Basic Concepts of Reliability and Security

EECE 571R – Lecture 2

Karthik Pattabiraman

Outline

- Reliability Basics
- Security Basics

Learning Objectives

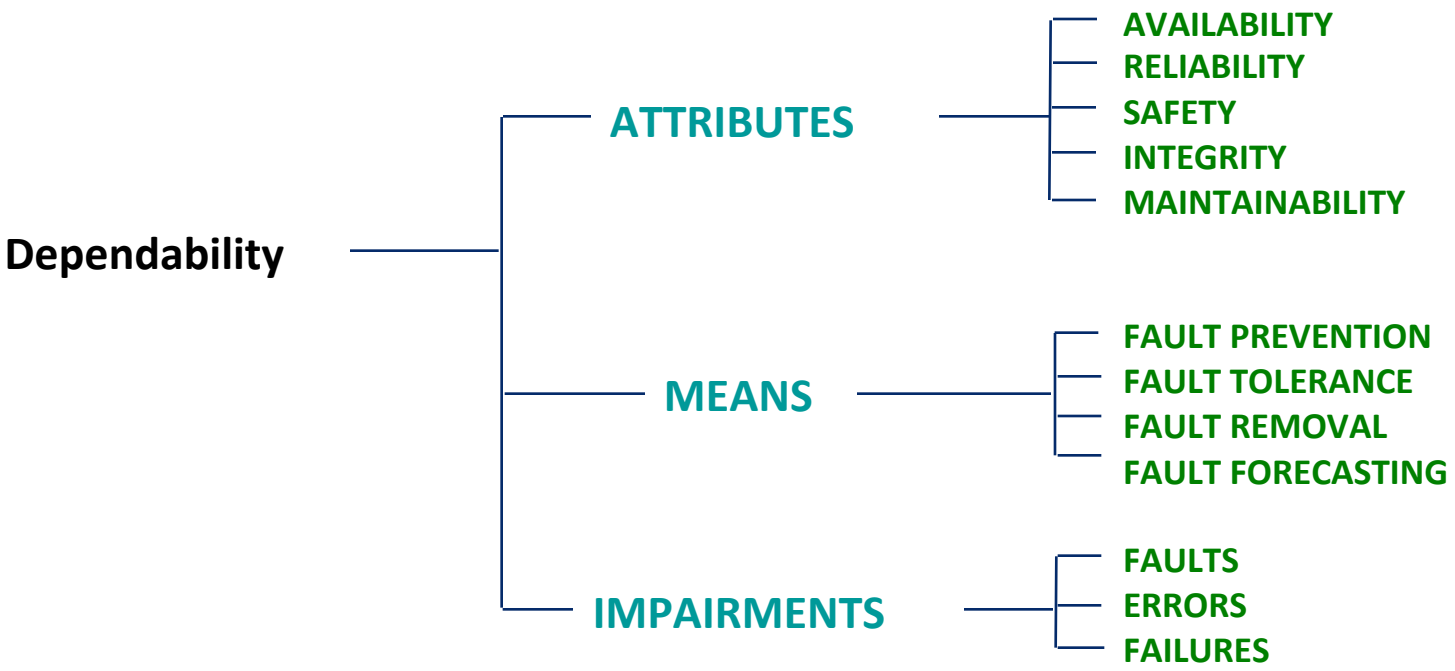
- At the end of this lecture, you should be able to:
  - **Define** fault-tolerance terms such as reliability, availability, safety and distinguish between them
  - **Identify** faults, errors and failures based on system descriptions and scenarios
  - **Classify** faults, errors and failures into various types
  - **Categorize** fault-tolerance techniques based on which phase they are applied
  - **Apply** common fault-tolerance strategies to problem scenarios and systems

What is dependability ?

- IFIP WG 10.4 on dependability
  - *[..] the trustworthiness of a computing system which allows reliance to be **justifiably** placed on the service it delivers*

- Incorporates the following notions:
  - Availability, Reliability, Maintainability (traditional)
  - Safety, security and Integrity (modern)

Dependability: Attributes, Means and Impairments



Dependability Attributes

- **Availability:** Readiness for correct service
- **Reliability**: Continuity for correct service
- **Safety:** Absence of catastrophic consequences
- **Integrity:** Absence of improper modifications
- **Maintainability:** ability to undergo modifications and repairs

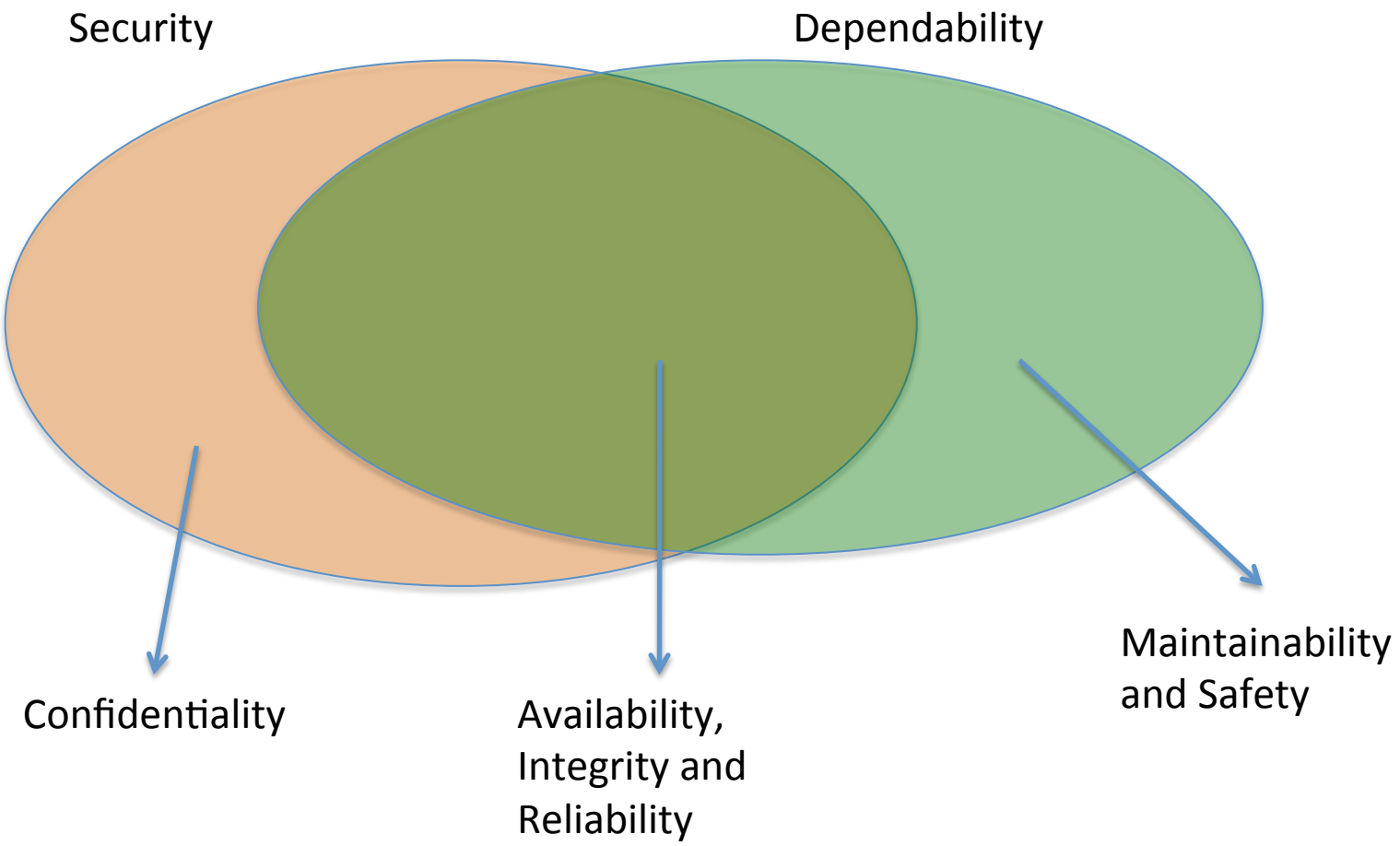
Each system does NOT satisfy one of the properties. Identify which property

- A database system fails every 1 minute, but recovers in 0.1 micro-seconds. The recovery is guaranteed to occur after every failure.
- A web server has downtime of 1 month a year, but it goes down at the same month every year and is down for the entire month
- A missile system has an expected downtime of 1 minute a year, and will hit its target with 99.999% certainity. However, occasionally it may backfire and hit an object close to the missile launcher itself.
- A nuclear power plant system will lock down whenever any improper change is made to it but will allow the change. Once locked down, it needs the sysadmin to initiate a complex control sequence to bring it up again. This operation may take anywhere from few hours to a few days.
- A computer system on the stock trading floor has only 3 minutes of downtime a year, and is always up during critical operations. It also prevents any modifications to it (including code updates) unless three different operators coordinate to simultaneously apply them.

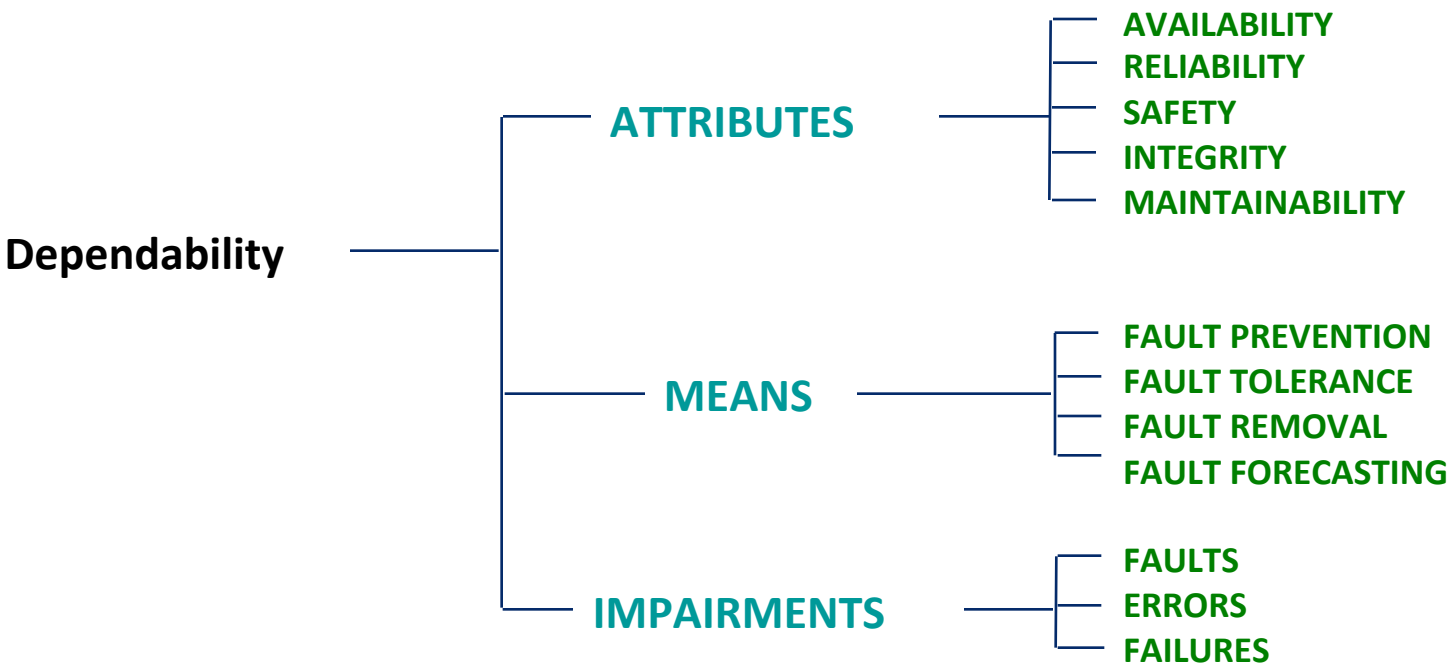
Solution

Type of system	Availability	Reliability	Safety	Integrity	Maintainability
Scenario 1		X			
Scenario 2	X				
Scenario 3			X		
Scenario 4				X	
Scenario 5					X

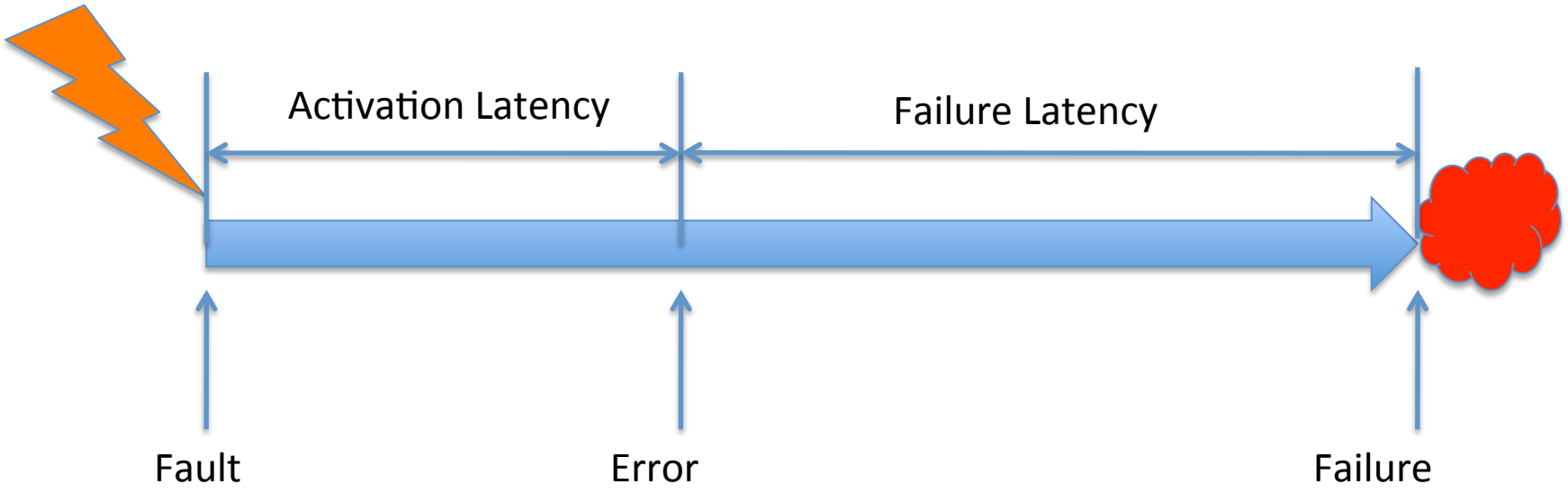
Dependability and Security



Dependability: Attributes, Means and Impairments



Dependability Impairments



Fault – Defect in the system (e.g., soft error)

Error – Deviation of system behavior from fault-free run

Failure – Violation of system's specification (e.g., crash)

Identify fault, error and failure

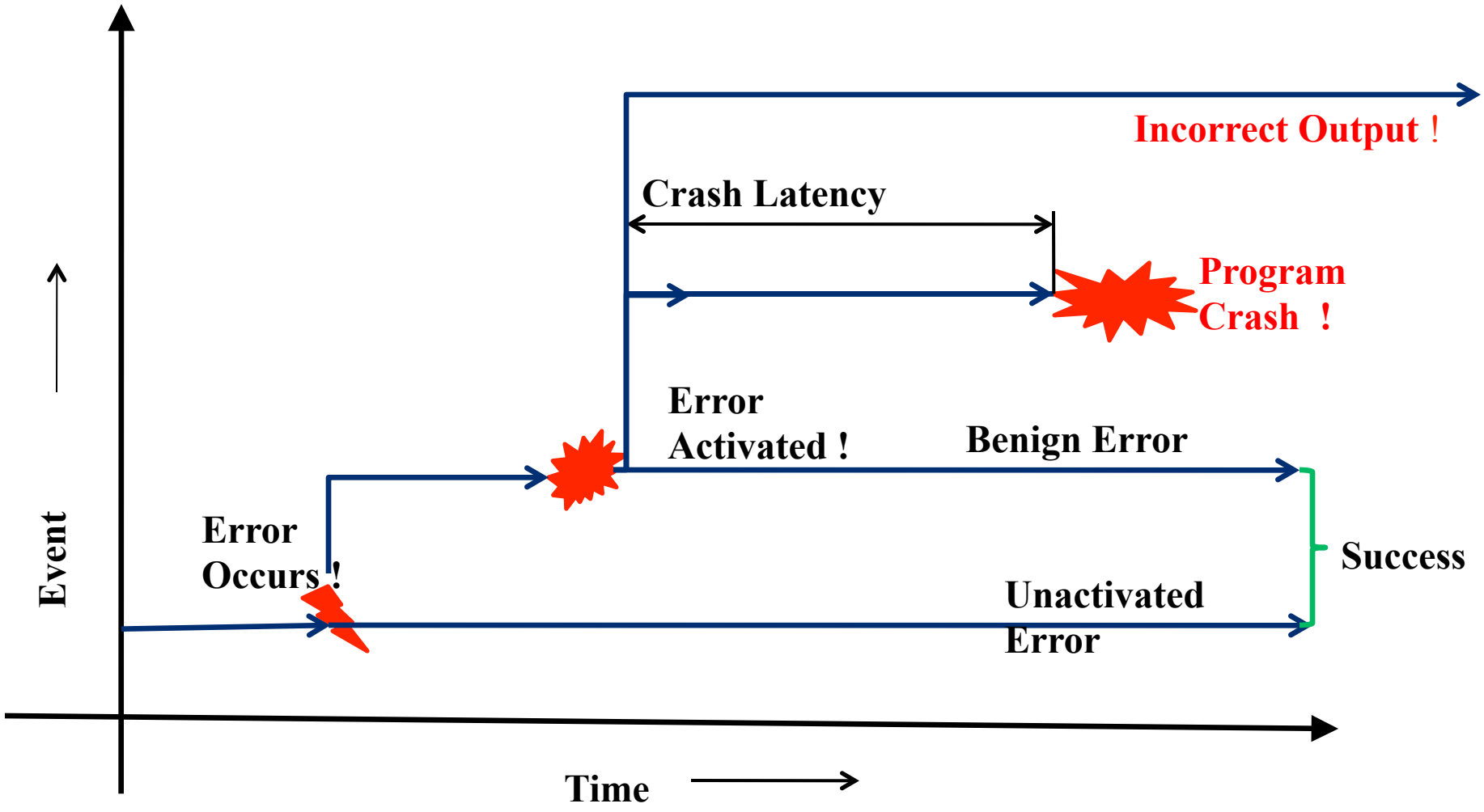
- A soft error in a processor causes corruption of a data word stored in the cache. This word is read and de-referenced in the program, which leads to an “out of bounds” exception (e.g., seg. fault).
- A program has a logical bug that is trigerred only by certain test cases (i.e., when they exercise it). When the bug is triggered, it causes the program to compute a value incorrectly and the wrong value is printed as part of the program’s output.

Fault masking and benign errors

- **Not all faults lead to errors**
  - Faults can be masked because they are not activated (e.g., faults in unread locations)
  - Faults can also be corrected before they lead to errors (e.g., memory scrubbing in ECC)

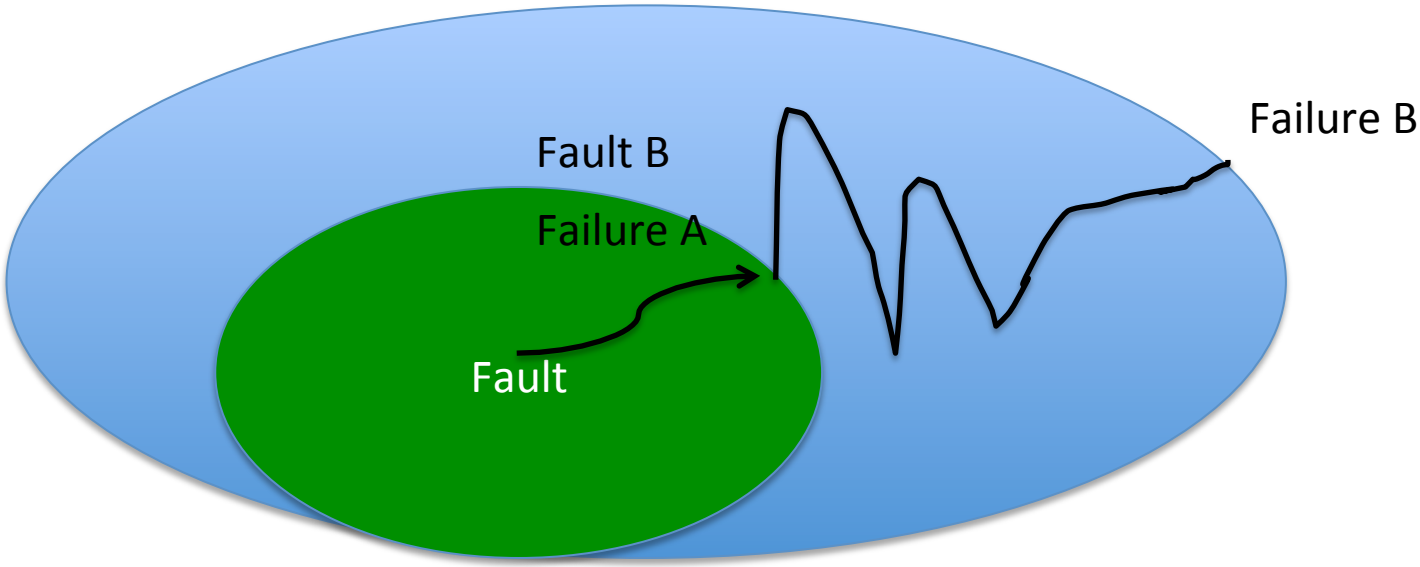
- **Not all errors lead to failures (benign errors)**
  - Error may affect inconsequential system state
  - System may have redundancy to correct error

Anatomy of an Error



Inter-connected Systems

- One system's failure (system A) may be another one's fault/error (system B)



Examples

- **Example 1**
  - A short in an integrated circuit is a failure (with respect to the function of the circuit)
  - The consequence (e.g., stuck at a Boolean value) is a fault that stays dormant until activated
  - Upon activation (invoking the faulty component by applying an input) the fault becomes active and produces an error
  - If and when the propagated error affects the delivered service (e.g., information content), a failure occurs
- **Example 2**
  - The result of an error by a programmer leads to a failure to write the correct instruction or data.
  - This results in a dormant fault in the written software (e.g., faulty instruction)
  - Upon activation the fault become active and produces an error
  - When the error affects the delivered service , a failure occurs

Fault Classification

Temporal

- Transient
  - Occurs only once at a location
  - Eg., cosmic ray strikes
- Intermittent
  - Periodically recur at a location
  - E.g., timing violations
- Permanent
  - Continuous and stable occurrence at a location
  - E.g., stuck-at-faults

Origin

- Physical faults:
  - Occur due to physical phenomena such as EM effects, threshold effects etc. or due to environmental conditions such as temperature or workload
- Human-made faults:
  - Programming errors, mis-configuration, human operator errors etc.

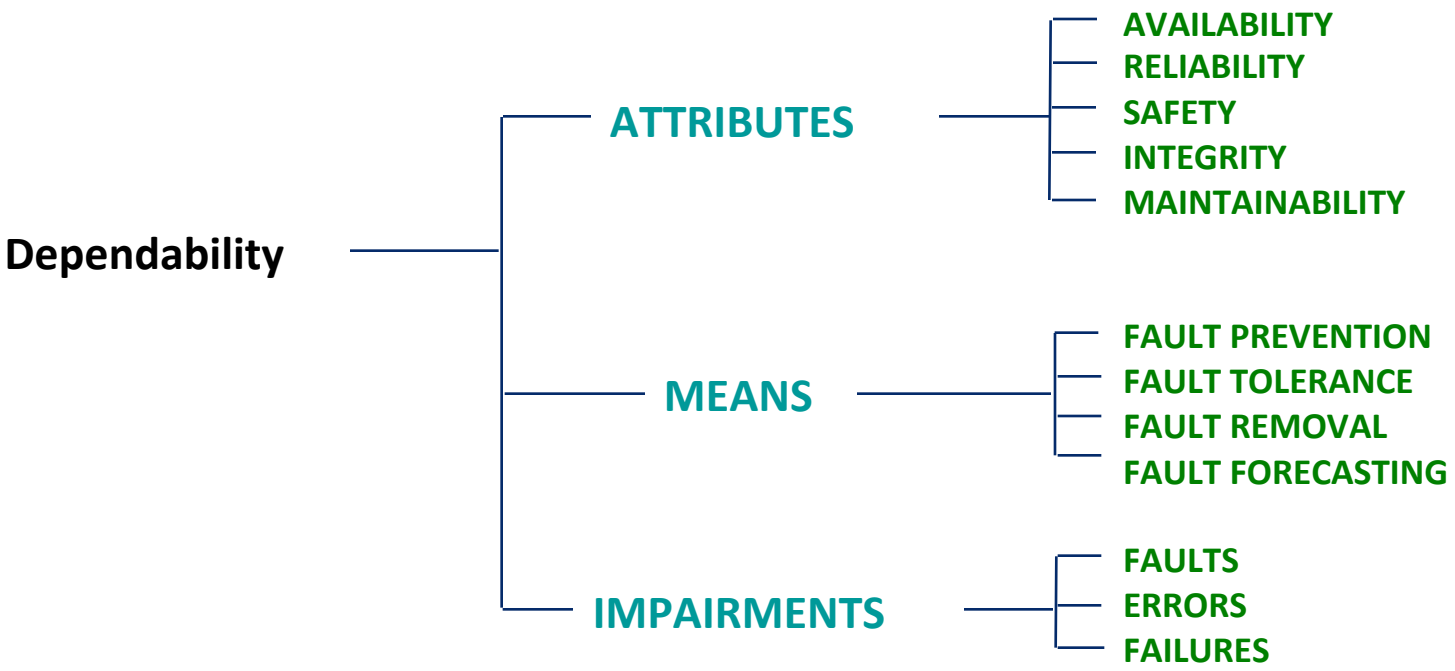
Classification of Failures

Criteria	Classification 1	Classification 2
Nature	Halt (fail-stop)	Erratic (Babbling)
Detection	Signaled	Un-signaled (fail silent)
Consistency	Consistent	Inconsistent
Severity	Minor	Catastrophic

Classification of Errors

Criteria	Classification 1	Classification 2
Domain	Timing	Content
Detection	Detected	Latent
Consistency	Consistent	Inconsistent
Severity	Minor	Catastrophic

Dependability: Attributes, Means and Impairments



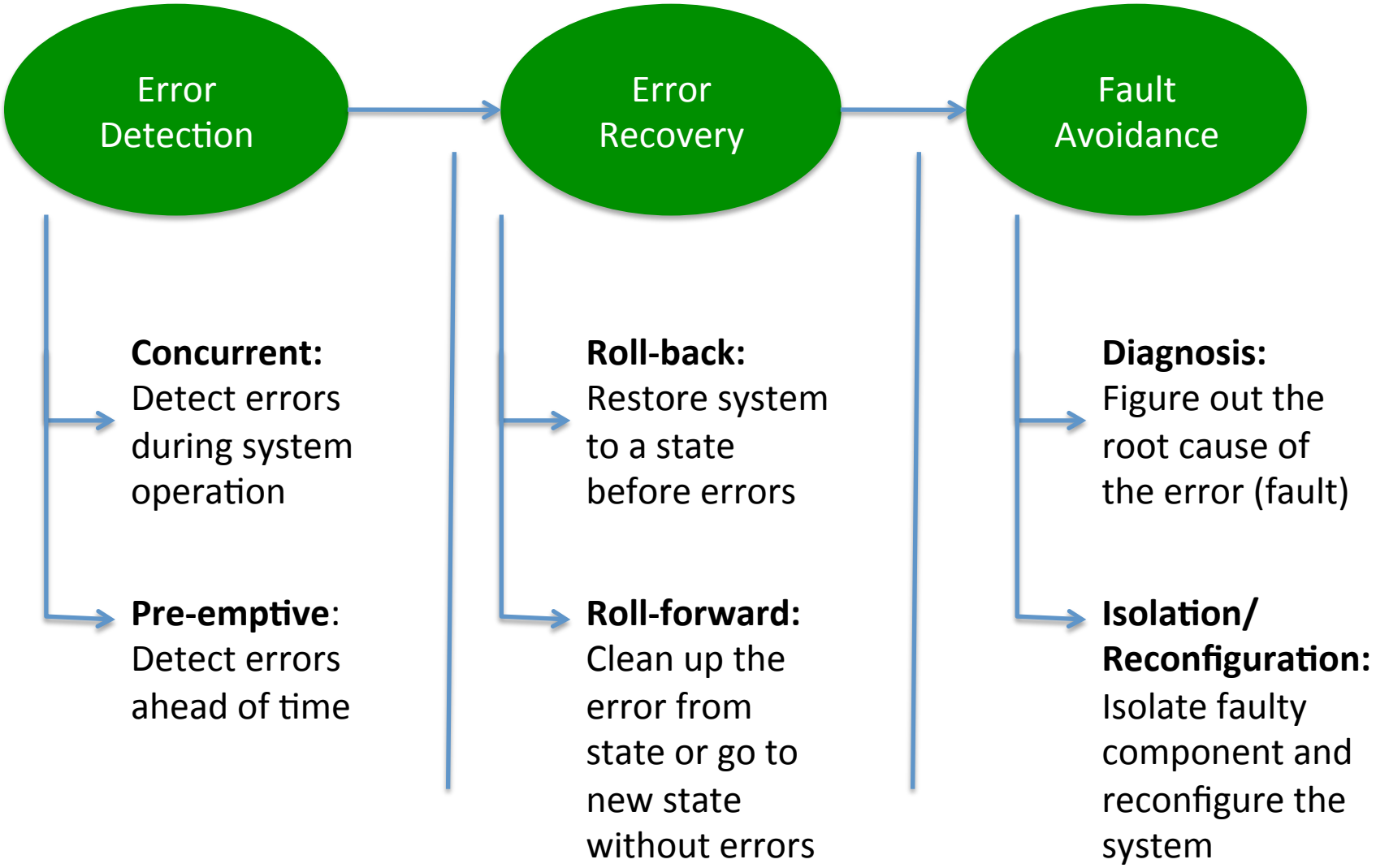
Dependability Means

- **Fault prevention**
  - Prevent occurrence of faults
- **Fault tolerance**
  - Avoid service failures in the presence of faults
- **Fault removal**
  - Reduce the number or occurrence of faults
- **Fault forecasting**
  - Predict how many faults remain and their likely consequences (increase assurance in system)

Examples

- Consider a space mission that has to be deployed for long periods of time (say a few years)
  - **Fault prevention**: Use of reliable coding practices, (e.g., defensive programming) and safe languages
  - **Fault-tolerance**: Provide ability to detect and repair failed components during missions
  - **Fault-removal**: Remove the faults at runtime or record the commonly observed faults for later removal (during debugging)
  - **Fault-forecasting:** Predict how likely a future mission is to succeed based on the lessons learned

Fault-tolerance techniques



Fault Tolerance

- **The ability to provide continued correct operation despite the presence of faults**
  - Encompasses a broad rage of techniques ranging from low-level devices to application software
  - Important to ensure that the service behaves as expected (if the fault belongs to fault-model)
  - There is no such thing as perfect fault-tolerance Every fault-tolerance technique has a coverage and a fault-model over which it is evaluated.

Error Detection

- **Concurrent detection during normal operation**
  - Watchdog timer
  - Software assertions
  - Process pairs
- **Preemptive detection preempts the failure**
  - Spare checking
  - Memory scrubbing
  - Software rejuvenation

Error Recovery

- **Rollback:** Restores the state of the system to one before the fault. Useful for transient and intermittent faults.

- **Rollforward:** Corrects the fault and allows system to make forward progress, if possible.

- **Compensation:** Leverage natural redundancy of the system to mask the error.

Fault Avoidance

- **Diagnosis**
  - Identify the root cause of the fault (location, type)
- **Isolation**
  - Physically/Logically excludes faulty component
- **Reconfiguration**
  - Switches in non-faulty components and remaps the tasks to the non-faulty components
- **Reinitialization**
  - Record the new system state and updates the external entities that interface with the system (if necessary)

Example: Hardware Fault Tolerance

- Multi-core system experiences a fault in one of the cores. Error is **detected** through a concurrent check. What are the possible recovery actions that can be taken ?

Example: Software Fault Tolerance

- Two identical copies of a software program are run concurrently to check each other. If the output of one differs from the output of the other, what recovery actions can be taken ?

Summary

- **Dependability concepts:** definitions, means, attributes and impairments, systems view

- Recommended reading:

Avizienis, A., Laprie, J., Randell, B., and Landwehr, C.

Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable and Secure Computing,* Vol 1, Issue 1, 2004, 11-33.

Outline

- Reliability Basics
- Security Basics

Chapter 1: Introduction

- Components of computer security
- Threats
- Policies and mechanisms
- The role of trust
- Assurance
- Operational Issues
- Human Issues

Basic Components

- Confidentiality
  - Keeping data and resources hidden
- Integrity
  - Data integrity (integrity)
  - Origin integrity (authentication)
- Availability
  - Enabling access to data and resources

Classes of Threats

- Disclosure
  - Snooping
- Deception
  - Modification, spoofing, repudiation of origin, denial of receipt
- Disruption
  - Modification
- Usurpation
  - Modification, spoofing, delay, denial of service

Policies and Mechanisms

- Policy says what is, and is not, allowed
  - This defines "security" for the site/system/*etc*.
- Mechanisms enforce policies
- Composition of policies
  - If policies conflict, discrepancies may create security vulnerabilities

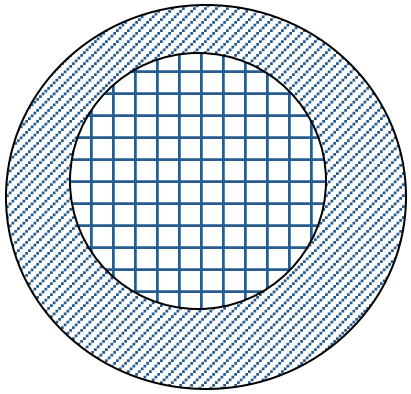
Goals of Security

- Prevention
  - Prevent attackers from violating security policy
- Detection
  - Detect attackers' violation of security policy
- Recovery
  - Stop attack, assess and repair damage
  - Continue to function correctly even if attack succeeds

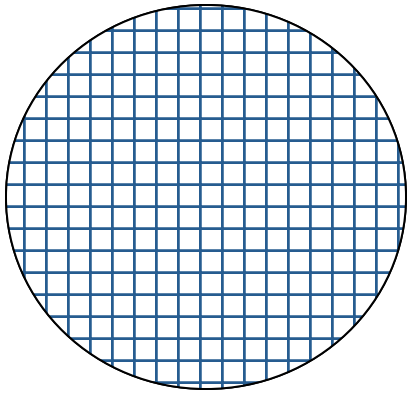
Trust and Assumptions

- Underlie *all* aspects of security
- Policies
  - Unambiguously partition system states
  - Correctly capture security requirements
- Mechanisms
  - Assumed to enforce policy
  - Support mechanisms work correctly

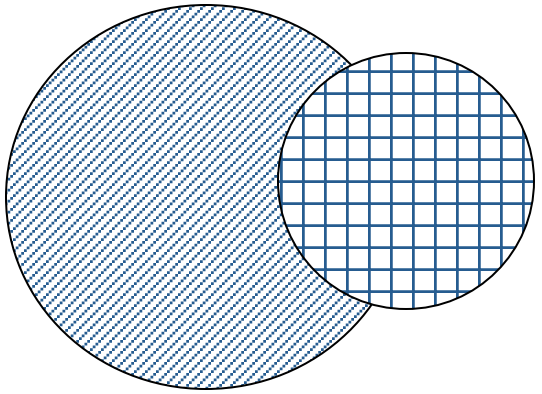
Types of Mechanisms



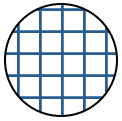
secure



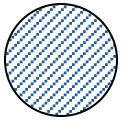
precise



broad



set of reachable states



set of secure states

Assurance

- Specification
  - Requirements analysis
  - Statement of desired functionality
- Design
  - How system will meet specification
- Implementation
  - Programs/systems that carry out design

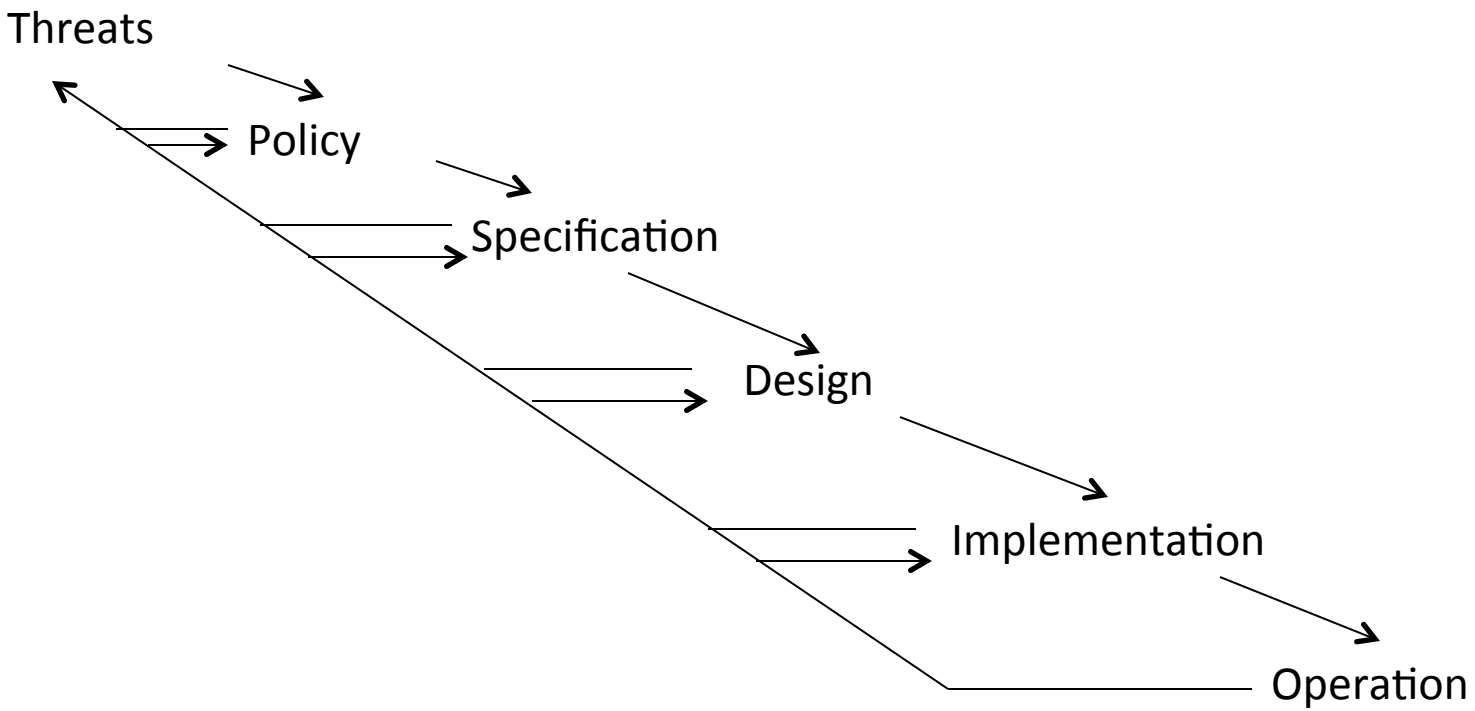
Operational Issues

- Cost-Benefit Analysis
  - Is it cheaper to prevent or recover?
- Risk Analysis
  - Should we protect something?
  - How much should we protect this thing?
- Laws and Customs
  - Are desired security measures illegal?
  - Will people do them?

Human Issues

- Organizational Problems
  - Power and responsibility
  - Financial benefits
- People problems
  - Outsiders and insiders
  - Social engineering

Tying Together



Key Points

- Policy defines security, and mechanisms enforce security
  - Confidentiality
  - Integrity
  - Availability
- Trust and knowing assumptions
- Importance of assurance
- The human factor

Chapter 13: Design Principles

- Overview
- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability

Overview

- Simplicity
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand
- Restriction
  - Minimize access
  - Inhibit communication

Least Privilege

- A subject should be given only those privileges necessary to complete its task
  - Function, not identity, controls
  - Rights added as needed, discarded after use
  - Minimal protection domain

Fail-Safe Defaults

- Default action is to deny access
- If action fails, system as secure as when action began

Economy of Mechanism

- Keep it as simple as possible
  - KISS Principle
- Simpler means less can go wrong
  - And when errors occur, they are easier to understand and fix
- Interfaces and interactions

Complete Mediation

- Check every access
- Usually done once, on first action
  - UNIX: access checked on open, not checked thereafter
- If permissions change after, may get unauthorized access

Open Design

- Security should not depend on secrecy of design or implementation
  - Popularly misunderstood to mean that source code should be public
  - "Security through obscurity"
  - Does not apply to information such as passwords or cryptographic keys

Separation of Privilege

- Require multiple conditions to grant privilege
  - Separation of duty
  - Defense in depth

Least Common Mechanism

- Mechanisms should not be shared
  - Information can flow along shared channels
  - Covert channels
- Isolation
  - Virtual machines
  - Sandboxes

Psychological Acceptability

- Security mechanisms should not add to difficulty of accessing resource
  - Hide complexity introduced by security mechanisms
  - Ease of installation, configuration, use
  - Human factors critical here

Key Points

- Principles of secure design underlie all security-related mechanisms
- Require:
  - Good understanding of goal of mechanism and environment in which it is to be used
  - Careful analysis and design
  - Careful implementation

References

- Matt Bishop's Book Chapters 1 and 13

"Computer Security Art and Science", Addison Wessley

(both chapters are freely available on the web)