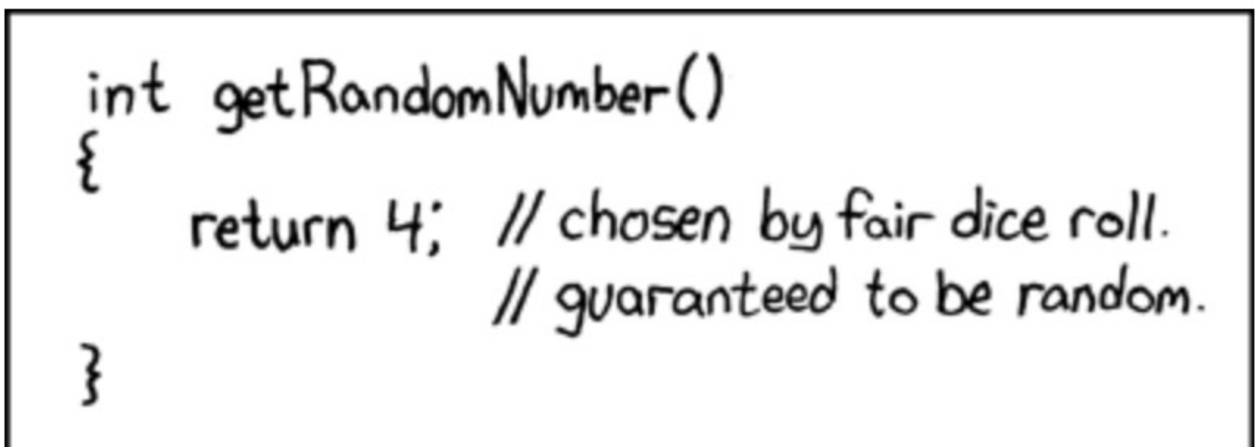
Michael Goodrich

CS 165

Some slides from CS 15-853:Algorithms in the Real World, Carnegie Mellon University

Random Numbers in the Real World

https://fitforrandomness.files.wordpress.com/2010/11/dilbert-does-randomness.jpg

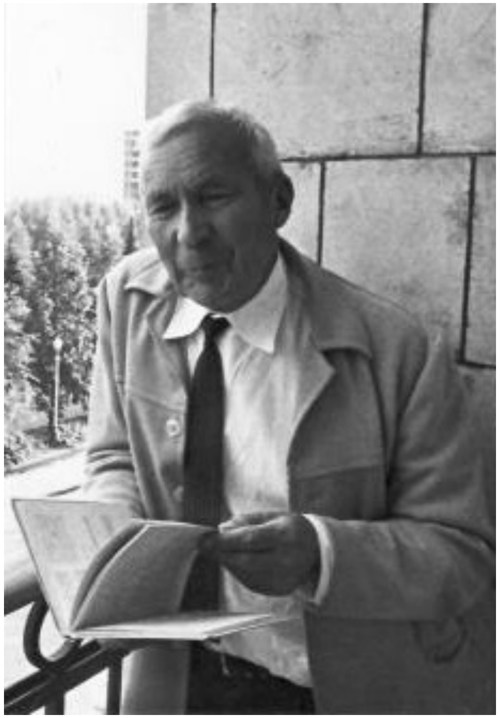


https://xkcd.com/221/

Random number sequence definitions

Randomness of a sequence is the Kolmogorov complexity of the sequence (size of smallest Turing machine that generates the sequence) – infinite sequence should require infinite size Turing machine.

This definition is useful for proving computational complexity results, but it is not as useful for algorithm experiments.



Andrey Kolmogorov

Random number sequence definitions

Each element is chosen independently from a probability distribution [Donald Knuth].

This definition is more usable for algorithm experiments.

A typical distribution is the **uniform** distribution, where every number in a range of numbers is equally likely.

Donald Knuth

Environmental Sources of Randomness

Radioactive decay http://www.fourmilab.ch/hotbits/

Radio frequency noise http://www.random.org

Noise generated by a resistor or diode.

- Canada http://www.tundra.com/ (find the data encryption section, then look under RBG1210. My device is an NM810 which is 2?8? RBG1210s on a PC card)
- Colorado http://www.comscire.com/
- Holland http://valley.interact.nl/av/com/orion/home.html
- Sweden http://www.protego.se

Inter-keyboard timings (watch out for buffering)

Inter-interrupt timings (for some interrupts)

Combining Sources of Randomness

Suppose $r_1, r_2, \ldots, r_k$ are random numbers from different sources. E.g.,

r_1 = from JPEG file

r_2 = sample of hip-hop music on radio

r_3 = clock on computer

$$b = r_1 \oplus r_2 \oplus \cdots \oplus r_k$$

If any one of $r_1, r_2, \ldots, r_k$ is truly random, then so is b.

Skew Correction

Von Neumann's algorithm – converts biased random bits to unbiased random bits:

Collect two random bits.

Discard if they are identical.

Otherwise, use first bit.

Efficiency?

John von Neumann

Chi Square Test

Experiment with k outcomes, performed n times.
$p_1, \ldots, p_k$ denote probability of each outcome
$Y_1, \ldots, Y_k$ denote number of times each outcome occured

$$\chi^2 = \sum_{1 \leq s \leq k} \frac{(Y_s - np_s)^2}{np_s}$$

Large X^2 indicates deviance from random chance

Analysis of random.org numbers

John Walker's Ent program

```
Entropy = 7.999805 bits per character.
Optimum compression would reduce the size of this
  1048576 character file by 0 percent.
 Chi square distribution for 1048576 samples is
  283.61, and randomly would exceed this value
  25.00 percent of the times.
 Arithmetic mean value of data bytes is 127.46
  (127.5 = random).
 Monte Carlo value for PI is 3.138961792 (error
  0.08 percent).
 Serial correlation coefficient is 0.000417
  (totally uncorrelated = 0.0
```

Analysis of JPEG file

Entropy = 7.980627 bits per character.

Optimum compression would reduce the size of this 51768 character file by 0 percent.

Chi square distribution for 51768 samples is 1542.26, and randomly would exceed this value 0.01 percent of the times.

Arithmetic mean value of data bytes is 125.93 (127.5 = random).

Monte Carlo value for Pi is 3.169834647 (error 0.90 percent).

Serial correlation coefficient is 0.004249 (totally uncorrelated = 0.0).

Pseudorandom Number Generators

- A pseudorandom number generator (PRNG) is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.
- The PRNG-generated sequence is not truly random, because it is completely determined by an initial value, called the PRNG's seed (which may include truly random values).
- Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom number generators are important in practice for their speed and reproducibility.

Pseudorandom Number Generators

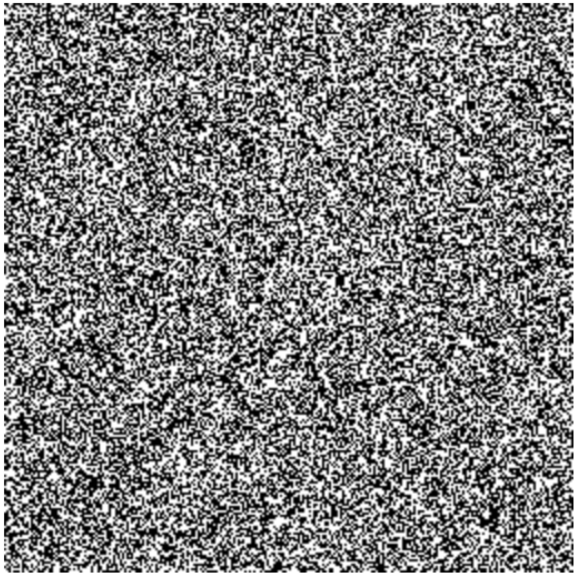
- PRNGs are central in applications such as simulations (e.g. for the Monte Carlo method), electronic games (e.g. for procedural generation), and cryptography.
- Cryptographic applications require the output not to be predictable from earlier outputs.

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

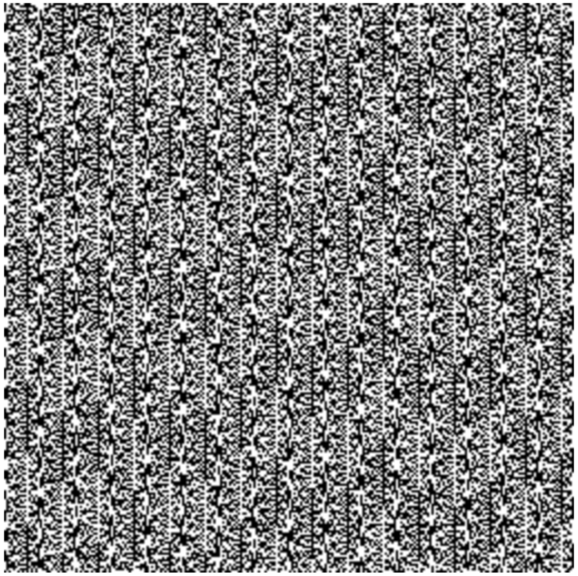
- John Von Neumann, 1951

Simple Visual Test

- Create a visualization of the consecutive tuples of numbers it produces.
- Humans are really good at spotting patterns.



RANDOM.ORG



PHP rand() on Microsoft Windows

Linear Congruential Generator (LCG)

x_0 = given, $x_{n+1} = P_1 x_n + P_2 \pmod N$ $n = 0,1,2,\ldots$ (*)

$x_0 = 79$, $N = 100$, $P_1 = 263$, and $P_2 = 71$

$x_1 = 79*263 + 71 \pmod{100} = 20848 \pmod{100} = 48$,
$x_2 = 48*263 + 71 \pmod{100} = 12695 \pmod{100} = 95$,
$x_3 = 95*263 + 71 \pmod{100} = 25056 \pmod{100} = 56$,
$x_4 = 56*263 + 71 \pmod{100} = 14799 \pmod{100} = 99$,

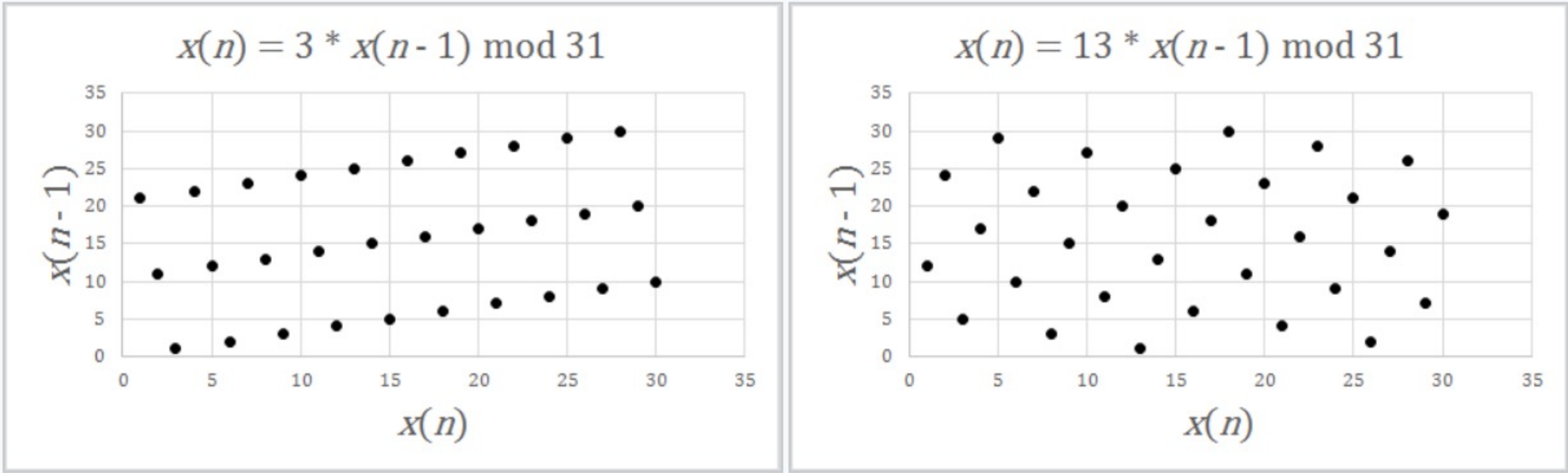
Sequence: 79, 48, 95, 56, 99, 8, 75, 96, 68, 36, 39, 28, 35, 76, 59, 88, 15, 16, 79, 48, 95

Park and Miller:
$P_1 = 16807$, $P_2 = 0$, $N = 2^{31}-1 = 2147483647$, $x_0 = 1$.

ANSI C rand():

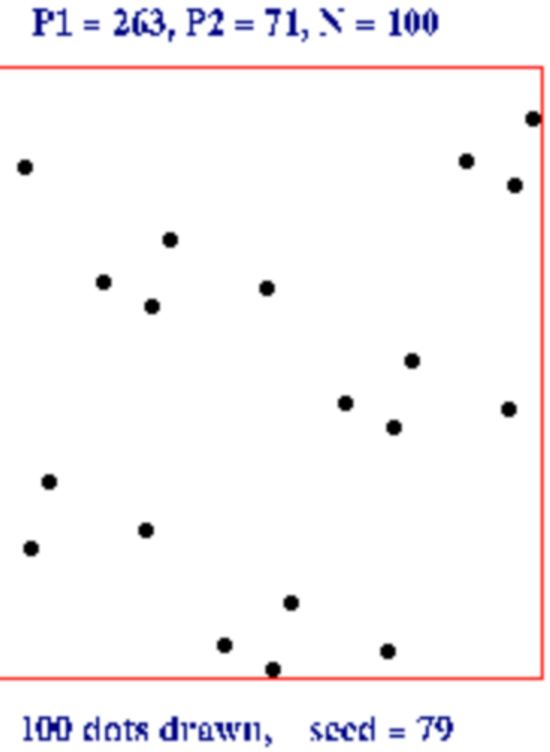
$P_1 = 1103515245$, $P_2 = 12345$, $N = 2^{31}$, $x_0 = 12345$

Example Comparison

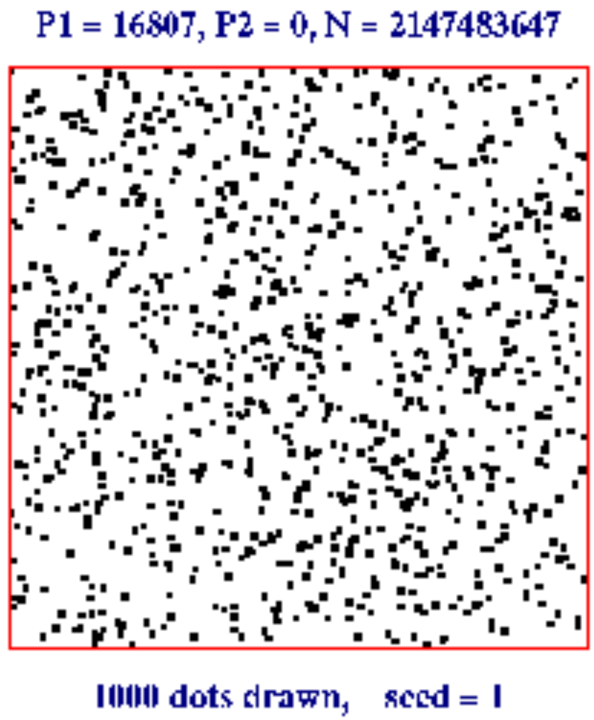


Despite the fact that both relations pass the Chi-squared test, the first LCG is less random than the second, as the range of values it can produce by the order it produces them in is less evenly distributed.

Plot (x_i, x_{i+1})



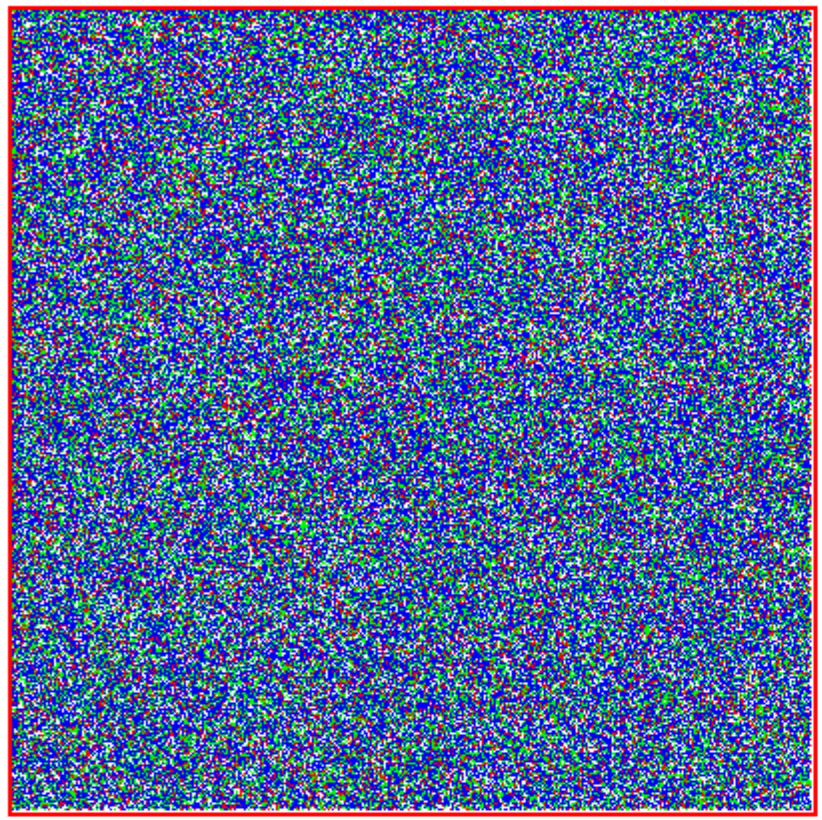
Plot (x_i, x_{i+1})



Park and Miller

(x_i, x_{i+1}), (x_i, x_{i+2}), (x_i, x_{i+2})

P1 = 16807, P2 = 0, N = 2147483647

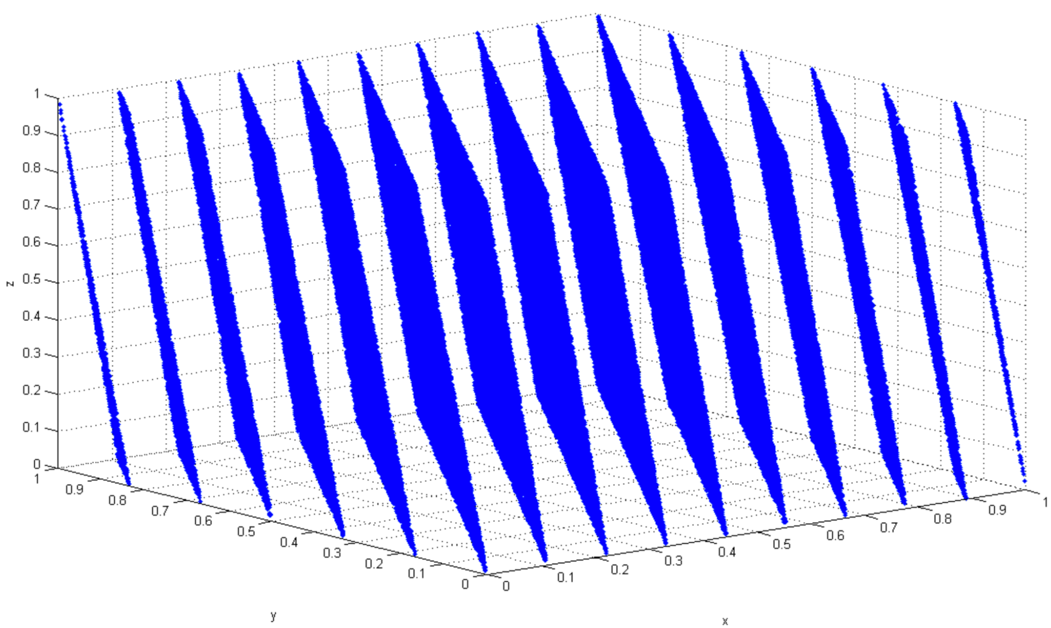


100000 dots drawn, seed = 1

http://www.math.utah.edu/~alfeld/Random/Random.html

Visual Test in 3D

- Three-dimensional plot of 100,000 values generated with IBM RANDU routine. Each point represents 3 consecutive pseudorandom values.
- It is clearly seen that the points fall in 15 two-dimensional planes.

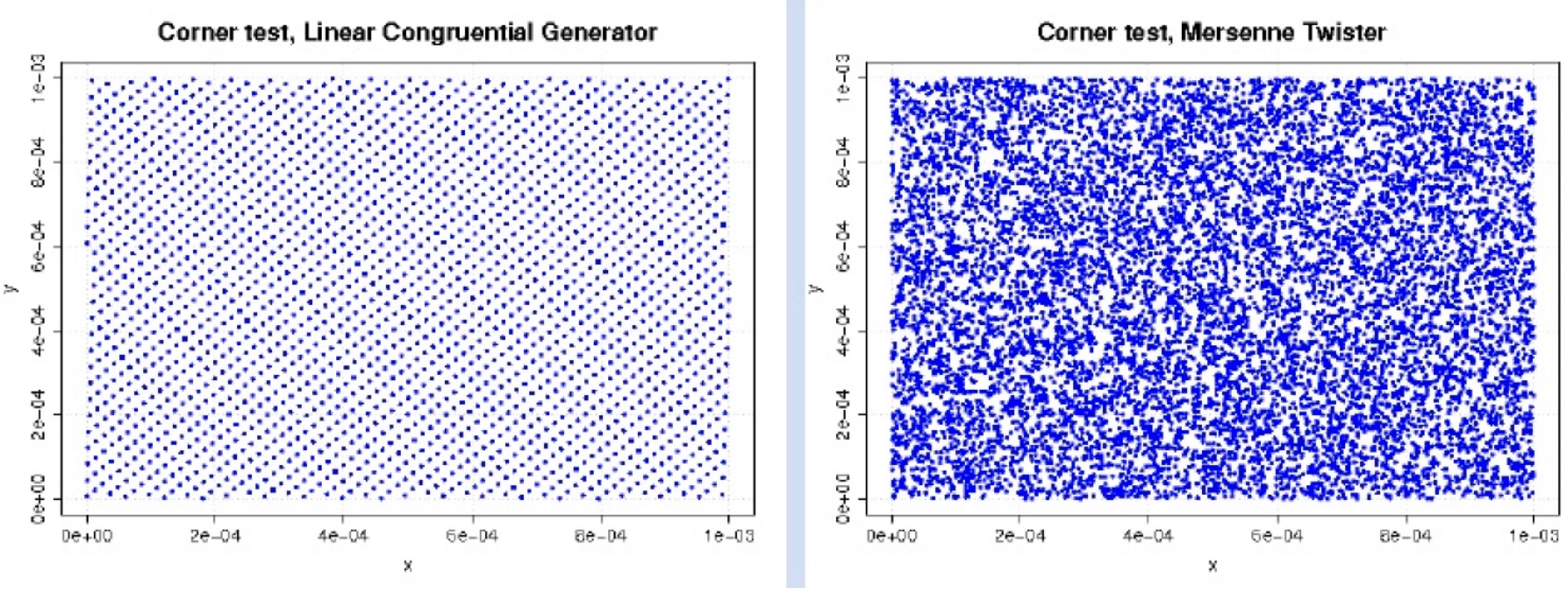


Matsumoto's Marsenne Twister

Considered one of the best linear congruential generators.

http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

Example Visual Test

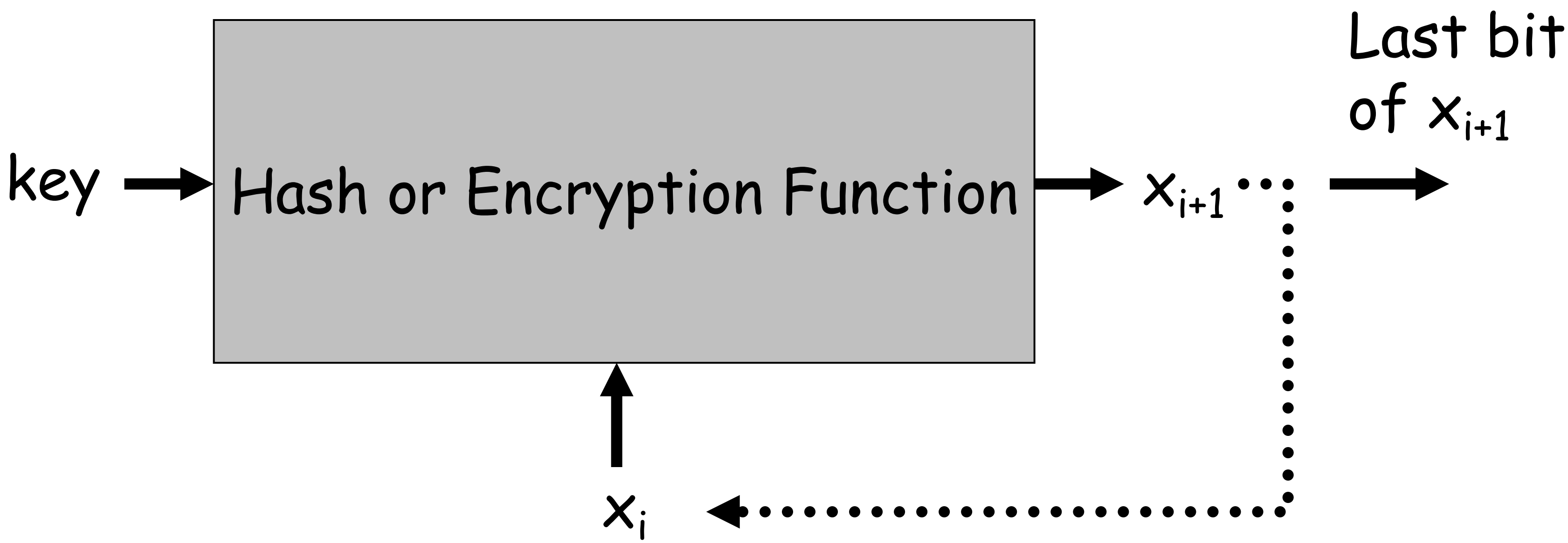


Cryptographically Strong Pseudorandom Number Generator

Next-bit test: Given a sequence of bits $x_1, x_2, \ldots, x_k$, there is no polynomial time algorithm to generate x_{k+1}.

Yao [1982]: A sequence that passes the next-bit test passes all other polynomial-time statistical tests for randomness.

Hash/Encryption Chains



(need a random seed x_0 or key value)

Some Cryptographic Hash Functions

- SHA-1 Hash function https://en.wikipedia.org/wiki/SHA-1
- MD5 Hash function https://en.wikipedia.org/wiki/MD5
- These functions are good pseudo-random number generators and when seeded with a random number generator, they provide good sequences for use in algorithm experiments.

BBS "secure" random bits

BBS (Blum, Blum and Shub, 1984)

- Based on difficulty of factoring, or finding square roots modulo n = pq.

Fixed

- p and q are primes such that p = q = 3 (mod 4)
- n = pq (is called a Blum integer)

For a particular bit seq.

- **Seed:** random x relatively prime to n.
- **Initial state:** $x_0 = x^2$
- **i^{th} state:** $x_i = (x_{i-1})^2$
- **i^{th} bit:** lsb of x_i

Note that: $x_0 = x_i^{-2^i \bmod \phi(n)} \pmod{n}$

Therefore knowing p and q allows us to find x_0 from x_i

Random Numbers in Python

https://docs.python.org/3/library/random.html

[Review this website]