Reading Assignment

• Please read chapter 9 of the course textbook by Schneier
Introduction

• So far the concept of 64-bit encryption has been introduced
• It turns out that a 64-bit ciphertext per 64-bit plaintext is problematic
• This lecture introduces a variety of encryption modes
Block Versus Stream Ciphers

• Block ciphers operate on blocks of plaintext or ciphertext, e.g., 64 bits at a time or 128 bits at a time

• Stream ciphers operate on as little as one bit at a time
  • May also consider one byte at a time or one word

• The vast majority of modern cryptography considers block ciphers

• Nevertheless, we will introduce some stream-based attempts later in this course
Notation

- $C_i$ is ciphertext message $i$
- $P_i$ is plaintext message $i$
- $E_k$ is encryption with key $k$
  - Note that $E$ could be symmetric or asymmetric
  - $E_k(P_i) = C_i$
- $D_k$ is decryption with key $k$
  - Note that $D$ could be symmetric or asymmetric
  - However, for asymmetric cryptographic, need distinct keys (a “key” may be a set of numbers, e.g., in RSA a “key” is a pair of numbers)
    - $E_{k1}$ and $D_{k2}$ where $k1$ is the public “key” and $k2$ is the private “key”
      - $E_{k1}(P_i) = C_i$
      - $D_{k2}(C_i) = P_i$
- $\{X\}$ is a set of elements of type $X$
- $\mid$ is “such that”; e.g., integer $i \mid 3 < i < 5$ implies that $i = 4$
Electronic Codebook (ECB) Mode

• One to one correspondence between plaintext and ciphertext
  • E.g., consider a message of 1280 bits broken up into 20 “blocks” each of 64 bits of plaintext
  • Each 64-bit $P_i$ is encrypted into a 64-bit $C_i = E_k(P_i)$

• Problem #1: codebook
  • Attacker can compile a codebook of known $P_i, C_i$ pairs without knowing the key
  • Over time and especially if the encrypted messages have significant redundancies, an attacker can glean a lot of information
    • Beginnings and endings of messages particularly vulnerable

• Problem #2: replay attack
  • Classic example: bank transactions
Classic Bank Example Replay Attack

- Attacker deposits $10 and then $100; only blocks 1 and 13 change
- Attacker deposits $10 again later; only block 1 changes; block 13 is $
- Now a variety of attacks on block 13 may commence...
Ciphertext Stealing Instead of Padding

Figure 9.1  Ciphertext stealing in ECB mode.
Cipher Block Chaining

- Use results of previous block encryption
- Typical use is based on exclusive-or (XOR)
  - For encryption where $i > 1$ (i.e., after the first block), $C_i = E_k(P_i \oplus C_{i-1})$
  - For decryption (except for the first block, i.e., $i \neq 1$), $P_i = C_{i-1} \oplus D_k(C_i)$

![Diagram of Cipher Block Chaining](image)

Figure 9.3  Cipher block chaining mode.

©Georgia Institute of Technology, 2018-2019
Initialization Vector

• Note that so far for the case where the first plaintext block is identical, the block will encrypt to the same ciphertext (assuming the same key)

• In fact, two identical messages will encrypt to the same ciphertext message

• Solution: use an *initialization vector (IV)*
  • E.g., timestamp
  • Note that after the first block, the attacker has all of the ciphertext blocks
  • Therefore using a plaintext IV does not provide the attacker with significant help
Figure 9.5  Ciphertext stealing in CBC mode.
Stream Ciphers

• We consider the case of converting plaintext to ciphertext one bit at a time
• One simple approach to a keystream generator is a one-time pad
Stream Ciphers

Figure 9.6 Stream cipher.
Weaknesses

• No diffusion or avalanche effect
• If the adversary has any correct (i.e., known) plaintext – ciphertext pairs, that portion of the keystream can be easily calculated
Keystream Generator Requires a Key

Figure 9.7 Inside a keystream generator.
Adding a Key

• Keystream generator output a function of a key
• Now users can update keys regularly
• Attack surface is reduced