Pseudo Random Number Generation Lab

Randomness

- essential in Cryptography!
  - key generation, unique numbers (e.g. nonces), seeds for ciphers...
- generation
  - excellent: physical source
    - inherent: radioactive decay, brownian movement, ...
    - depending on initial conditions: (non-biased) roulette or dice, ...
  - reasonable: algorithmic-based with physical seed
    - cryptographically secure pseudorandom number generators
      - use physical (≅ random) sources (e.g. mouse movements)[1]
  - bad: algorithmic-based
    - pseudorandom number generators (e.g. POSIX's `random()`)

1 Linux's `getrandom()` (`/dev/random`, `/dev/urandom`)

Cryptographically Secure Pseudorandom Number Generators

- start with an initial number as "random" as possible
- feed it to cryptography calculators
  - Hash Function-Based
    - repeatedly hash internal state and output parts of the hash
  - Block Cipher-Based
    - repeatedly encipher running counter and output part of the cipher
  - ...
- outputs should have uniform frequency distribution
  - and are used to seed other Cryptographically Secure Pseudorandom Number Generators!

Evaluation

- frequency analysis
  - determine the frequency distribution of digit or bit patterns of a sequence of values:
    - if (truly) random, each digit or bit occurs with approximately equal frequency
- entropy measurement[1]
  - measure of the unpredictability of the values in sequence:
    - if values are (truly) random, unpredictability (so, entropy) is maximum

1 Calculation (and concept) of entropy varies. In computing and cryptography, it is as the original, in Shannon's Information Theory: entropy E (in bits) = $-\sum_i$ [(probability of occurrence of value i)*$\log_2$ (probability of occurrence of value i)] , where i is a value from a set. If i is one bit, and its 0 or 1 value occurs with equal probability, $E = 1$ (bit).
If a cryptographic key has 128 (equally probable, random) bits, its entropy is 128 b, as the probability of a specific 128-b value is 1/number_of_possible_values = $1/ 2^{128}$.

...Randomness: evaluation...

- statistical tests
  - examination of properties such as uniformity, independence and distribution of sequence values. Examples: Chi-square[1], Kolmogorov-Smirnov[2], RUNS[3].
    - if sequence is (truly) random, results match specific value (type of test dependent)
- serial correlation measurement
  - check for correlations between successive values:
    - if (truly) random sequence of values, correlation should be zero
- *randomness* general tests
  - run specialized tests. Examples of test suites: NIST Statistical[4], Dieharder[5], ENT[6].
    - results will show degree of proximity to (true) randomness

1 en.wikipedia.org/wiki/Chi-squared_test
2 en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
3 en.wikipedia.org/wiki/Wald%E2%80%93Wolfowitz_runs_test
4 nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
5 webhome.phy.duke.edu/~rgb/General/dieharder.php
6 www.fourmilab.ch/random/

ENT, A Pseudorandom Number Sequence Test Program

- battery of tests:
  - frequency (ideal: all values with same number of occurrences)
  - entropy (ideal: 8 bits per byte)
  - compression (ideal: 0 % compression)
  - Chi-square (ideal:] ~10%, ~90% [)
  - arithmetic mean (ideal: 50% of possible values)
  - Monte Carlo value for Pi (ideal: Pi with very "low" error)
  - Serial correlation coefficient (ideal: 0)
- used in a SEED lab!