Cryptography and Security

Lecture 6

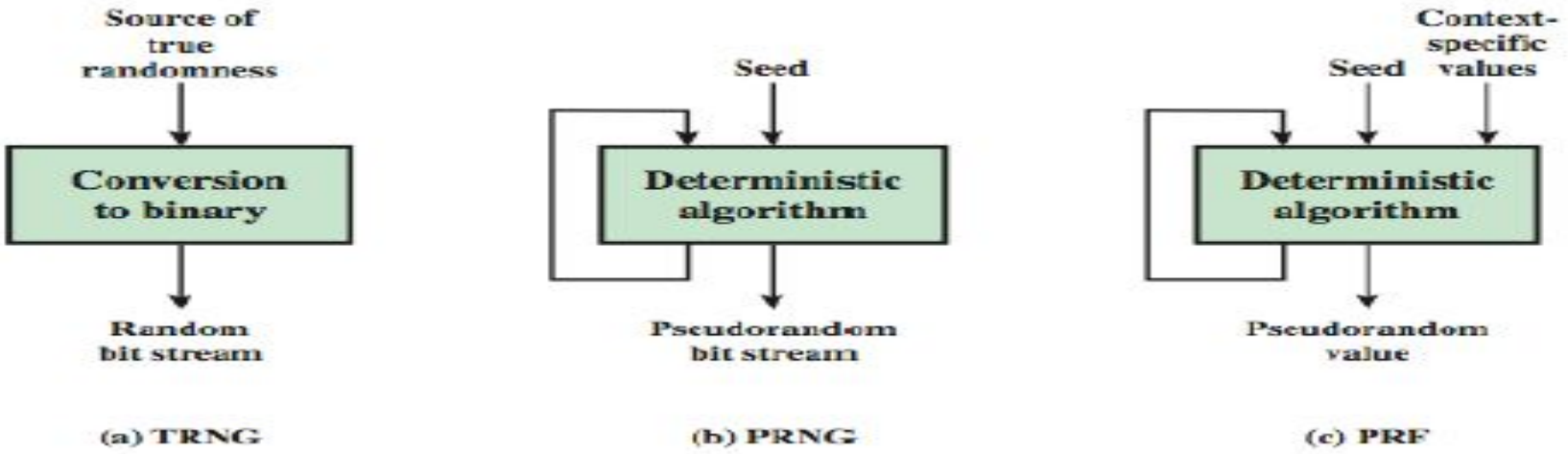
Pseudorandom Number Generation and Introduction to Stream Cipher

Random Number

- Many uses of **random numbers** in cryptography
  – Nonces in authentication protocols to prevent replay attack.
  – session keys
  – public key generation
  – keystream for a one-time pad
- Criteria of Random Number:
  – Uniform Distribution.
  – Independence
  – Unpredictability
- Care should be taken during the generation of pseudo random number.

(Pseudo) Random Number Generators

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



(Pseudo) Random Number Generators

- **Entropy Source:**
  - Source of true randomness.
  - Drawn from the physical environment of the computer such as keystroke timing patterns, disk electrical activity, mouse movements and instantaneous values of the system clock.
- **Seed:** Often generated by TRNG.
- If the adversary knows the algorithm and the seed, then s/he can reproduce the entire bit stream.

PRNG Requirements

- **Randomness**
  - uniformity, scalability, consistency
- **Unpredictability**
  - forward & backward unpredictability
- **Characteristics of the seed**
  - secure
  - if known adversary can determine output
  - so must be random or pseudorandom number

Linear Congruential 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 (knowing *a, c and m*)

Blum Blum Shub Generator

- Based on public key algorithms
- Use least significant bit from iterative equation:

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

- 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

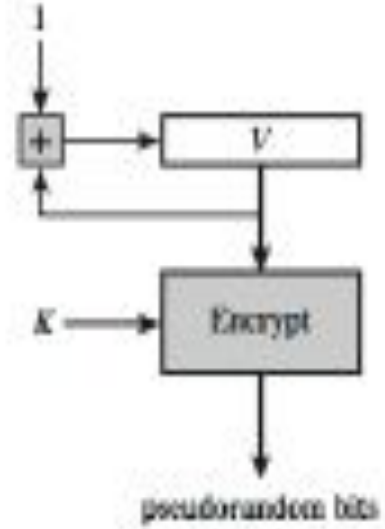
- For cryptographic applications, can use a block cipher to generate random numbers

- CTR

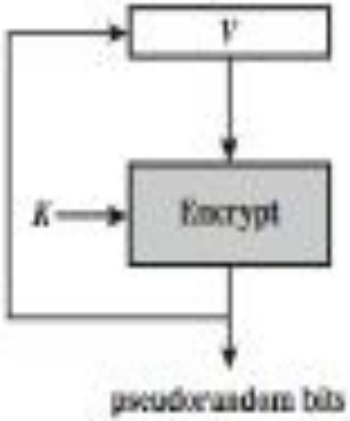
$$X_i = \mathrm{E}_K[\mathrm{V}_i]$$

- OFB

$$X_i = \mathrm{E}_K[X_{i-1}]$$



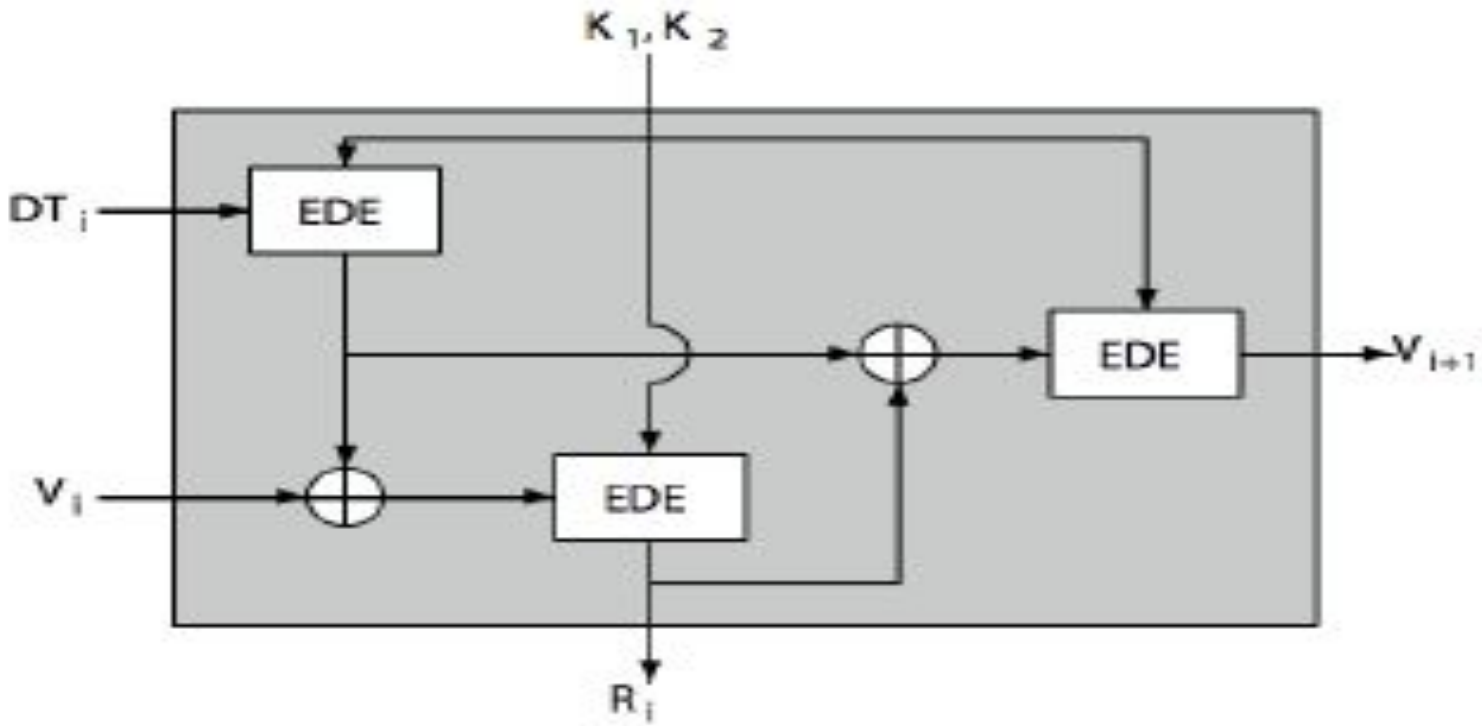
(a) CTR Mode



(b) OFB Mode

ANSI X9.17 PRNG

- A relatively complicated construction, including timestamps, and multiple encryptions
- 64 bit date and time (DT_i) and 64 bit seed value
- Uses 2-key (K_1,K_2) triple DES



Stream Cipher

- process message bit by bit (as a stream)
  - have a pseudo random **keystream**
- combined (XOR) with plaintext bit by bit
  - $C_i = M_i \; XOR \; StreamKey_i$

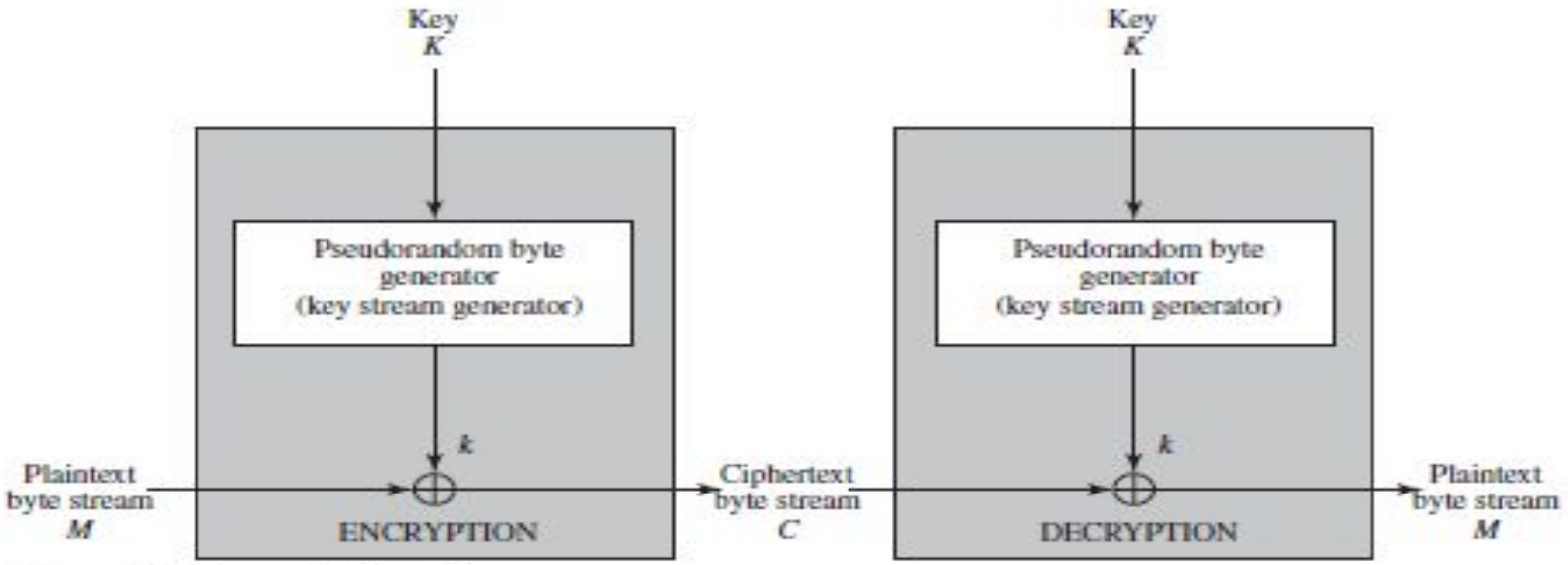


Figure 7.5 Stream Cipher Diagram

Stream Cipher

- Design Consideration for Stream Cipher:
  - The encryption period should have a large period.
  - Keystream should approximate the properties of a true random number stream as close as possible.
  - Key to generate the pseudorandom should be sufficiently large to defend the brute-force attack.
- Never reuse stream key
  - otherwise can recover messages (via XOR of msgs)
- Stream cipher can be used in data communication channel or a browser/web link.

RC4

- a proprietary cipher owned by RSA security.
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP/WPA)

RC4 Key Schedule

- Starts with an array **S** of numbers: 0....255
- Use key to well and truly shuffle **S**
- **S** forms **internal state** of the cipher

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

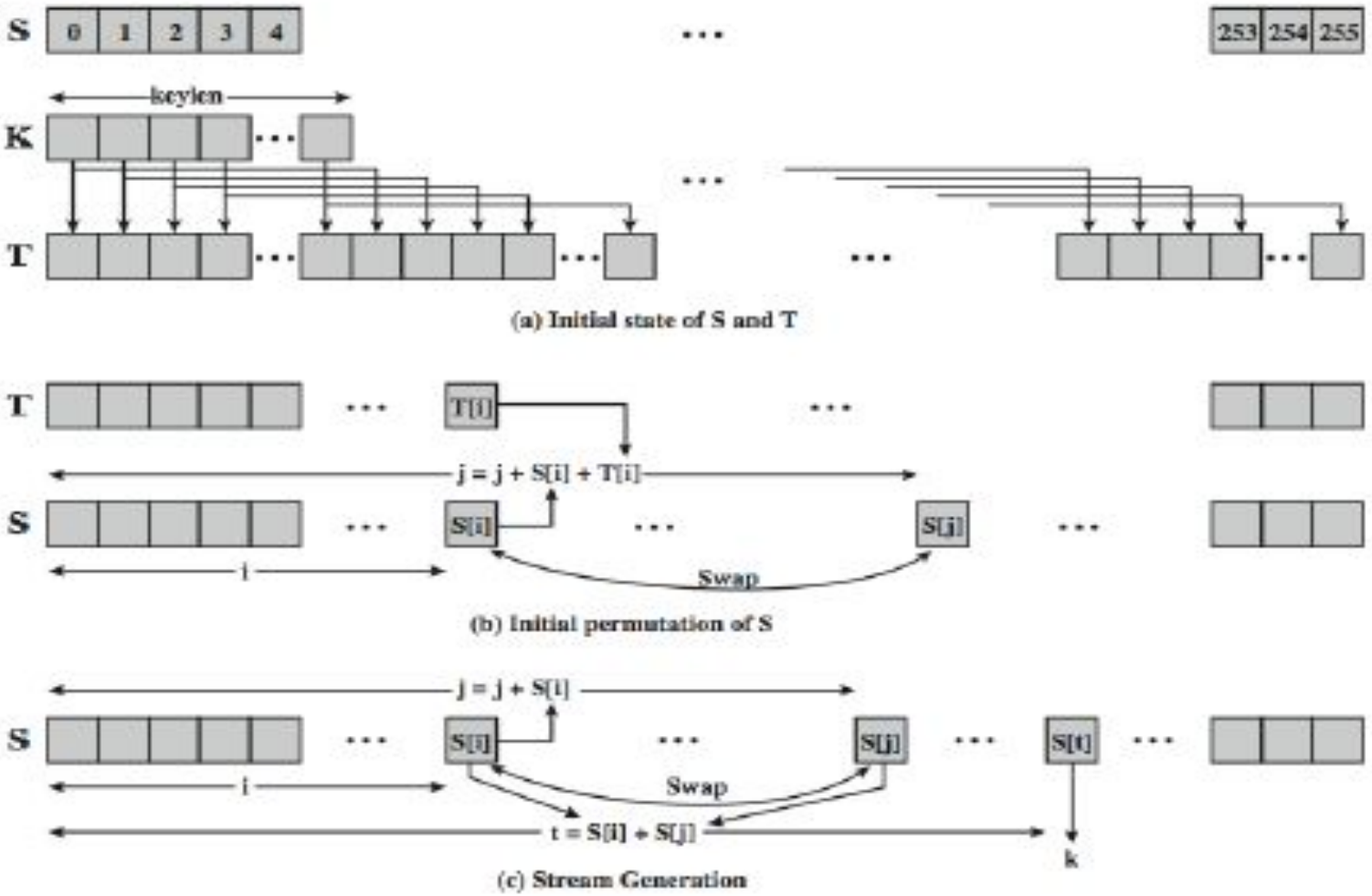
RC4 Encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value from permutation
- XOR **S[t]** with next byte of message to en/decrypt

```
for each message byte M_i
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap (S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    C_i = M_i XOR S[t]
```

- used in the WiFi Protected Access (WPA) .It is optional for use in Secure Shell (SSH) and Kerberos.

RC4 Overview



True Random Number Generator

- **Entropy Source:**
  - Non-deterministic source to generate randomness.
  - Measure unpredictable natural processes such as pulse detector of radiation event, gas discharge tubes and leaky capacitor.
  - Generators are commercially available.
  - Some sources for computers:
    - Sound / video input.
    - Disk drive.
    - **Skew:**
  - Output from TRNG may contains some bias.
  - Deskewing process:
    - » Pass though MD5 or SHA-1.
    - » Operating system has built-in mechanism to generate random number.

Comparison of PRNG and TRNG

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

- TRNG is used for key generation and for some applications where small number of random bits are required.

Chapter 8 from your textbook.