Computer Security

Reagan Shirk

September 17, 2020

Announcements

- We'll have a quiz in the next class (9/22)
  - AES/DES, everything up through this lecture is fair game

AES and DES stuff

- The most important factors are:
  - key size
    - DES: 56/64 bits
      - Subkey is 48 bits. . . ?
    - AES: 128, 192, or 256 bits
  - input size
    - DES: 64 bits
    - AES: 128, 192, or 256 bits
  - output size
    - DES: 64 bits
    - AES: 128, 192, or 256 bits
  - number of rounds
    - DES: 16
    - AES: 10, 12, or 14

Feistel Cipher

- Divide
- Put right of input to left of output
- Scramble right side of input
- XOR left side with scrambled right side

AES

AES 128 Overview

- You don't need to remember permutation tables or anything- it's all public. Just need to remember the mechanism
  - Won't be tested over the math either, we just need to be able to analyze the system
- Does AES have the avalanche property?
  - Avalanche property: if key changes a little, the ciphertext changes significantly. If the plaintext changes a little, the ciphertext changes significantly

AES Assessment

- There are no known successful attacks on full AES
  - Best attacks work on 7-9 rounds
- AES-128 requires much more effort than DES for a brute-force attack

Attacks on AES

- Differenetial Cryptanalysis: based on the differences in inputs and how they correlate with differences in outputs
  - This is reduced due to high number of rounds
- Linear Cryptanalysis: based on correlations between input and output
- Side channel attacks
- Timing attacks: measure the time it takes to do operations
  - Some operations/operands are much faster than other operations with other operand values
  - provides clues about what internal operations are being performed and what internal data values are being produced
- Power attacks: measures power to do operations
  - Changing a single bit requires less power than changing a byte

Modes of Operation

- What does block cipher mean?
  - Encrypts plaintext in blocks

Processing with Block Ciphers

- How do we chain the ciphertext together when we encrypt in blocks?
- Modes of operation:
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - OFB ()

Issues for Block Chaining Mode

- Block chaining mode = the process of chaining blocks together I think, and can be any of the modes of operation listed above that I missed
- Ciphertext manipulation
  - Can the attacker modify the ciphertext blocks in a way that will produce a predictable/desired change in the decrypted plaintext blocks?
  - Assume the structure of the plaintext is known, like the first block is the salary of employee #1 and the second block is the salary of employee #2
- Information leakage
  - Does the ciphertext reveal info about the plaintext blocks?
- Parallel/Sequential
  - Can we encrypt all of the blocks simultaneously or do we need to encrypt one by one? What about decryption?
- Error propagation
  - If there is an error in a plaintext/ciphertext block, will there be an encryption/decryption error in more than one ciphertext/plaintext block?

- Fang said that making blocks independent and using different keys will make things “very interesting”
- He said something about picking our own pictures. . . ?
- We don’t need answers to these questions rn, we just need to keep them in mind during our analysis

Electronic Code Book

- Normally not used b/c it’s not good
- One of the modes of operations for block chaining
- If the total length is not a multiplier of 64, we can add padding
- We want to use the same key
- It looks like you split your message into 64 bit blocks, encrypt each block using the same key, and then you have ciphertext blocks that are 64 bits
- This is the easiest mode of operation because each block is independently encrypted
- Decryption is the reverse
- Don’t forget to ask:
  - Does information leak?
  - Can ciphertext be manipulated?
  - Is parallel processing possible?
  - Are there ciphertext errors?

Cipher Block Chaining (CBC)

- Chains all of the blocks
- You split your input into 64 bit blocks
  - You encrypt the blocks and XOR the output of the previous block with the input for the next block before the next encryption
    - You need an initialization vector for the first round XOR since you don’t have an output yet
  - The key is the same for all rounds
- Chaining dependency: each ciphertext block depends on *all of the preceding* plaintext blocks
- How many ciphertext blocks does each plaintext block depend on for decryption?
  - You need 2- the current and the previous
- Don’t forget to ask:
  - Does information leak?
    - Identical plaintext blocks will produce different ciphertext blocks
  - Can ciphertext be manipulated?
    - He literally has question marks on the slide
  - Is parallel processing possible?
    - No for encryption, yes for decryption
  - Are there ciphertext errors?
    - Yes for encryption, a little for decryption

Initialization Vectors

- Also written as IV
- Used alongside the key, not secret
- For a given plaintext, changing either the key or the IV will produce a different ciphertext. Why is this useful?