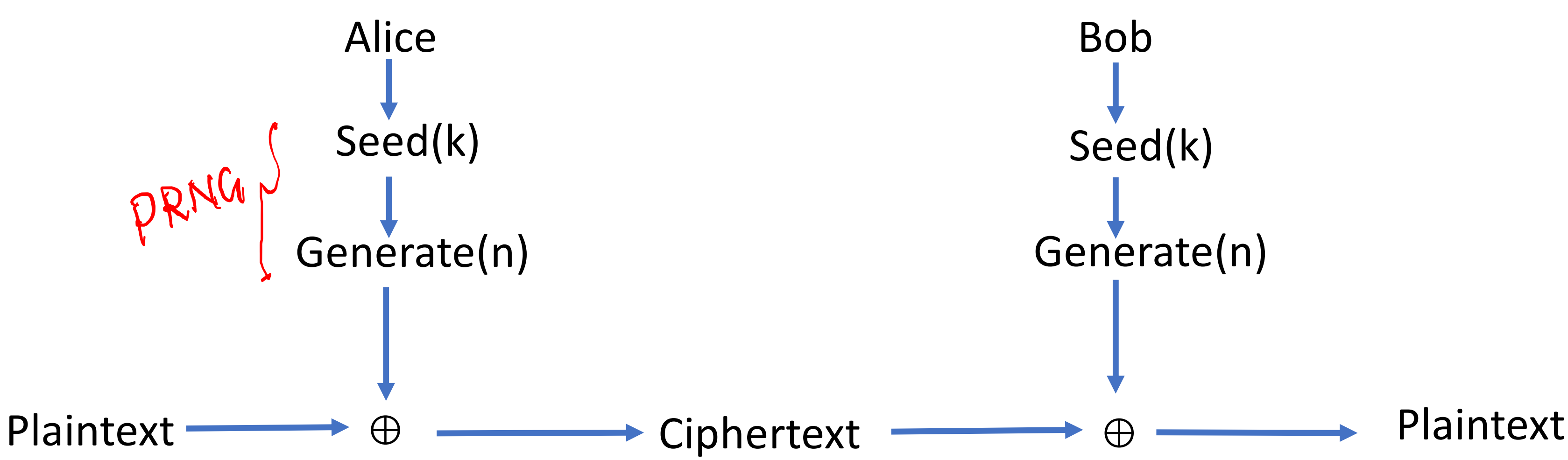
Stream Ciphers

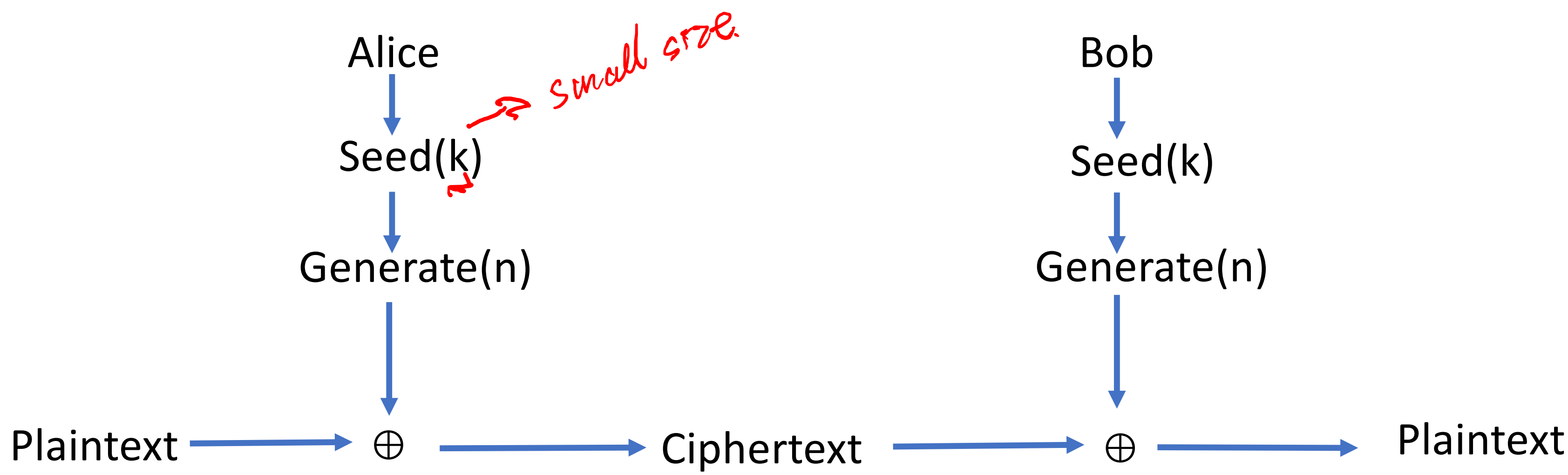
- Protocol: Alice and Bob both seed a secure PRNG with their symmetric secret key, and then use the output as the key for stream key

k → pre-share



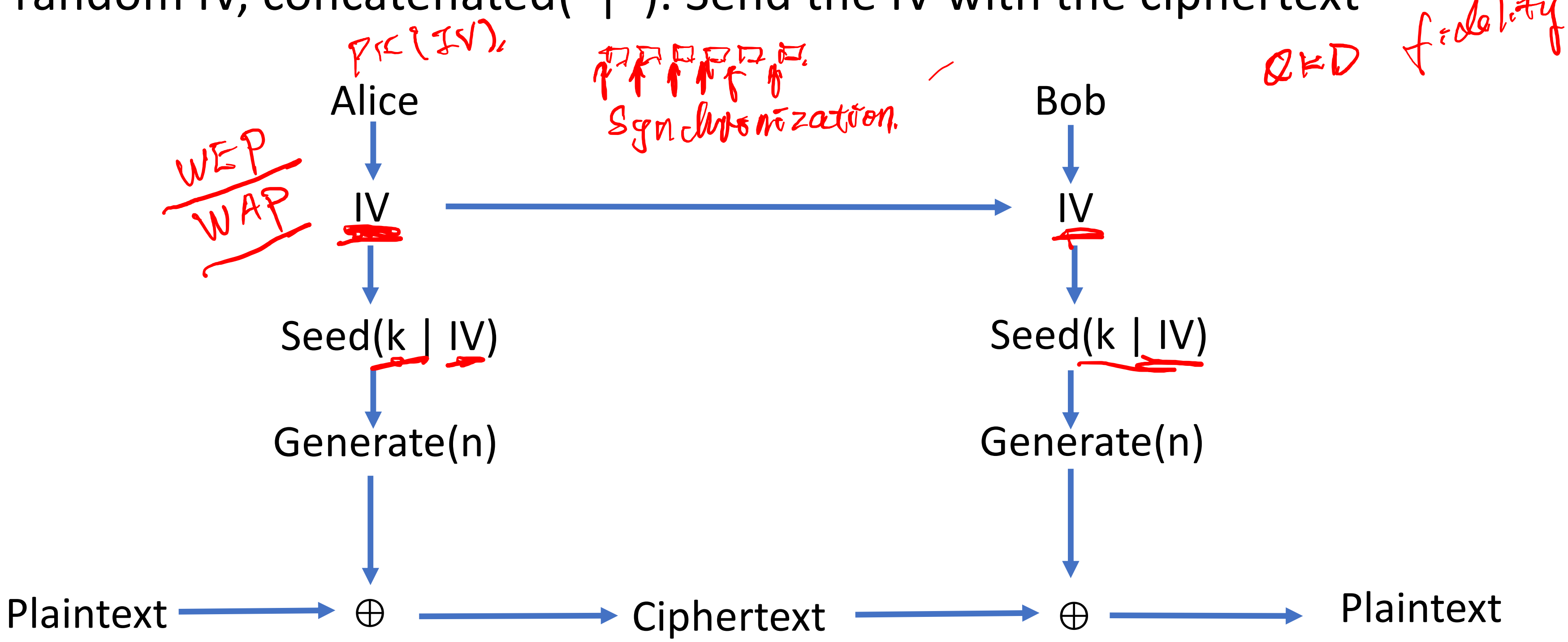
Stream Ciphers: Encrypting Multiple Messages

- How do we encrypt multiple messages without key reuses?



Stream Ciphers: Encrypting Multiple Messages

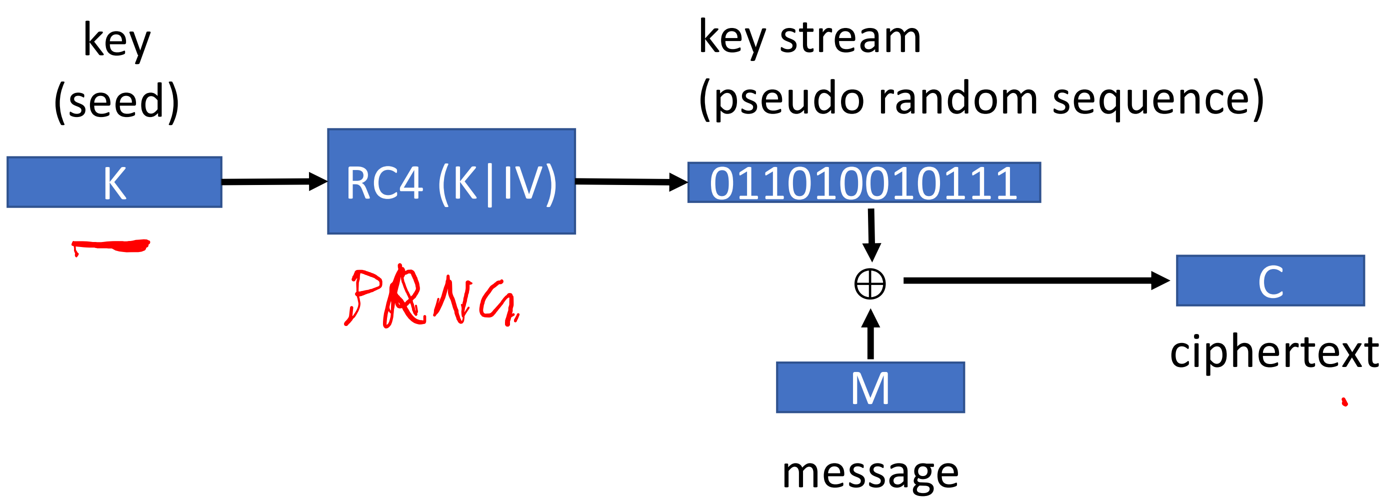
- Solution: For each message, seed the PRNG with the key and a random IV, concatenated(“|”). Send the IV with the ciphertext



Real-world example: RC4

- a proprietary cipher designed in 1987
- Extremely simple but effective! → fast
- Very fast - especially in software → IV
- Easily adapts to any key length, byte-oriented stream cipher
- widely used (web SSL/TLS, wireless WEP, WAP)
- A trade secret by RSA Security → RSA 1984
- uses that permutation to scramble input info processed a byte at a time
  swap( )

RC4 Stream Cipher



①

RC4 Key Schedule

② Encryption.

- starts with an array S of numbers: 0...255
- S forms internal state of the cipher
- given a key k of length l bytes
- use key to well and truly shuffle S

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

K || IV > 256

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
  j = (j + S[i] + T[i]) mod 256;
  Swap (S[i], S[j]);
```

Throw away ~~T & K~~, retain S

$T[i] = K[i \text{ mod } keylen]$ $\qquad$ i mod keylen

if keylen $\geq i$ $\Rightarrow T[i] = K[i]$

$K = [1, 2, 3, 4]$ $\qquad$ keylen $= 4$

$0 \;\; 1 \;\; 2 \;\; 3$

else keylen $< i$

$i=0$ $T[0] = 1$

$i=1$ $T[1] = 2$

$i=2$ $T[2] = K[2\%4] = K[2] = 3$

$i=3$ $T[3] = K[3\%4] = K[3] = 4$

$i=5$ $T[5] = K[5\%4] = K[1] = 2$

$i=6$ $T[6] = K[6\%4] = K[2] = 3$

$\vdots$

$i=7$ $\qquad = 4$

$T[\,] = 1, 2, 3, 4, 2, 3, 4$ $\rightarrow$ repeatitive pattern key reuse

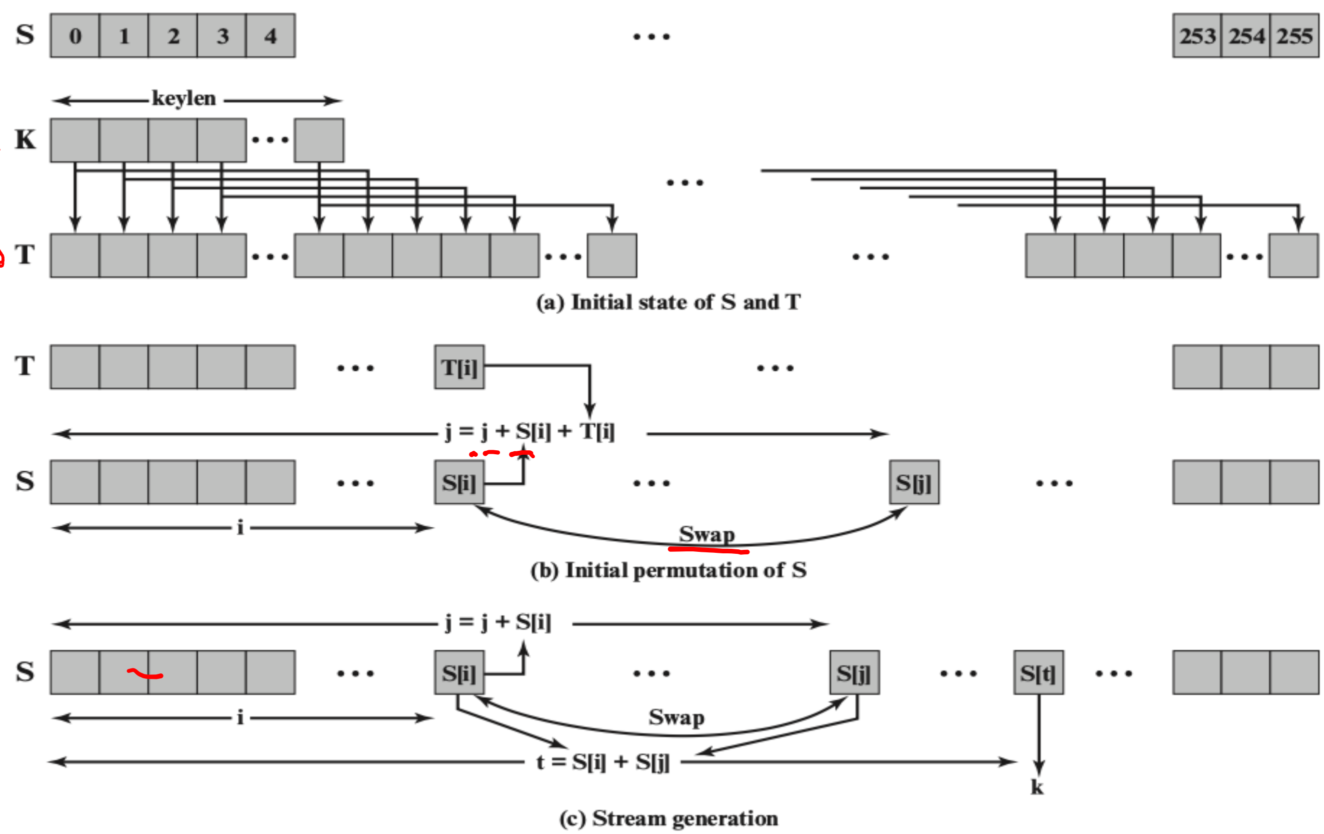
$\therefore$ Add keylen.

RC4 Encryption

- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value
- XOR with next byte of message to en/decrypt

```
i = j = 0
for each message byte Mi
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap(S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    Ci = Mi XOR S[t]
```

RC4



S
0
1
2
3
4
253
254
255
keylen
K
T
(a) Initial state of S and T
T
T[i]
j = j + S[i] + T[i]
S
S[i]
S[j]
i
Swap
(b) Initial permutation of S
j = j + S[i]
S
S[i]
S[j]
S[t]
i
Swap
t = S[i] + S[j]
k
(c) Stream generation

S

RC4 Security

brute force

- claimed secure against known attacks
- since RC4 is a stream cipher, must never reuse a key
- have a concern with WEP, but due to key handling rather than RC4 itself

block cipher in counter mode

WAP IV WAP1 → WAP2.

- RC4 is widely used, in SSL for secure web transactions amongst other uses. Currently it's regarded as secure, if used correctly.
  - Extensively studied, not a completely secure PRNG, first part of output biased, when used as stream cipher, should use RC4-Drop[n]
    - Which drops first n bytes before using the output
    - Conservatively, set n=3072

Summary – Chapter 2

- Symmetric block cipher
  - DES, 3DES
  - AES
- Random number
  - true random number
  - pseudorandom number
- Stream cipher
- The security of symmetric encryption depends on the secrecy of the key
- Symmetric encryption: pros and cons

Reading material

- Encryption: Strengths and Weaknesses of Public-key Cryptography

Homework 1 - individual

- Chapter 1 & 2
- **Deadline**: Thursday, October 2, 11:59 PM
- We will use the Canvas submission time as your final timestamp
- 10% penalty per day for late submission

Modular Arithmetic

- Definition (congruent modulo):
  - given b – a = km for some k ϵ Z, then a $\equiv b$ (mod m)
- Given a $\equiv b$ (mod m) and c $\equiv d$ (mod m), then
  - a + b ≡ c + d (mod m)
  - a - b ≡ c - d (mod m)
  - a + c ≡ b + d (mod m)
  - a × c ≡ b × d (mod m)
  - $a^k \equiv b^k$ (mod m)
  - ka = kb (mod m)
  - p(a) ≡ p(b) (mod m), any polynomial *p*(*x*) with integer coefficients
- $\mathrm{A} \oplus B \oplus B$ = A

Thank you!