



# Random Bit Generators

Gianluca Dini


Dept. of Ingegneria dell'Informazione – University of Pisa

Email: [gianluca.dini@unipi.it](mailto:gianluca.dini@unipi.it)

Version: 2024-02-28

1

## Random Bit Generator



UNIVERSITÀ DI PISA

- DEFINITION. A Random Bit Generator (RBG) outputs a sequence of **statistically independent** and **unbiased** bits
  - **Statistically independent** means that the probability of emitting a bit value (1 or 0) does not depend on the previous bits
  - **Unbiased** means that the probability of emitting a bit value (1 or 0) is equal to 0.5

Feb-24

Random Generators

2

2

## Random Bit Generators



UNIVERSITÀ DI PISA

- **Random Number Generators (RNGs)** can be used to generate uniformly distributed random numbers
- A random number in the interval  $[0, n]$  can be obtained by generating a bit sequence of length  $\lceil \lg n \rceil + 1$  and converting it to an integer;
  - If the resulting number exceeds  $n$ , one possible option is to discard it and generate another random bit sequence

Feb-24

Random Generators

3

3

## Random Bit Generators



UNIVERSITÀ DI PISA

- **Classes of RBGs**
  - True random bit generators (TRBG)
  - Pseudorandom Bit Generator (PRBG)
  - Cryptographically Secure Pseudorandom Bit Generator (CSPRBG)

Feb-24

Random Generators

4

4

## True Random Bit Generators



UNIVERSITÀ DI PISA

- Based on a physical process
  - Coin flipping, rolling a dice, semiconductor noise, clock jitter, radioactive decay
- The output «cannot» be reproduced
  - $\text{Pr}[\text{flipping a coin 100 times and generate a given 100-long sequence}] = 1/2^{100}$
- Classification
  - Hardware-based generators
  - Software-based generators

Feb-24

Random Generators

5

5

## TRBG – Hardware-based



UNIVERSITÀ DI PISA

- Physical phenomena
  - elapsed time between emission of particles during radioactive decay
  - thermal noise from a semiconductor diode or resistor
  - the frequency instability of a free running oscillator
  - the amount a metal-insulator semiconductor capacity is charged during a fixed period of time
  - air turbulence within a sealed disk drive which causes random fluctuations in disk drive sector read latency times
  - sound from a microphone or video from a camera

Feb-24

Random Generators

6

6

## TRBG – Hardware-based



UNIVERSITÀ DI PISA

- Example: Intel Digital Random Number Generator
- Introduced in Intel CPUs since 2012
- Based on [NIST SP 800-90](#)
- Exploits thermal noise fluctuations with the CPU
- [DRNG and RDRAND/RDSEED assembly instructions](#)
- Partially documented

Feb-24

Random Generators

7

7

## TRBG – Hardware-based



UNIVERSITÀ DI PISA

- Subject to external influence and malfunction
  - Subject to observation and manipulation
- Periodic tests
- Defective generators
  - Biased: Probability of emitting a 1 is not equal to 0.5
  - Correlated: Probability of emitting a 1 depends on previous bit emitted
- De-skewing techniques: generate truly random bit sequences from the output bits of a defective generator
  - A practical technique is to pass the sequence through a cryptographically secure hash function

Feb-24

Random Generators

8

8

## TRBG – Hardware-based



UNIVERSITÀ DI PISA

- Deskewing: an example
  - Suppose that i) a generator produces biased but uncorrelated bits; ii) the probability of a 1 is  $p$ , and the probability of a 0 is  $1-p$ , where  $p$  is unknown but fixed,  $0 < p < 1$ . Remove biases in output bits.
- Solution
  - Group the output sequence into pairs of bits, with
    - a 10 pair transformed to a 1,
    - a 01 pair transformed to a 0, and
    - 00 and 11 pairs are discarded
  - The resulting sequence is both unbiased and uncorrelated

Feb-24

Random Generators

9

9

## TRBG – Software-based



UNIVERSITÀ DI PISA

- Processes
  - the system clock
  - elapsed time between keystrokes or mouse movement
  - content of input/output buffers
  - user input
  - operating system values such as system load and network statistics

Feb-24


Random Generators

10

10

## TRBG – Software-based

- Subject to observation and manipulation
- Use as many sources of randomness as possible
  - Mixing functions
    - E.g., Cryptographically secure hash functions (SHA-1, MD5)



UNIVERSITÀ DI PISA

Feb-24

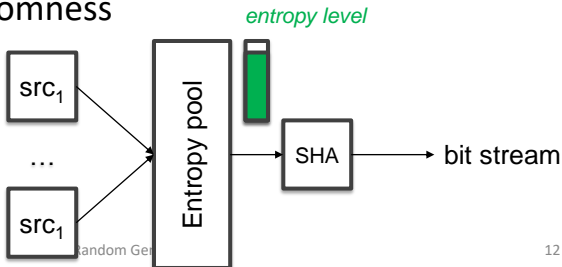
Random Generators

11

11

## Sw-based RNG: the Linux case

- $src_i$ : i-th source of randomness
  - Inter-key press timing, inter-interrupt timing,...
- Two char devices
  - `/dev/random`: higher-quality, blocking
  - `/dev/urandom`: lower quality, not-blocking
- Boot time randomness



```
graph LR; src1[src1] --> pool[Entropy pool]; dots[...]; src2[src2] --> pool; pool -- "entropy level" --> sha[SHA]; sha --> stream[bit stream];
```

The diagram illustrates the Linux software-based random number generation process. Multiple sources of randomness ( $src_1, \dots, src_n$ ) feed into a central 'Entropy pool'. The pool's state is represented by a vertical bar with a green fill level, labeled 'entropy level'. The output of the pool is processed by a SHA (Secure Hash Algorithm) block, which then produces a 'bit stream'.

Feb-24

Random Gen

12

12

Random Bit Generators

PSEUDORANDOM BIT GENERATORS


Feb-24

Random Generators

13

13

Pseudo Random Bit Generator



UNIVERSITÀ DI PISA


- DEFINITION. A Pseudo Random Bit Generator is a *deterministic* algorithm that, given a truly random binary sequence of length  $k$  (*seed*), outputs a binary sequence of length  $L$  (pseudorandom bit sequence),  $L \gg k$ 
  - The number of possible sequences is at most  $2^k$ , i.e., a fraction  $2^k/2^L$  of all possible sequences

Feb-24

Random Generators

14

14

  
UNIVERSITÀ DI PISA

## Pseudo Random Bit Generator


- SECURITY INTUITION. A “small” seed is expanded into a “large” *pseudorandom* sequence in such a way that an adversary cannot “efficiently” distinguish between outputs of a PRBG and outputs of a TRG
- MINIMUM SECURITY REQUIREMENT. The length  $k$  of the seed is sufficiently large so that it is “infeasible” to search over  $2^k$  possible output sequences (necessary condition)

Feb-24

Random Generators

15

15

  
UNIVERSITÀ DI PISA

## Formalization

For all attackers (tests)  $\mathcal{A}$ , there is negligible function  $\varepsilon(n)$ , s.t.:

$$\left| P_{x \leftarrow U_k} [A(G(x)) = 1] - P_{y \leftarrow U_{\ell(k)}} [A(y) = 1] \right| \leq \varepsilon(n)$$

where  $G: \{0, 1\}^k \rightarrow \{0, 1\}^{\ell(k)}$ ,  $x$  is uniformly sampled from  $\{0, 1\}^k$ , and  $U_{\ell(k)}$  is the uniform distribution on  $\ell(k)$ -bit strings

Feb-24

Random Generators


16

16



# PRBG

- Typically
  - $s_0 = \text{seed}$
  - $s_{i+1} = f(s_i), i = 0, 1, 2, \dots$
- A generalization
  - $s_0 = \text{seed}$
  - $s_{i+1} = f(s_i, s_{i-1}, s_{i-2}, \dots, s_{i-t}, )$



UNIVERSITÀ DI PISA

Feb-24


Random Generators

17

17

# PRBG

- Linear Congruential Generator
  - A popular example largely used in simulation and testing
  - Definition
    - $s_0 = \text{seed}$
    - $s_{i+1} = (a \cdot s_i + b) \bmod m, i = 0, 1, 2, \dots$
    - where  $a, b, m$  are integer constants
  - ANSI C rand()
    - $s[0] = 12345;$
    - $s[i] = 1103515245 s[i-1] + 12345 \times 2^{31}$



UNIVERSITÀ DI PISA

Feb-24

Random Generators

18

18

## LCG predictability



- Assume a prefix  $s_r, s_{r+1}, s_{r+2}$  is known
- Define
  - $s_{r+2} = a \cdot s_{r+1} + b \bmod m$
  - $s_{r+1} = a \cdot s_r + b \bmod m$
  - which is a linear system of two linear equations in two unknowns (a and b) that can be “easily” solved

Feb-24

Random Generators

19

19

## PRBG



- Linear Congruential Generator has good statistical properties
  - Output approximates a sequence of true random bits
  - It passes a variety of statistical tests
- Not suitable for cryptography because it is predictable

Feb-24

Random Generators

20

20

Random Bit Generator

CRYPTOGRAPHICALLY SECURE  
PSEUDORANDOM BIT GENERATOR


Feb-24

Random Generators

21

21

CSPRBG

  
UNIVERSITÀ DI PISA

- Informally, a **CSPRNG is an unpredictable PRNG**
  - The need for unpredictability is unique for cryptography
- Informally,
  - Given a sequence of bits  $s_i, s_{i+1}, \dots, s_{i+n-1}$  (a prefix), for some integer  $n$ , it is «difficult» to compute the subsequent bits  $s_{i+n}, s_{i+n+1}, \dots$  (or any preceding bits  $s_{i-1}, s_{i-2}, \dots$ )
- More formally
  - Given a sequence of bits  $s_i, s_{i+1}, \dots, s_{i+n-1}$  (a prefix), there exist no polynomial time algorithm that can predict the next bit  $s_{i+n}$  with better than 50% chance of success

Feb-24


Random Generators

22

22

# CRPRBG

- General Security Requirements
  - A PRBG is said to pass all polynomial-time statistical tests if no polynomial-time algorithm can correctly distinguish between an output sequence of the generator and a truly random sequence of the same length with probability significantly greater than 0.5
  - A PRBG is said to pass the next-bit test if there is no polynomial-time algorithm which, on input of the first  $t$ -bits of an output sequence  $s$ , can predict the  $(t + 1)$ -st bit of  $s$  with probability significantly greater than 0.5
  - Polynomial-time statistical tests and next-bit test are equivalent



UNIVERSITÀ DI PISA

Feb-24


Random Generators

24

24

# CRPRBG

- CSPRBG DEFINITION. A PRBG that passes<sup>(\*)</sup> the next-bit test is called cryptographically secure pseudorandom bit generator
  - <sup>(\*)</sup> possibly under some plausible but unproven mathematical assumption such as the intractability of factoring integers



UNIVERSITÀ DI PISA


Feb-24

Random Generators

25

25

# CSPRBG



UNIVERSITÀ DI PISA

- **Ad-hoc methods**, based on one-way functions
  - Hash functions, block ciphers
  - ANSI X9.17, FIPS 186
  - They have not been proven to be CSPRBG, however they are sufficient for most applications
- Based on presumed intractability of number-theoretic problem
  - RSA PRBG (integer factorization)
  - Blum-Blum-Shub PRBG (integer factorization)

Feb-24

Random Generators

26

26

Random Bit Generators

# STATISTICAL TESTS


Feb-24

Random Generators

27

27

## Statistical tests



UNIVERSITÀ DI PISA

- A set of **statistical tests** have been devised to measure the quality of an RBG
  - It is not possible to prove whether a generator is indeed an RBG; tests detect weaknesses
  - Tests provide **necessary conditions**
    - Each test operates on a given output sequence and **probabilistically** determines whether it possesses a certain attribute that a truly random sequence would exhibit
  - A generator may be either **rejected** or **accepted** (i.e., not rejected)


Feb-24

Random Generators

28

28

## Statistical tests



UNIVERSITÀ DI PISA

- Five basic tests
  - Frequency test (monobit test).
    - Determine whether the number of 0's and 1's are approximately the same
  - Serial test (two-bit test).
    - Determine whether the number of occurrences of 00, 01, 10, 11 are approximately the same
  - Poker test.
    - Determine whether the sequences of length m each appear approximately the same number of times

Feb-24

Random Generators

29

%

29

## Statistical tests



UNIVERSITÀ DI PISA

- Basic tests
  - Runs test.
    - Determine whether the number of runs of various length is as expected for a random sequence
  - Autocorrelation test.
    - Check correlations between the sequence and shifted (non-cyclic) versions of it

Feb-24

Random Generators

30

30

## Statistical tests



UNIVERSITÀ DI PISA

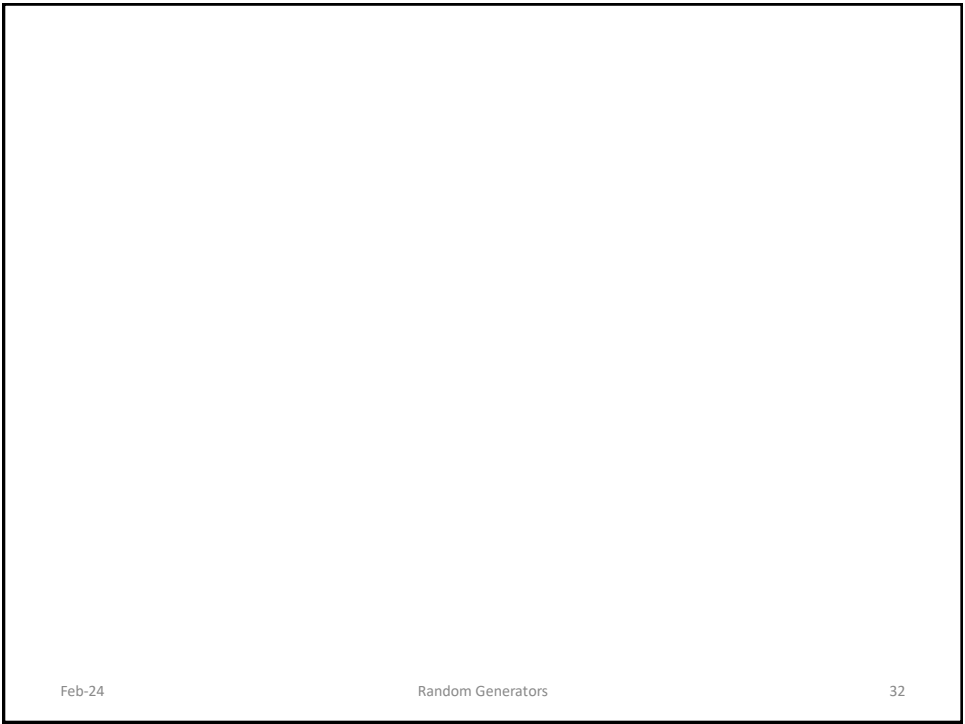
- Maurer's universal statistical test
  - Intuition: It is not possible to **significantly compress** (without loss of information) the output sequence of a random generator
  - Determine a very general class of possible defects (universality)
    - Including defects detectable by basic tests
  - Require a longer sequence than basic tests but more efficient than basic tests

Feb-24

Random Generators

31

31



32