Random and Pseudorandom Bit Generators

- Random bit generators
- Pseudorandom bit generators
- Cryptographically Secure PRBG
- Statistical tests

Unpredictable quantities

- The security of many cryptographic systems depends on the generation of **unpredictable** quantities
- These quantities must be of **sufficient size** and **random** in the sense that
  - the probability of any particular value being selected must be sufficiently small to preclude an adversary from gaining advantage through optimizing a search strategy based on such probability
Random bit generator

- RBG requires a naturally occurring source of randomness

**RBG** sequence of statistically independent and unbiased bits

Probability of emitting 1 does not depend on the previous bits

Probability of emitting 1 is equal to 0.5

Hardware-based generators

- HW-based RBGs exploit the randomness in some 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
Software-based generators

- Random processes used by SW-based RBGs include
  - 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
- A well-designed SW-based RBG uses as many sources as available

Design and implementation problems

- RBG must not be subject to observation and manipulation by an adversary
- The natural source of randomness is subject to influence by external factors and to malfunction
- RBG must be tested periodically
De-skewing techniques

- A natural source of randomness may be defective and produce biased and correlated output bits.
- De-skewing techniques make it possible to generate truly random bit sequences from the output bits of a defective generator.
- De-skewing techniques:
  - provable
  - practical

Pseudorandom bit generation

RBGs raise problems

- Generation of (truly) random bits is an inefficient procedure in most practical systems.
- Storage and transmission of a large number of random bits may be impractical.

- These problems can be ameliorated by substituting a RBG with a Pseudorandom Bit Generator (PRBG).
Pseudorandom Bit Generator

- PRBG is a **deterministic** algorithm
- An adversary must not **efficiently** distinguish between output sequences of PRBG and truly random bit sequences

**Requirements**

- **Minimum security requirement**
  - $k$ should be sufficiently large to make an exhaustive search over $2^k$ seeds practically infeasible

- **General requirements**
  - A PRBG passes 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 passes the **next-bit test** if there is no polynomial-time algorithm which, on input of the first $l$-bits of an output sequence $s$, can predict the $(l + 1)^{st}$ bit of $s$ with probability significantly greater than 0.5
  - These two requirements are equivalent

- A PRBG that passes tests is said **cryptographically secure**
Ad-hoc PRBG

- **One-way functions** can be used to generate pseudo-random bit sequences

  ![Diagram](image)

  (initial value random and secret)

  ![Diagram](image)

  (initial value equal to 0)

- Although **ad-hoc techniques have not proven** to be cryptographically secure, they appear sufficient for most applications

Ad-hoc PRBG: ANSI X9.17 generator

X9.17 generator is used to pseudorandomly generate keys and initialization vectors for use with DES

Let $s$ be a 64-bit random seed, $m$ be an integer, $k$ be DES E-D-E encryption key, and $D$ be a 64-bit representation of time/date

1. Let $I = E_k(D)$
2. For $i = 1$ to $m$ do
   1. Let $x_i \leftarrow E_k(I \oplus s)$
   2. Let $s \leftarrow E_k(x_i \oplus s)$
3. Return($x_1, x_2, \ldots, x_m$)
Ad-hoc PRBG: FIPS 186

- FIPS-approved methods for pseudo-randomly generating
  - DSA private key $a$
  - DSA per-message secret $k$

- Both algorithms use a *randomly generated secret seed* $s$ and *one-way function* constructed by using either SHA-1 or DES

CSPRBG

- The security of CSPRBGs relies on the presumed intractability of an underlying number-theoretic problem

- **RSA pseudorandom bit generator** is a CSPRBG under the assumption that **RSA problem** is intractable

- Blum-Blum-Shub pseudorandom bit generator is a CSPRBG under the assumption that **integer factorization** is intractable

- These CSPRBGs make use of modular multiplication which makes them relatively slower than ad-hoc PRBG
RSA CSPRBG

1. Generate two primes $p$ and $q$, and compute $n = pq$ and $\phi = (p - 1)(q - 1)$. Select a random integer $e$, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.

2. Select a random number $x_0$ (the seed) in the interval $[1, n-1]$

3. For $i = 1$ to $l$ do
   1. Let $x_i \leftarrow x_{i-1}^e \mod n$
   2. Let $z_i \leftarrow \text{lsb}(x_i)$

4. Return($z_1, z_2, \ldots, z_l$)

Statistical tests

- A set of statistical tests have been devised to measure the quality of a random bit generator
- While it is not possible to prove whether a generator is indeed a random bit generator, these tests detect certain kinds of weaknesses the generator may have
- Each test takes a sample output sequence and probabilistically determines whether it possesses a certain attribute that a truly random sequence would be likely to exhibit
  - A sequence should roughly have the same number of 1’s as 0’s
- A generator may be rejected or accepted (not rejected)
Statistical tests: basic tests

- **Frequency test** (monobit test). The purpose of this test is to determine whether the number of 0's and 1's are approximately the same.

- **Serial test** (two-bit test). The purpose of this test is to determine whether the number of occurrences of 00, 01, 10, 11 are approximately the same.

- **Poker test**. The purpose of this test is to determine whether the sequences of length \( m \) each appear approximately the same number of times.

- **Runs test**. The purpose of this test is to determine whether the number of runs of various length is as expected for a random sequence.

- **Autocorrelation test**. The purpose of this test is to check correlations between the sequence and shifted versions of it.

---

Statistical tests

- **Statistical tests give only necessary conditions** for a periodic pseudorandom sequence to look random:
  - *Linear congruential pseudorandom generator*
    \[ x_n = a \ x_{n-1} + b \mod n \], \( n \geq 1 \)
  
  passes statistical tests
  
  However, it is **predictable** and hence entirely **insecure** for cryptographic purposes.

- **FIPS 140-1** specifies statistical tests for randomness.