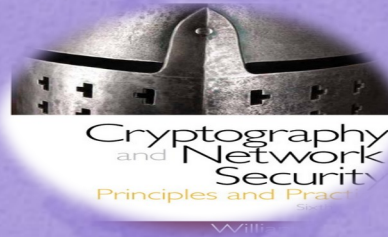


Cryptography and Network Security

Eighth Edition
by William Stallings

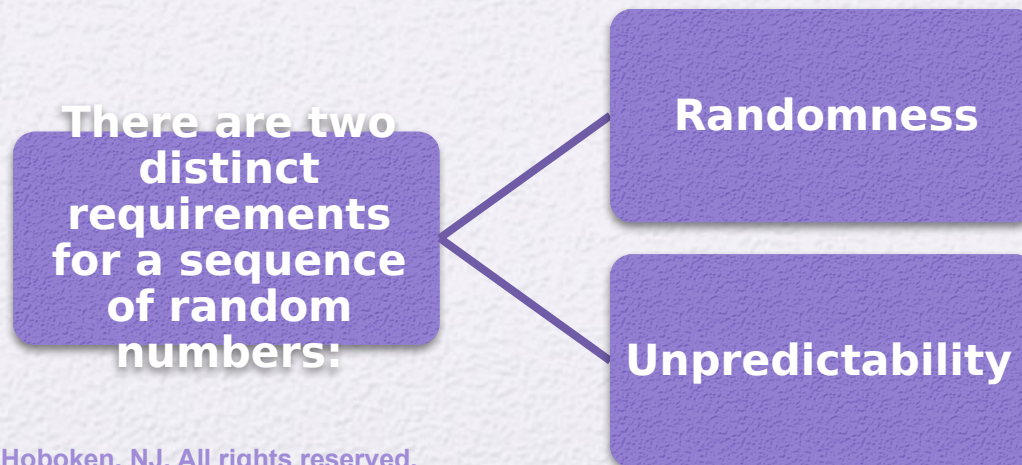


Chapter 8

Random Bit Generation and Stream Ciphers

Random Numbers

- A number of network security algorithms and protocols based on cryptography make use of random binary numbers:
 - Key distribution and reciprocal authentication schemes
 - Session key generation
 - Generation of keys for the RSA public-key encryption algorithm
 - Generation of a bit stream for symmetric stream encryption



Randomness

- The generation of a sequence of allegedly random numbers being random in some well-defined statistical sense has been a concern

Two criteria are used to validate that a sequence of numbers is random:

Uniform distribution

- The frequency of occurrence of ones and zeros should be approximately equal

Independence

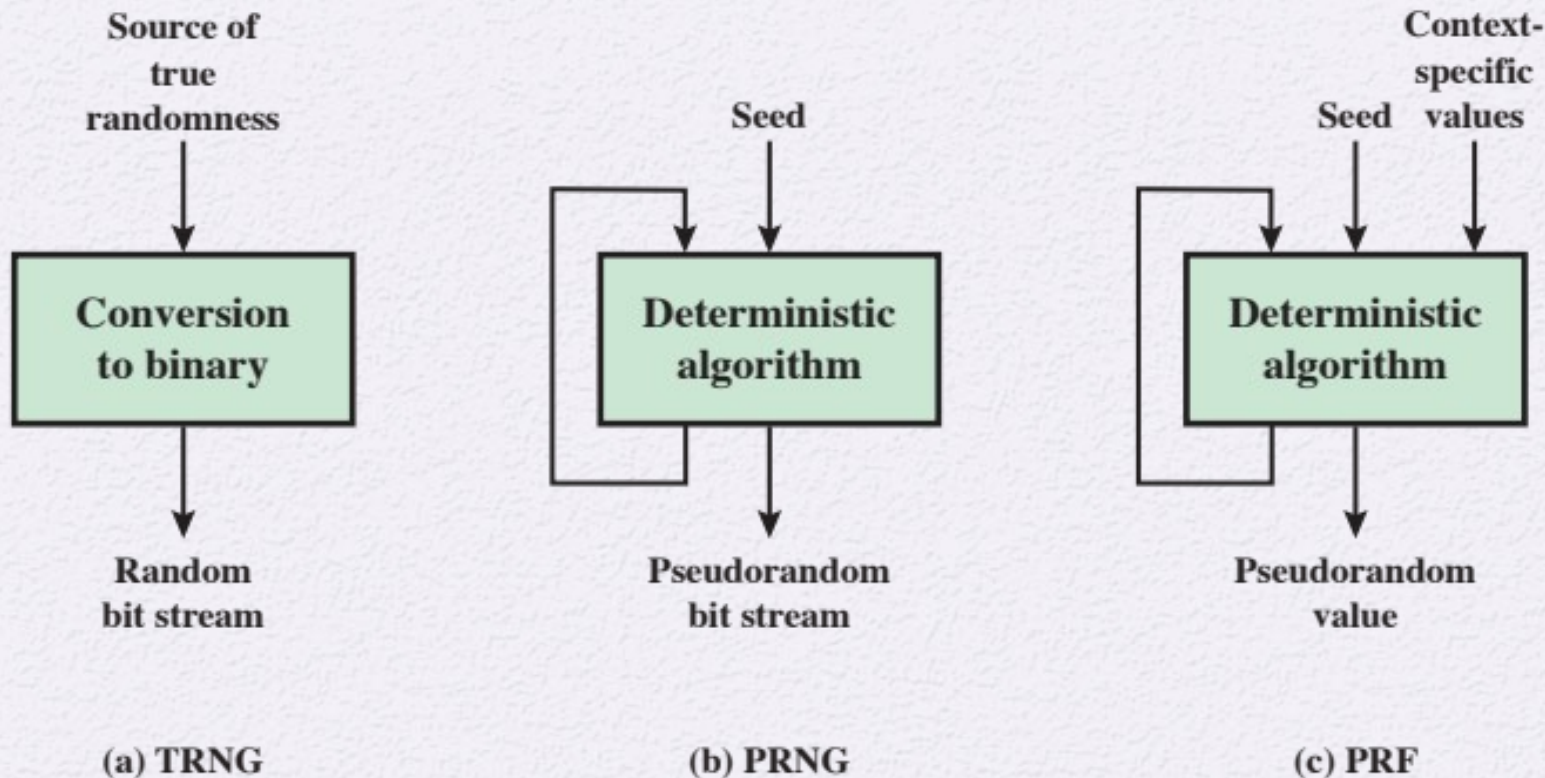
- No one subsequence in the sequence can be inferred from the others

Unpredictability

- The requirement is not just that the sequence of numbers be statistically random, but that the successive members of the sequence are unpredictable
- With “true” random sequences each number is statistically independent of other numbers in the sequence and therefore unpredictable
 - True random numbers have their limitations, such as inefficiency, so it is more common to implement algorithms that generate sequences of numbers that appear to be random
 - Care must be taken that an opponent not be able to predict future elements of the sequence on the basis of earlier elements

Pseudorandom Numbers

- Cryptographic applications typically make use of algorithmic techniques for random number generation
- These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random
- If the algorithm is good, the resulting sequences will pass many tests of randomness and are referred to as pseudorandom numbers



TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

Figure 8.1 Random and Pseudorandom Number Generators

True Random Number Generator (TRNG)

- Takes as input a source that is effectively random
- The source is referred to as an *entropy source* and is drawn from the physical environment of the computer
 - Includes things such as keystroke timing patterns, disk electrical activity, mouse movements, and instantaneous values of the system clock
 - The source, or combination of sources, serve as input to an algorithm that produces random binary output
- The TRNG may simply involve conversion of an analog source to a binary output
- The TRNG may involve additional processing to overcome any bias in the source

Pseudorandom Number Generator (PRNG)

- Takes as input a fixed value, called the *seed*, and produces a sequence of output bits using a deterministic algorithm
 - Quite often the seed is generated by a TRNG
- The output bit stream is determined solely by the input value or values, so an adversary who knows the algorithm and the seed can reproduce the entire bit stream
- Other than the number of bits produced there is no difference between a PRNG and a PRF

Two different forms of PRNG

Pseudorandom number generator

- An algorithm that is used to produce an open-ended sequence of bits
- Input to a symmetric stream cipher is a common application for an open-ended sequence of bits

Pseudorandom function (PRF)

- Used to produce a pseudorandom string of bits of some fixed length
- Examples are symmetric encryption keys and nonces

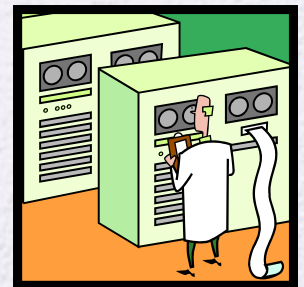
PRNG Requirements

- The basic requirement when a PRNG or PRF is used for a cryptographic application is that an adversary who does not know the seed is unable to determine the pseudorandom string
- The requirement for secrecy of the output of a PRNG or PRF leads to specific requirements in the areas of:
 - Randomness
 - Unpredictability
 - Characteristics of the seed



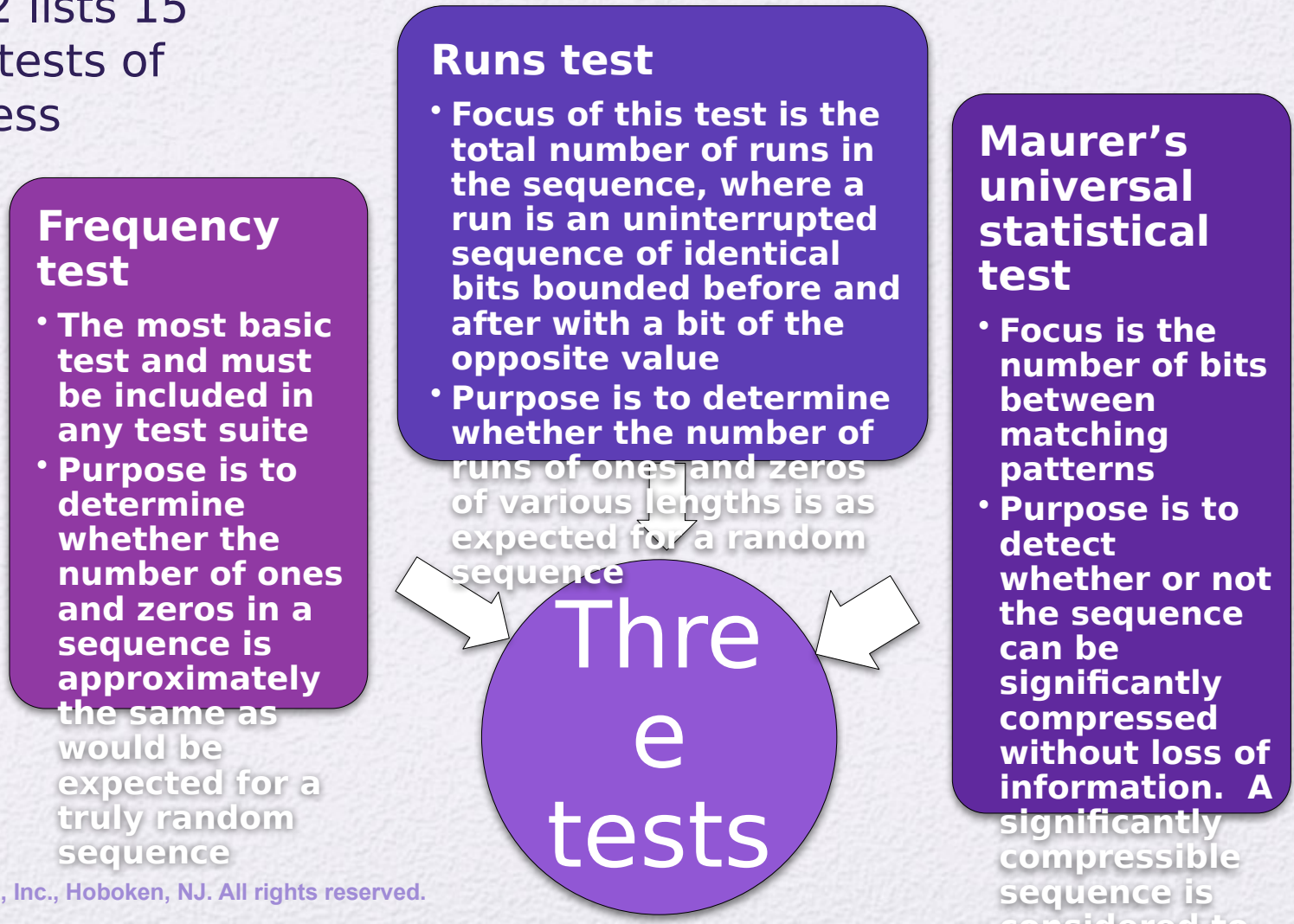
Randomness

- The generated bit stream needs to appear random even though it is deterministic
- There is no single test that can determine if a PRNG generates numbers that have the characteristic of randomness
 - If the PRNG exhibits randomness on the basis of multiple tests, then it can be assumed to satisfy the randomness requirement
- NIST SP 800-22 specifies that the tests should seek to establish three characteristics:
 - Uniformity
 - Scalability
 - Consistency



Randomness Tests

- SP 800-22 lists 15 separate tests of randomness



Unpredictability

- A stream of pseudorandom numbers should exhibit two forms of unpredictability:
 - Forward unpredictability
 - If the seed is unknown, the next output bit in the sequence should be unpredictable in spite of any knowledge of previous bits in the sequence
 - Backward unpredictability
 - It should not be feasible to determine the seed from knowledge of any generated values
 - No correlation between a seed and any value generated from that seed should be evident
 - Each element of the sequence should appear to be the outcome of an independent random event whose probability is $1/2$
- The same set of tests for randomness also provides a test of unpredictability
 - A random sequence will have no correlation with a fixed value (the seed)

Seed Requirements

- The seed that serves as input to the PRNG must be secure and unpredictable
- The seed itself must be a random or pseudorandom number
- Typically the seed is generated by TRNG



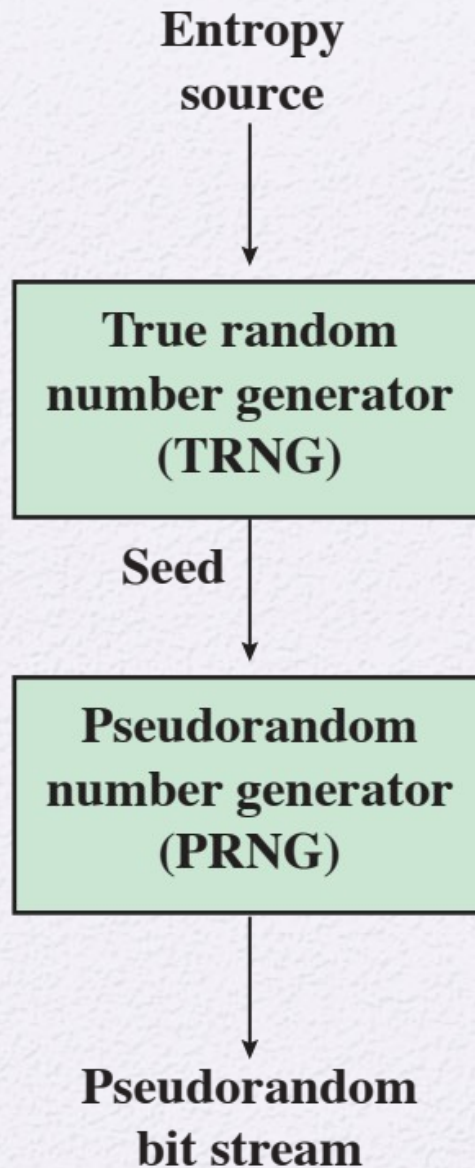


Figure 8.2 Generation of Seed Input to PRNG

Algorithm Design

- Algorithms fall into two categories:
 - Purpose-built algorithms
 - Algorithms designed specifically and solely for the purpose of generating pseudorandom bit streams
 - Algorithms based on existing cryptographic algorithms
 - Have the effect of randomizing input data

Three broad categories of cryptographic algorithms are commonly used to create PRNGs:

- Symmetric block ciphers
- Asymmetric ciphers
- Hash functions and message authentication codes

Stream Cipher Design Considerations

The encryption sequence should have a large period

- A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats; the longer the period of repeat the more difficult it will be to do cryptanalysis

The keystream should approximate the properties of a true random number stream as close as possible

- There should be an approximately equal number of 1s and 0s
- If the keystream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often

A key length of at least 128 bits is desirable

- The output of the pseudorandom number generator is conditioned on the value of the input key
- The same considerations that apply to block ciphers are valid

With a properly designed pseudorandom number generator a stream cipher can be as secure as a block cipher of comparable key length

- A potential advantage is that stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than block ciphers

RC4

- Designed in 1987 by Ron Rivest for RSA Security
- Variable key size stream cipher with byte-oriented operations
- Based on the use of a random permutation
- Eight to sixteen machine operations are required per output byte and the cipher can be expected to run very quickly in software
- RC4 is used in the WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard
- It is optional for use in Secure Shell (SSH) and Kerberos
- RC4 was kept as a trade secret by RSA Security until September 1994 when the RC4 algorithm was anonymously posted on the Internet on the Cypherpunks anonymous remailers list

Strength of RC4

- A fundamental vulnerability was revealed in the RC4 key scheduling algorithm that reduces the amount of effort to discover the key
- Recent cryptanalysis results exploit biases in the RC4 keystream to recover repeatedly encrypted plaintexts
- As a result of the discovered weaknesses the IETF issued RFC 7465 prohibiting the use of RC4 in TLS
- In its latest TLS guidelines, NIST also prohibited the use of RC4 for government use

Stream Ciphers Using Feedback Shift Registers

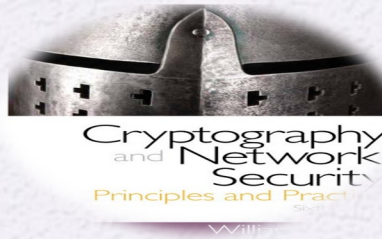
With the increasing use of highly constrained devices there has been increasing interest in developing new stream ciphers that take up minimal memory, are highly efficient, and have minimal power consumption requirements

Most of the recently developed stream ciphers are based on the use of feedback shift registers (FSRs)

- FSRs exhibit the desired performance behavior, are well-suited to compact hardware implementation, and there are well-developed theoretical results on the statistical properties of the bit sequences they produce
- An FSR consists of a sequence of 1-bit memory cells
- Each cell has an output line, which indicates the value currently stored, and an input line
- At discrete time instants, known as clock times, the value in each storage device is replaced by the value indicated by its input line
- The effect is as follows: The rightmost (least significant) bit is shifted out as the output bit for this clock cycle; the other bits are shifted one bit position to the right; the new leftmost (most significant) bit is calculated as a function of the other bits in the FSR

Summary

- Explain the concepts of randomness and unpredictability with respect to random numbers
- Present an overview of requirements for pseudorandom number generators
- Explain the significance of skew
- Present an overview of stream ciphers and RC4



- Understand the differences among true random number generators, pseudorandom number generators, and pseudorandom functions
- Explain how a block cipher can be used to construct a pseudorandom