Chapter 7

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

There are two distinct requirements for a sequence of random numbers:

- Randomness
- Unpredictability

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

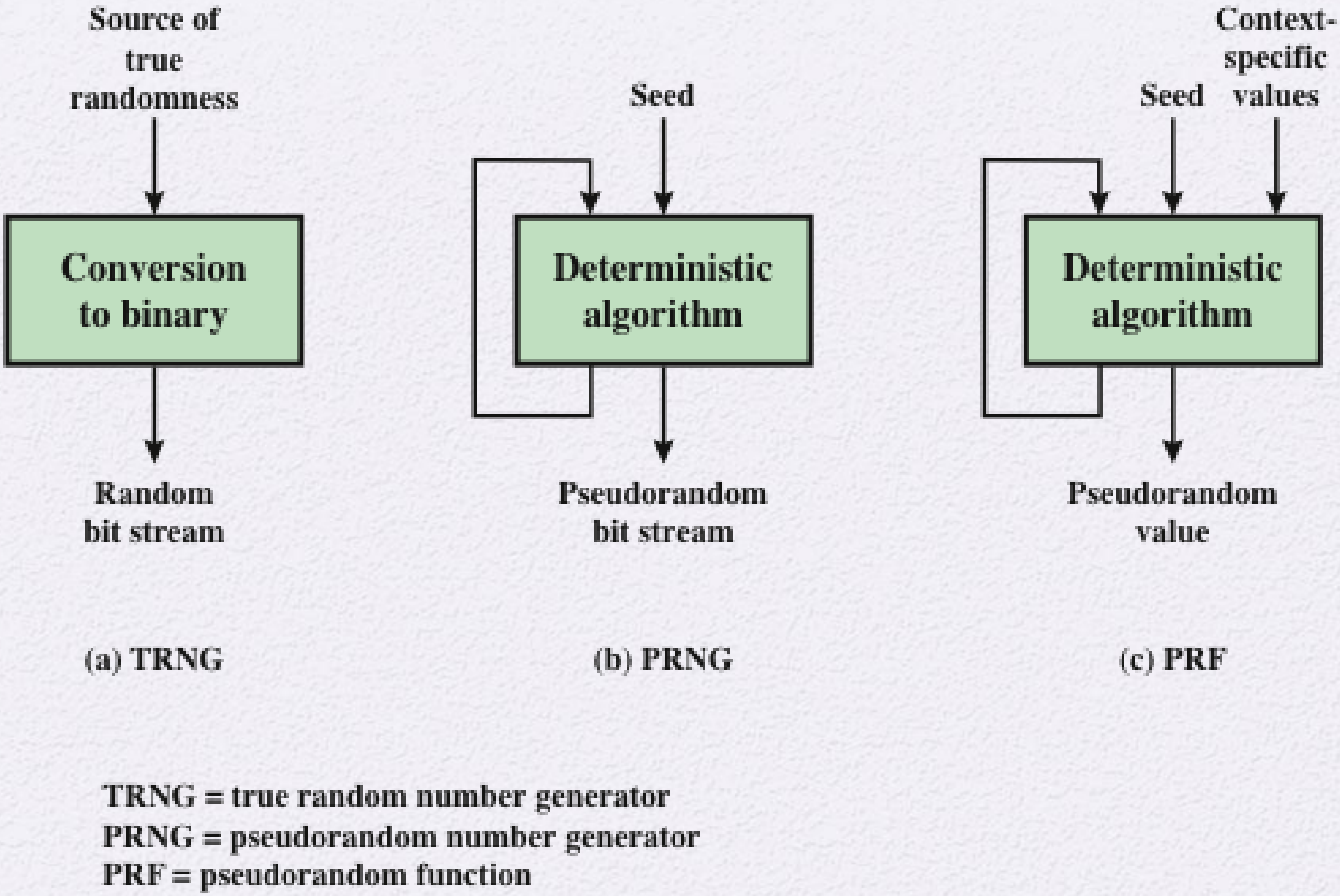


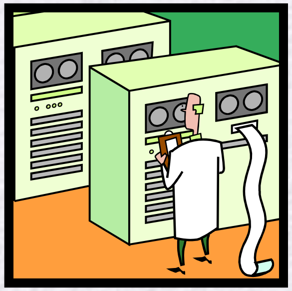
Figure 7.1 Random and Pseudorandom Number Generators

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

Frequency test

- The most basic test and must be included in any test suite
- Purpose is to determine whether the number of ones and zeros in a sequence is approximately the same as would be expected for a truly random sequence

Runs test

- Focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits bounded before and after with a bit of the opposite value
- Purpose is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence

Maurer's universal statistical test

- Focus is the number of bits between matching patterns
- Purpose is to detect whether or not the sequence can be significantly compressed without loss of information.

Three tests

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

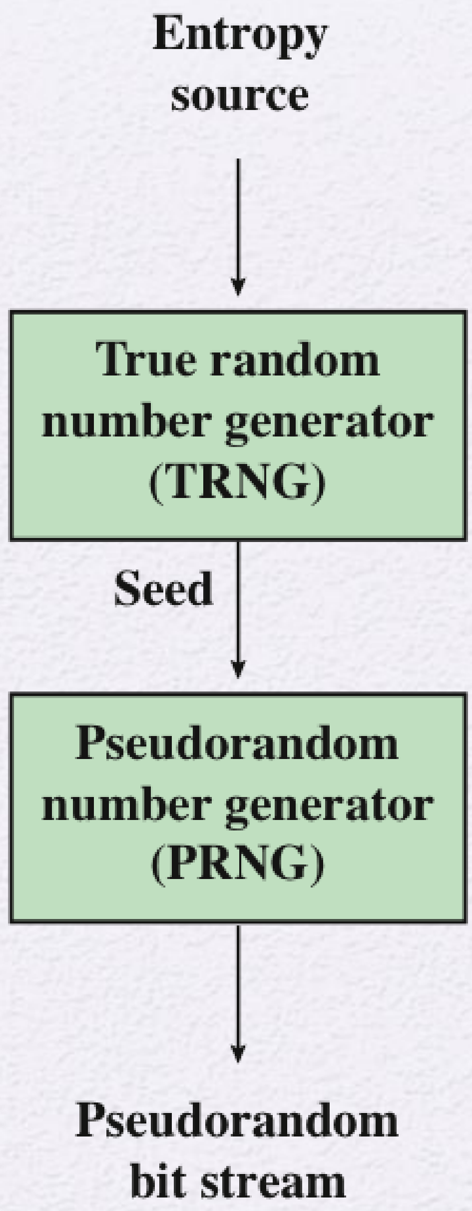


Figure 7.2 Generation of Seed Input to PRNG

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

- 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	$0 \le c < m$
X_0	the starting value, or seed	$0 \le X_0 < m$

- 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 $0 \le X_n < m$
- The selection of values for a , c , and m is critical in developing a good random number generator

Blum Blum Shub (BBS) Generator

- Has perhaps the strongest public proof of its cryptographic strength of any purpose-built algorithm
- Referred to as a *cryptographically secure pseudorandom bit generator* (CSPRBG)
  - A CSPRBG is defined as one that passes the *next-bit-test* if there is not a polynomial-time algorithm that, on input of the first k bits of an output sequence, can predict the $(k + 1)$st bit with probability significantly greater than 1/2
- The security of BBS is based on the difficulty of factoring n

```
X0     = s^2 mod n
for i  = 1 to ∞
   Xi  = (Xi-1)^2 mod n
   Bi  = Xi mod 2
```

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$X_i = (X_{i-1})^2 \bmod n$$
$$B_i = X_i \bmod 2$$

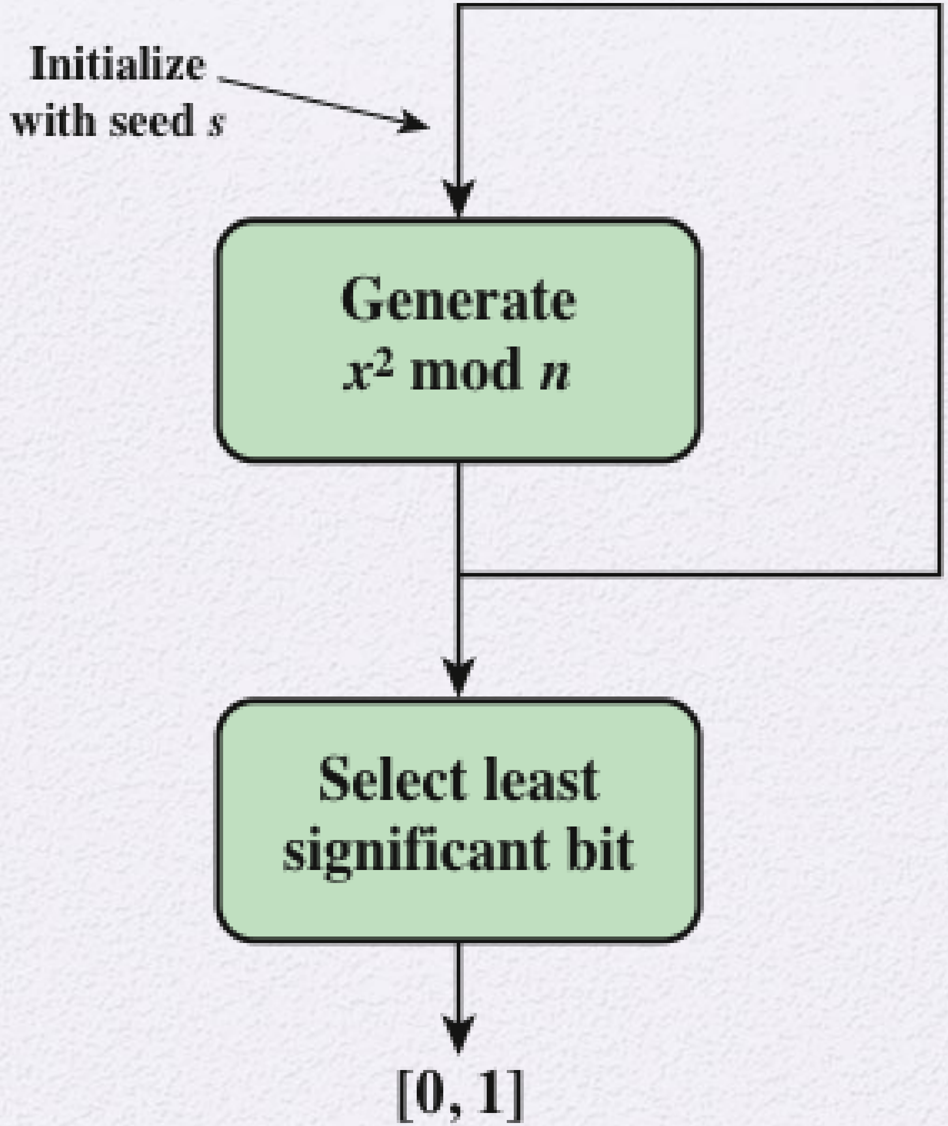


Figure 7.3 Blum Blum Shub Block Diagram

Table 7.1 Example Operation of BBS Generator

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

n = 192649 = 383 * 503, and the seed s = 101355.

PRNG Using Block Cipher Modes of Operation

- Two approaches that use a block cipher to build a PNRG have gained widespread acceptance:
  - CTR mode
    - Recommended in NIST SP 800-90, ANSI standard X.82, and RFC 4086
  - OFB mode
    - Recommended in X9.82 and RFC 4086

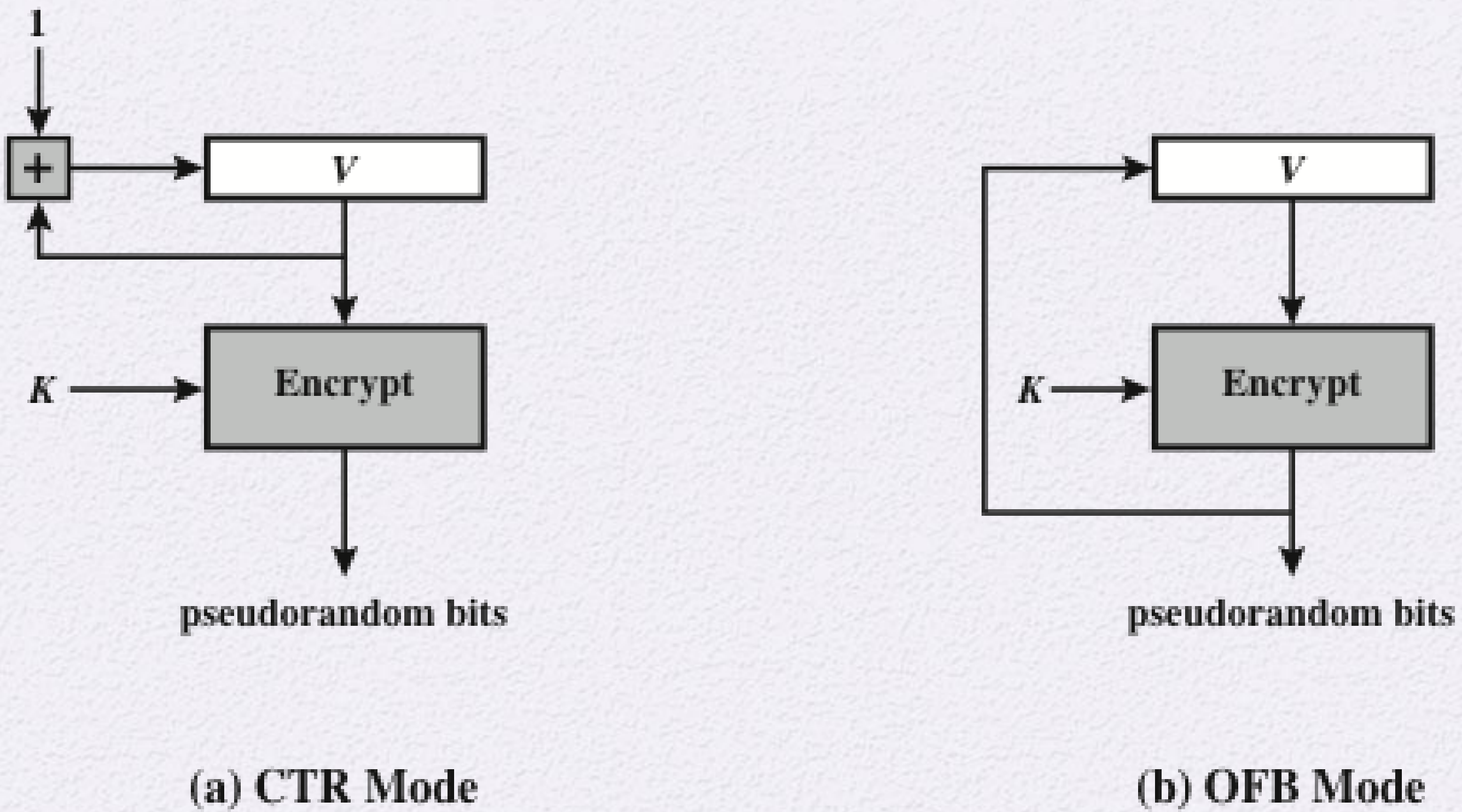


Figure 7.4 PRNG Mechanisms Based on Block Ciphers

Table 7.2

Key:	cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

Output Block	Fraction of One Bits	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
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

Example Results for PRNG Using OFB

Table 7.3

Key:	cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

Output Block	Fraction of One Bits	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

Example Results for PRNG Using CTR

ANSI X9.17 PRNG

- One of the strongest PRNGs is specified in ANSI X9.17
  - A number of applications employ this technique including financial security applications and PGP

Input

- Two pseudorandom inputs drive the generator. One is a 64-bit representation of the current date and time. The other is a 64-bit seed value; this is initialized to some arbitrary value and is updated during the generation process.

The algorithm makes use of triple DES for encryption. Ingredients are:

Keys

- The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56-bit keys, which must be kept secret and are used only for pseudorandom number generation.

Output

- The output consists of a 64-bit pseudorandom number and a 64-bit seed value.

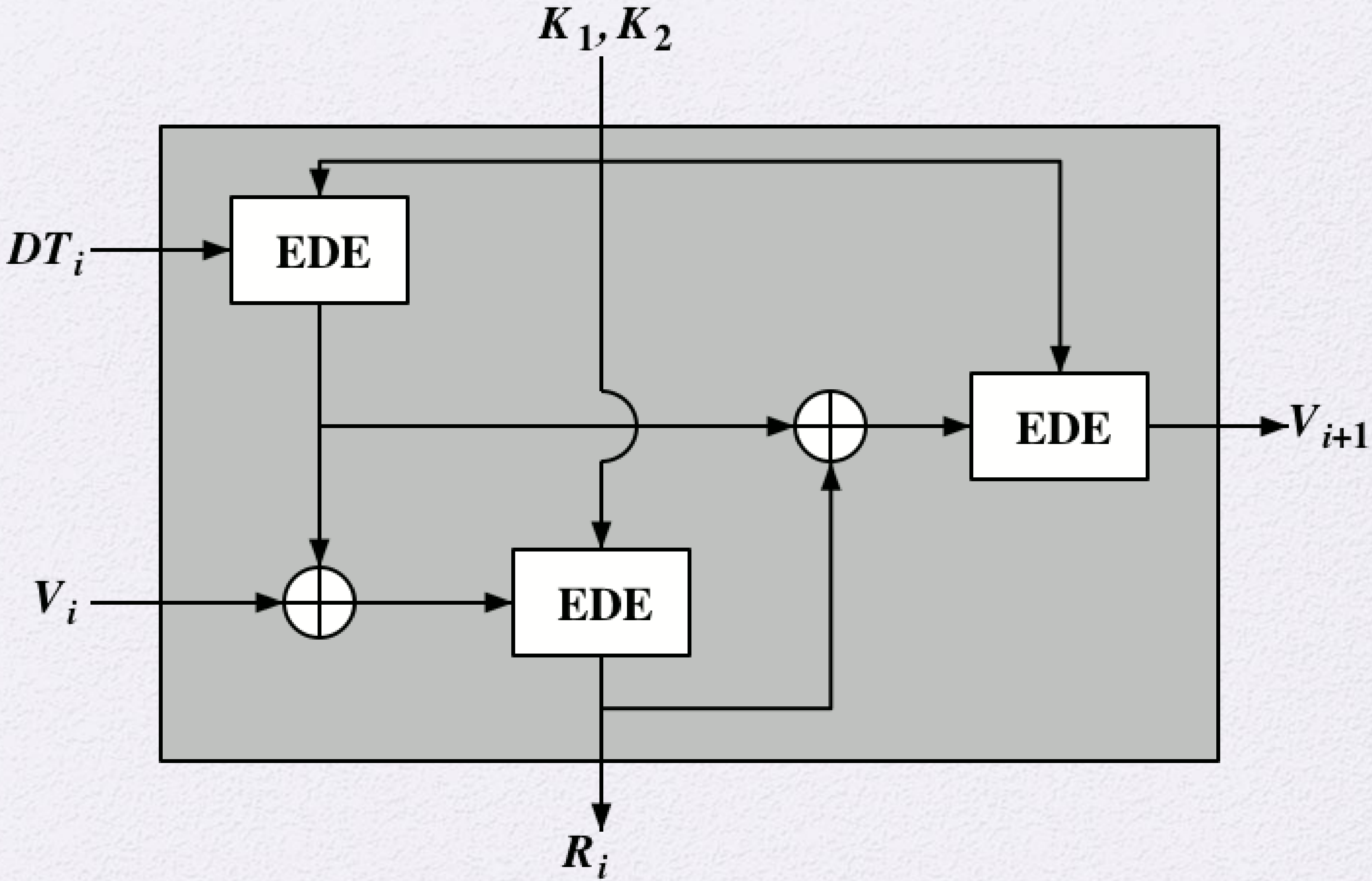


Figure 7.5 ANSI X9.17 Pseudorandom Number Generator

NIST CTR_DRBG

- Counter mode-deterministic random bit generator
- PRNG defined in NIST SP 800-90 based on the CTR mode of operation
- Is widely implemented and is part of the hardware random number generator implemented on all recent Intel processor chips
- DRBG assumes that an entropy source is available to provide random bits
    - Entropy is an information theoretic concept that measures unpredictability or randomness
- The encryption algorithm used in the DRBG may be 3DES with three keys or AES with a key size of 128, 192, or 256 bits

Table 7.4

	3DES	AES-128	AES-192	AES-256
outlen	64	128	128	128
keylen	168	128	192	256
seedlen	232	256	320	384
reseed_interval	$\leq 2^{32}$	$\leq 2^{48}$	$\leq 2^{48}$	$\leq 2^{48}$

CTR_DRBG Parameters

Stream Ciphers

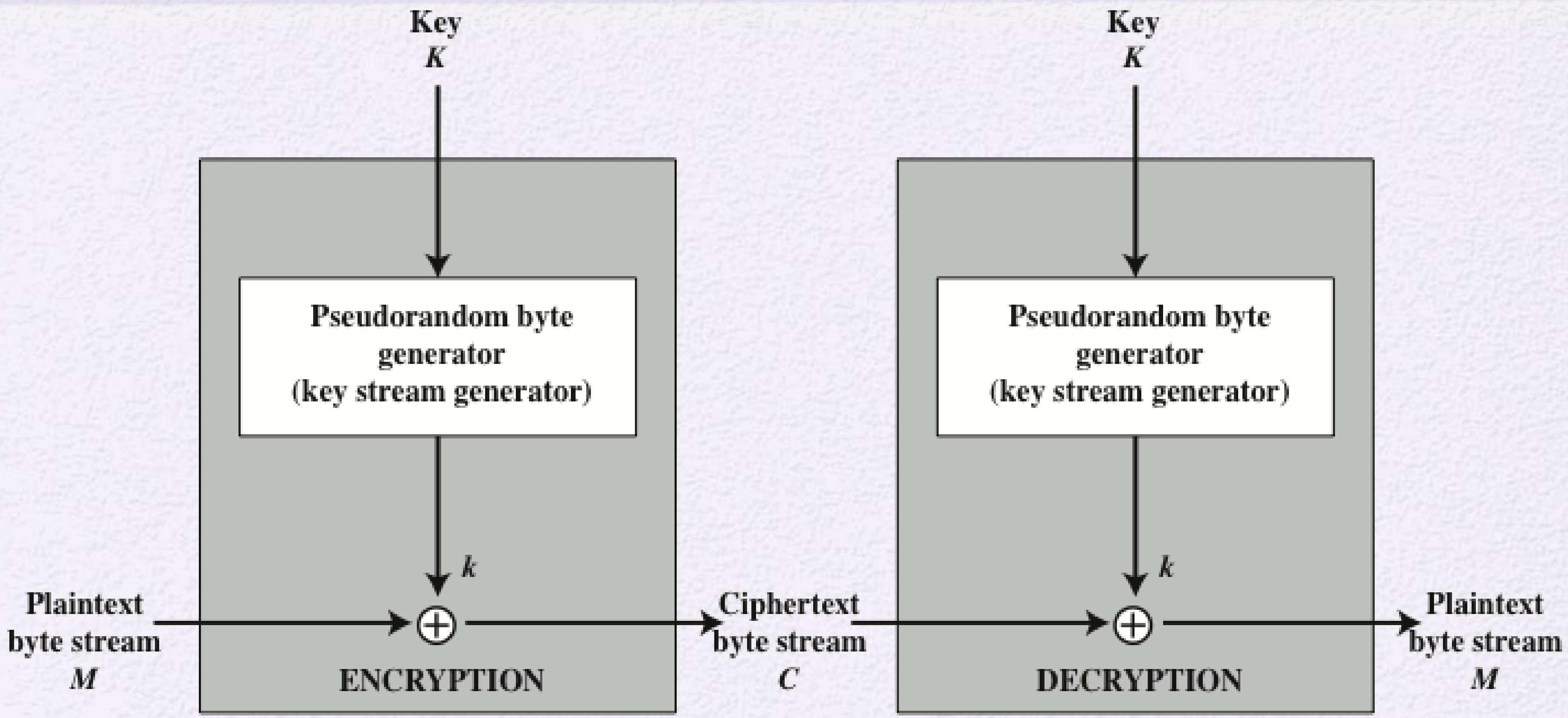


Figure 7.7 Stream Cipher Diagram

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

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
- Used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers
- Is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard

Strength of RC4

A number of papers have been published analyzing methods of attacking RC4

- None of these approaches is practical against RC4 with a reasonable key length

A more serious problem is that the WEP protocol intended to provide confidentiality on 802.11 wireless LAN networks is vulnerable to a particular attack approach

- The problem is not with RC4 itself, but the way in which keys are generated for use as input
- Problem does not appear to be relevant to other applications and can be remedied in WEP by changing the way in which keys are generated
- Problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them

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

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

Skew

- A TRNG may produce an output that is biased in some way, such as having more ones than zeros or vice versa
  - Deskewing algorithms
    - Methods of modifying a bit stream to reduce or eliminate the bias
    - One approach is to pass the bit stream through a hash function such as MD5 or SHA-1
    - RFC 4086 recommends collecting input from multiple hardware sources and then mixing these using a hash function to produce random output
- Operating systems typically provide a built-in mechanism for generating random numbers
  - Linux uses four entropy sources: mouse and keyboard activity, disk I/O operations, and specific interrupts
    - Bits are generated from these four sources and combined in a pooled buffer
    - When random bits are needed the appropriate number of bits are read from the buffer and passed through the SHA-1 hash function

Summary

- Principles of pseudorandom number generation
  - The use of random numbers
  - TRNGs, PRNGs, and PRFs
  - PRNG requirements
  - Algorithm design
- Pseudorandom number generators
  - Linear congruential generators
  - Blum Blum Shub generator
- Pseudorandom number generation using a block cipher
  - PRNG using block cipher modes of operation
  - ANSI X9.17 PRNG
  - NIST CTR_DRBG

- Stream ciphers
- RC4
  - Initialization of S
  - Stream generation
  - Strength of RC4
- True random number generators
  - Entropy sources
  - Comparison of PRNGs and TRNGs
  - Skew
  - Intel digital random number generator
  - DRNG hardware architecture
  - DRNG logical structure