

# Random Number Generation and Stream Ciphers

Yih-Kuen Tsay

Department of Information Management  
National Taiwan University

# The Use of Random Numbers

- 🌐 Random numbers are used by a number of security algorithms for:
  - ☀ Nonces (used in authentication protocols)
  - ☀ Session key generation (by the KDC or an end system)
  - ☀ Key generation for the RSA algorithm
- 🌐 Two requirements: **randomness** and **unpredictability**.

# Pseudorandom Numbers

- 🌐 True random numbers are hard to come by.
- 🌐 Cryptographic applications typically use **algorithmic techniques** for random number generation.
- 🌐 These algorithms are deterministic and therefore produce sequence of numbers that are not statistically random.
- 🌐 If the algorithm is good, the resulting sequences will pass reasonable tests for randomness.
- 🌐 Such numbers are often referred to as **pseudorandom numbers**.

# The Linear Congruential Method

$m$	the modulus	$m > 0$
$a$	the multiplier	$0 \leq a < m$
$c$	the increment	$0 \leq c < m$
$X_0$	the starting value (seed)	$0 \leq X_0 < m$

- Iterative equation:  $X_{n+1} = (aX_n + c) \bmod m$
- Larger values of  $m$  imply higher potential for a long period.
- For example,  $X_{n+1} = (7^5 X_n) \bmod (2^{31} - 1)$  has a period of  $2^{31} - 2$ .
- What are the weakness and the remedy?

# The Blum Blum Shub (BBS) Generator

- Choose two large prime numbers  $p$  and  $q$  such that  $p \equiv q \equiv 3 \pmod{4}$ . Let  $n = p \times q$ .
- Choose a random number  $s$  relatively prime to  $n$ .
- Bit sequence generating algorithm:

$$\begin{aligned} X_0 &= s^2 \pmod{n} \\ \text{for } i &= 1 \text{ to } \infty \\ X_i &= (X_{i-1})^2 \pmod{n} \\ B_i &= X_i \pmod{2} \end{aligned}$$

- The BBS generator passes the [next-bit test](#).




# 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

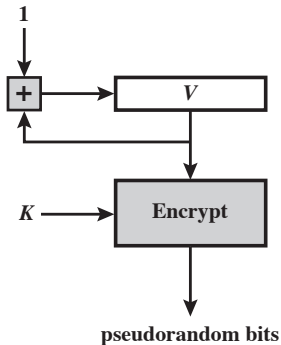
$i$	$X_i$	$B_i$
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

Source: Table 7.1, Stallings 2010

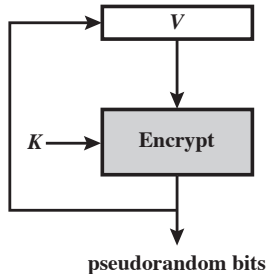
# Cryptographical Generation

-  **Cyclic encryption:** use an arbitrary block cipher. Full-period generating functions are easily obtained.
-  **DES Output Feedback Mode:** the successive 64-bit outputs constitute a sequence of pseudorandom numbers.
-  **ANSI X9.17 Pseudorandom number generator (PRNG):** make use of triple DES. Employed in financial security applications and PGP.

# Pseudorandom Number Generation



(a) CTR Mode



(b) OFB Mode

Source: Figure 7.3, Stallings 2010



# Results from CTR Mode

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

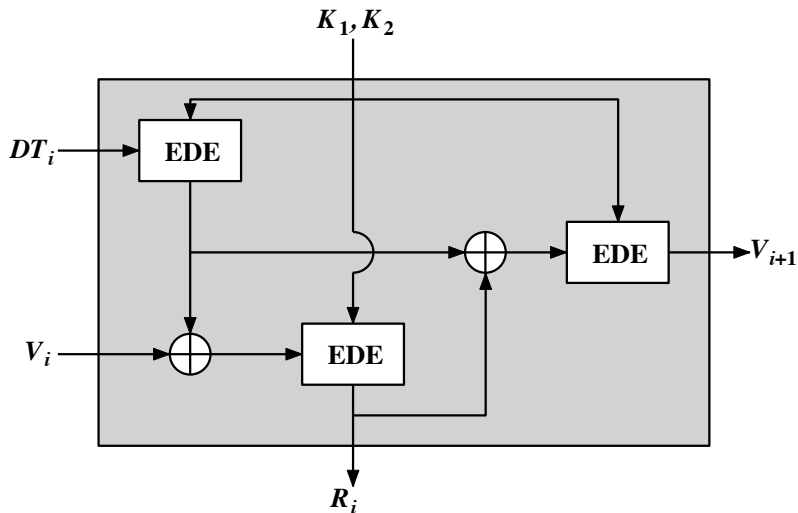
Source: Table 7.3, Stallings 2010

# Results from OFB Mode

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

Source: Table 7.2, Stallings 2010

# ANSI X9.17 PRNG

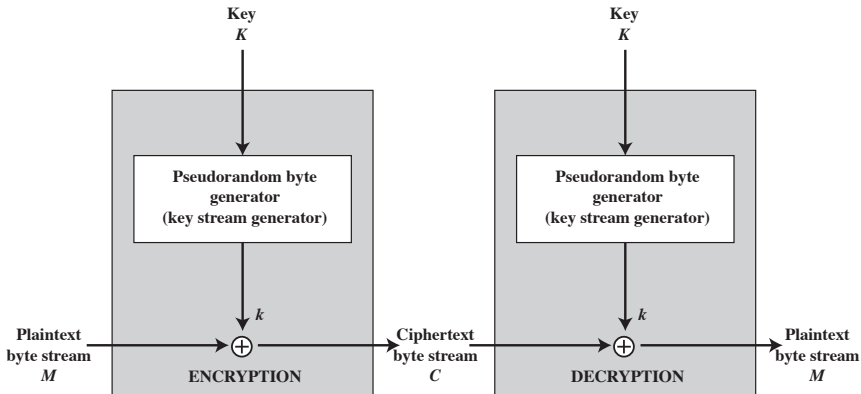


Source: Figure 7.4, Stallings 2010

# Stream Ciphers

- 🌐 Encrypt plaintext **one byte** at a time; other units are possible.
- 🌐 Typically use a **keystream** from a pseudorandom byte generator (conditioned on the input key).
- 🌐 Decryption requires the same pseudorandom sequence.
- 🌐 Usually are **faster** and use far less code than block ciphers.
- 🌐 Design considerations:
  - ☀️ The encryption sequence should have a **large period**.
  - ☀️ The keystream should approximate a **truly random** stream.
  - ☀️ The input key needs to be sufficiently long.

# Stream Cipher Diagram



Source: Figure 7.5, Stallings 2010

- Probably the most widely used stream cipher, e.g., in SSL/TLS and in WEP (part of IEEE 802.11)
- Developed in 1987 by Ron Rivest for RSA Security Inc.
- Variable key size with byte-oriented operations
- Based on the use of random permutation
- The period of the cipher likely to be  $> 10^{100}$
- Simple and fast
- Proprietary, though its algorithm has been disclosed

# Comparisons of Symmetric Ciphers

<b>Cipher</b>	<b>Key Length</b>	<b>Speed (Mbps)</b>
DES	56	9
3DES	168	3
RC2	Variable	0.9
RC4	Variable	45

Source: Table 7.4, Stallings 2010

# Stream Generation in RC4

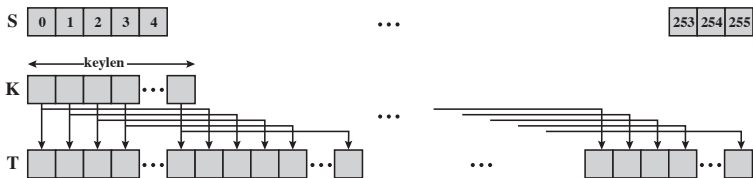
```
i, j = 0;  
while (true)  
    i = (i + 1) mod 256;  
    j = (j + S[i]) mod 256;  
    Swap (S[i], S[j]);  
    t = (S[i] + S[j]) mod 256;  
    k = S[t];
```



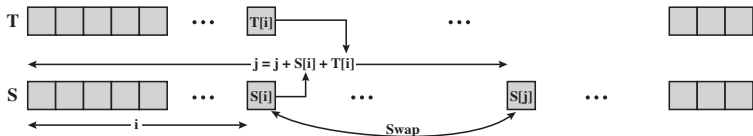
# Initialization of S in RC4

```
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]);
```

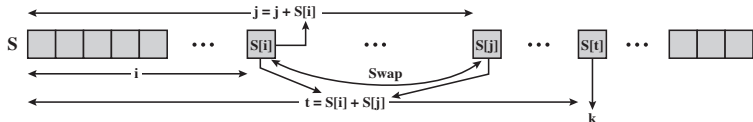
# RC4 in Picture



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation