Stream Ciphers:

process the message bit by bit (as a stream)

•typically have a (pseudo) random **stream key**

•combined (XOR) with plaintext bit by bit

•randomness of **stream key** completely destroys any statistically properties in the message

•$C_i = M_i$ XOR $StreamKey_i$

•what could be simpler!!!!

•but must never reuse stream key

•otherwise, can remove effect and recover messages, M⊕K⊕K = M

•Idea: replace "rand" by "pseudo rand"

•Use Pseudo Random Number Generator

•A secure PRNG produces output that looks indistinguishable from random

•An attacker who can't see the internal PRNG state can't learn any output

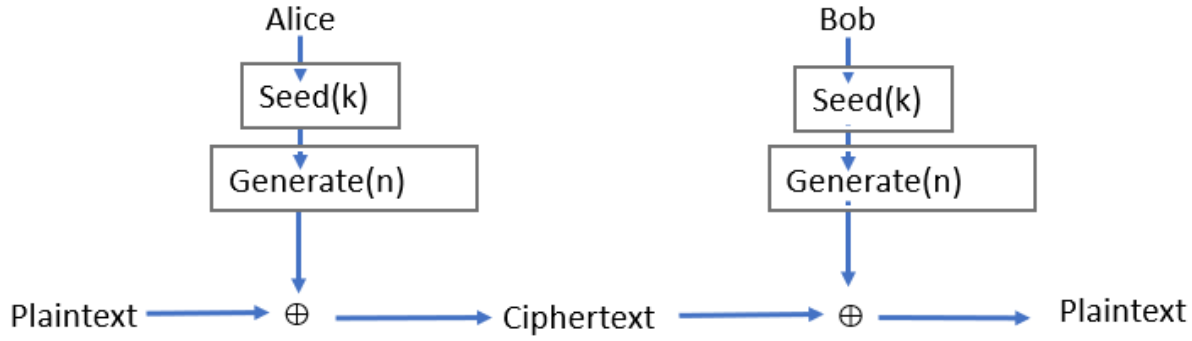
•PRNG: $\{0,1\}^s$ ® $\{0,1\}^n$

•expand a short (e.g., 128-bit) random seed into a long (typically unbounded) string that "looks random"

•Secret key is the seed

•Basic encryption method: $E_{key}[M]$ = M Å PRNG(key)

•**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



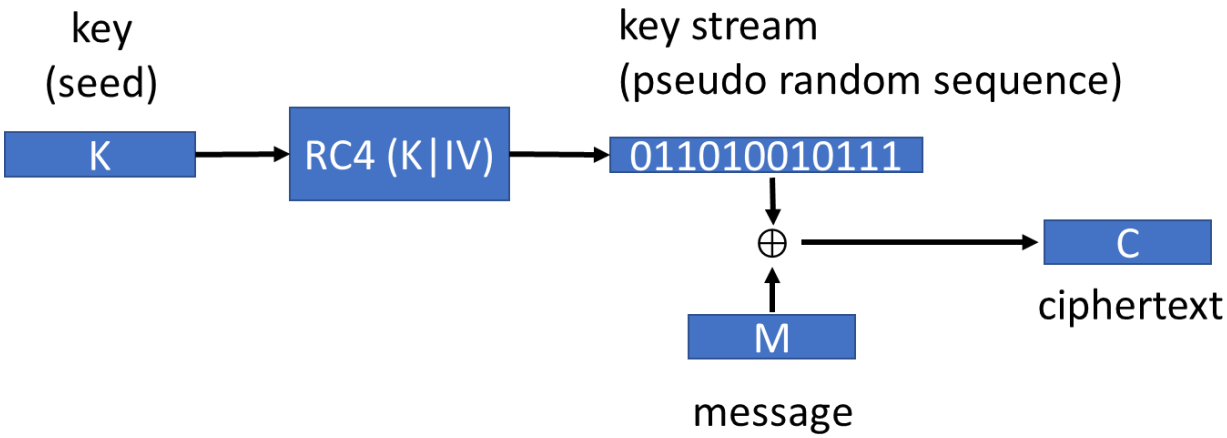
How do we encrypt multiple messages without key reuses?

Real-world example: RC4

- •A proprietary cipher designed in 1987
- •Extremely simple but effective!
- •Very fast - especially in software
- •Easily adapts to any key length, byte-oriented stream cipher
- •Uses that permutation to scramble input info processed a byte at a time

•Widely used (web SSL/TLS, wireless WEP, WPA)

RC4 Stream Cipher



RC4 Key Schedule

•starts with an array S of numbers: 0…255

•use key to well and truly shuffle

•S forms internal state of the cipher

•given a key k of length L bytes

RC4 Encryption

•encryption continues shuffling array values

•sum of shuffled pair selects "stream key" value

•XOR with next byte of message to en/decrypt