Computer Security

Reagan Shirk

September 10, 2020

Secrety Key Cryptography

- The basic technique is a product cipher
  - Applications of interleaved substitutions and permutations
- For secret key cryptography, the ciphertext is about as long as the plaintext
- You have stream ciphers: RC4
  - encrypting each character one by one
- You have block ciphers: DES, IDEA, AES
  - encrypting blocks of characters

Public Key Cryptography

- Invented in 1975
- There is a public/private key pair rather than just a private key
  - the public key can be publicly known (shock)
  - the private key is secret (shock)
- Slower but more secure
- Also known as asymmetric

Hash Algorithms (other name: Message Digests)

- Output is a fixed length
- Desirable properties of hash functions:
  - Performance: easy to compute $H(m)$
  - One-way property: If you're given $H(m)$, it's very difficult to figure out what m is
  - Weak collision free: Given $H(m)$, it's hard to find m' such that $H(m') = H(m)$
  - Strong collision free: Computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$

Summary

- Cryptography is a fundamental and carefully studied part of security
  - normally it isn't the weak link or the department with the budget cuts
- "Perfectly secure" ciphers are possible but defs too expensive in practice, not feasible
- The early ciphers (like caesar cipher, monoalphabetic cipher) are not nearly strong enough, I can break that shit so you know it's not secure because I'm not very smart
- Key distribution/management is hard af no matter what
- That's it for this topic lit

Topic 3.1: Some shit he changed the slide too fast

Key Sizes

- Keys should be chosen from a large set so we don't have to worry about brute force attacks
  - i.e. you want to have at least 4 digits in a passcode so that it's hard to guess
    - a one digit passcode has 9 possibilites, a 2 digit passcode as 100 possibilities, etc
  - If a key is 3 bits, how many possible keys are there? $2^3 = 8$
- An attacker can do a brute force search to find a key if they have a plaintext/ciphertext pair
  - brute force = trying each possibility one by one
  - if a key has n bits, how many possibilities would the attacker need to try to succeed with a brute force approach? 2^n
- What is a good key size?
  - For today: **128 bits**
  - In the 80's: 56 bits was aight
  - In the 70's: 40 bits was aignt
- **If computers increase in power by 40% per year, we need roughly 5 more key bits per decade to stay sufficiently hard to break**

Principles for Cipher Design

- Confusion
  - Make the relationship between . . . he changed slides we'll come back to this later
- Diffusion

Exploiting the Principles

- The idea is using multiple, alternating permutations and substitutions
  - This isn't any safer- doing multiple permutations is basically just doing one long permutation
    - i.e. +3, +2, -1 permutation is just a +4 permutation
  - Doing multiple substitutions doesn't really change any patterns
- Confusion is accomplished by substitution
- Diffusion is accomplished by permutation
  - I need to come back and get those definitions for above

Feistel Ciphers

- Probs going to be confusing if we haven't kept up with class so far
- Good to pay attention now for the midterm - this is going to be like 10 pts on the exam
- An influental template for designing a block cipher
- Major benefit: encryption and ecryption take the same amount of time and can be performed on the same hardware
  - Why do they take the same time? This would be impossible with a hash function. Didn't get the answer as to why. . . I'll look it up
- Examples: DES, RC5

One "Round" of Encryption

- You break the input block into two: the left and right halves

- Copy the right part to create the output half of the left part
- The right half of the input gets scrambled with a key to create a function f
- XOR the scramble result with the left half of the input to create the right half of the output block
- Which part is encrypted? The left half of the input block
- Why does encryption take the same time as decryption? You do this whole process backwards
- He has an equation written down that I'll type up later but I can't write LaTex that fast

Complete Feistel Cipher (Multiple Rounds of Encryption)

- You keep doing the above process as many times as you need to
- At the very end, you swap the two halves of the output block (both encryption and decryption)
- What does the final swap step do- how does it help?
  - No swap means you can't decrypt the message
- **Only need to swap for the final step- YOU DO NOT NEED TO SWAP EVERY ROUND**

Parameters

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- Scrambling function f

Summary

- Decryption is the same as encryption- you just have to do the process backwards
  - the reversibility is from the XOR operation
- The scrambling function f can be *anything*
  - hopefully easy to compute
  - no need to invert the function

DES (Data Encryption Standard)

- Standardized in 1976 by NBS
- Proposed by IBM, type of feistel cipher

Official Criteria

- Provides high level security
- Security has to reside in the key, not in the algorithm. The algorithm isn't secret
- Not patented
- Efficient to implement in hardware
- Slow to execute in software

Basicts

- Blocks: 64 bit for plaintext input and ciphertext output
- 16 rounds

- Key is 64 or 56 bits. Both are right. Why? Every 8th bit is a parity check, so even though there are 64 total bits only 56 of them matter

Exams

- Give a scenario, what kind of security requirements would be needed?
  - Designing a security scheme
  - Important for us to understand the applications of public/private key security