Chapter 8

Random Bit Generation and Stream Ciphers

Random numbers

- Numbers that are generated independently and uniformly
- Random numbers (bits) are used for
  - Key distribution
  - Reciprocal authentication schemes
  - Session key generation
  - Key generation for cryptosystems
  - Bit stream for stream ciphers
  - …

Independence and uniformity

- Bit sequence: $b_1, b_2, \ldots$
- Independence
  - B_k is the random variable for random bit $b_k, k \geq 1$
  - B_i is independent of other random bits if

$$\Pr[B_i = b] = \Pr[b_i = b | B_1 \ldots B_{i-1} B_{i+1} \ldots = b_1 \ldots b_{i-1} b_{i+1} \ldots]$$

for any $b, b_j \in \{0,1\}, j \geq 1, j \neq i$

- Uniformity
  - any segment $b_i b_{i+1} \ldots b_{i+k-1}$ is uniform over $\{0,1\}^k$ for any $k \geq 1$

Unpredictability

- Bit sequence: $b_1, b_2, \ldots$
- Unpredictability
  - Forward unpredictability
    - Given $b_1, b_2, \ldots, b_{i-1}$, the probability of predicting b_i is ½ despite of any knowledge of the way of sequence generation
  - Backward unpredictability
    - Given$b_{i+1}, b_{i+2}, \ldots, b_n$, the probability of predicting b_i is ½ despite of any knowledge of the way of sequence generation

True random number sequence

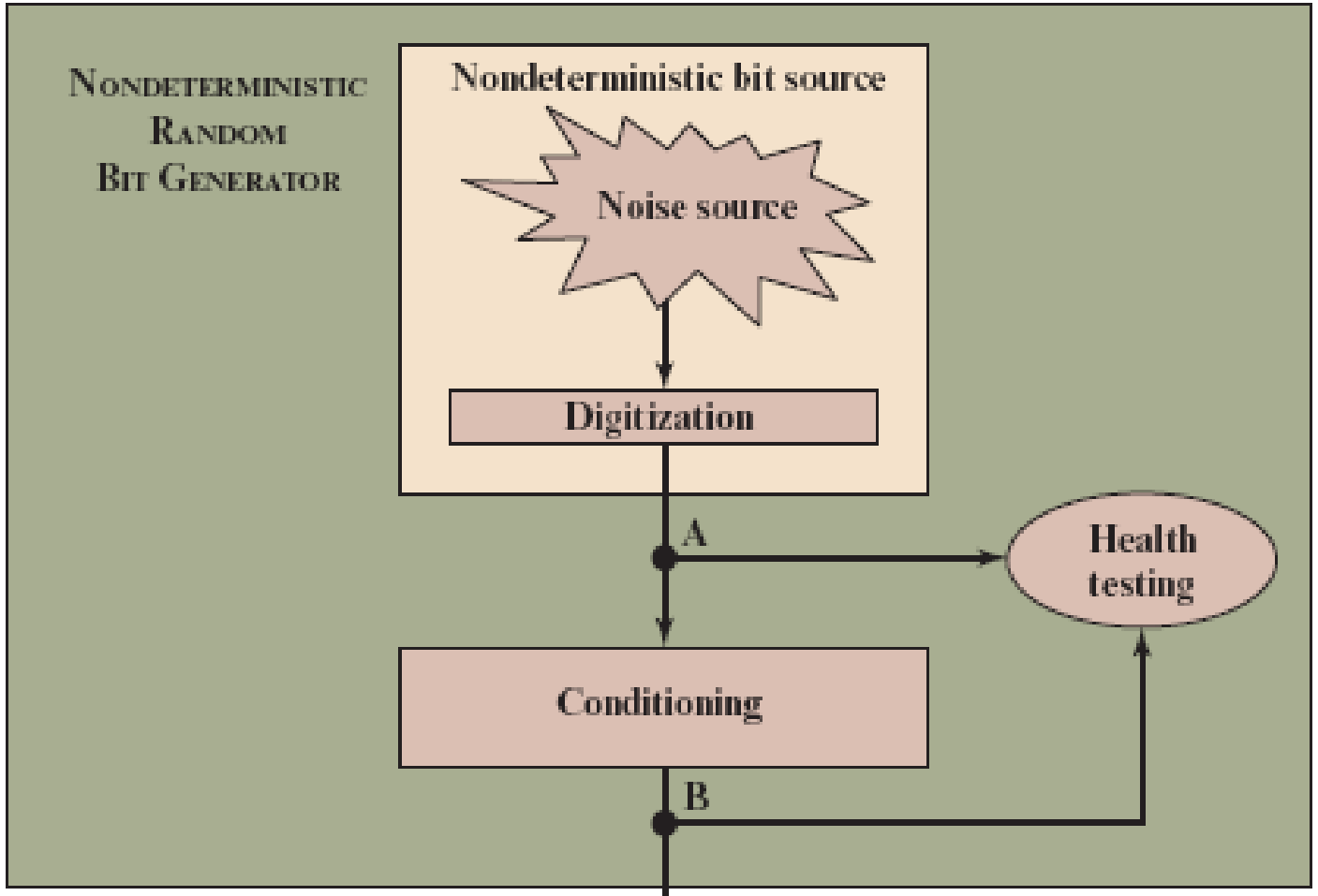
- Bit sequence: $b_1, b_2, \ldots$
- It satisfies the property of unpredictability
- Disadvantages of use
  - Inefficient to generate
  - Hard to store since they cannot be compressed

Entropy source

- A physical source of information whose output signals are noisy and unpredictable
  - keystroke timing pattern
  - mouse movement
  - instantaneous values of system parameters
  - disk electrical activity
  - twinkling of stars
  - …
- Raw entropy source problems
  - bias
  - low entropy

True random number generator

- TRNG converts noisy signals from entropy source to a true random number sequence
  - eliminate bias and enhance entropy of entropy source
  - the resulting number sequence passes $all\ randomness\ tests$
- Conditioning (de-skewing) algorithm
  - remove bias and enhance entropy of a sequence with uncertainty (entropy source)


Nondeterministic Random Bit Generator
Nondeterministic bit source
Noise source
Digitization
A
Health testing
Conditioning
B
Output

Conditioning algorithm: hashing

- A hash function produces an n-bit output from an input of arbitrary length
  - will be introduced later
- Conditioning by hash function
  - input: m-bit block
  - output: n-bit block, $n < m$

Pseudorandom number sequence

- Bit sequence: $b_1, b_2, \ldots$
- Do not satisfy independence, uniformity and unpredictability
- Satisfy the property of "indistinguishability"
  - Computers within reasonable time cannot distinguish it from truly random sequences
- Pass $all\ statistical$ randomness tests
- It is as effective as true random number sequence in practical use

Pseudorandom number generator

- A $deterministic$ algorithm with input of a true random seed and output of a bit sequence
  - output is much longer than seed
  - output is a pseudorandom bit sequence
- Easier to achieve since seed is short, compared to the length of generated pseudorandom sequence
- Security
  - Do not satisfy unpredictability since knowledge of the seed results in predicting the whole sequence
  - Do not satisfy uniformity and independence due to the way of generation

- Toy example
  - $G(b_1 b_2 b_3)$
    $= (b_1 \oplus b_3)(b_2 \oplus b_3)(b_1 \oplus b_2 \oplus \bar{b}_3)(\bar{b}_1 \oplus b_2)(\bar{b}_1 \oplus \bar{b}_2 \oplus \bar{b}_3)$

seed s	G(s)
000	00111
001	11010
010	01000
011	10101
100	10000
101	01101
110	11111
111	00010

Statistical randomness tests

- NIST SP 800-22 lists 15 statistical randomness tests
  - Frequency (Monobits) Test, Frequency Test within a Block, Runs Test, Test for the Longest Run of Ones in a Block, Binary Matrix Rank Test, Discrete Fourier Transform (Specral) Test, Non-Overlapping Template Matching Test, Overlapping Template Matching Test, Maurer's "Universal Statistical" Test, Linear Complexity Test, Serial Test, Approximate Entropy Test, Cumulative Sums (Cusum) Test, Random Excursions Test, Random Excursions Variant Test
- NIST has a software suite for these tests
  - Search the Internet

PRNG: linear congruential generator

- Parameters
  - $m > 0$: the modulus
  - a: the multiplier, $0 < a < m$
  - c: the increment, $0 \leq c < m$
  - X_0: the starting value (seed), $0 \leq X_0 < m$
- Generator

$$X_{n+1} = (aX_n + c) \bmod m$$

- Choices of a, c, m are critical for LCG

- Examples: choices of a, c and m
  - $a = 7,\ c = 0,\ m = 32, X_0 = 1 \rightarrow\ 7, 17, 23, 1, 7, 17, \ldots$
    - Period=4 is short, compared with m=32
  - $a = 5, c = 0, m = 32, \mathrm{X}_0 = 1 \rightarrow\ 5, 25, 29, 17, 21, 9, 13, 1, 5\ , \ldots$
- m is better prime
  - $m = 2^{32} - 1, a = 7^5 = 16807 \rightarrow\ \text{period} = m - 1$
- Typical practice
  - fix a and m
  - each run chooses different X_0
- Vulnerable: from $X_i, X_{i+1}, X_{i+2}, X_{i+3}$, solve (a, c, m) from
  - $X_{i+1} = (aX_i + c) \bmod m$
  - $X_{i+2} = (aX_{i+1} + c) \bmod m$
  - $X_{i+3} = (aX_{i+2} + c) \bmod m$

PRNG: BBS generator

- Parameters
  - $n = pq$: product of two large random primes p and q
  - X_0: the seed
- Generator

$$X_i = X_{i-1}^2 \bmod n,$$
$$B_i = last\text{–}bit\ (X_i)$$

- Also called cryptographically secure pseudorandom bit generator (CSPRBG)
- Pass the next-bit test under hardness assumption of factorization

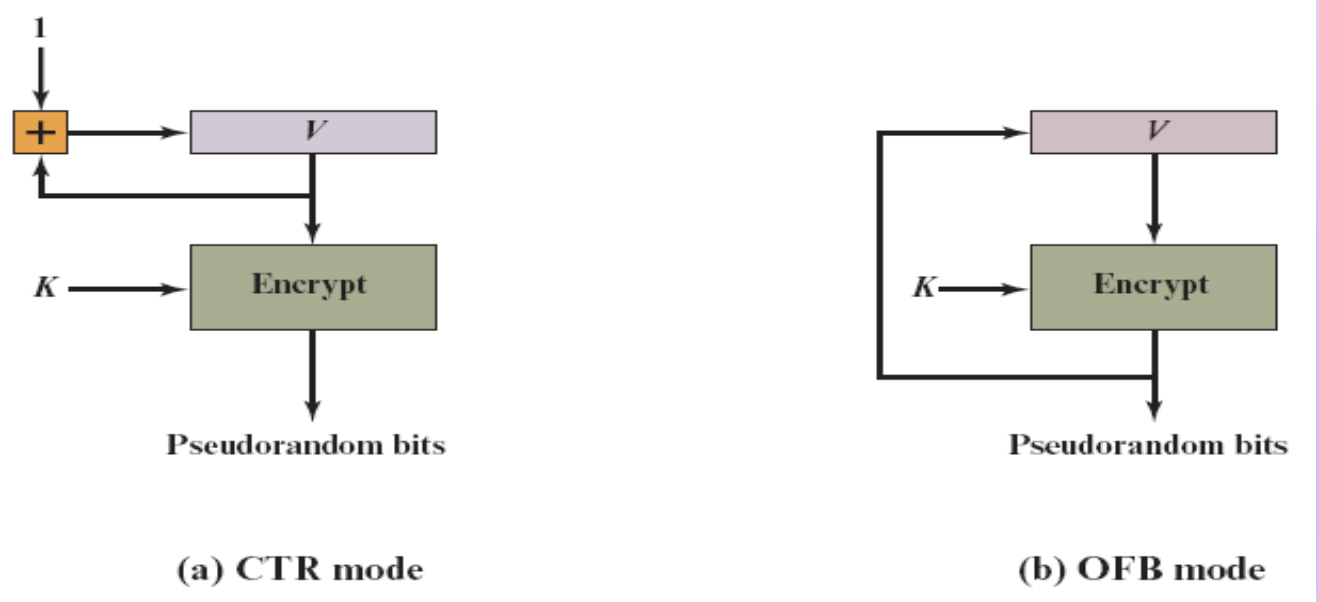
- n=192649=383x503

i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0

i	X_i	B_i
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

PRNG: block cipher modes

- Two block ciphers are used for PRNG
  - CTR mode: recommended in NIST SP 800-90, ANSI standard X.82, and RFC 4086
  - OFB mode: recommended in X9.82 and RFC 4086



- PRNG by block cipher with OFB mode
  - Key: cfb0 ef31 08d4 9cc4 562d 5810 b0a9 af60
  - V: 4c89 af49 6176 b728 ed1e 2ea8 ba27 f5a4

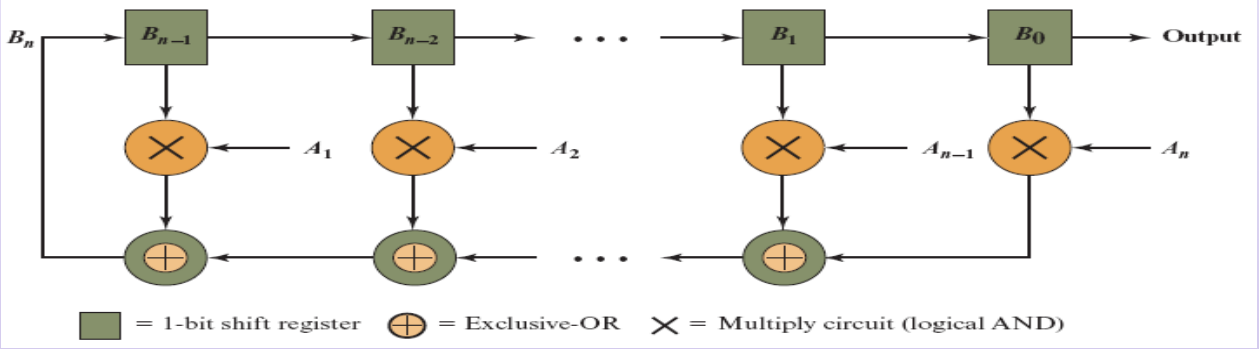
Output block	Fraction of bits 1	Fraction of bits that match with preceding block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e230195429 62e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

- PRNG by block cipher with CTR mode

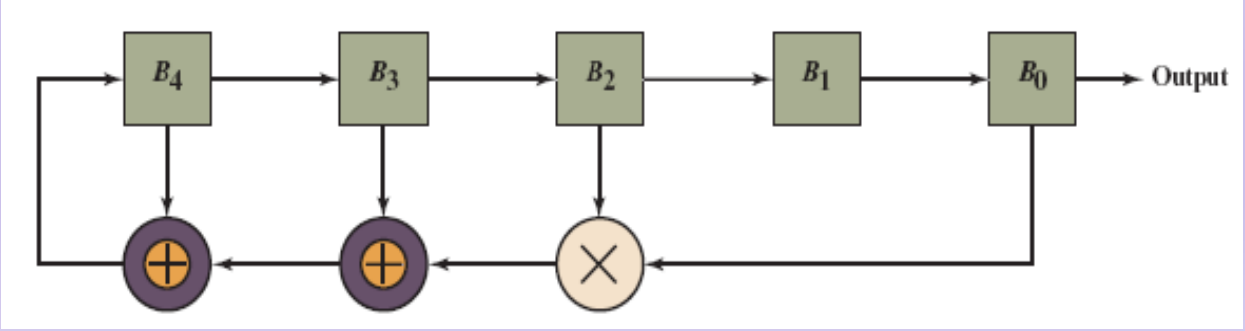
Output block	Fraction of bits 1	Fraction of bits that match with preceding block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
f4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eebac658a3911515c3c66	0.47	0.45

PRNG: feedback shift register

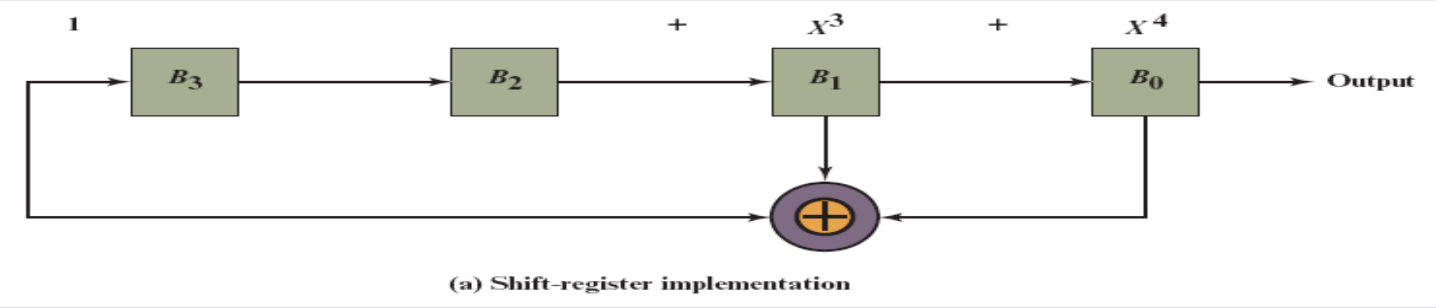
- Hardware-based
- Linear feedback shift register



- Non-linear feed shift register



LFSR example



(a) Shift-register implementation

State	B_3	B_2	B_1	B_0	B_0 Å B_1	output
Initial = 0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	1	0
3	1	0	0	1	1	1
4	1	1	0	0	0	0
5	0	1	1	0	1	0
6	1	0	1	1	0	1
7	0	1	0	1	1	1
8	1	0	1	0	1	0
9	1	1	0	1	1	1
10	1	1	1	0	1	0
11	1	1	1	1	0	1
12	0	1	1	1	0	1
13	0	0	1	1	0	1
14	0	0	0	1	1	1
15 = 0	1	0	0	0	0	0

(b) Example with initial state of 1000

PRNG for stream cipher