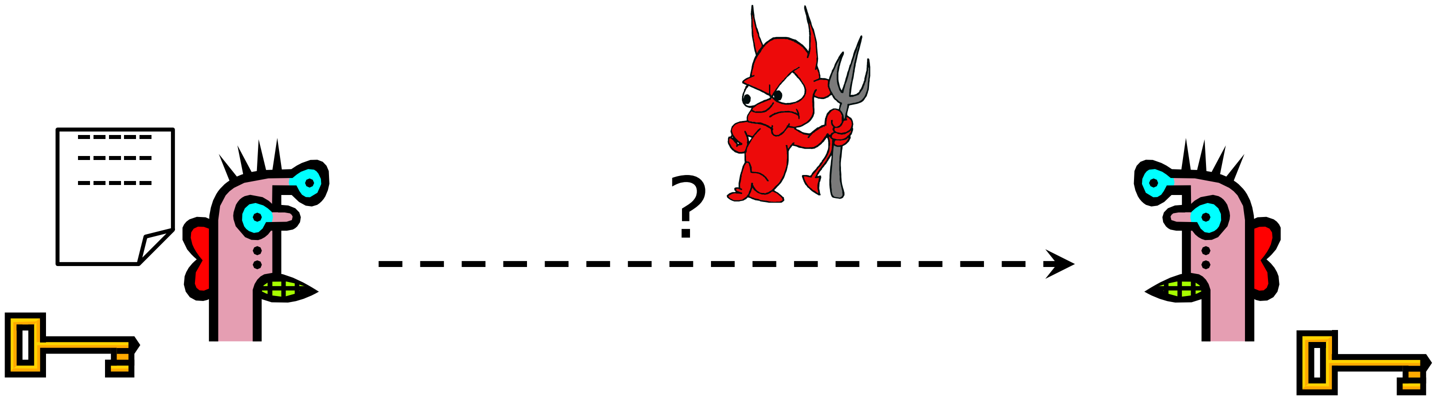
Stream Cipher

EJ Jung
ejung@cs.usfca.edu

Basic Problem



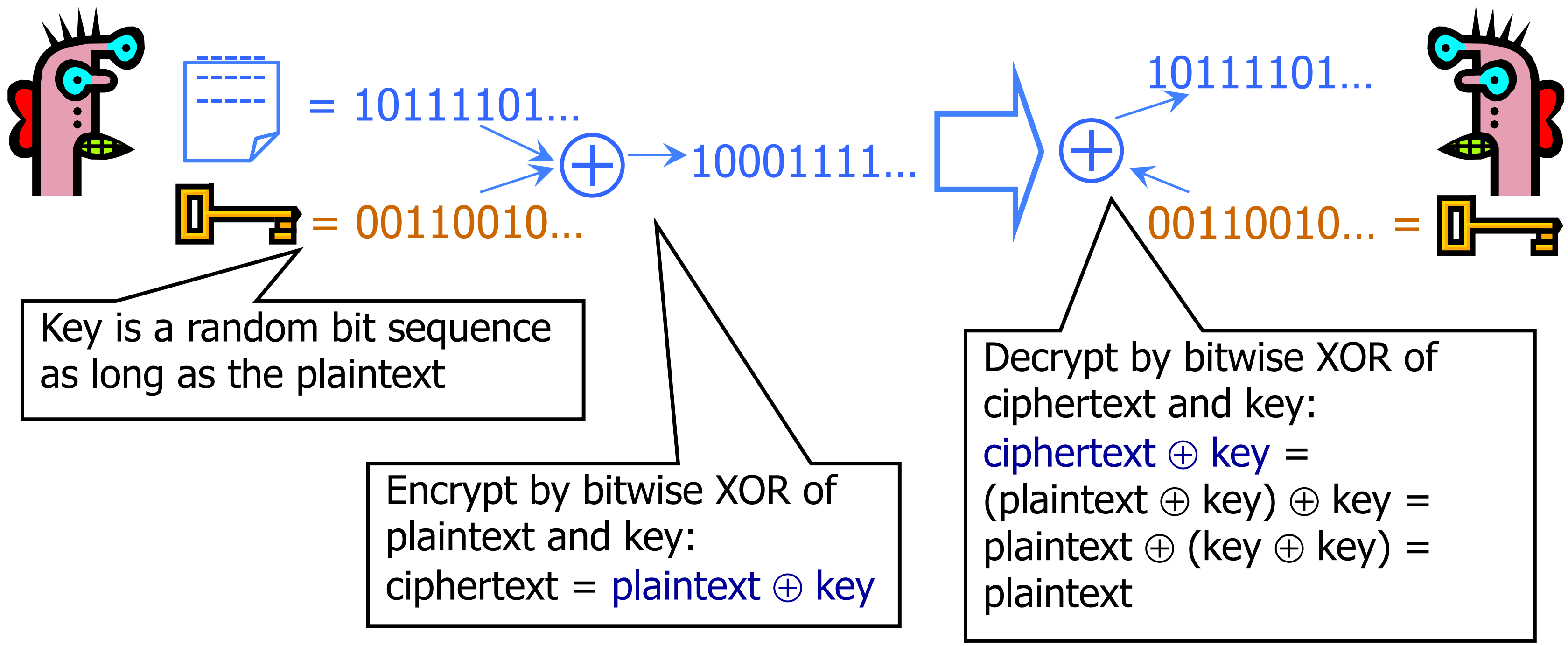
Given: both parties already know the same secret

Goal: send a message confidentially

How is this achieved in practice?

Any communication system that aims to guarantee confidentiality must solve this problem

One-Time Pad



Cipher achieves perfect secrecy if and only if
there are as many possible keys as possible plaintexts, and
every key is equally likely (Claude Shannon)

Advantages of One-Time Pad

- Easy to compute
  - Encryption and decryption are the same operation
  - Bitwise XOR is very cheap to compute
- As secure as theoretically possible
  - Given a ciphertext, all plaintexts are equally likely, regardless of attacker's computational resources
  - …as long as the key sequence is truly random
    - True randomness is expensive to obtain in large quantities
  - …as long as each key is same length as plaintext
    - But how does the sender communicate the key to receiver?

Problems with One-Time Pad

- Key must be as long as plaintext
  - Impractical in most realistic scenarios
  - Still used for diplomatic and intelligence traffic
- Does not guarantee integrity
  - One-time pad only guarantees confidentiality
  - Attacker cannot recover plaintext, but can easily change it to something else
- Insecure if keys are reused
  - Attacker can obtain XOR of plaintexts

Stream Ciphers

- One-time pad

  Ciphertext(Key,Message)=Message$\oplus$Key

  - Key must be a random bit sequence as long as message
- Idea: replace "random" with "pseudo-random"
  - Encrypt with pseudo-random number generator (PRNG)
  - PRNG takes a short, truly random secret seed (key) and expands it into a long "random-looking" sequence
    - E.g., 128-bit key into a 10^6-bit pseudo-random sequence
- Ciphertext(Key,Message)=Message$\oplus$PRNG(Key)
  - Message processed bit by bit, not in blocks

Properties of Stream Ciphers

- Usually very fast
  - Used where speed is important: WiFi, SSL, DVD
- Unlike one-time pad, stream ciphers do not provide perfect secrecy
  - Only as secure as the underlying PRNG
  - If used properly, can be as secure as block ciphers
- PRNG must be unpredictable
  - Given the stream of PRNG output (but not the seed!), it's hard to predict what the next bit will be
    - If PRNG(unknown seed)=$b_1 \ldots b_i$, then b_{i+1} is "0" with probability ½, "1" with probability ½

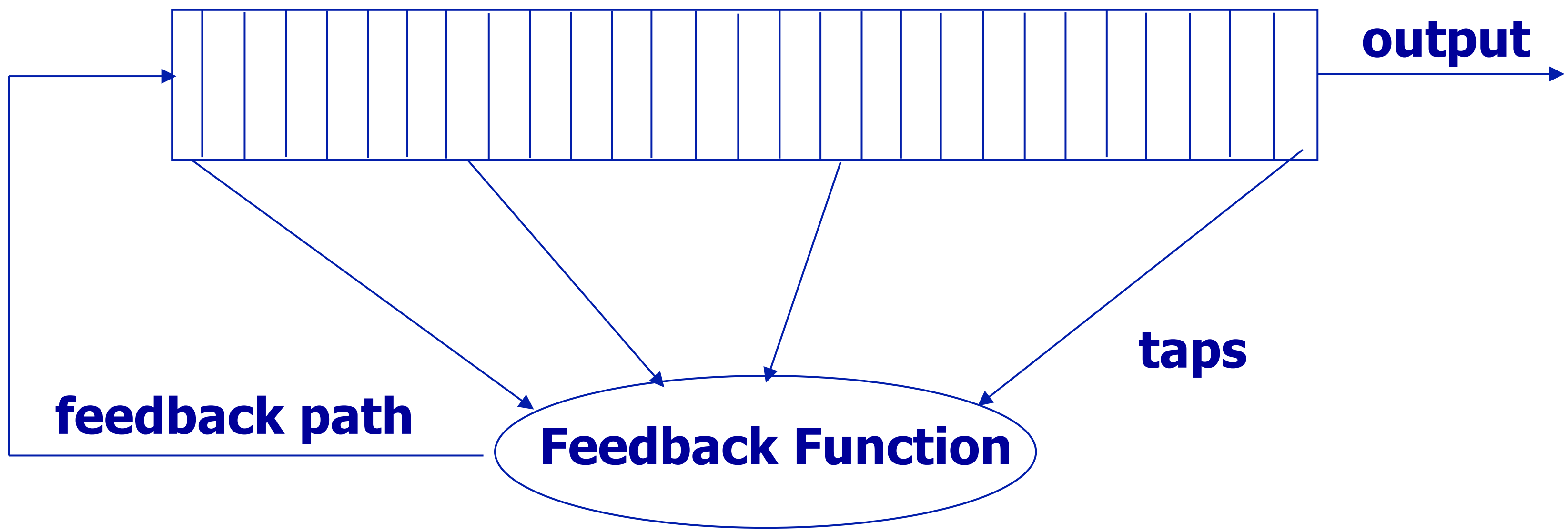
Weaknesses of Stream Ciphers

- No integrity
  - Associativity & commutativity: $(X \oplus Y) \oplus Z = (X \oplus Z) \oplus Y$
  - $(M_1 \oplus PRNG(key)) \oplus M_2 = (M_1 \oplus M_2) \oplus PRNG(key)$
- Known-plaintext attack is very dangerous if keystream is ever repeated
  - Self-cancellation property of XOR: $X \oplus X = 0$
  - $(M_1 \oplus PRNG(key)) \oplus (M_2 \oplus PRNG(key)) = M_1 \oplus M_2$
  - If attacker knows M_1, then easily recovers M_2
    - Most plaintexts contain enough redundancy that knowledge of M_1 or M_2 is not even necessary to recover both from $M_1 \oplus M_2$

Stream Cipher Terminology

- Seed of pseudo-random generator often consists of initialization vector (IV) and key
  - IV is usually sent with the ciphertext
  - The key is a secret known only to the sender and the recipient, not sent with the ciphertext
- The pseudo-random bit stream produced by PRNG (IV,key) is referred to as keystream
- Encrypt message by XORing with keystream
  - ciphertext = message $\oplus$ keystream

Generic LFSR



- The register is ***seeded*** with an initial value.
- **At each clock tick, the feedback function is evaluated using the input from the *tapped bits*. The result is shifted into the leftmost bit of the register. The rightmost bit is shifted into the output.**