Recovery

- Techniques thus far allow failure handling
- Recovery: operations that must be performed after a failure to recover to a correct state
- Techniques:
  - Checkpointing:
    - Periodically checkpoint state
    - Upon a crash roll back to a previous checkpoint with a consistent state

Independent Checkpointing

- Each processes periodically checkpoints independently of other processes
- Upon a failure, work backwards to locate a consistent cut
- Problem: if most recent checkpoints form inconsistent cut, will need to keep rolling back until a consistent cut is found
- Cascading rollbacks can lead to a domino effect.
Coordinated Checkpointing

- Take a distributed snapshot [discussed in Lec 11]

- Upon a failure, roll back to the latest snapshot
  - All processes restart from the latest snapshot

Message Logging

- Checkpointing is expensive
  - All processes restart from previous consistent cut
  - Taking a snapshot is expensive
  - Infrequent snapshots => all computations after previous snapshot will need to be redone [wasteful]

- Combine checkpointing (expensive) with message logging (cheap)
  - Take infrequent checkpoints
  - Log all messages between checkpoints to local stable storage
  - To recover: simply replay messages from previous checkpoint
    - Avoids recomputations from previous checkpoint
Recovery Oriented Computing

- Cheaper to optimize for recover than to design the system to prevent faults
- Need to restart the system upon failure

- Naive case: reboot
- Reboot part of the system: modular system, where components can be restarted independently
  - Unix /etc/rc service
- Stateful recovery
  - Database recovery
  - Use of checkpointing

Security in Distributed Systems

- Introduction
- Cryptography
- Authentication
- Key exchange
- Readings: Tannenbaum, chapter 9
  Ross/Kurose, Ch 7
Intruder may
• eavesdrop
• remove, modify, and/or insert messages
• read and playback messages
• Security threats
  – Interception, Interruption, Modification, Fabrication

Important issues:
• *Encryption/ cryptography*: secrecy of info being transmitted
• *authentication*: proving who you are and having correspondent prove his/her/its identity
• *Authorization*: verify you have rights to perform requested action
• *Auditing*: log actions and do post-facto analysis (forensics)
Security in Computer

**User resources:**
- login passwords often transmitted unencrypted in TCP packets between applications (e.g., telnet, ftp)

**Security Issues**

**Network resources:**
- often completely unprotected from intruder eavesdropping, injection of false messages
- mail spoofs, router updates, ICMP messages, network management messages

**Bottom line:**
- intruder attaching his/her machine (access to OS code, root privileges) onto network can override many system-provided security measures
- users must take a more active role
Encryption

plaintext: unencrypted message
ciphertext: encrypted form of message

Intruder may
• intercept ciphertext transmission
• intercept plaintext/ciphertext pairs
• obtain encryption decryption algorithms

A simple encryption algorithm

Substitution cipher:

abcdefghijklmnopqrstuvwxyz

poiuytrewqasdfghjklmnbcxz
• replace each plaintext character in message with matching ciphertext character:

plaintext: Charlotte, my dear
ciphertext: iepksgmmmy, dz uypk
Encryption Algo (contd)

- key is pairing between plaintext characters and ciphertext characters
- **symmetric key**: sender and receiver use same key
- 26! (approx $10^{26}$) different possible keys: unlikely to be broken by random trials
- substitution cipher subject to decryption using observed frequency of letters
  - 'e' most common letter, 'the' most common word

DES: Data Encryption Standard

- encrypts data in 64-bit chunks
- encryption/decryption algorithm is a published standard
  - everyone knows how to do it
- substitution cipher over 64-bit chunks: 56-bit key determines which of 56! substitution ciphers used
  - substitution: 19 stages of transformations, 16 involving functions of key
- Replacements: DES3 and now AES
Symmetric Cryptosystems: DES (1)

- The principle of DES
- Outline of one encryption round

Symmetric Cryptosystems: DES (2)

• Details of per-round key generation in DES.
**Key Distribution Problem**

**Problem:** how do communicant agree on symmetric key?
- N communicants implies N keys

**Trusted agent distribution:**
- keys distributed by centralized trusted agent
- any communicant need only know key to communicate with trusted agent
- for communication between i and j, trusted agent will provide a key

---

**Key Distribution**

We will cover in more detail shortly
Public Key Cryptography

- separate encryption/decryption keys
  - receiver makes *known* (!) its encryption key
  - receiver keeps its decryption key secret
- to send to receiver B, encrypt message M using B's publicly available key, EB
  - send EB(M)
- to decrypt, B applies its private decrypt key DB to receiver message:
  - computing DB( EB(M) ) gives M

Knowing encryption key does not help with decryption; decryption is a non-trivial inverse of encryption
- only receiver can decrypt message

**Question:** good encryption/decryption algorithms
RSA: public key encryption/

RSA: a public key algorithm for encrypting/decrypting
Entity wanting to receive encrypted messages:

- choose two prime numbers, \( p, q \) greater than \( 10^{100} \)
- compute \( n=pq \) and \( z=(p-1)(q-1) \)
- choose number \( d \) which has no common factors with \( z \)
- compute \( e \) such that \( ed = 1 \mod z \), i.e.,
  \[
  \text{integer-remainder}( \frac{ed}{(p-1)(q-1)}) = 1, \text{ i.e.,} \]
  \[
  ed = k(p-1)(q-1) + 1
  \]
- three numbers:
  - \( e, n \) made public
  - \( d \) kept secret

RSA (continued)

to encrypt:
- divide message into blocks, \( \{b_i\} \) of size \( j: 2^j < n \)
- encrypt: \( \text{encrypt}(b_i) = b_i^e \mod n \)

to decrypt:
- \( b_i = \text{encrypt}(b_i)^d \)

to break RSA:
- need to know \( p, q \), given \( pq=n \), \( n \) known
- factoring 200 digit \( n \) into primes takes 4 billion years using known methods
RSA example

- choose $p=3$, $q=11$, gives $n=33$, $(p-1)(q-1) = z = 20$
- choose $d = 7$ since 7 and 20 have no common factors
- compute $e = 3$, so that $ed = k(p-1)(q-1)+1$
  (note: $k=1$ here)

Further notes on RSA

why does RSA work?
- crucial number theory result: if $p$, $q$ prime then
  $b_i^{((p-1)(q-1))} \mod pq = 1$
- using mod $pq$ arithmetic:
  $(b^e)^d = b^{ed}$
  
  $= b^{k(p-1)(q-1)+1}$ for some $k$
  $= b \ b^{(p-1)(q-1)} \ b^{(p-1)(q-1)} \ ... \ b^{(p-1)(q-1)}$
  $= b \ 1 \ 1 \ ... \ 1$

  $= b$
  
  **Note:** we can also encrypt with $d$ and encrypt with $e$.
- this will be useful shortly
How to break RSA?

Brute force: get B's public key
- for each possible \( b_i \) in plaintext, compute \( b_i^e \)
- for each observed \( b_i^e \), we then know \( b_i \)
- moral: choose size of \( b_i \) "big enough"

Breaking RSA

man-in-the-middle: intercept keys, spoof identity:

1: get \( EB \)
2: return my \( EI \)
3: intercept \( b^{**}El \)
   compute \( b = DI (El(b)) \)
   send \( b^{**}EB \)