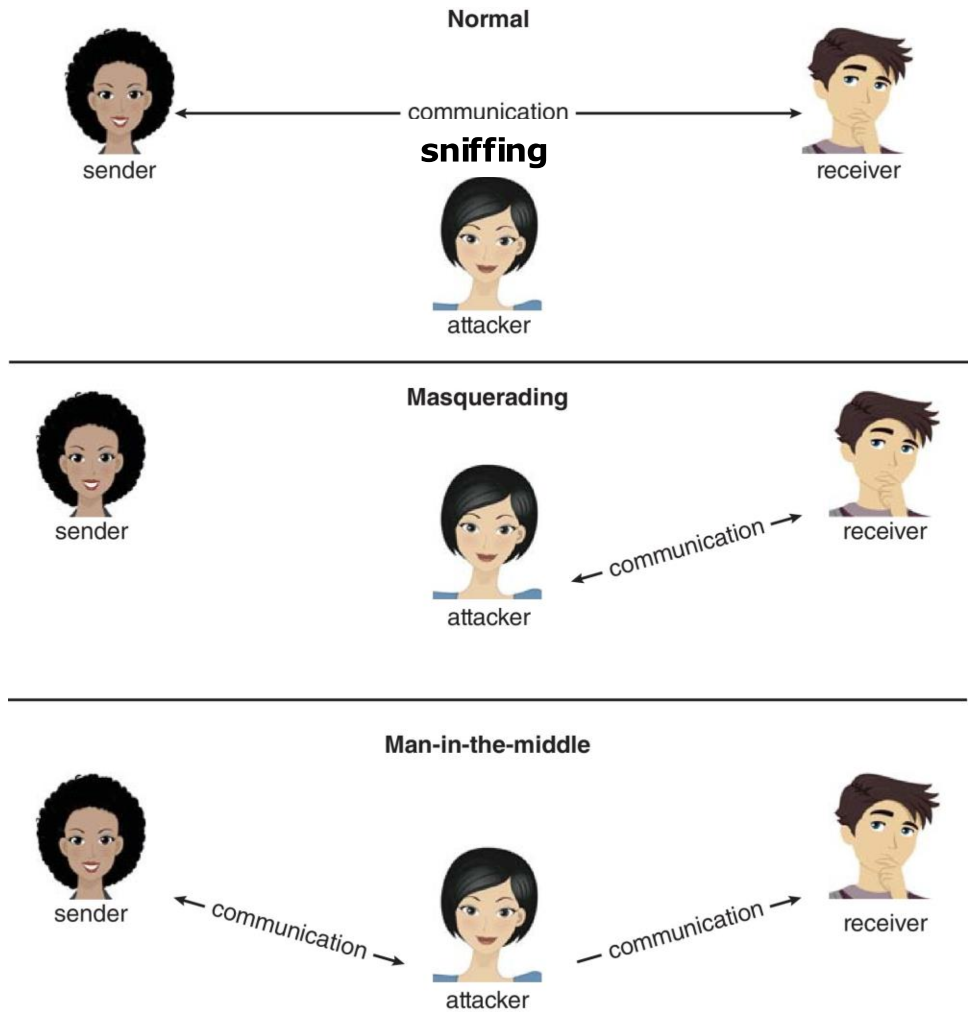
Recap

- VMM
  - System call
  - Memory virtualization
- Container
- Basic of Security: CIA

System and Network Threats



System and Network Threats (Cont.)

Denial of Service

- Overload the targeted computer preventing it from doing any useful work
- **Distributed Denial-of-Service** (**DDoS**) come from multiple sites at once
- Consider the TCP-connection handshake
  - How many connections can the OS handle?
- Consider traffic to a web site
  - How can you tell the difference between being a target and being really popular?

Port scanning

- Automated attempt to connect to a range of ports on one or a range of IP addresses
- Detection of running services in order to identify vulnerabilities
- Detection of OS and version running on system

Basic of Cryptography

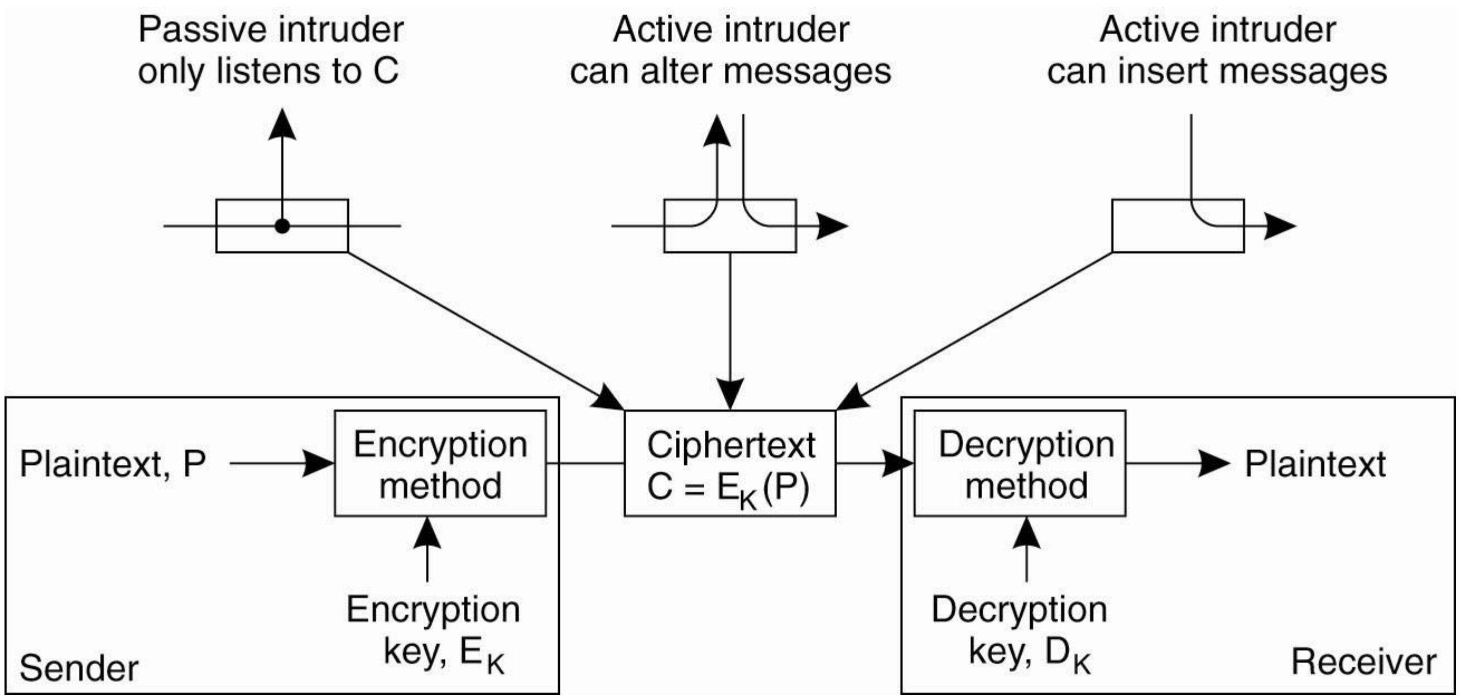
Cryptography

Goal: keep information from those who aren't supposed to see it.

- Do this by encrypting the data
- Encryption constrains the set of possible receivers of a message
- Algorithms have two inputs: data and key(s)
- Some keys must be kept secret

A good encryption algorithm should never depend on the secrecy of its implementation (assume attackers know the details of the algorithm), only the keys are secret.

Basics of Cryptography



- plaintext: unencrypted message
- ciphertext: encrypted form of message

Cryptosystems

Cryptosystems are either symmetric or asymmetric

Symmetric system: $E_k = D_k$, so the key must be kept secret

Asymmetric system (aka **public-key system**): $E_k \neq D_k$, E_k can be made public; D_k is secret and can't easily be derived from E_k

- E_k is a **public key**
- D_k is a **private key**

Symmetric Encryption Algorithms

Same key is used to encrypt and decrypt

- Therefore key *k* must be kept secret

Data Encryption Standard (**DES**) was most commonly used symmetric block-encryption algorithm (created by US Government)

- 56-bit keys
- Encrypts a block of data at a time, 64-bit block size
- Keys too short so now considered insecure

In 2001 NIST adopted a new block cipher - Advanced Encryption Standard (**AES**)

- Keys of 128, 192, or 256 bits, works on 128-bit blocks
- A machine that could crack 56-bit DES in one second would take 149 trillion years to crack a 128-bit AES key!

Asymmetric Encryption

- **Public-key encryption** based on each user having two keys:
  - **public key** – published key used to encrypt data
  - **private key** – key known only to individual user used to decrypt data
- Most common is **RSA** block cipher
  - No efficient algorithm is known for finding the prime factors of a number

Authentication

- Question: how does the receiver know that remote communicating entity is who it is claimed to be?

Authentication Protocol (AP)

AP 1.0

- Alice to Bob: “I am Alice”
- Problem: intruder “Trudy” can also send such a message

AP 2.0

- Authenticate source IP address is from Alice’s machine
- Problem: IP Spoofing (send IP packets with a false address)

AP 3.0: use a secret password

- Alice to Bob: “I am Alice, here is my password” (e.g., telnet)
- Problem: Trudy can intercept Alice’s password by sniffing packets

Authentication Protocol

AP 3.1: encrypt the password

- Use a symmetric key known to Alice and Bob

    A to B: “I am A”, and A’s encrypted password

    B: if decrypted password is correct

        then A is verified

        else A is fraudulent

- Failure scenario: playback attack
    - Trudy can intercept Alice’s message and masquerade as Alice at a later time

Authentication Using Nonces

- Problem with AP 3.1: same password is used for all sessions
- **Solution:** pick a "once-in-a-lifetime" number (nonce) for each session

AP 4.0

- A to B: msg1 = "I am A" /* note: unencrypted message! */
- B to A: once-in-a-lifetime value, n
- A to B: msg2 = encrypt(n) /* use symmetric keys */
- B computes: if decrypt(msg2)==n

  then A is verified

  else A is fraudulent

Authentication Using Public Keys

AP 4.0 uses symmetric keys for authentication

Question: can we use public keys?

Symmetricity in public key crypto: DA(EA(n)) = EA(DA(n))

- DA using private key of A, EA using public key of A

AP 5.0

- A to B: msg = "I am A"
- B to A: once-in-a-lifetime value, n
- A to B: msg2 = DA(n)
- B computes: *if EA(DA(n))== n*
  then A is verified
  else A is fraudulent

Problems with AP 5.0

Bob needs Alice's public key for authentication

- Trudy can impersonate Alice to Bob
  - Trudy to Bob: msg1 = "I am Alice"
  - Bob to Alice: nonce n (Trudy intercepts this message)
  - Trudy to Bob: msg2= DT(n) where DT uses its (Trudy's) private key
  - Bob to Alice: send me your public key (Trudy intercepts)
  - Trudy to Bob: send Trudy's public key (claiming it is Alice's)
  - Bob: verify ET(DT(n)) == n and authenticates Trudy as Alice!!

AP 5.0 is only as "secure" as public key distribution!

PKI – Public Key Infrastructure is to solve the problem. Trusted CA (certificate authority) certificate public keys for others.

Digital Signatures Using Public Keys

Goals of digital signatures:

- Sender cannot repudiate message ("I never sent that")
- Receiver cannot fake a received message

Suppose A wants B to "sign" a message M:

- B sends M and *DB(M)* to A, where DB is decryption with B's private key
- A checks if *EB(DB(M)) == M*, where EB is encryption with B's public key

    If yes, then B has signed M