Random numbers

- Random numbers are important
  - key generation for PKS
    - primality testing
  - key generation for symmetric ciphers
  - nonces (one–time values)
- Randomness makes guessing impossible

Requirements on a sequence of random numbers

- Randomness (statistical)
  1. Uniform distribution: relative frequency curve flat
  2. Independence: no single value can be inferred from others in the sequence
- Unpredictability (practical)
  - future elements not predictable from earlier
  - even though sequence is generated by deterministic algorithm

Sources of randomness

- True randomness
  - physical noise generators
    - radiation event detectors etc
    - impractical, slow, low precision
- Tables of statistically random numbers
  - limited in size
  - predictable
- Algorithms
  - deterministic: not statistically random
  - pseudo–randomness suffices (if good enough)

Requirements on random number generation function

- Should generate full period [0,m] before repeating the sequence
- Should pass reasonable tests on statistical randomness
- Should be efficiently implemented

Linear Congruences

- Lehmer, 1951:
  $x_{n+1} = (a\, x_n + c) \bmod m$, given x_0, a, c and m
- Examples:
  - $a=c=1$ gives +1 mod m
  - $a=7$, $c=0$, $m=32$, $x_0=1$ gives $\{7,17,23,1\}$
- If m prime, $c = 0$, some a pass all three tests
  - Ex: $m=2^{31}-1$, $a=7^5$ widely used for statistics

Linear congruences (cont)

- Linear congruences are fast, simple, pass requirements
- Linear congruences are predictable
  - given the parameters *a, c, m*, a single *x* makes the rest predictable
  - given part of the sequence, parameters can be found
  - Ex: given x_n, x_{n+1}, x_{n+2}, x_{n+3}
    $x_{n+1} = (a\, x_n + c) \bmod m$
    $x_{n+2} = (a\, x_{n+1} + c) \bmod m$
    $x_{n+3} = (a\, x_{n+2} + c) \bmod m$

Linear Feedback Shift Registers

- Shift register R=(r_n,..., r_1) of bits
  Tap sequence T=(t_n,..., t_n) of bits
- Output: r_1
- Feedback:
  $r'_i = r_{i+1}$ for $i \in [1,n-1]$
  $r'_n = TR = \sum_{i=1}^{n} t_i r_i \bmod 2 = t_1 r_1 \oplus \ldots \oplus t_n r_n$
- So $R' = HR$ mod 2, where H is a $n \times n$ matrix, whose first row is T, and the rest has 1 on the subdiagonal, 0 otherwise

LFSR (cont)

- An n–bit LFSR generates a pseudo–random bit sequence of length 2^n–1 if T causes R to cycle through all non–zero values before repeating
- This happens if the polynomial
  $T(x) = t_n x^n + t_{n-1} x^{n-1} + \dots + t_1 x^1 + 1$
  is primitive
- A primitive polynomial of degree n is an irreducible polynomial that divides $x^{2n-1}+1$ but not x^d+1 for any d that divides 2^n–1

LFSR example

- $T = (1,0,0,1)$
  $H =$
  $$\begin{matrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix}$$
- $T(x) = x^4+x+1$ is primitive: given non−zero R, generates all 15 non−zero values of $\boldsymbol{Z}_{16}$:
  0001, 1000, 1100, 1110, 1111, 0111, 1011, 0101, 1010, 1101, 0110, 0011, 1001, 0100, 0010
- Output stream (rightmost bits):
  100011110101100

LFSR for encryption

- LFSR can be used in Vernam ciphers
  $c_i = m_i \oplus k_i$
- Easily broken: $2n$ pairs of (c,m) sufficient:
  - $m_i \oplus c_i = m_i \oplus (m_i \oplus k_i) = k_i$ for $i \in [1,2n]$
  - Let $X=((k_n,\ldots, k_1),(k_{n+1},\ldots, k_2),\ldots,(k_{2n-1},\ldots,k_n))$
    and $Y=((k_{n+1},\ldots, k_2),(k_{n+2},\ldots, k_3),\ldots,(k_{2n},\ldots,k_{n+1}))$
  - $Y = HX$ mod 2, and since X is always nonsingular,
    $H = YX^{-1}$ mod 2, and T is the first row of H.
  - Inverting X is $O(n^3)$: 1 day for n=1000, 1 MIPS

LFSR (cont)

- Combinations of LFSR:
  - Geffe: $z=(a\otimes b)\oplus(-b\otimes c)$
    where a=LFSR(7), b=LFSR(5), c=LFSR(8)
    gives period $(2^7-1)(2^5-1)(2^8-1) > 10^9$
  - Still weak: $p(z=a) = 3/4$, $p(z=c) = 1/4$
  - GSM uses "A5" with LFSRs of length 19, 22, 23.
- LFSRs are **fast**!

Cryptographic random number generators

- In cryptography, we want to reduce redundancy and give minimal information about m given c.
- Use this for random number generation!
- Examples:
  - Cyclic encryption: $x_i = E_k(n_i \bmod m)$
    where $n_{i+1} = n_i + 1$
    Since $n_i \neq n_{i+1}$, $x_i \neq x_{i+1}$, and decryption without k is hard,
    so the sequence is (computationally) unpredictable!
  - E.g, use DES in OFB mode, use pseudo–random generator instead of counter

ANSI X9.17 PRNG

- Uses three triple DES encryptions (112−bit key)
  - two "random" sources: date/time and seed
  - feedback of seed value
  - random value R_i does not reveal seed V_{i+1}

Blum Blum Shub

- p, q large primes s.t. $p \equiv q \equiv 3 \pmod 4$
  $n = pq$
  s random s.t. $\gcd(n,s)=1$
- Output: bit sequence B_i
- $x_0 = s^2 \bmod n$
  for (i = 1; i>0; i++) {
  $\quad x_i = (x_{i-1})^2 \bmod n$;
  $\quad B_i = x_i \bmod 2$;
  }

BBS is a CSPRBG

- The BBS is a cryptographically secure pseudo–random bit generator (CSPRBG): it passes the *next–bit* test:
  - Given the first k bits, there is no polynomial algorithm to predict the next bit with probability > ½
- Security based on factorization of n.