Computer Security

Reagan Shirk

October 15, 2020

Introduction to Cryptography

- CIA: Confidentiality, integrity, availability
  - Confidentiality: secret information. Info should only be known by those who are allowed to know
  - Integrity: Information should not be updated unless it is explicitly required by the owner of the information
  - Availability: information should be easily available to those who are allowed to access it
- Secret Key Cryptography: sender and receiver share a secret key
  - Examples:
    - Caesar cipher
    - Mono-alphabetic cipher
    - Vigenere Cipher
    - Hill cipher
    - Permutation cipher
    - One-time pad
    - DES
    - AES
    - IDEA
    - RC4
  - Secret key cryptography = symmetric cryptography
  - The basic technique is multiple applications of interweaved permutations and substitutions
  - The produced ciphertext is about as long as the plaintext
  - Applications:
    - Transmitting over an insecure channel
    - Authentication
    - Integrity
  - Problems:
    - How do you share the key?
    - Easy to attack
    - Catch 22: Need a key to secure the message but need a secure channel to share the key
- Public Key Cryptography: slower but more secure than secret key
  - Invented in 1975
  - There is a public and a private key
  - Known as asymmetric cryptography
  - Applications:
    - Data transmission
    - Storage
    - Authentication
    - Digital signatures
    - Key exchange

DES

- DES (Data Encryption Standard) is a feistel cipher
- Proposed by IBM, standardized in 1976 by NBS

- No longer good enough
- Official Criteria:
  - Provide a high level of security
  - Security must reside in the key, not the algorithm (algorithm is public)
  - Not patended
  - Efficient to implement **in hardware**
  - Slow to execute **in software**
- Basics
  - Input text and output text are 64 bits in length
  - 16 rounds
  - Key is 56 bits
    - In total it is 64, but every 8^{th} bit is a parity bit so we say it's 56 bits
  - Subkey is 48 bits
- High-Level Overview
  - You have an initial permutation that the 64-bit input runs through
  - The 56-bit key produces multiple 48-bit keys
  - The 48-bit keys are used to encrypt - each round has its own key
  - After the final round, the two halves are swaped
  - The final permutation is performed
  - The 64-bit output is produced
- Initial permutation:
  - The initial permutation is fully specified and independent of the key
    - This means it doesn't improve security... so why do we need it?
      - I missed the answer to this in class, but Google says that it makes the hardware implementation easier. It can accumulate the bits into eight shift registers which is more efficient than a single 64-bit register. This process "naturally" performs the initial permutation of DES.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- Final permutation:
  - The final permutation is the inverse of the initial permutation
  - You make the same table above but the numbers represent the output bits instead of the input bits
  - The final permutation is needed to make DES a Feistel cipher (i.e. decryption is the inverse of encryption)
- Sub-Key Generation: First permutation
  - First step is to through out the 8 parity bits (8, 16, 24, ...) and permutate the resulting 56 bits (in table below)
  - Processing per round:
    - On rounds 1, 2, 9, 16: left circular shift 1 bit
    - On other rounds, left circular shift 2 bits
    - Split the entire 56 bit key into two sets of 28 bits and do the shift
      - Perform the permutation/discarding to create the key for the round
      - Take the two sets of shifted bits and use them as the starting place for the next round

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Modes of Block Cipher Operations

- ECB - electronic code block
- CBC - cipher block chaining
- OFB - output feedback
- CFB - cipher feedback
- CTR - counter

Double/Triple DES

- Understand how it works

Meet in the Middle Attack

Hash Functions and Applications

- Hash function has no key
- Known as:
  - Message digests
  - One-way transformations
  - One-way functions
  - Hash functions
- $H(m)$ is shorter than m
- Usually fixed lengths (128-160 bits)
- The primary application is for generating/verifying digital signatures
- Desirable properties:
  - Easy to compute $H(m)$ (performance)
  - If all you have is $H(m)$, it is computationally infeasible to compute m (one-way)
  - If all you have is $H(m)$, it is computationally infeasible to find m' such that $H(m) = H(m')$ (weak collision free)
  - It is computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$ (strong collision free)

Applications

- File Authentication:
  - Creating a file F and storing it's hash, $H(F)$. To check if the file was changed, take it's hash again. If the hashes are not equal, the file was changed.
- User Authentication:

  - Receiver picks a random number R. The sender calculates $Y = H(R|K)$ and the receiver comfirms
- Committment Protocols:
  - I don't entirely understand the practical applications of this so if someone could explain something other than the given example that would be great
  - Sender computes $Z = H(x)$. Receiver picks and sends Y, sender sends X, receiver verifies $H(x) = Z$
- Message encryption:
  - Sender and receiver share a secret key K. Sender sends random, encrypted number R_1 and receiver sends random, encrypted number R_2. $R_1|R_2$ is used like an IV for an OFB Mode, but the encryption is a concatentation of the key and hash
    * The concatenation is only used for the first round, afterwards it's just the hash.
- Message Integrity:
  - Goal is to authenticate without encryption. The sender wants to authenticate M, shares the secret key K with the receiver. Pick a random number R and compute $Y = H(M|K|R)$. The reciever is sent M, R, Y and confirms that $Y = H(M|R|K)$
- Digital Signatures:
  - Hash a message and encrypt the hash to create a digital signature. Encryption is done with a private key. To verify the signature, you take the hash and signature and do something with the public key? Then you can determine if the signature is valid.

MD5

SHA