Random Bit Generators

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 2022-03-03

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

March 22 FoC - Random Generators 2

2

Random Bit Generators

- 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 $\lfloor \lg n \rfloor + 1$ 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

Random Bit Generators

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

True Random Bit Generators

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

March 22 FoC - Random Generators 5

5

TRBG – Hardware-based

- 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

March 22 FoC - Random Generators 6

6

Example: Intel Digital Random Number Generator

- Introduced in Intel CPUs since 2012
- Based on NIST SP 800-90
- Exploit thermal noise fluctuations with the CPU
- RDRAND assembly instructions
- Partially documented

TRBG – Hardware-based

- 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

TRBG – Exercise on deskewing

UNIVERSITÀ DI PISA

- The problem
  - Suppose that a generator produces biased but uncorrelated bits. Suppose that 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

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
- Use as many sources of randomness as possible
  - E.g., Cryptographically secure hash functions

TRBG – Software-based

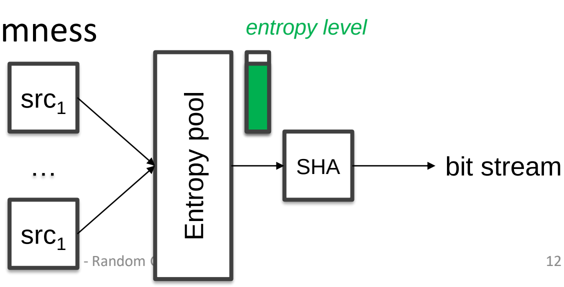
UNIVERSITÀ DI PISA

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

Sw-based RNG: the Linux case

UNIVERSITÀ DI PISA

- 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



Random Bit Generators

PSEUDORANDOM BIT GENERATORS

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 lenght k (*seed*), outputs a binary sequence of length L (pseudorandom bit sequence), L >> k
  - The number of possible sequences is at most 2^k, i.e., a fraction $2^k/2^L$ of all possible sequences

Pseudo Random Bit Generator

UNIVERSITÀ DI PISA

- 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)

March 22 FoC - Random Generators 15

15

Formalization

UNIVERSITÀ DI PISA

For all attackers (tests) A, there is negligible function ε(n), s.t.:

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

March 22 FoC - Random Generators 16

16

PRBG

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

March 22 FoC - Random Generators 17

17

PRBG

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

March 22 FoC - Random Generators 18

18

LCG predictabily

- 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

PRBG

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

Random Bit Generator

CRYPTOGRAPHICALLY SECURE PSEUDORANDOM BIT GENERATOR

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}, …, 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}, … (or any preceding bits s_{i-1}, s_{i-2}, …)
- More formally
  - Given a sequence of bits s_i, s_{i+1}, …, 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

CRPRBG

UNIVERSITÀ DI PISA

- 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

March 22 FoC - Random Generators 24

24

CRPRBG

UNIVERSITÀ DI PISA

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

March 22 FoC - Random Generators 25

25

CSPRBG

- 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)

March 22 FoC - Random Generators 26

26

Random Bit Generators

STATISTICAL TESTS

March 22 FoC - 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)

March 22 FoC - 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

%

March 22 FoC - Random Generators 29

29

Statistical tests

- 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

Statistical tests

- 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

32