Cryptography and Network Security Chapter 7

Fifth Edition

by William Stallings

Lecture slides by Lawrie Brown

Random Numbers

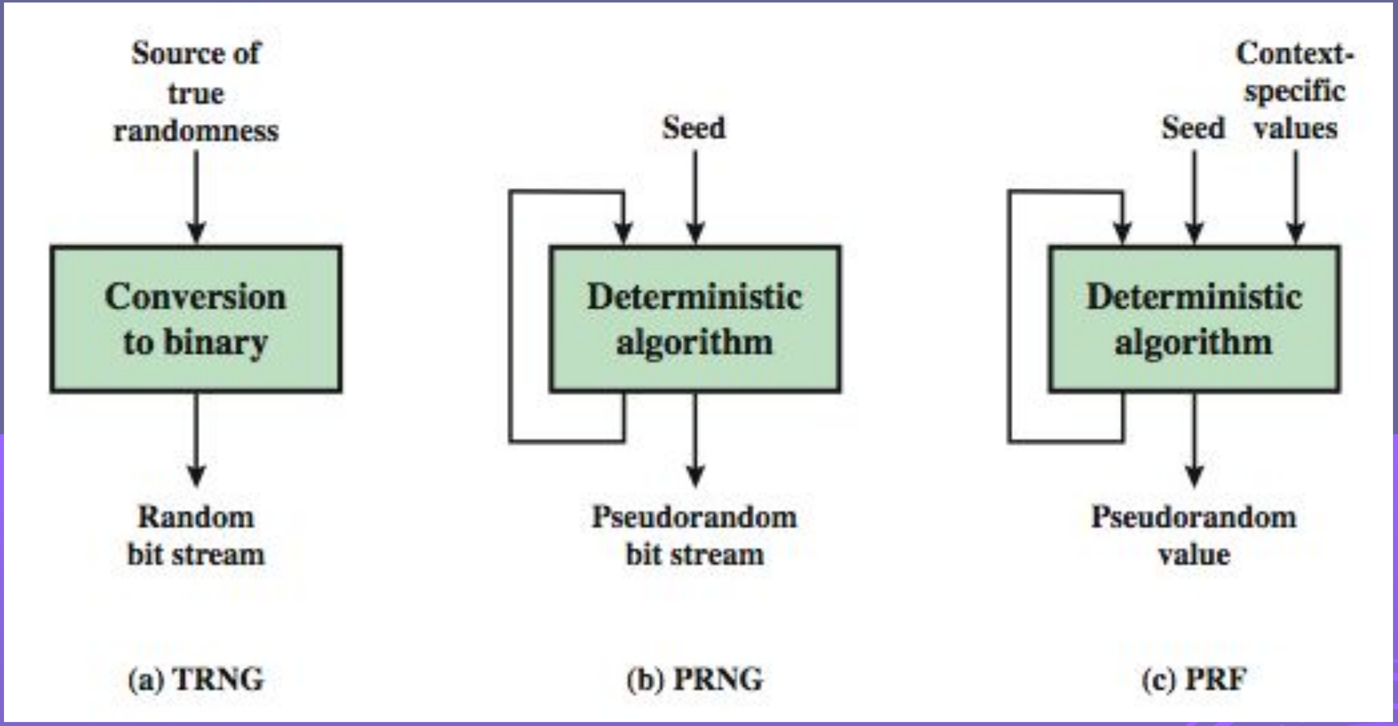
- many uses of **random numbers** in cryptography
  - nonces in authentication protocols to prevent replay
  - session keys
  - public key generation
  - keystream for a one-time pad
- in all cases its critical that these values be
  - statistically random, uniform distribution, independent
  - unpredictability of future values from previous values
- true random numbers provide this
- care needed with generated random numbers

Pseudorandom Number Generators (PRNGs)

- often use deterministic algorithmic techniques to create “random numbers”
  - although are not truly random
  - can pass many tests of “randomness”
- known as “pseudorandom numbers”
- created by “Pseudorandom Number Generators (PRNGs)”

Reference: R. Hegadi and A. P. Patil, "A Statistical Analysis on In-Built Pseudo Random Number Generators Using NIST Test Suite," 2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India, 2020, pp. 1-6, doi: 10.1109/ICCCS49678.2020.9276849.

Random & Pseudorandom Number Generators



PRNG Requirements

- randomness
  - uniformity, scalability, consistency
- unpredictability
  - forward & backward unpredictability
  - use same tests to check
- characteristics of the seed
  - secure
  - if known adversary can determine output
  - so must be random or pseudorandom number

PRNG Requirements

- Randomness
  - Uniformity – at any point in the generation of the PRN sequence, the occurrence of a zero or a one is equally likely (i.e., p = 0.5)
  - Scalability – any test for randomness applicable to a sequence can also be applied to any random subsequence (it should pass)
  - Consistency – the characteristics of the PRN sequence of the PRNG must not depend on the seed used

Linear Congruential RN Generator

- common iterative technique using:

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

- given suitable values of parameters can produce a long random-like sequence
- suitable criteria to have are:
  - function generates a full-period
  - generated sequence should appear random
  - efficient implementation with 32-bit arithmetic
- note that an attacker can reconstruct sequence given a small number of values have possibilities for making this harder

Blum Blum Shub RN Generator

- based on public key algorithms
- use least significant bit from iterative equation:
  - $x_i = x_{i-1}^2 \bmod n$
  - where $n=p.q$, and primes p,q
- unpredictable, passes **next-bit** test
- security rests on difficulty of factoring N
- is unpredictable given any run of bits
- slow, since very large numbers must be used
- too slow for cipher use, good for key generation

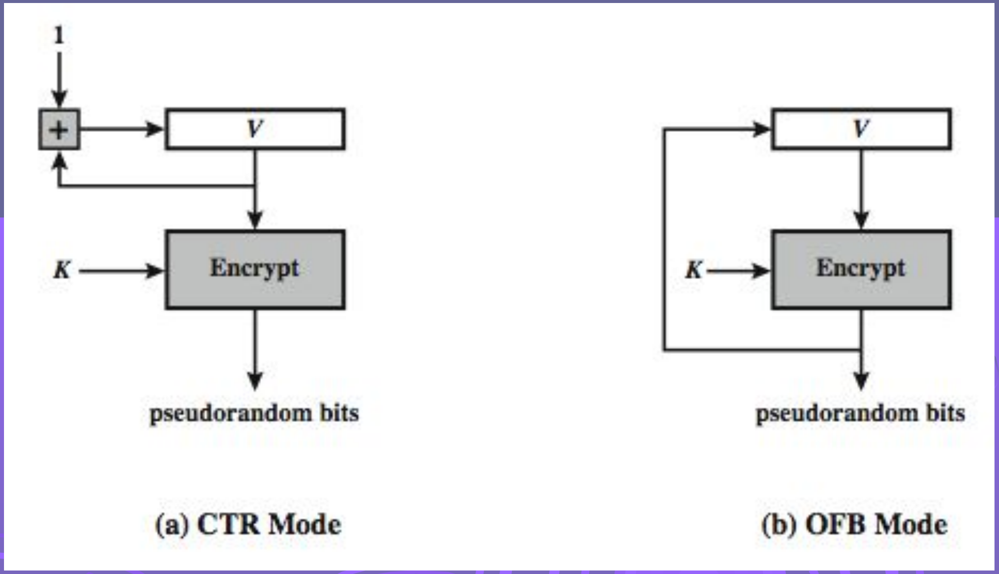
Using Block Ciphers as PRNGs

- for cryptographic applications, can use a block cipher to generate random numbers
- often for creating session keys from master key
- CTR (Counter mode)

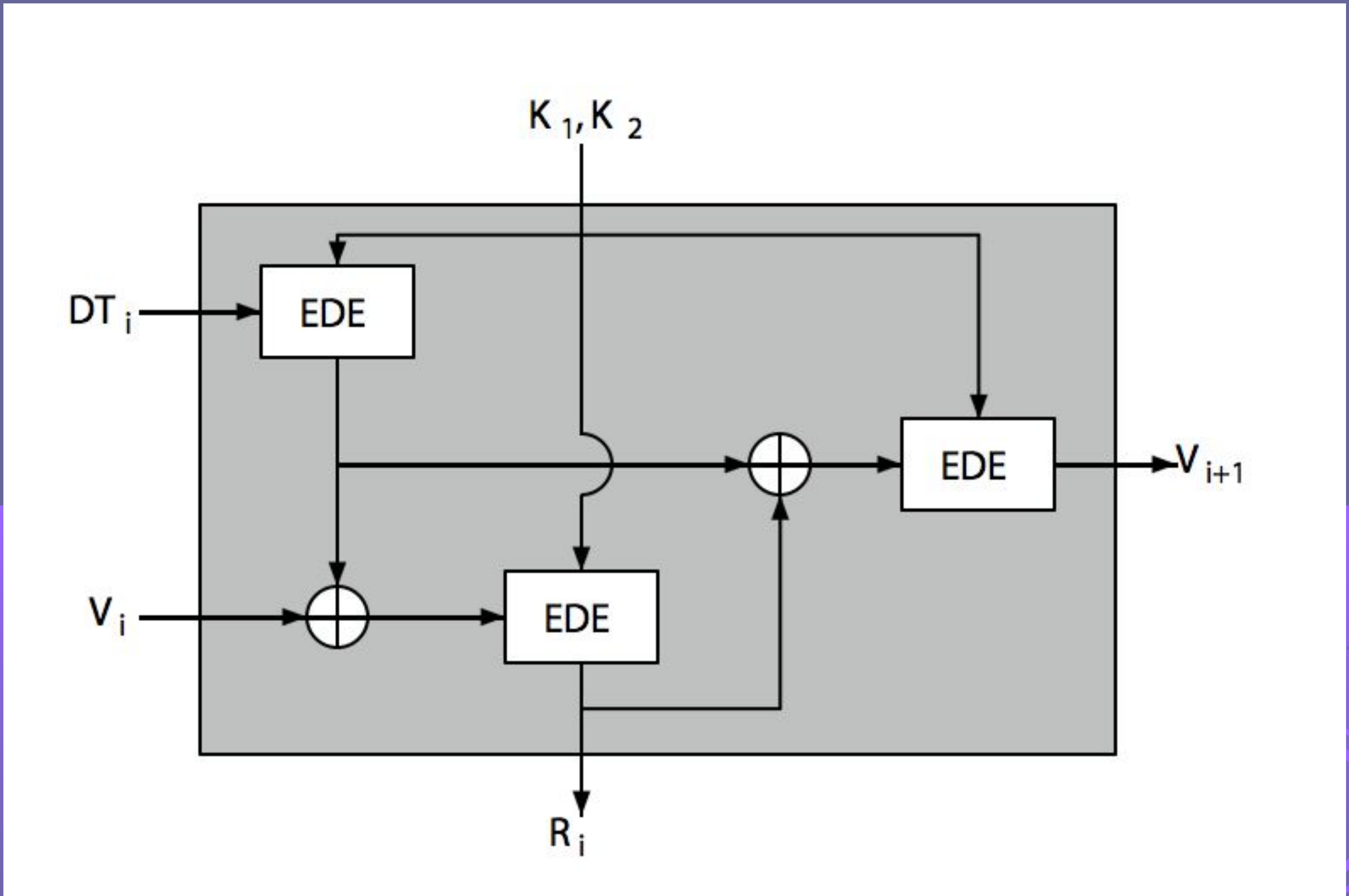
  $X_i = E_K[V_i]$

- OFB(o/p feedback mode)

  $X_i = E_K[X_{i-1}]$



ANSI X9.17 PRG

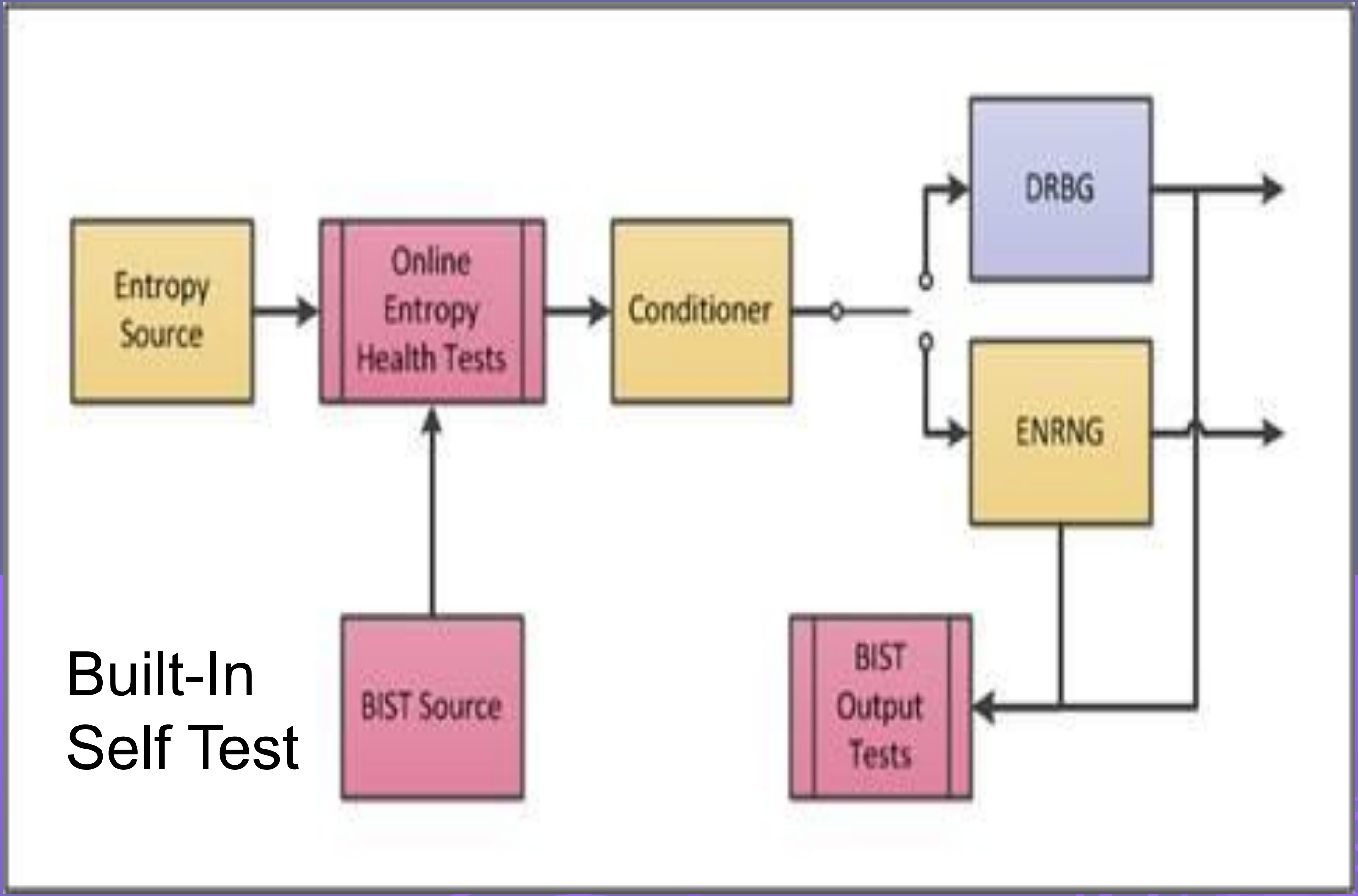


Dti = date and time

Natural Random Noise

- best source is natural randomness in real world
- find a regular but random event and monitor
- do generally need special h/w to do this
  - eg. radiation counters, radio noise, audio noise, thermal noise in diodes, leaky capacitors, mercury discharge tubes etc
- starting to see such h/w in new CPUs
  - Intel Ivy Bridge, Via Padlock
- problems of **bias** or uneven distribution in signal
  - have to compensate for this when sample, often by passing bits through a hash function
  - best to only use a few noisiest bits from each sample
  - RFC4086 recommends using multiple sources + hash

Intel Ivy Bridge TRNG



Health tests are basic, ad hoc, but detect RNG failure
Output zero with carry zero on failure (can't read a 0)

Trustworthiness of TRNGs

- Design appears to be very sound
- Possible back doors
  - Snowden leaks show NSA coerced hardware and software manufacturers to introduce back doors
  - Intel and Via hardware considered suspect
- Approach by FreeBSD
  - Use hardware PRNG as a source
  - Pass through Yarrow PRNG algorithm when producing cryptographically significant random numbers

See: https://www.schneier.com/paper-yarrow.pdf

Published Sources

- a few published collections of random numbers
- Rand Co, in 1955, published 1 million numbers
  - generated using an electronic roulette wheel
  - has been used in some cipher designs cf Khafre
- earlier Tippett in 1927 published a collection
- issues are that:
  - these are limited
  - too well-known for most uses

Summary

- pseudorandom number generation
- stream ciphers
- true random numbers