Random Number Generators

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 2021-03-08

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 (1 or 0) value 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

2

Random Bit Generators

- Random Number Generators (RNGs)
  - Random Bit Generators 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 (TRBGs)

UNIVERSITÀ DI PISA

- 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] = $1/2^{100}$
- Used to generate session keys

TRBGs

UNIVERSITÀ DI PISA

- Classification
    - Hardware-based generators
    - Software-based generators
- Remarks
    - May be subject to observation or manipulation by an adversary
    - May be subject to influence of external factors, and also to malfunction
    - Need to be tested periodically

TRBG – Hardware-based

- Examples
  - 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

TRBG – Hardware-based

- 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
  - Make it possible to 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 – Software-based

- Examples of random 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 mixing functions
  - E.g. Cryptographically secure hash functions

Random Generators

EXERCISE

On Deskewing

UNIVERSITÀ DI PISA

- Exercise
  - 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.

On Deskewing

UNIVERSITÀ DI PISA

- Exercise #1 – Solution
  - Group the output sequence of such a generator 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
  - Then the resulting sequence is both unbiased and uncorrelated

Pseudo Random Bit Generator (PRBG)

- 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
- Security intuition
  - A “small” seed is expanded in a “large” pseudorandom sequence in such a way that an adversary cannot efficiently distinguish between outputs of a PRBG and outputs of a TRG

PRBG

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

PRBG

UNIVERSITÀ DI PISA

- Linear Congruential Generator
  - A popular example
  - 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 $\times 2^{31}$

PRBG

UNIVERSITÀ DI PISA

- Linear Congruential Generator
  - Properties
    - Good statistical properties
      - Output approximates a sequence of true random number
      - Mathematical tests (e.g., chi-square test)
    - Largely used in simulation and testing
    - Not suitable for cryptography because it's predictable

LCG predicatibily

- 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)

CSPRBG

- A CSPRNG is an unpredictable PRNG
  - Unpredictability
    - Informally - Given a sequence of bits s_i, s_{i+1}, …, s_{i+n-1} (a prefix), for some integer n, it is computationally infeasible to compute the subsequent bits s_{i+n}, s_{i+n+1}, …
    - 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
    - Another property - Given a sequence of bits s_i, s_{i+1}, …, s_{i+n-1} (a prefix), it is infeasible to compute the preceding bits s_{i-1}, s_{i-2},…
  - The need for unpredictability is unique for cryptography

CSPRBG

- Security Requirements
  - Minimum security requirement
    - Seed length k is so large that a search over 2^k seeds is infeasible
  - General security requirements
    - The output sequence of the generator should be indistinguishable from truly random sequences
    - The output bits should be unpredictable to an adversary with limited computational resources

CRPRBG

- General Security Requirements
  - A PRBG is said to pass the polynomial-time statistical test 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

CRPRBG

UNIVERSITÀ DI PISA

- Universality of the next-bit test
  - A PRBG passes the next-bit test if and only if it passes all polynomial-time statistical tests

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

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
- CSPRBG
  - Assumption that integer factorization is intractable
    - RSA PRBG
    - Blum-Blum-Shub PRBG

Statistical tests

- 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 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)

Statistical tests – 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

%

Statistical tests – basic tests [↓]

- Poker test.
  - Determine whether the sequences of length m each appear approximately the same number of times
- 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 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