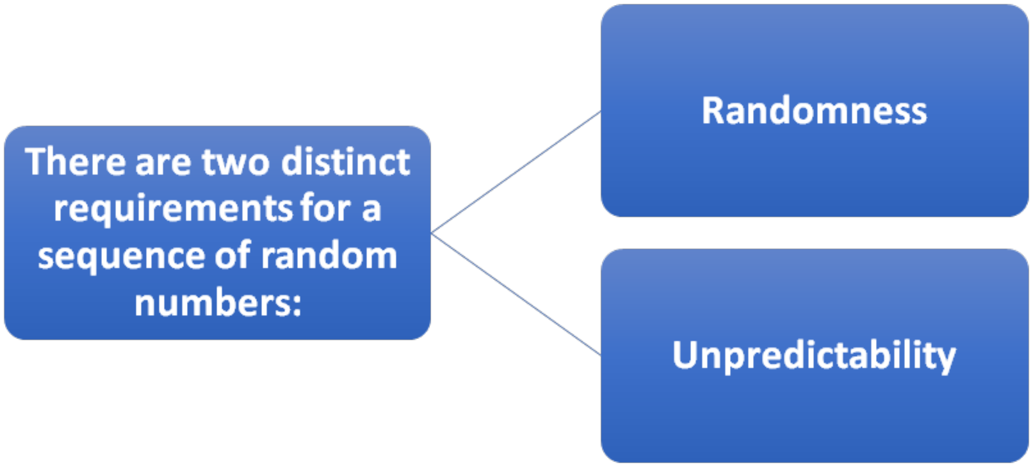
Pseudorandom Number Generation and Stream Ciphers

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*

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

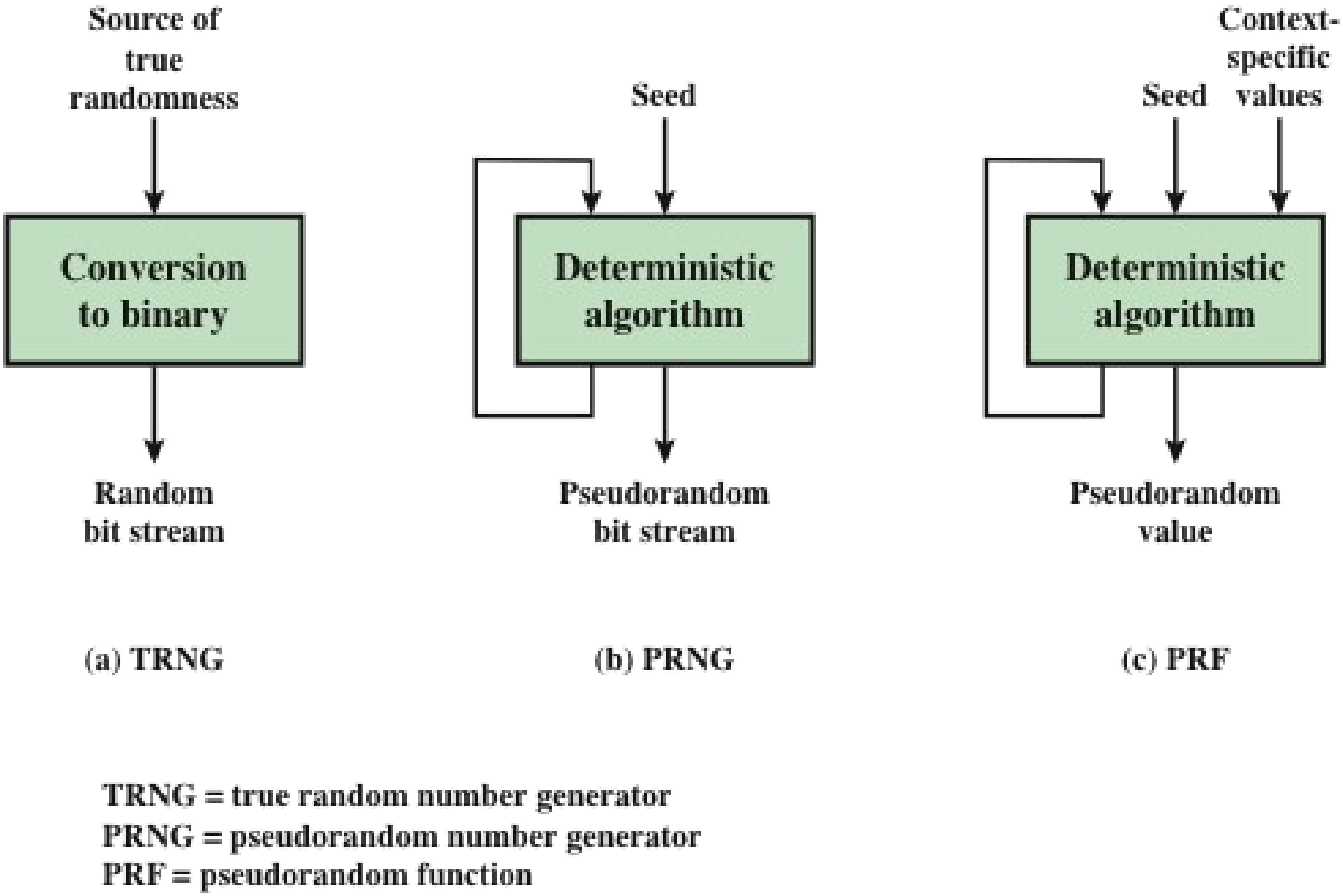


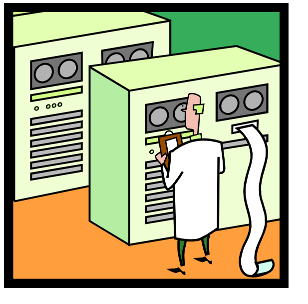
Figure 7.1 Random and Pseudorandom Number Generators

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

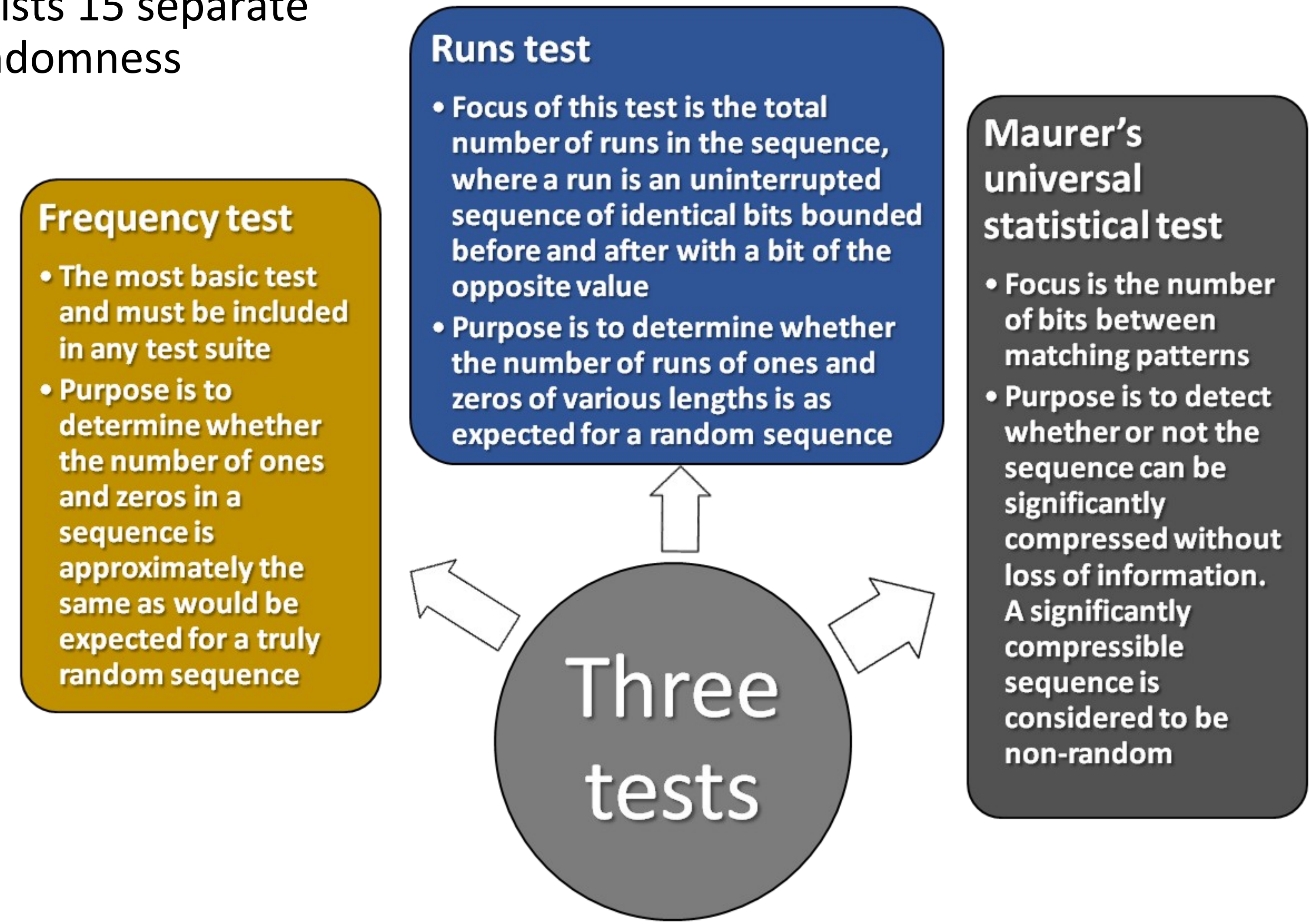
Randomness

- 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

Generation of Seed Input to PRNG

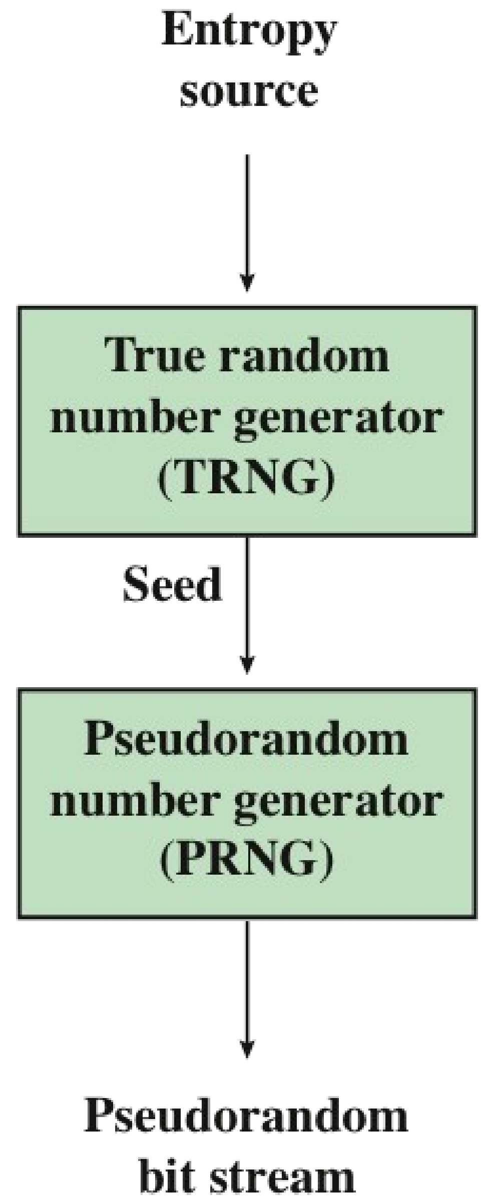


Figure 7.2 Generation of Seed Input to PRNG

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

Linear Congruential Generator Parameters

- ***An algorithm first proposed by Lehmer that is parameterized with four numbers:***

m	**the modulus**	**$m > 0$**
a	the multiplier	$0 < a < m$
c	the increment	$0 \leq c < m$
X_0	the starting value, or **seed**	$0 \leq X_0 < m$

- **The sequence of random numbers $\{X_n\}$ is obtained via the following iterative equation:**

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

- If m , a , c , and X_0 are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq X_n < m$
- ***The selection of values for a , c , and m is critical in developing a good random number generator***

PRNG Using Block Cipher Modes of Operation

- Two approaches that use a block cipher to build a PNRG have gained widespread acceptance:
  - **CTR mode (Counter)**

  Counter mode turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a "counter". The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual increment-by-one counter is the simplest and most popular.

    - Recommended in NIST SP 800-90, ANSI standard X.82, and RFC 4086
  - **OFB mode(Output Feedback)**

  The *Output Feedback* (OFB) mode makes a block cipher into a synchronous stream cipher. It generates keystream blocks, which are then XORed with the plaintext blocks to get the ciphertext.

    - Recommended in X9.82 and RFC 4086

Stream Ciphers

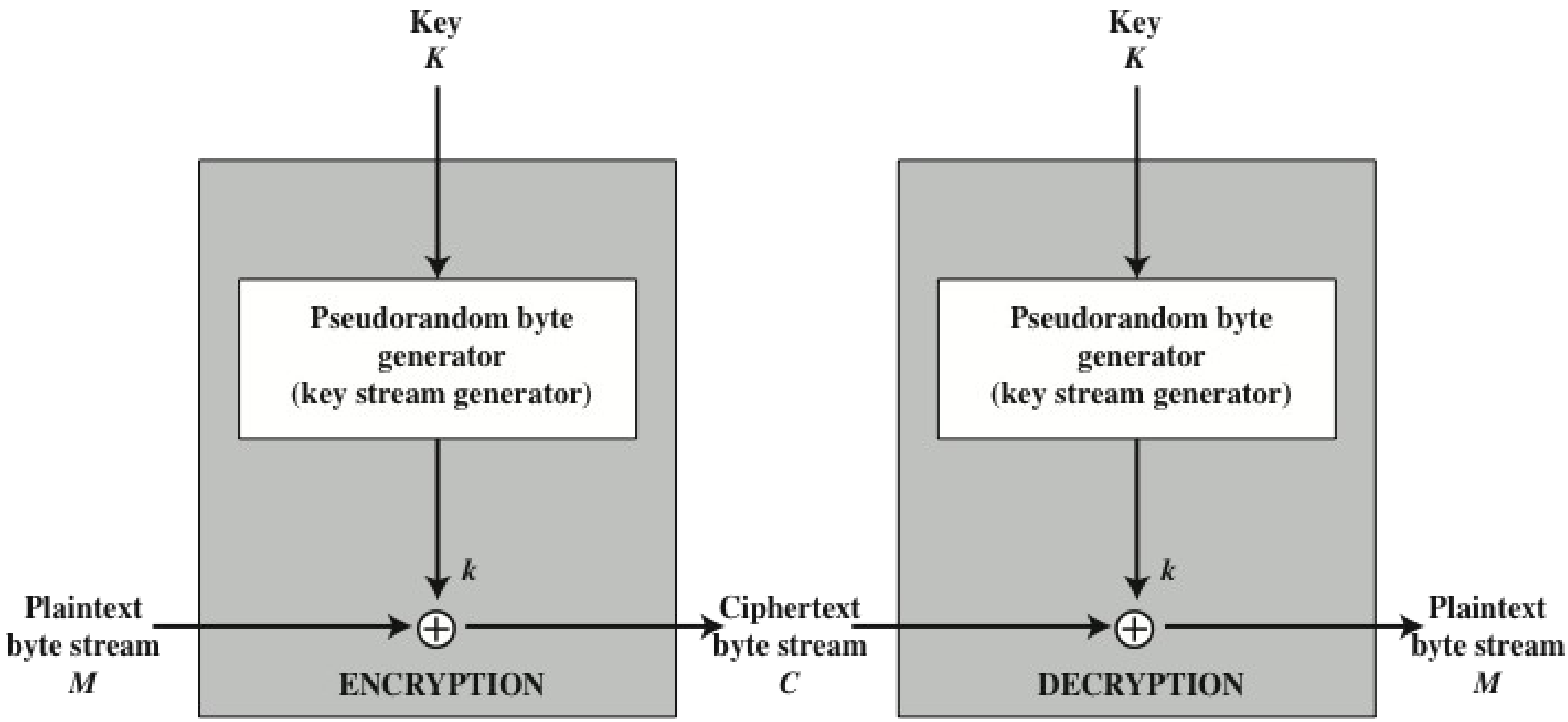


Figure 7.7 Stream Cipher Diagram

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

Entropy Sources

- A true random number generator (TRNG) uses a nondeterministic source to produce randomness
- Most operate by measuring unpredictable natural processes such as pulse detectors of ionizing radiation events, gas discharge tubes, and leaky capacitors
- Intel has developed a commercially available chip that samples thermal noise by amplifying the voltage measured across undriven resistors
- LavaRnd is an open source project for creating truly random numbers using inexpensive cameras, open source code, and inexpensive hardware
  - The system uses a saturated CCD in a light-tight can as a chaotic source to produce the seed; software processes the result into truly random numbers in a variety of formats

Possible Sources of Randomness

RFC 4086 lists the following possible sources of randomness that can be used on a computer to generate true random sequences:

Sound/video input

- The input from a sound digitizer with no source plugged in or from a camera with the lens cap on is essentially thermal noise
- If the system has enough gain to detect anything, such input can provide reasonable high quality random bits

Disk drives

- Have small random fluctuations in their rotational speed due to chaotic air turbulence
- The addition of low-level disk seek-time instrumentation produces a series of measurements that contain this randomness

There is also an online service (random.org) which can deliver random sequences securely over the Internet

Table 7.5

	Pseudorandom Number Generators	True Random Number Generators
Efficiency	Very efficient	Generally inefficient
Determinism	Deterministic	Nondeterministic
Periodicity	Periodic	Aperiodic

Comparison of PRNGs and TRNGs