Chapter 8

Random Bit Generation and Stream Ciphers

Random Numbers

- Random (binary) numbers are used for
  - Key distribution and reciprocal authentication schemes
  - Session key generation
  - Key generation for the RSA public-key encryption algorithm
  - A bit stream for symmetric stream encryption
- Distinct requirements for a sequence of random numbers:
  - Uniform in all senses
  - Unpredictability (independence)

Truly random

- “True” random sequences: each number is statistically independent of the other numbers in the sequence, and therefore **unpredictable**
- True random numbers have their limitations
  - Inefficient to generate
  - Hard to store: cannot be compressed

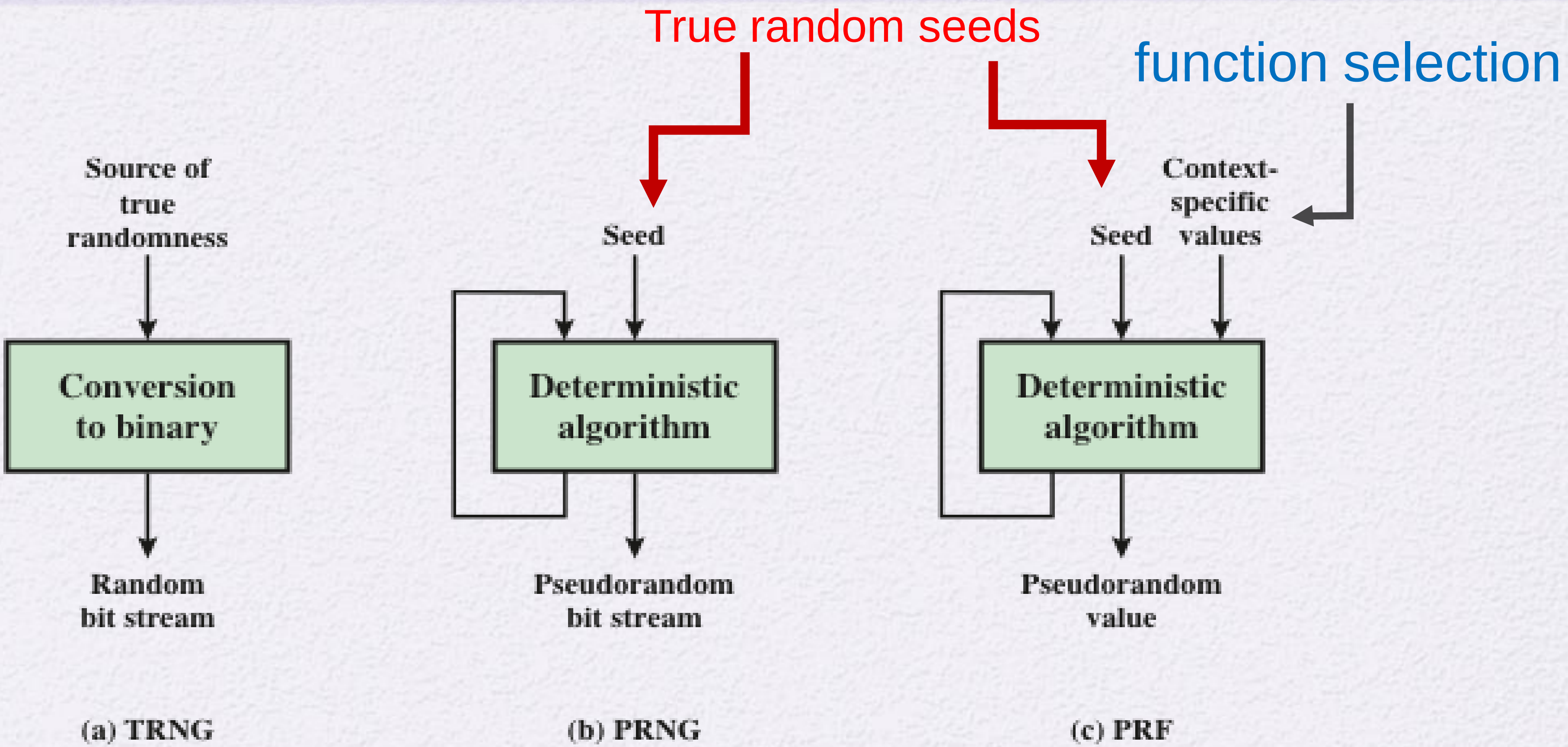
Unpredictability

- Unpredictability:
  - Forward unpredictability
    - The next output bit in the sequence should be unpredictable in spite of any knowledge of previous bits in the sequence
  - Backward unpredictability
    - Given b_{i+1} , b_{i+2}, … , b_n, one cannot predict b_i with probability higher than ½.
  - A random sequence will have no correlation with a fixed value

Pseudorandom Numbers

- Use “deterministic” algorithms and random seeds for random number generation
  - Not truly (statistically) random
- But, the resulting sequences will pass many computational (statistical) tests of randomness
  - *pseudorandom numbers*

Types of RNG



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

True Random Number Generator (TRNG)

- A random source for producing random numbers
- The random source (*entropy source*) is the physical environment of the computer
  - Eg, keystroke timing patterns, disk electrical activity, mouse movements, instantaneous values of the system clock, CPU usage, …
- TRNG may involve additional processing to overcome any bias in the source
- TRNG may simply involve conversion of an analog source to a binary output

Pseudorandom Number Generator (PRNG)

- A deterministic algorithm (program)
- Input: a *seed*
  - The seed must be secure and unpredictable
  - The seed itself must be a random or pseudorandom number
  - Typically the seed is generated by TRNG
- Output: a bit stream, determined solely by the algorithm and the seed
- **Security**: an adversary without seeds cannot predict the next number (bit) from the preceding numbers (bits)

PRNG Requirements

- Basic requirement: an adversary cannot distinguish a pseudorandom sequence from a true random sequence

- The output should
  - Look random
  - Unpredictable
  - Pass required computational and statistical tests

Randomness Tests

- NIST SP 800-22 lists 15 tests of randomness
- NIST has a software for randomness test suite
- See the supplementary slides

PRNG: Linear Congruential Generator

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	$0 \leq X_0 < m$

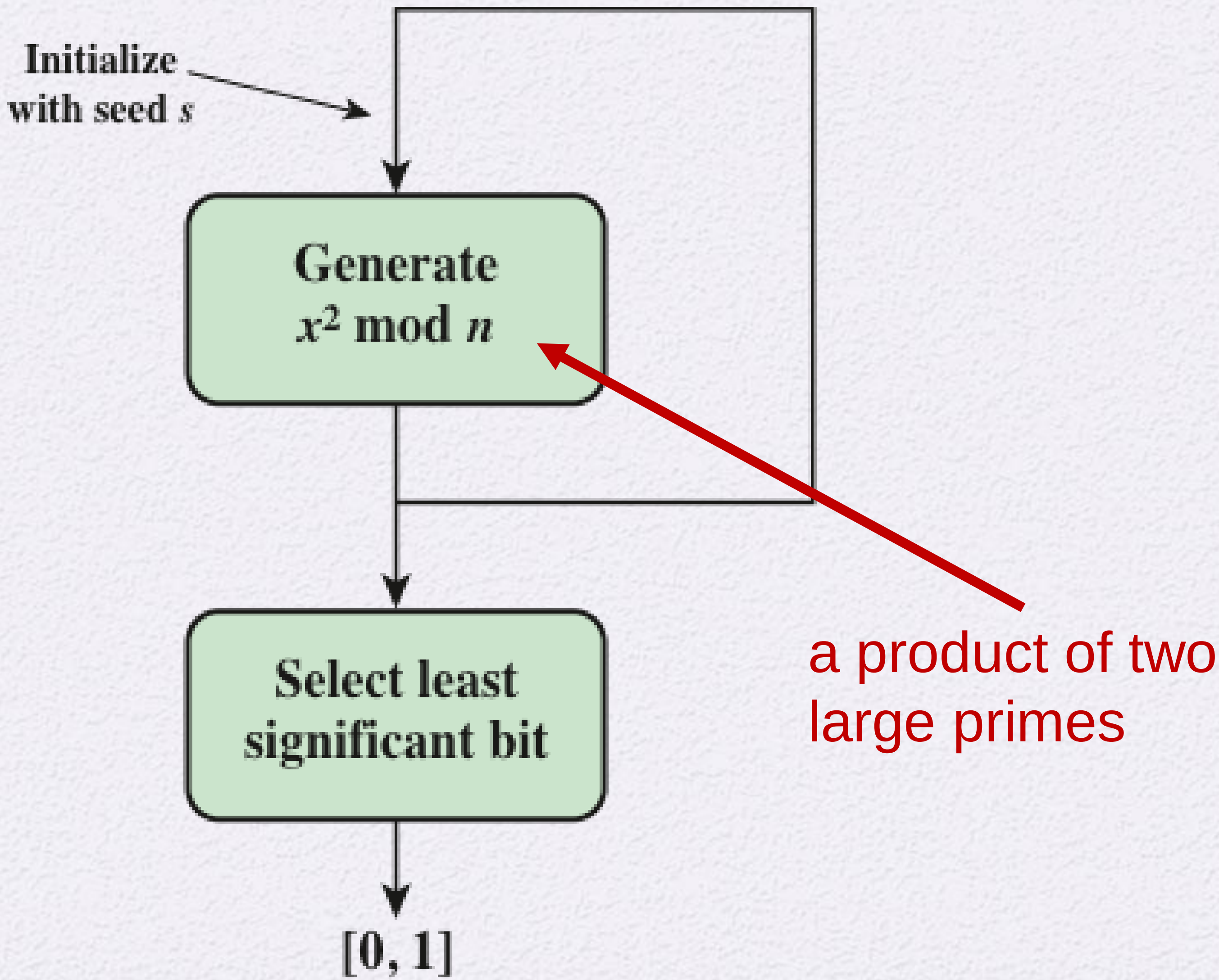
- The sequence:

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

- The selection of values for a, c, and m is critical for a good random number generator

PRNG: BBS generator

- Lenore Blum, Manuel Blum, Mike Shub
- The strength is proved based on the difficulty of factoring n
- Also called a *cryptographically secure pseudorandom bit generator* (CSPRBG)
  - It passes the *next-bit-test.*
  - *No* polynomial-time algorithm, on input of the first k bits of an output sequence, can predict the $(k + 1)$th bit with probability significantly greater than ½


Initialize
with seed s
Generate
x² mod n
Select least
significant bit
[0, 1]
a product of two
large primes

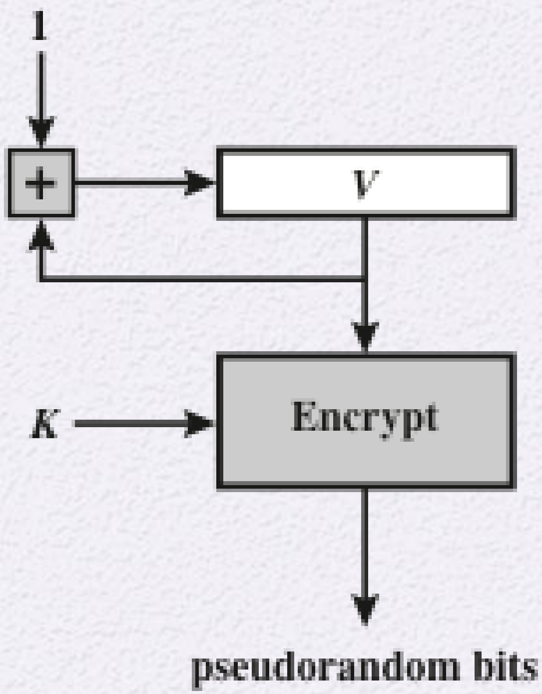
n: 192649 = 383x503

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

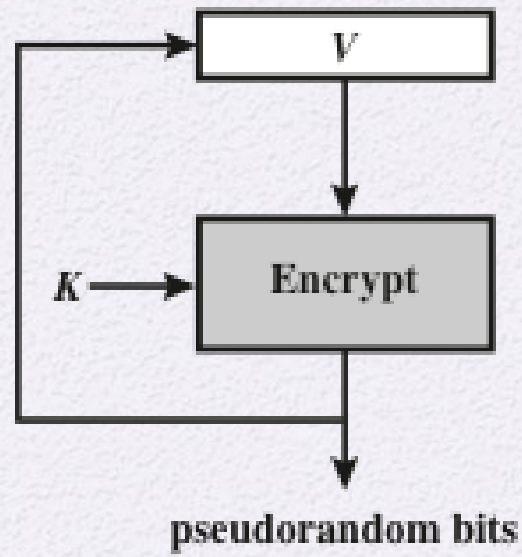
i	X_i	B_i
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

PRNG: Block Cipher Modes

- Two block ciphers are used to PNRG
  - CTR mode
    - Recommended in NIST SP 800-90, ANSI standard X.82, and RFC 4086
  - OFB mode
    - Recommended in X9.82 and RFC 4086



(a) CTR Mode



(b) OFB Mode

PRNG Using OFB

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

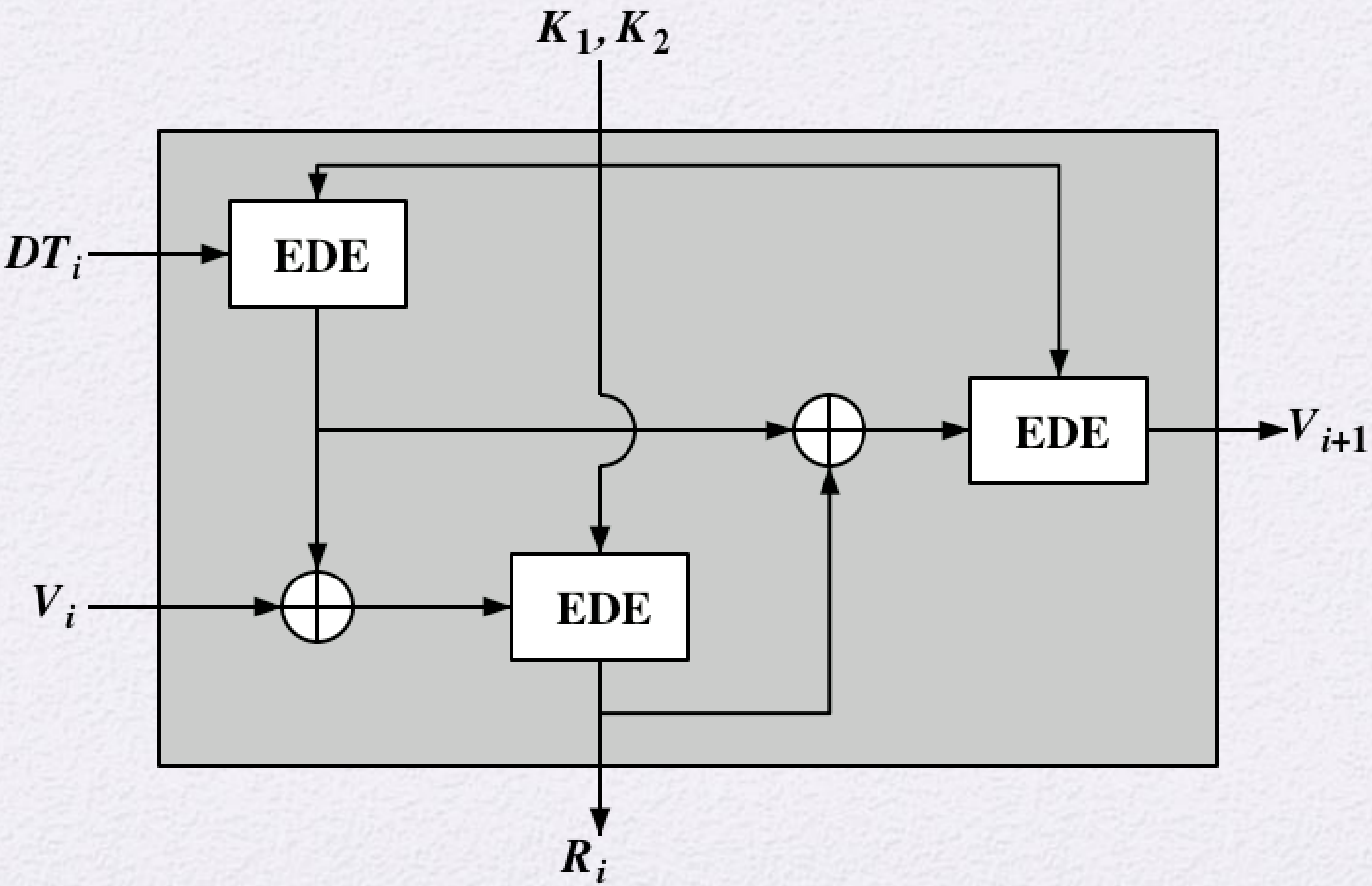
Seed (256 bits)

- Key: cfb0 ef31 08d4 9cc4 562d 5810 b0a9 af60
- V: 4c89 af49 6176 b728 ed1e 2ea8 ba27 f5a4

PRNG Using CTR

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

PRNG: ANSI X9.17

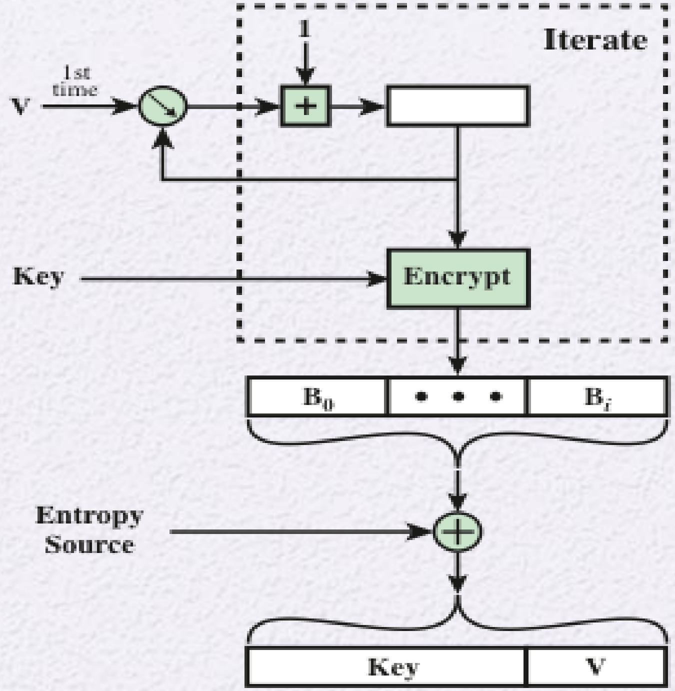


DRBG: NIST CTR_DRBG

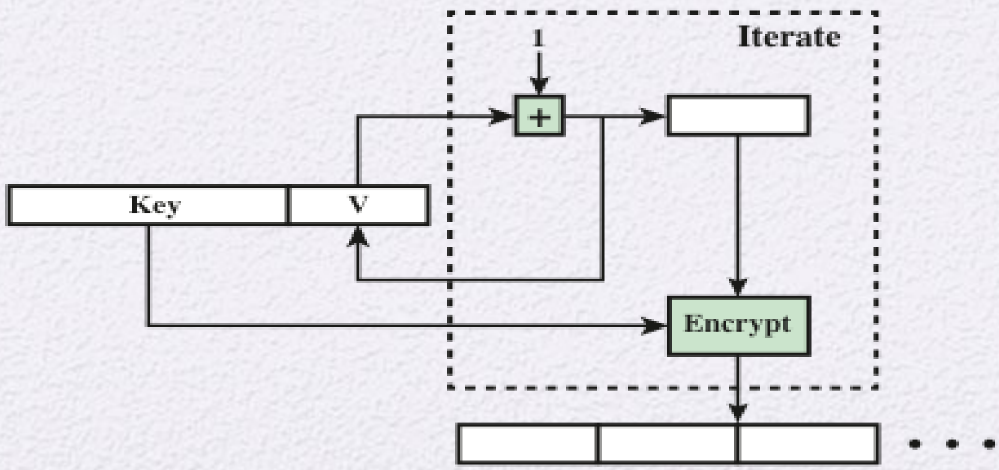
- Counter mode-deterministic random bit generator
- NIST SP 800-90: PRNG based on the CTR mode of operation
- Hardware implemented in all recent Intel processor chips
- DRBG assumes that an entropy source is available to provide random bits for seeds
- The encryption algorithm used in the DRBG may be 3DES with three keys or AES with a key size of 128, 192, or 256 bits

CTR_DRBG Parameters

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

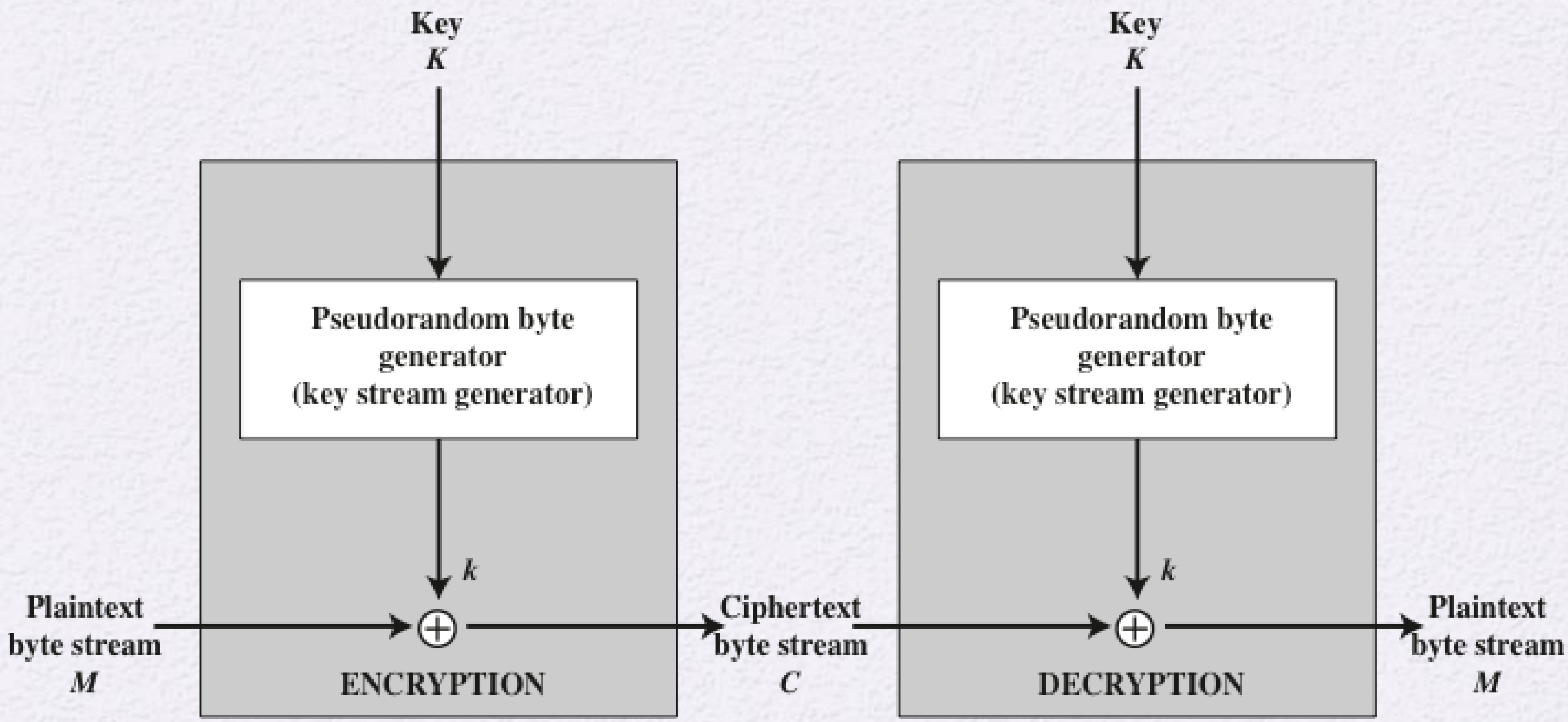


(a) Initialize and update function



(b) Generate function

PRNG for Stream Cipher



Stream cipher: RC4

- Designed in 1987 by Ron Rivest for RSA Security
- Variable key size stream cipher with byte-oriented operations
- Use a pseudorandom permutation
- Eight to sixteen machine operations per output byte: fast in software
- Used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards
- Also used in the Wired Equivalent Privacy (WEP) and the newer WiFi Protected Access (WPA) protocol, IEEE 802.11 wireless LAN standard

RC4: algorithm

1.RC4 initialization:

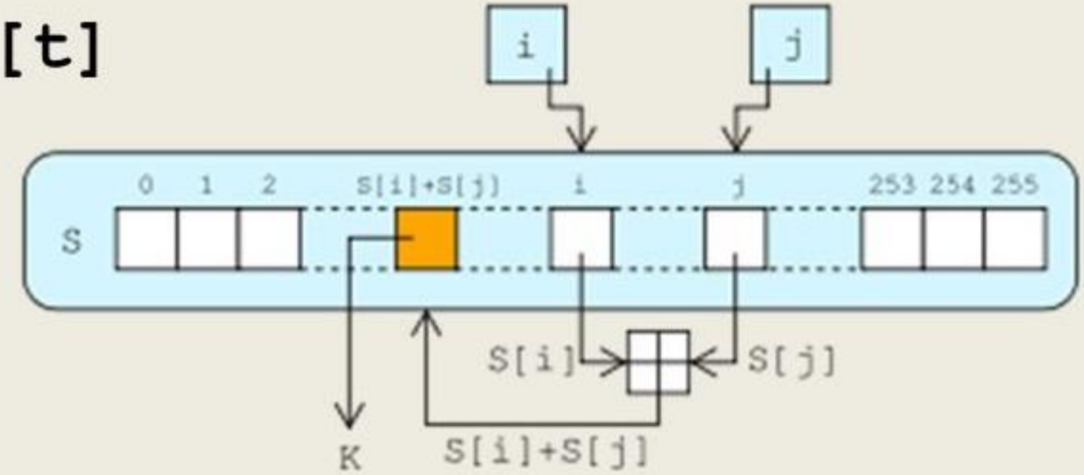
```
for i = 0 to 255
   S[i] = i
   T[i] = K[ i mod keylen];
next i
j = 0
for i = 0 to 255
   j = ( j + S[i] + T[i] ] ) mod 256
   swap ( S[i] , S[j] )
next i
```

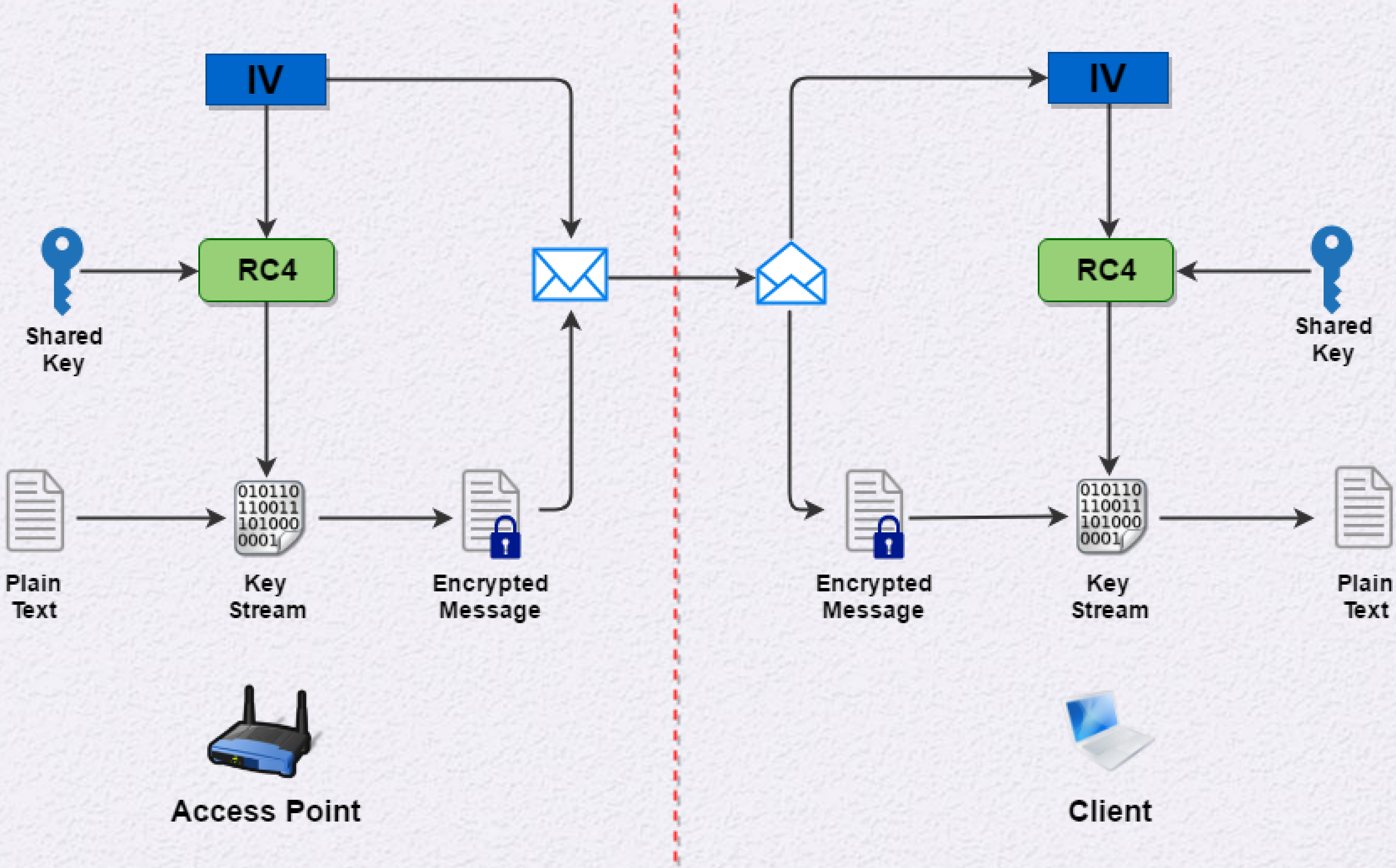
The key K[…] is used only for initializing S arrays.

RC4: algorithm

2. RC4 key stream byte:

1. i = j = 0
2. i = (i + 1) mod 256
3. j = (j + S[i]) mod 256
4. swap (S[i] , S[j])
5. t = (S[i] + S[j]) mod 256
6. keystreamByte = S[t]




IV
RC4
Shared Key
Plain Text
Key Stream
010110
110011
101000
0001
Encrypted Message
Access Point
IV
RC4
Shared Key
Encrypted Message
Key Stream
010110
110011
101000
0001
Plain Text
Client

Strength of RC4

- A number of published papers attacks RC4. But, none of these approaches is practical against RC4 with a reasonable key length
- Vulnerability
  - The way that keys are generated for use as input
  - Practically breakable
- IETF RFC7465 prohibits use of RC4 in TLS.
- NIST SP800-52 prohibits use of RC4 in government.

Entropy Sources

- A nondeterministic source to produce true randomness
- Measure 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 un-driven resistors

Possible Sources of Randomness

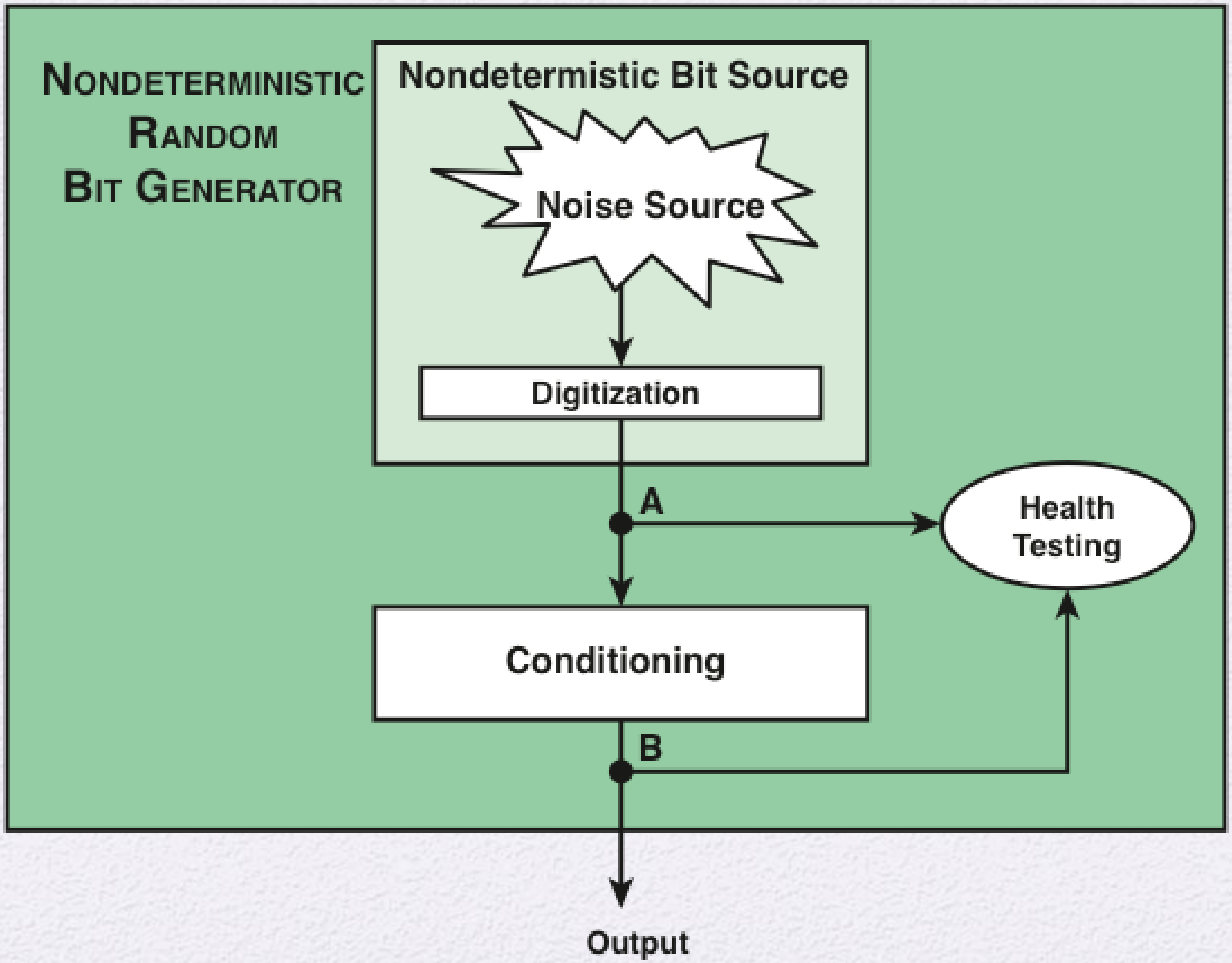
- RFC 4086 lists possible sources of randomness that can be used on a computer to generate true random sequences.
  - Sound/video input
  - Disk divers

Conditioning

- Problems of (raw) entropy source
  - Biased
  - Low entropy
- Conditioning algorithms / de-skewing algorithms
  - Modify a bit stream to further randomize the bits
  - Such as, hash function or symmetric block cipher

Conditioning by Hash Function

- A hash function produces an n-bit output from an input of arbitrary length
- Conditional by hash function:
  - Blocks of m input bits, with $m \geq n$, are passed through the hash function and the n output bits are used as random bits
  - Successive input blocks pass through the hash function to produce successive hashed output blocks


NONDETERMINISTIC
RANDOM
BIT GENERATOR
Nondetermistic Bit Source
Noise Source
Digitization
A
Health
Testing
Conditioning
B
Output

Intel: Digital Random Number Generator

- Intel offered the first commercial TRNG in May 2012
- It is implemented entirely in hardware on the same multicore chip as the processors

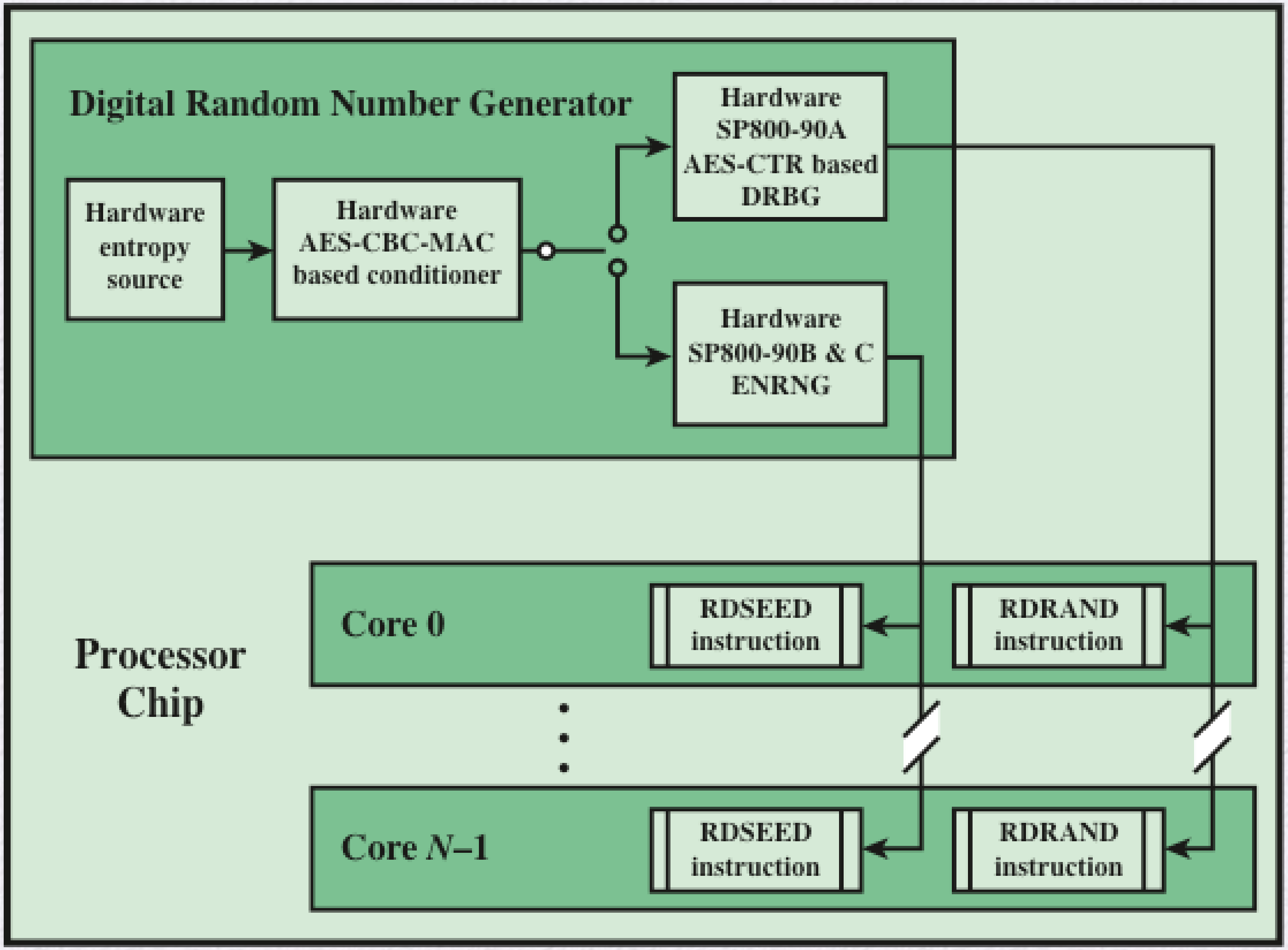


Figure 8.10 Intel Processor Chip with Random Number Generator

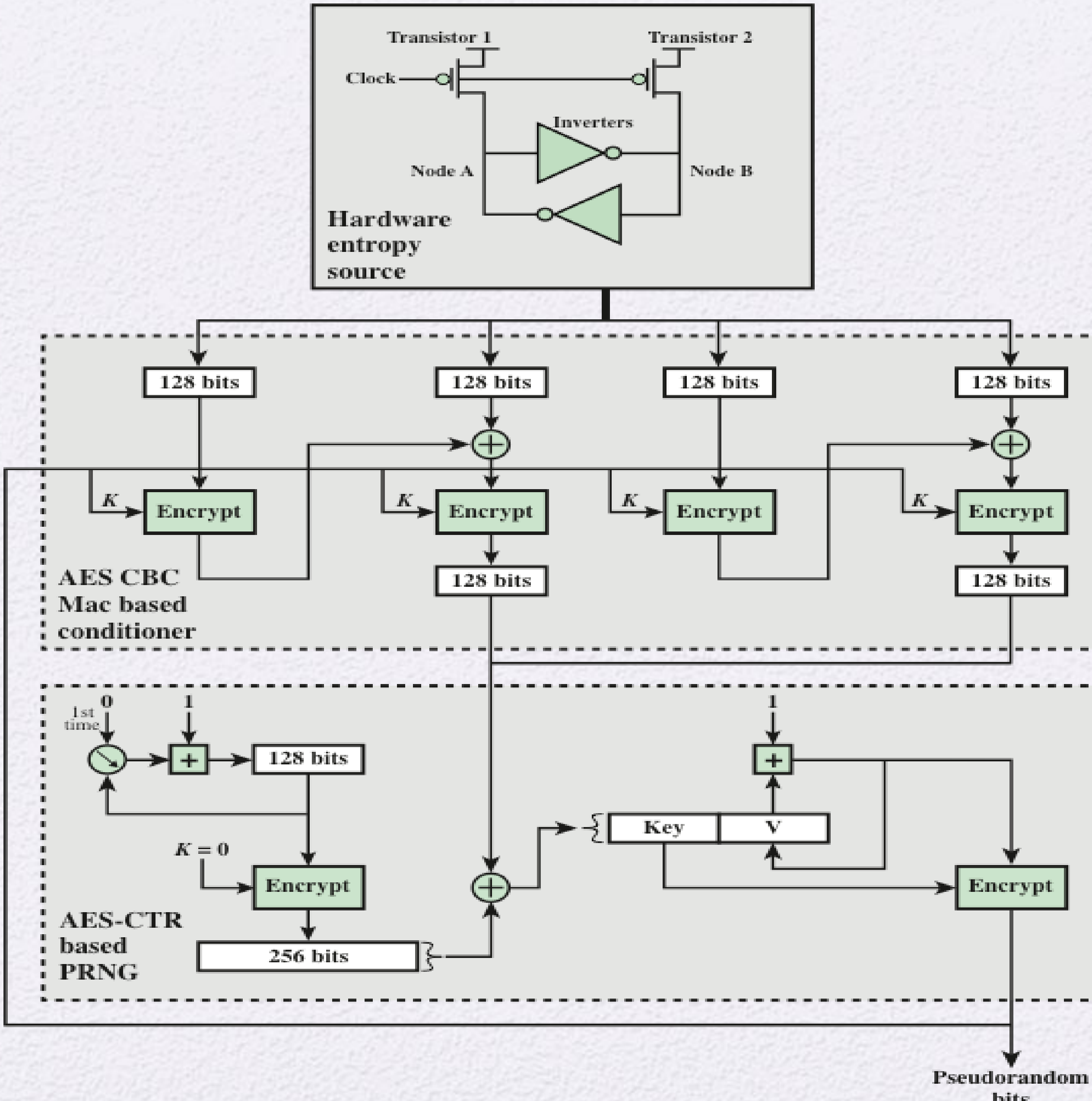


Figure 8.11 Intel DRNG Logical Structure

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
  - Conditioning
  - Intel digital random number generator