Security properties

- Confidentiality (C)
  - Data secrecy: Keep data secret from unauthorized subjects
  - Privacy: Individuals control who can access their information
- Integrity (I)
  - Aka “authenticity”
  - Data integrity: Keep data from being modified
  - System integrity: Keep systems functioning as intended
- Availability (A)
  - Keep the system running and responsive to legitimate clients

cryptography

- Authentication
  - verify that sender sent the message + not changed (digital signature)
- Integrity
  - message has not been modified
- Non-repudiation
  - sender cannot deny that she indeed sent the message

functions

- One-way hash functions h= H(M): no key
  - for integrity
  - Plaintext -> fingerprint
  - Brute force attack -> choose a large output size 128-160
  - Take a variable-length input *M* and produce fixed-length output
  - Public
  - weak collision resistance: Given *M* it is very hard to find M ′ such that H(M)=H(M′)
    - cheating after send: once the message is sent, sender cannot change it and claim they sent a different message
  - strong collision resistance: hard to find two random messages M_1 and M_2 such that H(M_1) = H(M_2)
    - cheating before send: sender cannot prepare two messages that have the same hash, send one and claim they have sent the other

- Symmetric crypto: one key
  - for confidentiality
  - only Alice and Bob know the same *shared key*, sender will encrypt with that key and receiver will decrypt
  - Plaintext -> ciphertext
  - Substitution
    - Monoalphabetic – each character is replaced with another character
    - Homophonic – each character is replaced with a character chosen randomly from a subset

- Polygram – each sequence of characters of length *n* is replaced with another sequence of characters of length *n*
- Polyalphabetic – many monoalphabetic ciphers are used sequentially, stream ciphers: a bit or a byte at a time
    - Assume XOR (weak but fast) with the key (long and random)
    - A xor K = B
    - A xor B = K
- One-time pad - Polyalphabetic cipher with **infinite** key, **randomly** generated, perfectly synchronized
- Block ciphers: polygram, block size = key size
    - Electronic Code Book (ECB) No dependency, do in parallel
        - Mallory can detect mapping, replay, fabricate, add/drop/replace and not be detected
    - Cipher Block Chaining (CBC) Dependency on earlier blocks
        - same plaintext blocks will encrypt to different ciphertext blocks, encryption/decryption cannot be parallelized
        - Replay is still possible if we don't use timestamps
        - Initialization vector (IV)

- Transposition(shuffling)

- Asymmetric crypto: two keys
  $D_{K2}(E_{K1}(M)) = M$
    - for confidentiality and authenticity
    - Alice has *public key* and *private key*
    - Everyone knows Alice's public key but only Alice knows her private key
    - Plaintext -> ciphertext
    - Confidentiality, A use B's public key to encrypt, B use its private key to decrypt
    - Authenticity, A use A's private key to sign, anyone use A's public key to verify
    - Functionality is greater but much slower
    - RSA
        - modular exponentiation in Galois Field GF(n) is efficient
        - factor large number is hard for attackers
    - digital signature EprivA(H(M)) or EprivA(M), provide non-repudiation

Attacks for symmetric/asymmetric crypto

- Cyphertext-only attack: gather and analyze enough **ciphertext** to learn decryption key, recognize the plaintext
- known-plaintext attack: observe many **ciphertexts** for **known plaintexts** to learn decryption key
- chosen-plaintext attack: feed chosen messages M into encryption algorithm and look at resulting ciphertexts C. Learn decryption key/messages M that produce C

- Man-in-the-middle attack: substitute, modify, drop, replay messages
- Brute-force attack: caught a ciphertext and try every possible key to decrypt -> choosing a large keyspace

Shared Key Exchange

- Diffie-Hellman
  - A sends g^a mod n
  - B sends g^b mod n
  - Shared key is $\boldsymbol{g^{ab} \bmod n}$
  - Hard to guess a and b with a large n
  - Man in the middle, no authentication
- KDC (Using a trusted third party)
  - Secrets should never be sent in clear, preconfigured keys $K_{KDC,C}$ and $K_{KDC,S}$
  - Use nonces to prevent replay attacks
  - Challenge response 3 flavors: check if S knows the same shared key Kc,s
  - We should prevent reuse of old keys -> tickets expire after some time, **validity period** in ticket
- Kerberos
  - Authentication server (AS) authenticates users, issues a ticket for clients to talk to TGS
  - Ticket Granting Server (TGS) Issue tickets for clients to talk to servers
  - **TGS+AS = KDC**
  - Each ticket has a validity period: timestamp and lifetime
  - Each service request (client to server) has an authenticator (nonce): timestamp + client identity, encrypted with a session key
- public key cryptography
  - A sends $EPub_B(K_{AB})$ to B, much slower

Public Key Exchange

- Need a trusted third party – Trent, aka *Certificate Authority (CA)*
- Everyone knows Trent's public key
- Trent signs Public-key Certificate: E privateKeyTrent (public key, Id(dns name))
- Certificate-Based Key Exchange
- Recovery from Stolen Private Keys

Password Authentication

- Dictionary attack – guessing passwords
  - Offline: **comparing** guesses to a list of precomputed hashes of **popular** pass. -> use a random number salt with a password in hashing
  - Online: trying popular passwords **manually** on server UI -> hard because password table is stored in the file /etc/shadow, only accessible by superuser
- Personal information attack

- Online: Try to guess a specific user's password by using **personal information** about the user manually on server UI
- Reuse attack
  - Steal a password from one server, try it at other servers

- Lamport hash or S-KEY – time-varying password
  - Someone **sniffing** on the network can learn the password if it is transmitted in clear
    - Encrypt communication
    - If **replay** attacks are possible, someone can still steal passwords and reuse them
  - Host sets password $x_0 = h(R)$, $x_1 = h(h(R))$, $x_2 = h(h(h(R)))$,..., x_{100}
  - User log on with x100, x99, …
- Shared-Key Authentication challenge response
- Public-Key Authentication
- Single Sign-On SSO
- One Identity Provider
  - Various systems share trust with **one** identity provider
  - Identity provider returns an encrypted token to the system, using a shared key
- Multiple Identity Providers
  - Various systems share trust with **all** identity providers
  - System decides which identity provider (IdP) to use, redirects user to identity provider
- Cookie-based Authentication
  - Placed into browser cache by servers
  - persistent sign in, shopping cart, user preferences
- Token-based Authentication
- Biometric Authentication

Access Control

- Discretionary (owner grant access)
  - Access Control Matrix
  - Access Control Lists (ACLs) For each object
  - Capabilities For each principal
- Mandatory (by system, government/army)
  - Each object has a **classification** and each subject has a **clearance**
  - **top secret > secret** > confidential > **restricted** > unclassified
  - **no read-up**
  - **no append-down**
  - Append is allowed on objects of the **same or higher** classification
  - Write is allowed **only** on objects of the **same** classification
  - Trusted subjects – the "no write-down" rule does not apply to them

- Bell-LaPadula Policy Model

    - Combine mandatory and discretionary
    - Intersection of Access Control Matrix and level check
- Role-based
    - depends on one's role in the organization
    - Maps to organization structure(companies)
- Attribute-based
    - depends on *attributes* assigned to user and object, and on *environment attributes*