Security Bootcamp

Konstantin (Kosta) Beznosov

EECE 512 "Topics in Computer Security"

outline

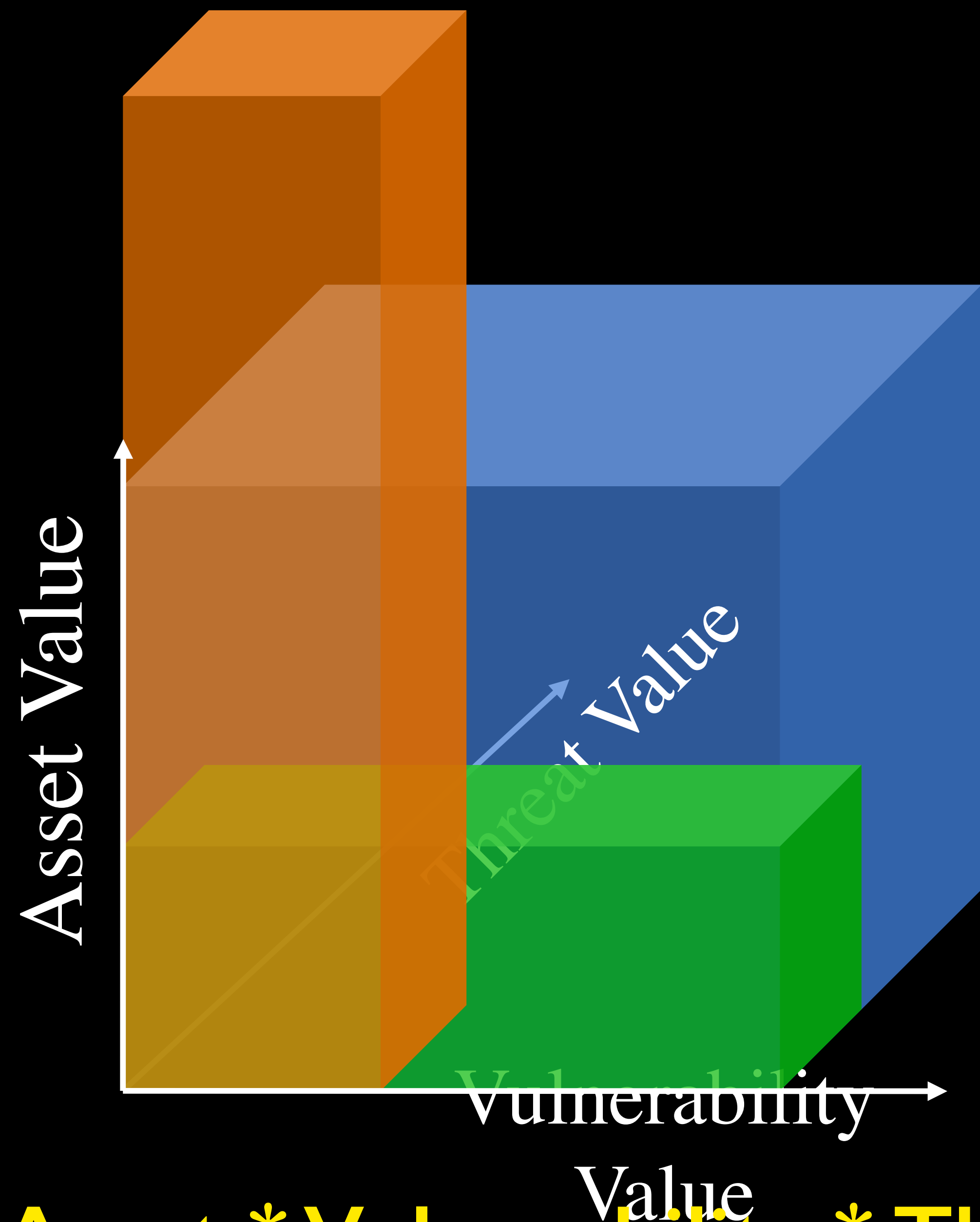
- very quick intro to computer security
- principles of designing secure systems
- security architectures: policies and mechanisms

Very Quick Intro to Computer Security

What is Security?

- security -- "safety, or freedom from worry"
- how can it be achieved?
  - Make computers too heavy to steal
  - Buy insurance
  - Create redundancy (disaster recovery services)

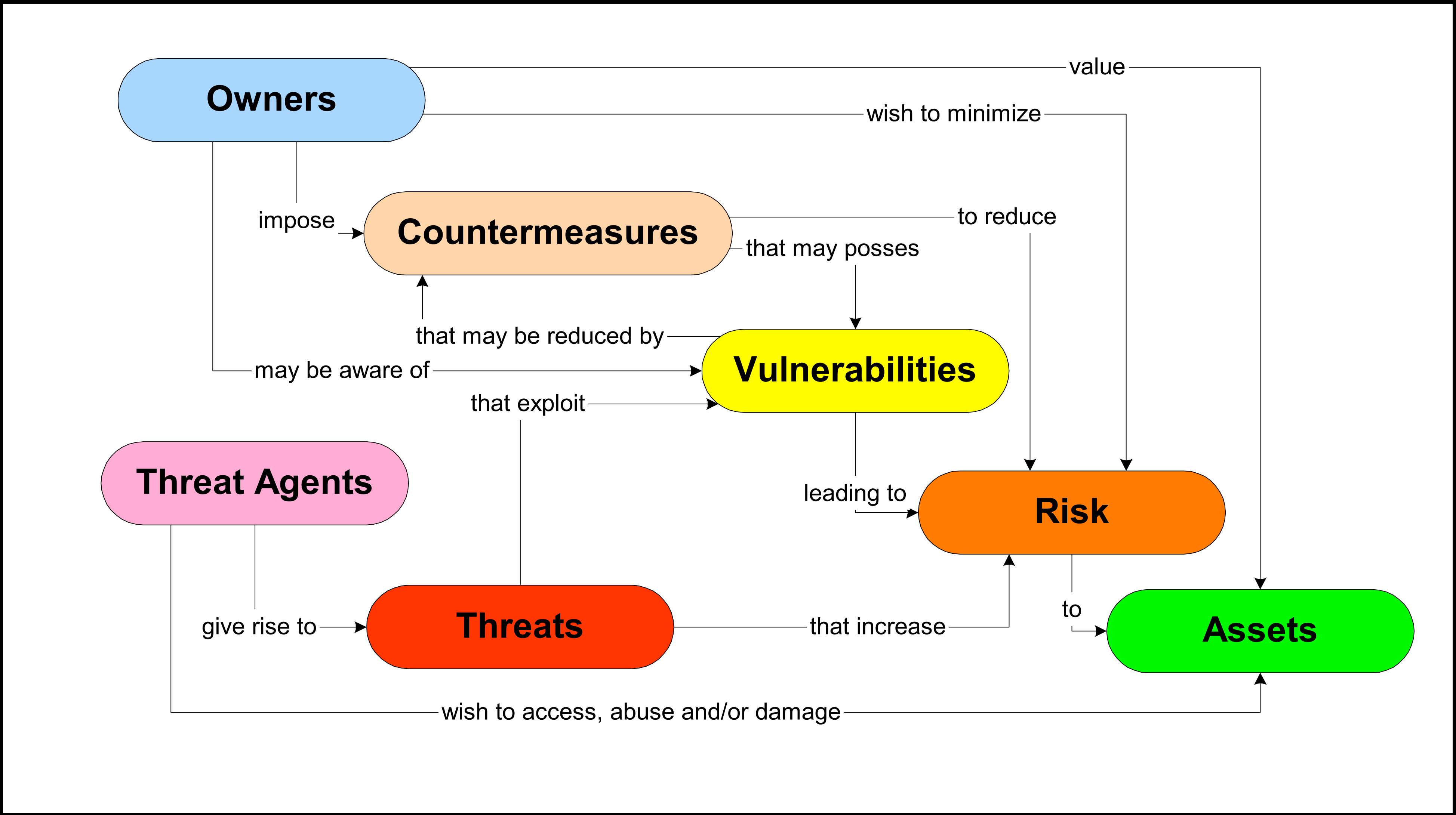
it’s all about risk management



Risk = Asset * Vulnerability * Threat

What can be done about risk?

- Accept
- Avoid
- Transfer
- Reduce



Source: Common Criteria for Information Technology Security Evaluation. 1999

Analyze

1. Assets at risk and their value
2. Threats to these assets
3. Threat agents

Classes of Threats

- Disclosure
  - snooping
- Deception
  - modification
  - spoofing
  - repudiation of origin
  - denial of receipt
- Disruption
  - modification
  - denial of service
- Usurpation
  - modification
  - spoofing
  - delay
  - denial of service

Goals of Security

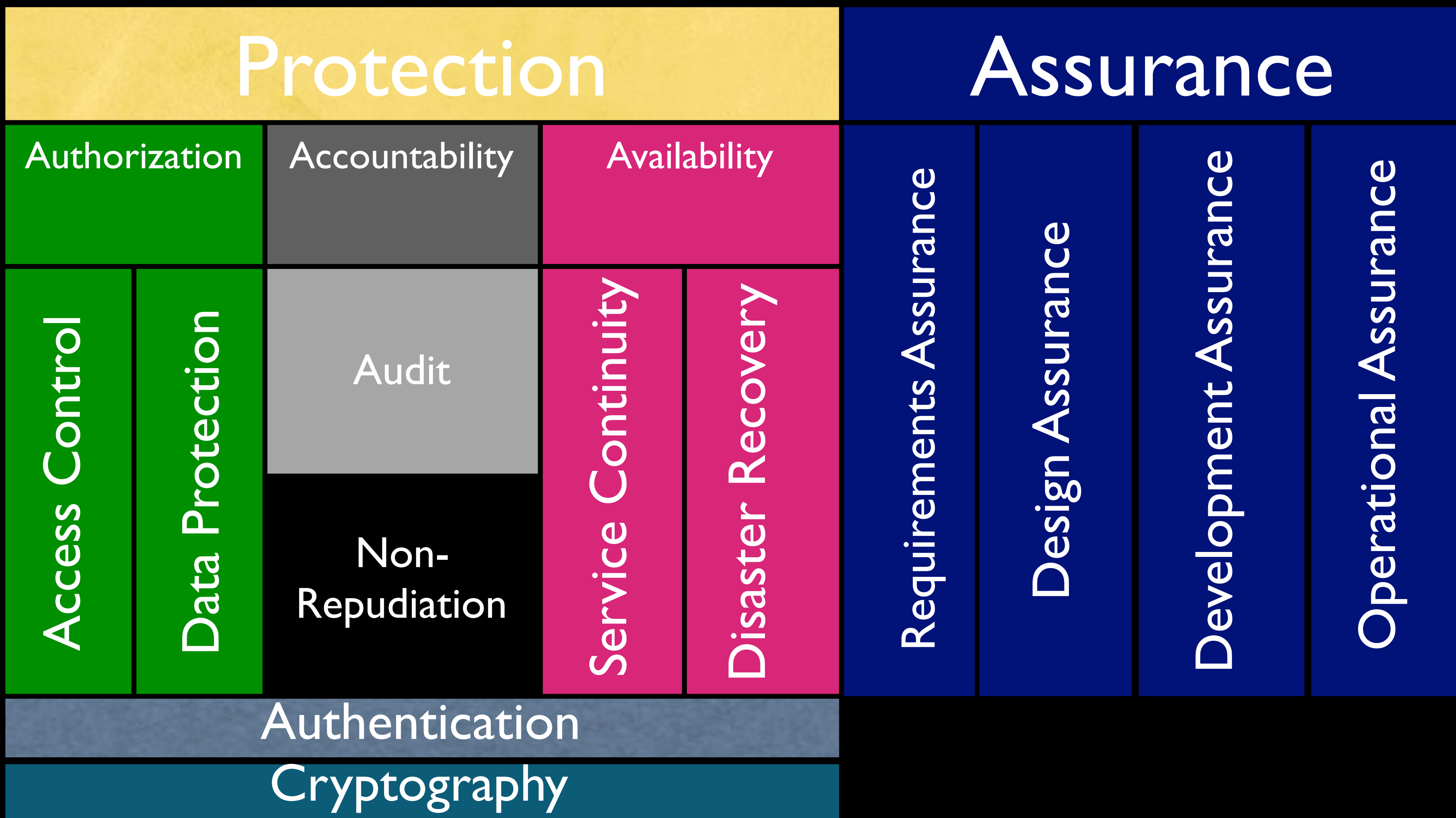
- Deterrence
  - Deter attacks
- Prevention
  - Prevent attackers from violating security policy
- Detection
  - Detect attackers' violation of security policy
- Recovery
  - Stop attack, assess and repair damage
  - Continue to function correctly even if attack succeeds
- Investigation
  - Find out how the attack was executed: forensics
  - Decide what to change in the future to minimize the risk

What Computer Security Policies are Concerned with?

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

CIA

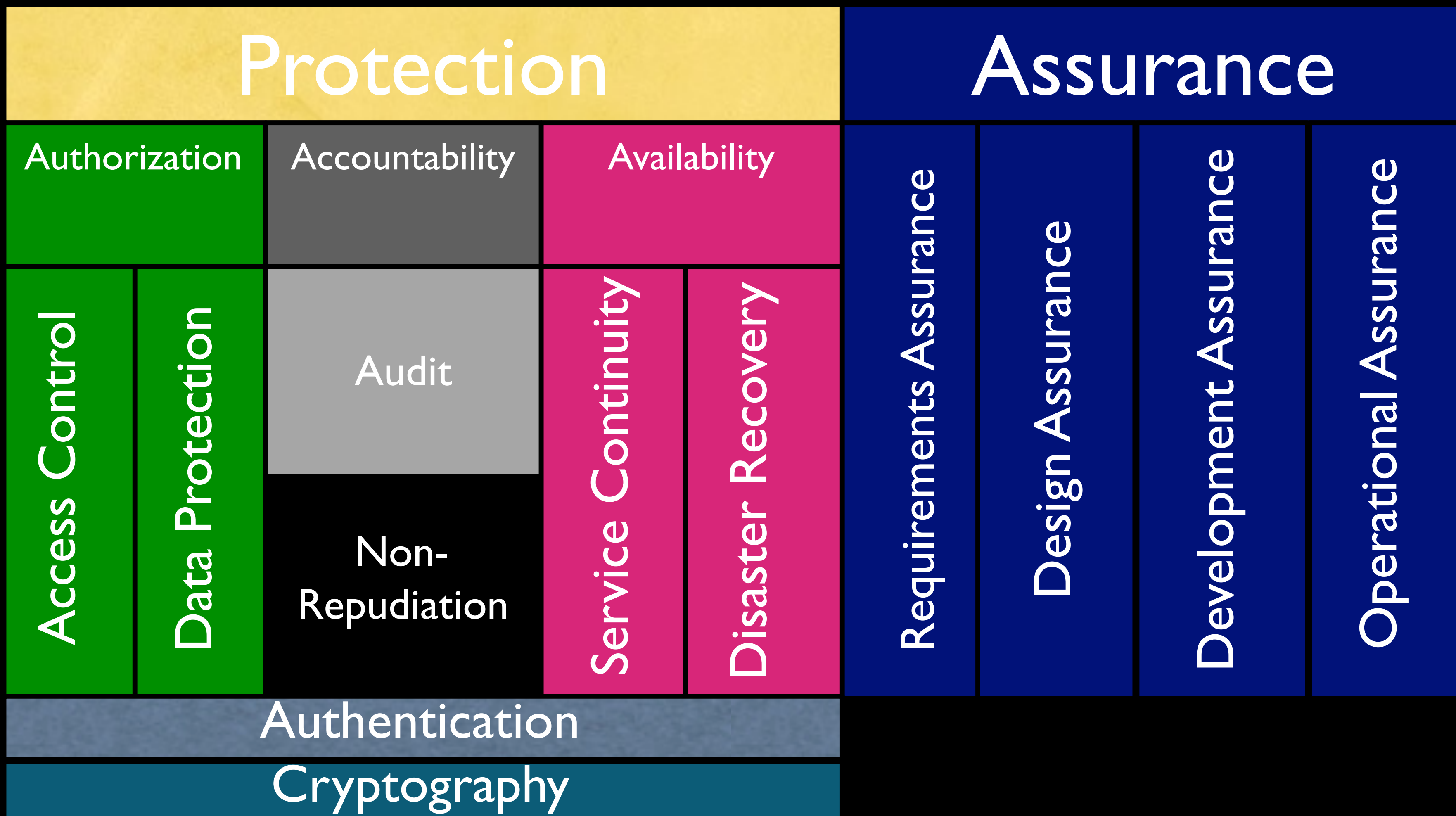
Conventional Approach to Security



Protection

provided by a set of mechanisms (countermeasures) to prevent bad things (threats) from happening

Conventional Approach to Security



Authentication

What is Authentication?

- Real-world and computer world examples?
- What is a result of authentication?
- What are the means for in the digital world?

Basics and Terminology

definition

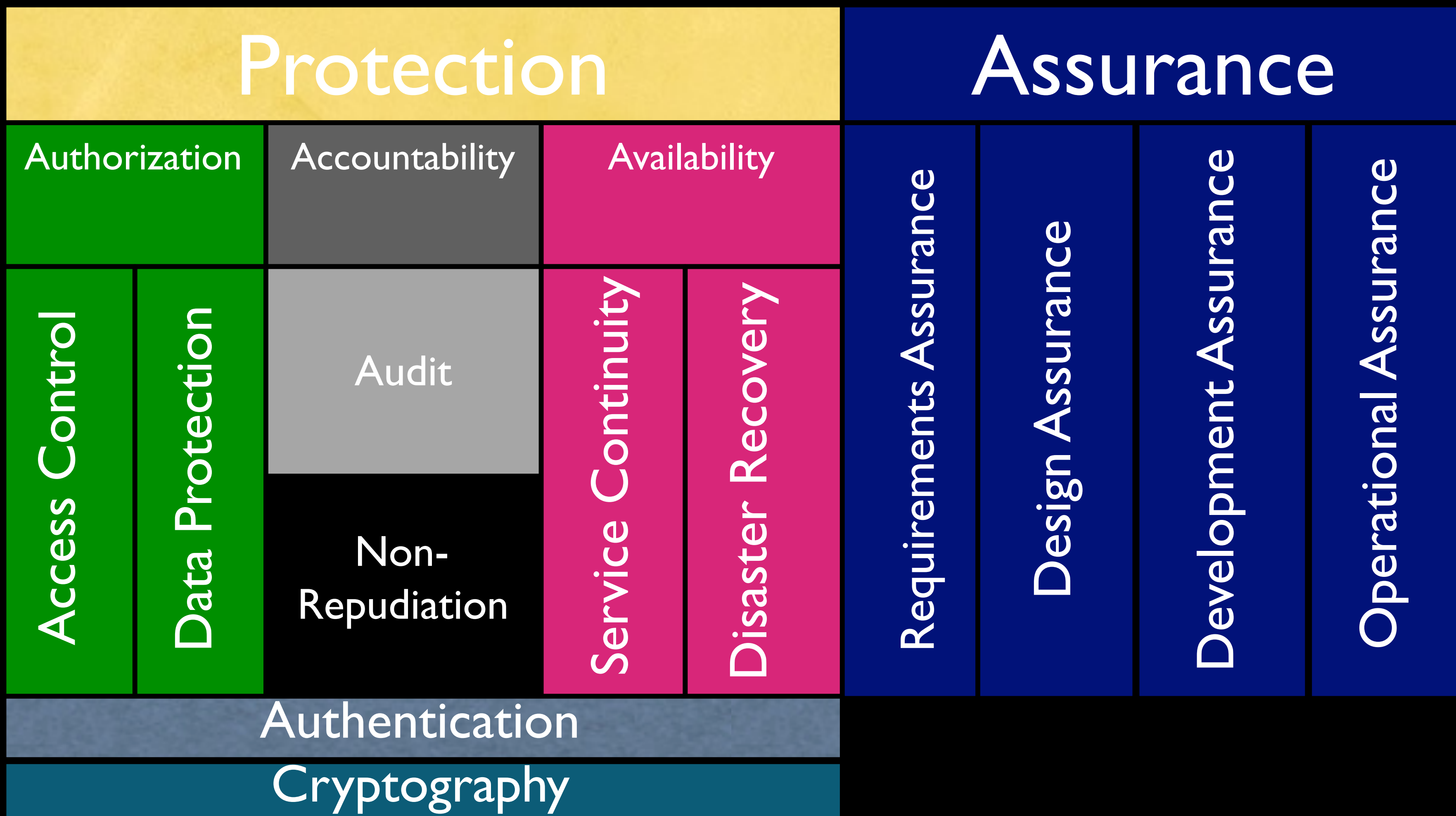
authentication is binding of identity to subject

- Identity is that of external entity
- Subject is computer entity
- Subject a.k.a. principal

What Authentication Factors are used?

- What you know
- What you have
- What you are

Conventional Approach to Security



Authorization

protection against breaking rules

- Rule examples:
  - No one outside the company can read proprietary data
  - Tellers can initiate funds transfers of up to $500;
    Managers -- up to $5,000
    Transfers over $5,000 must be initiated by a VP
  - Attending physician can read patient HIV status

Authorization Mechanisms: Access Control

Definition: **enforces the rules, when rule check is possible**

Subject
Principal
User, Client
Initiator

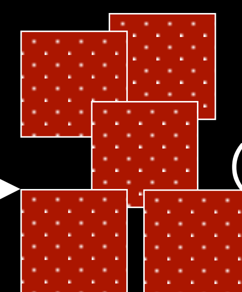
Action

Authorization Engine
Access Decision
Function
PDP

Authorization Decision Entitlement

Reference Monitor PEP

Security Subsystem



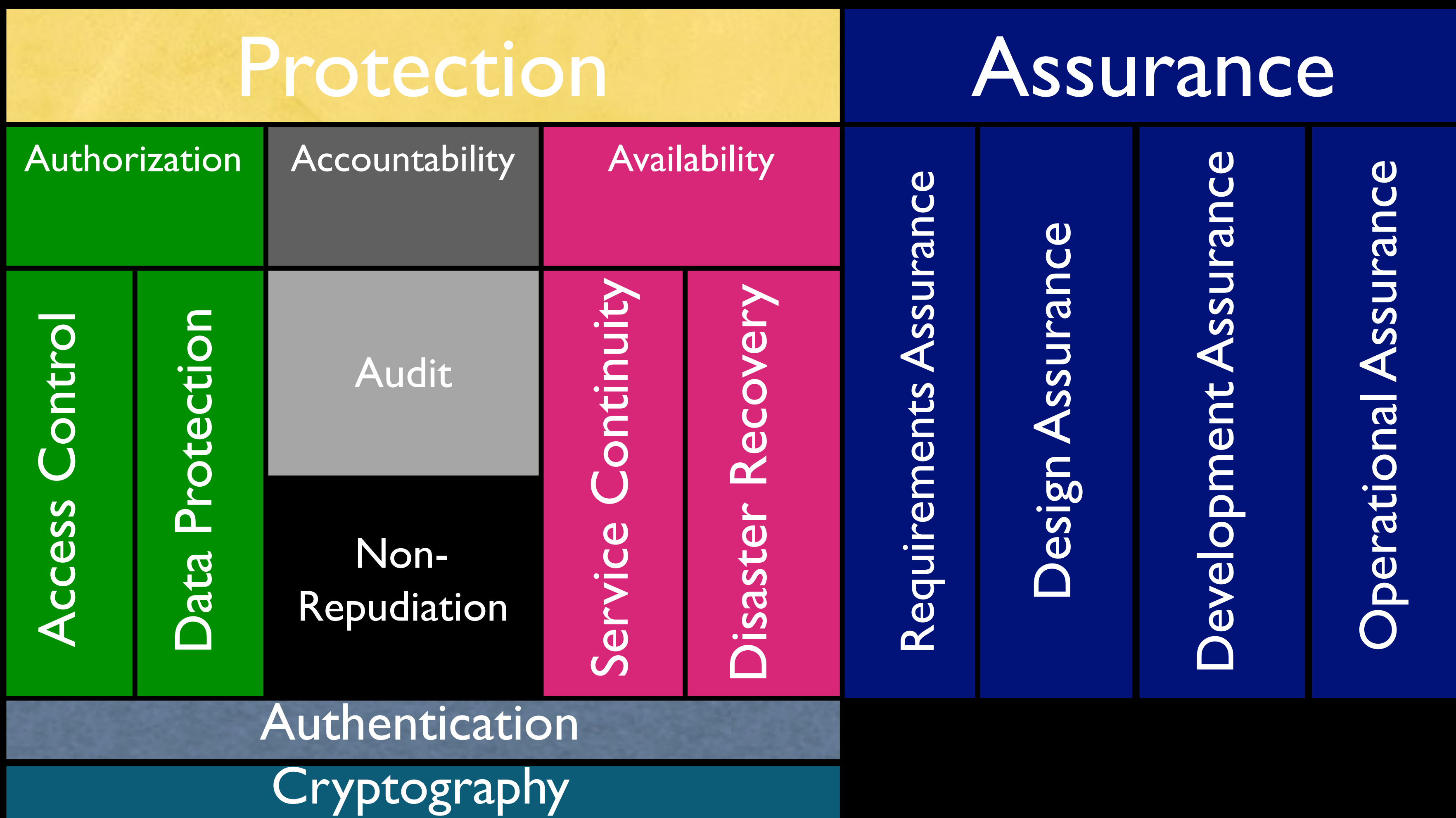
Object
Resource
(data/methods
menu item)
Target

Mix of terms:

Authorization == Access Control Decision

Authorization Engine == Policy Engine

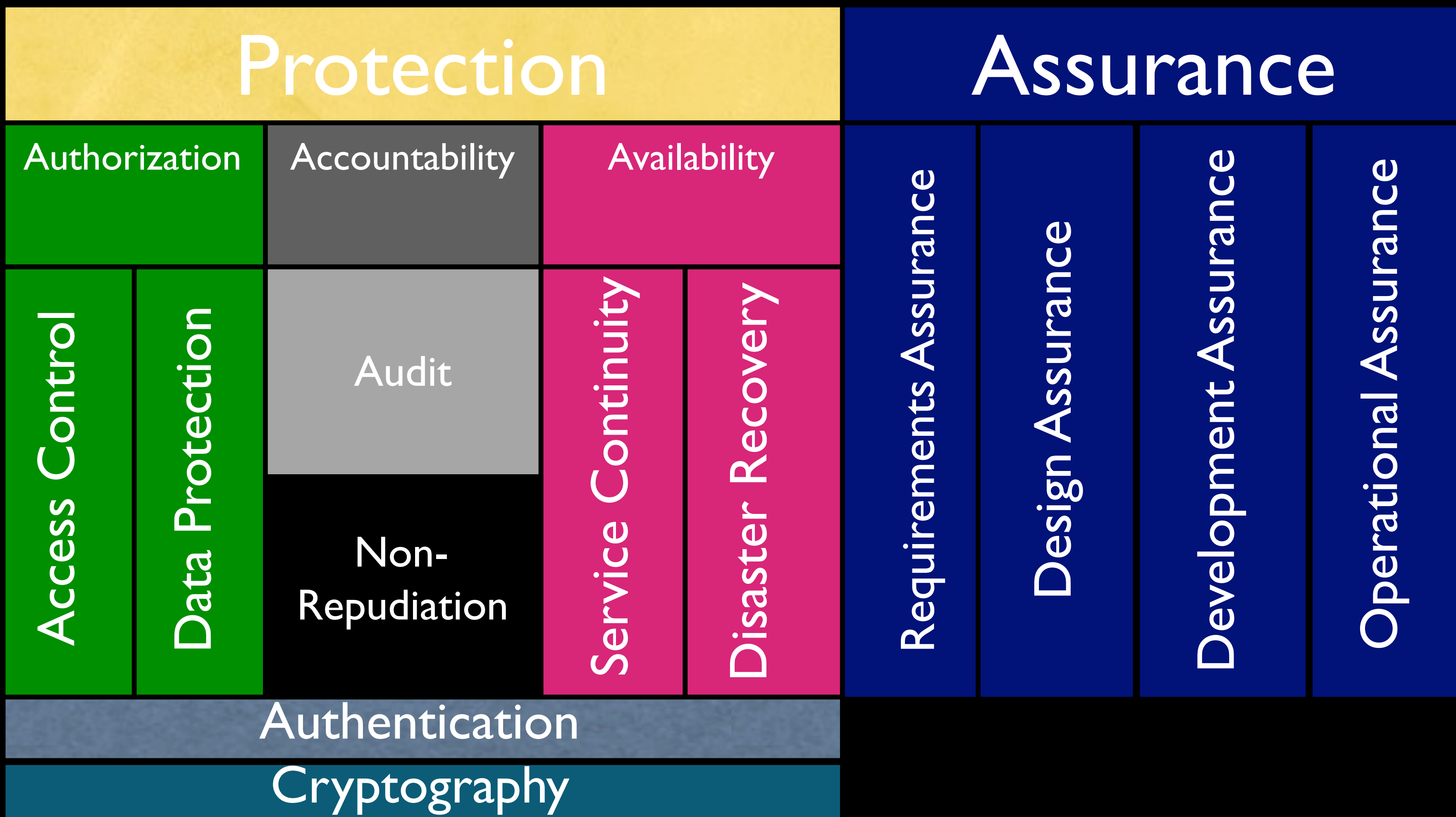
Conventional Approach to Security



Authorization Mechanisms: Data Protection

- No way to check the rules
  - e.g. telephone wire
- No trust to enforce the rules
  - e.g. MS-DOS

Conventional Approach to Security

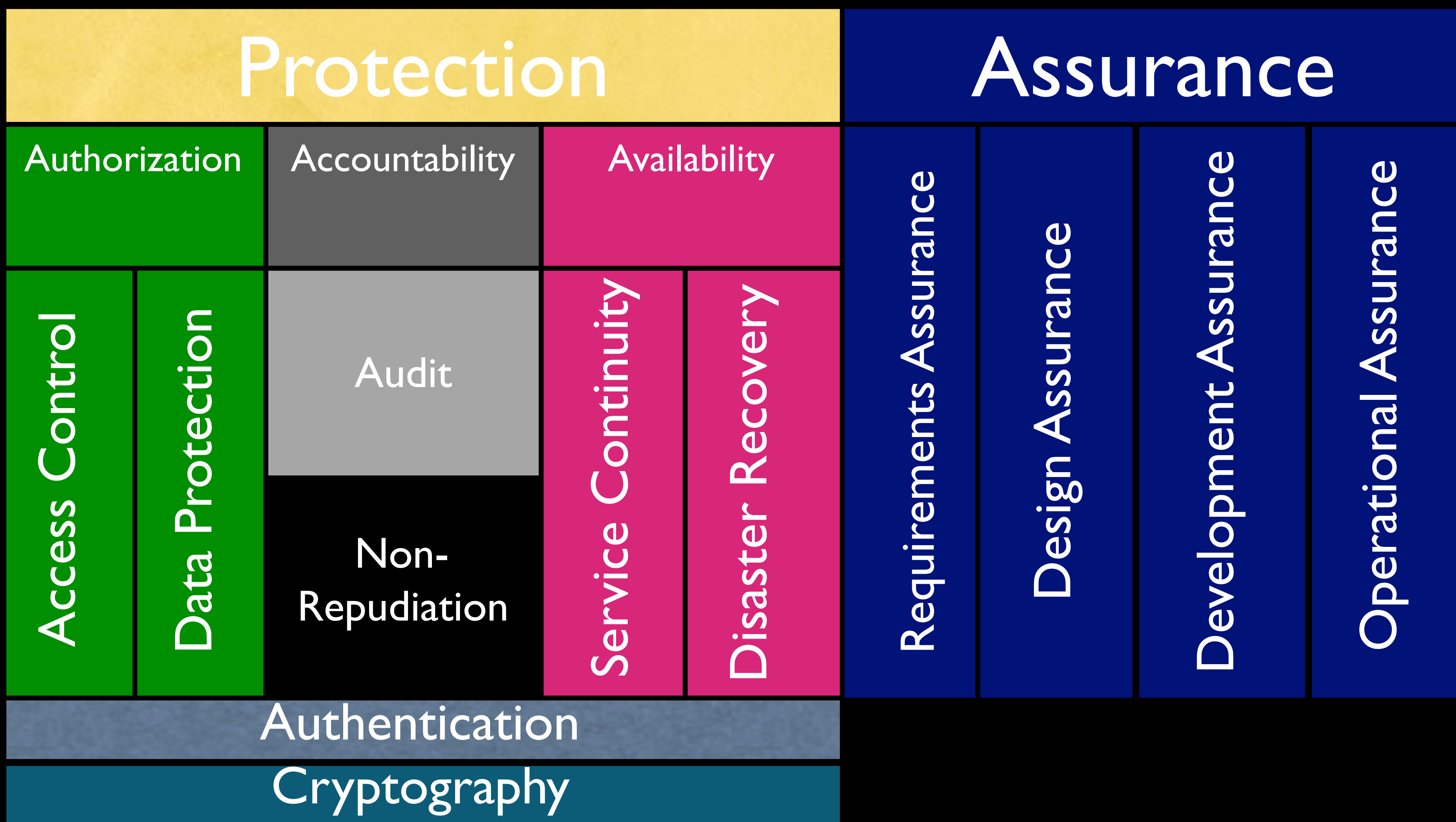


Accountability

You can tell who did what when

- Audit -- actions are recorded in audit log
- Non-Repudiation -- evidence of actions is generated and stored

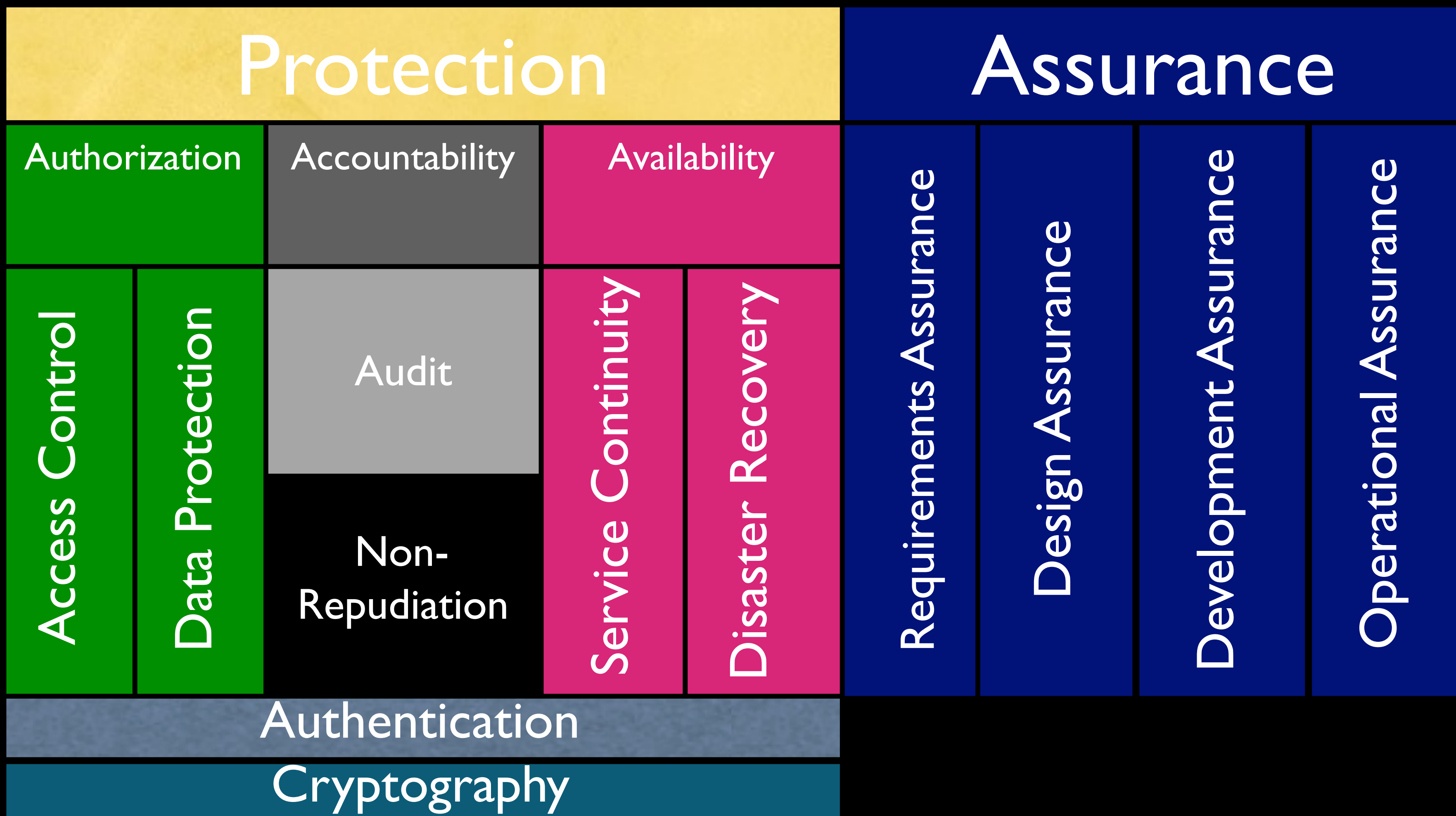
Conventional Approach to Security



Availability

- Service continuity -- you can always get to your resources
- Disaster recovery -- you can always get back to your work after the interruption

Conventional Approach to Security



Assurance

What's Assurance?

Set of things the system builder and the operator of the system do to convince you that it is really safe to use.

- the system can enforce the policy you are interested in, and
- the system works

Assurance Methods

- testing
- verification
- validation

Testing

Advantages

- actual product--not some abstraction or product precursor

Limitations

- negative nature of security properties
  - demonstrates the existing of the problem, but not the absence of it
- expensive and complex because of the combinatorial explosion of inputs and internal states
- black-box testing does not ensure completeness
- white-box testing affects the product’s behavior ==> new vulnerabilities
- non-determinism makes it hard to reproduce problems

Penetration Testing

a.k.a., tiger/red team analysis, ethical hacking

- experts try to crack the tested system
- mechanic inspects a used car
- automation tools for testing web servers, NOSs, firewalls, etc.

Verification

checks the (security) quality of the implementation

Formal Verification

1. system is modeled ==> model
2. system properties are described as assertions
3. model + assertions = theorem
4. theorem is proved

- popular in verifying cryptographic protocols

Validation

assures that the developers are building the right product

Ways to Validate a System

- requirements checking
- design and code reviews
- system testing
- system verification

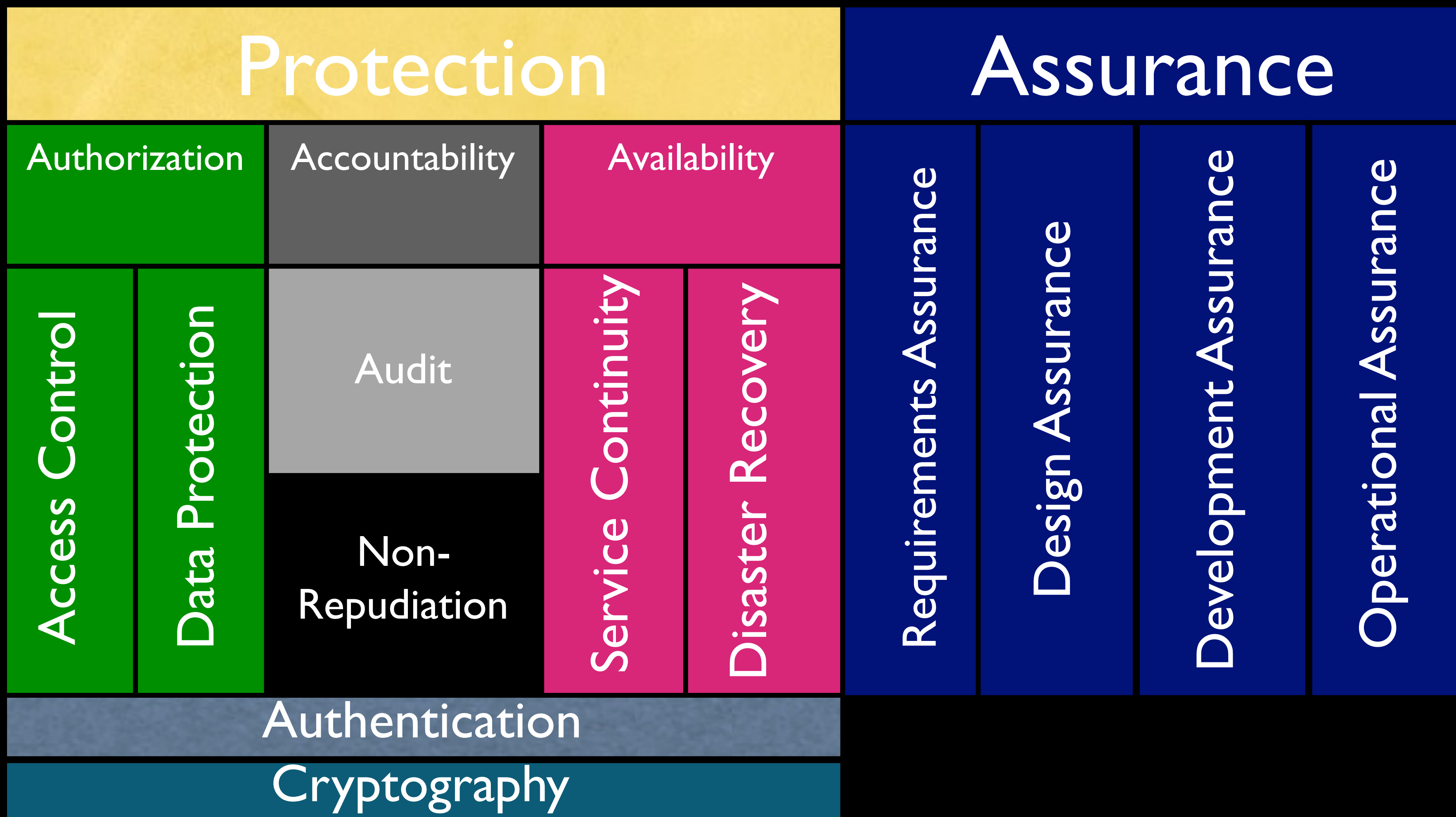
Validation Efforts

- Common Criteria

Steps of Improving Security

1. analyze risks
   - asset values
   - threat degrees
   - vulnerabilities
2. develop/change policies
3. choose & develop countermeasures
4. assure
5. go back to the beginning

Key Points



Protection
Assurance
Authorization
Accountability
Availability
Access Control
Data Protection
Audit
Non-Repudiation
Service Continuity
Disaster Recovery
Requirements Assurance
Design Assurance
Development Assurance
Operational Assurance
Authentication
Cryptography

Key Points (cont-ed)

- Risk = Asset * Vulnerability * Threat
- Steps of improving security
- Classes of threats
  - Disclosure
  - Deception
  - Disruption
  - Usurpation

Principles of Designing Secure Systems

Quick Overview

Principles

1. Least Privilege
2. Fail-Safe Defaults
3. Economy of Mechanism
4. Complete Mediation
5. Open Design
6. Separation of Duty
7. Least Common Mechanism
8. Psychological Acceptability
9. Defense in depth
10. Question assumptions

Overarching Goals

- Simplicity
  - Less to go wrong
  - Fewer possible inconsistencies
  - Easy to understand
- Restriction
  - Minimize access
    - “need to know” policy
  - Inhibit communication to minimize abuse of the channels

Principle 1: Least Privilege

Every program and every user of the system should operate using the least set of privileges necessary to complete the job

- Rights added as needed, discarded after use

- Limits the possible damage
- Unintentional, unwanted, or improper uses of privilege are less likely to occur
- Guides design of protection domains

Example: IIS in Windows Server 2003

- before -- all privileges
- in Windows Server 2003 and later -- low-priveleged account

Principle 2: Fail-Safe Defaults

Base access decisions on permission rather than exclusion.

suggested by E. Glaser in 1965

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

Example: IIS in Windows Server 2003

crashes if attacked using buffer overflow

Principle: Economy of Mechanism

Keep the design as simple and small as possible.

- KISS Principle
- Rationale?
  - Essential for analysis
  - Simpler means less can go wrong
    - And when errors occur, they are easier to understand and fix

Example: Trusted Computing Base (TCB)

- temper-proof
- non-bypassable
- small enough to analyze it

Principle 4: Complete Mediation

Every access to every object must be checked for authority.

If permissions change after, may get unauthorized access

Example: forgetting security checks in new/modified code

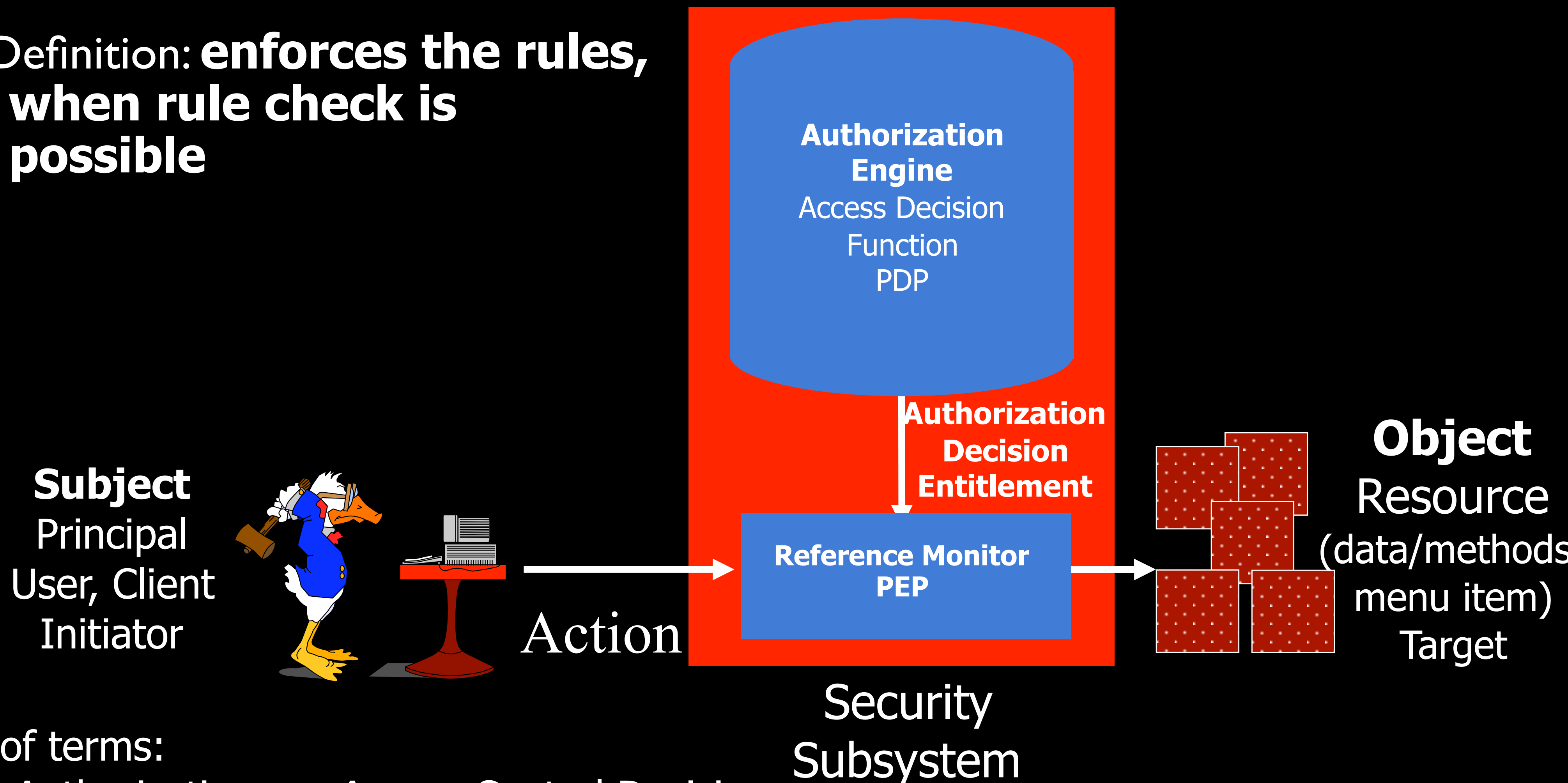
If an application mixes business and security logic, developers are prone to omitting security checks by mistakes

Example:
Multiple reads after one check

- Process rights checked at file opening
- No checks are done at each read/write operation
- Time-of-check to time-of-use

Authorization Mechanisms: Access Control

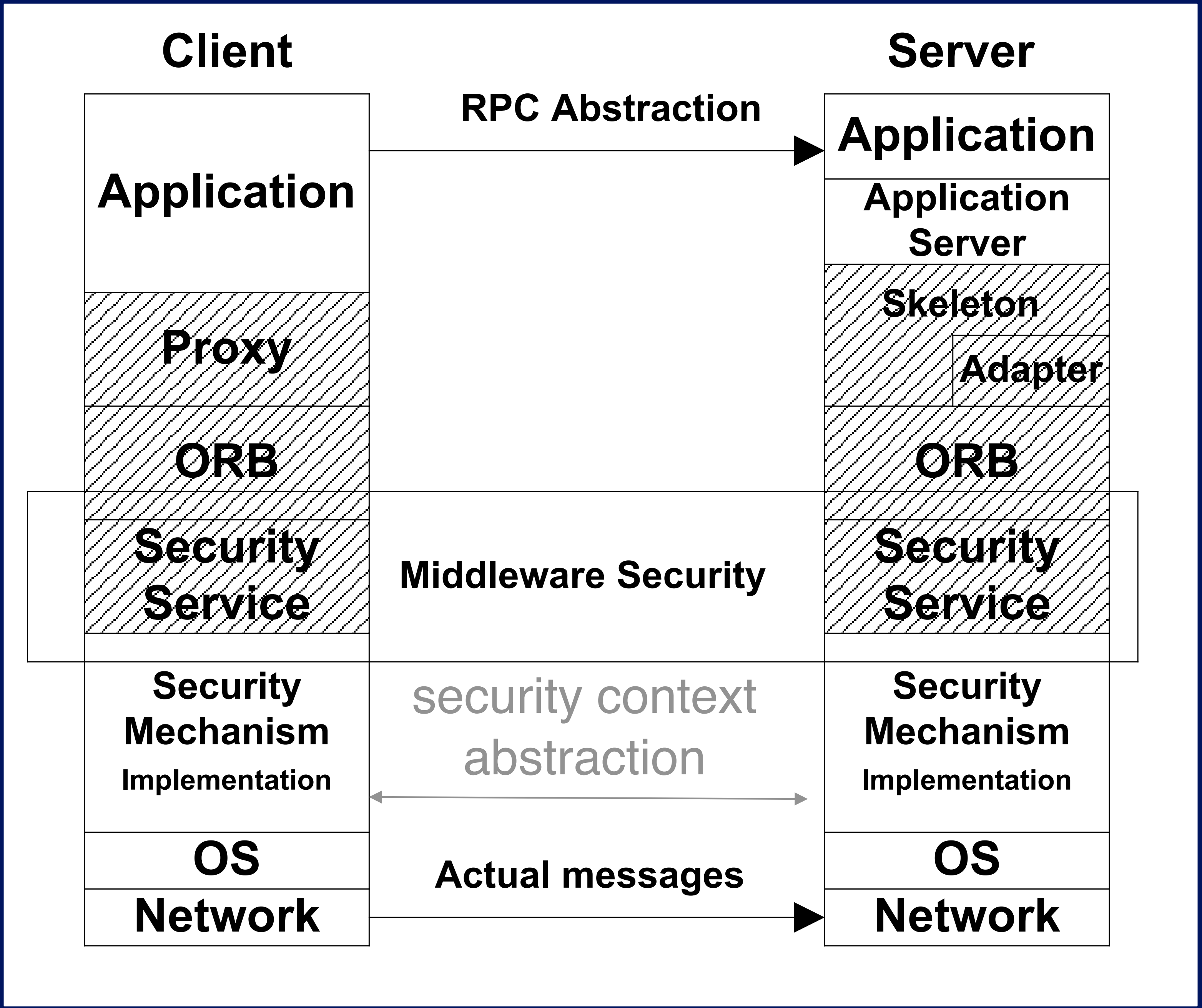
Definition: **enforces the rules, when rule check is possible**



Mix of terms:

- Authorization == Access Control Decision
- Authorization Engine == Policy Engine

Middleware Security Stack



Kerckhoff’s Principle

“The security of a cryptosystem must not depend on keeping secret the crypto-algorithm. The security depends only on keeping secret the key”

Auguste Kerckhoff von Nieuwenhof

Dutch linguist

1883

Principle 5: Open Design

Security should not depend on secrecy of design or implementation

P. Baran, 1965

- no "security through obscurity"
- does not apply to secret information such as passwords or cryptographic keys

Example: secretly developed GSM algorithms

- COMP128 hash function
  - later found to be weak
    - can be broken with 150,000 chosen plaintexts
  - attacker can find GSM key in 2-10 hours
- A5/1 & A5/2 weak

Example: Content Scrambling System

DVD content

- SecretEncrypt(K_D,K_{p1})
- …
- SecretEncrypt(K_D,K_{pn})
- Hash(K_D)
- SecretEncrypt(K_T,K_D)
- SecretEncrypt(Movie,K_T)

1999

- Norwegian group derived SecretKey by using K_{Pi}
- Plaintiff's lawyers included CSS source code in the filed declaration
- The declaration got out on the internet

Principle 6: Separation of Duty

Require multiple conditions to grant privilege

R. Needham, 1973

a.k.a. "separation of privilege"

example: SoD constraints in RBAC

- static SoD
  - if a user is assigned role “system administrator” then the user cannot be assigned role “auditor”
- dynamic SoD
  - a user cannot activate two conflicting roles, only one at a time

Principle 7: Least Common Mechanism

Mechanisms should not be shared

- Information can flow along shared channels in uncontrollable way
- Covert channels
- solutions using isolation
  - Virtual machines
  - Sandboxes

example: network security

- switches vs. repeaters
- security enclaves

Principle 8: 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

example: Switching between user accounts

- Windows NT -- pain in a neck
- Windows 2000/XP -- “Run as …”
- Unix -- “su” or “sudo”

Principle 9: Defense in Depth

Layer your defenses

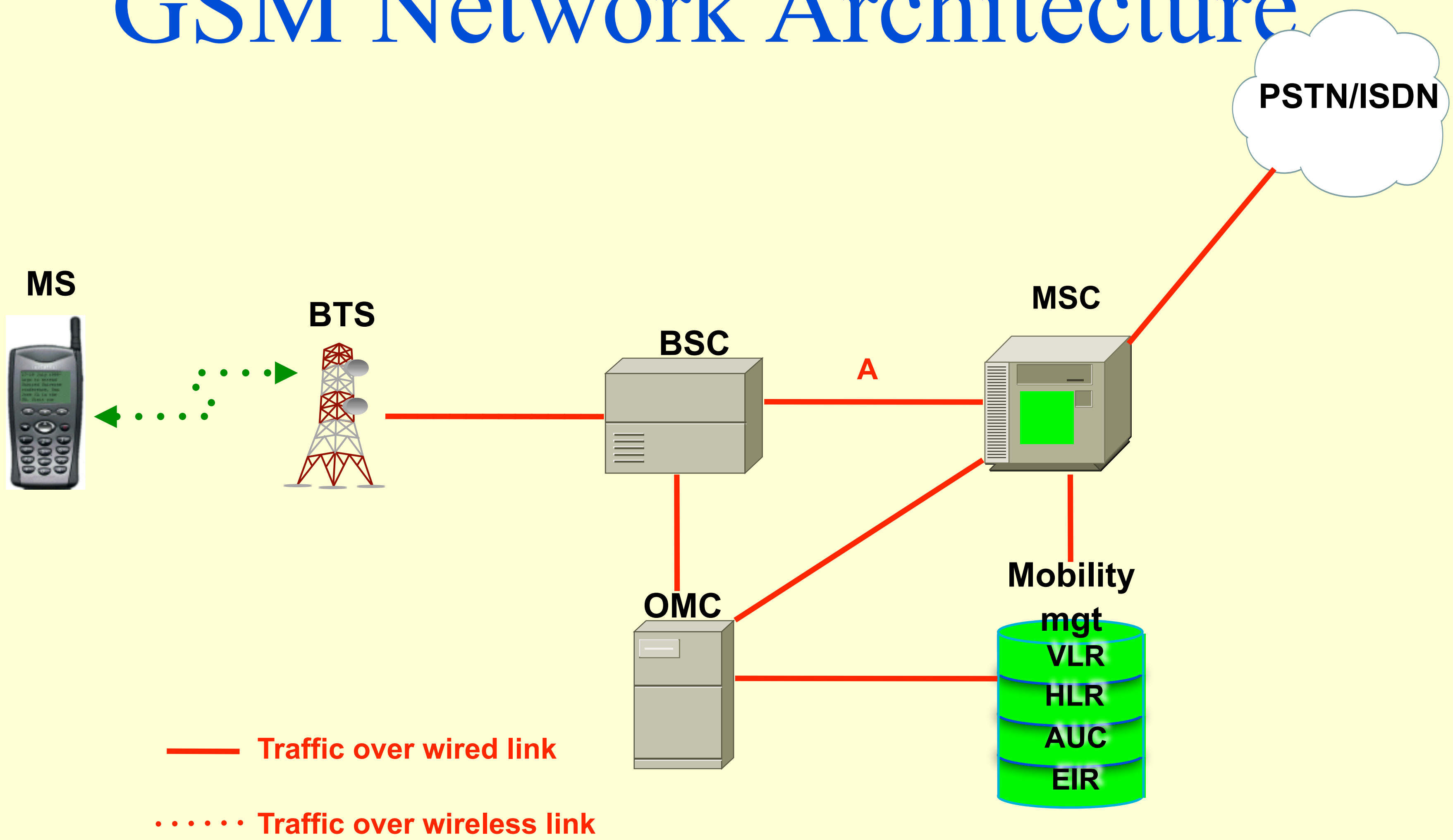
example: Windows Server 2003

Potential problem	Mechanism	Practice
Buffer overflow	defensive programming	check preconditions
Even if it were vulnerable	IIS 6.0 is **not** up by default	no extra functionality
Even if IIS were running	default URL length 16 KB	conservative limits
Even if the buffer were large	the process crashes	fail-safe
Even if the vulnerability were exploited	Low privileged account	least privileged

Principle 10: Question Assumptions

Frequently re-examine all the assumptions about the threat agents, assets, and especially the environment of the system

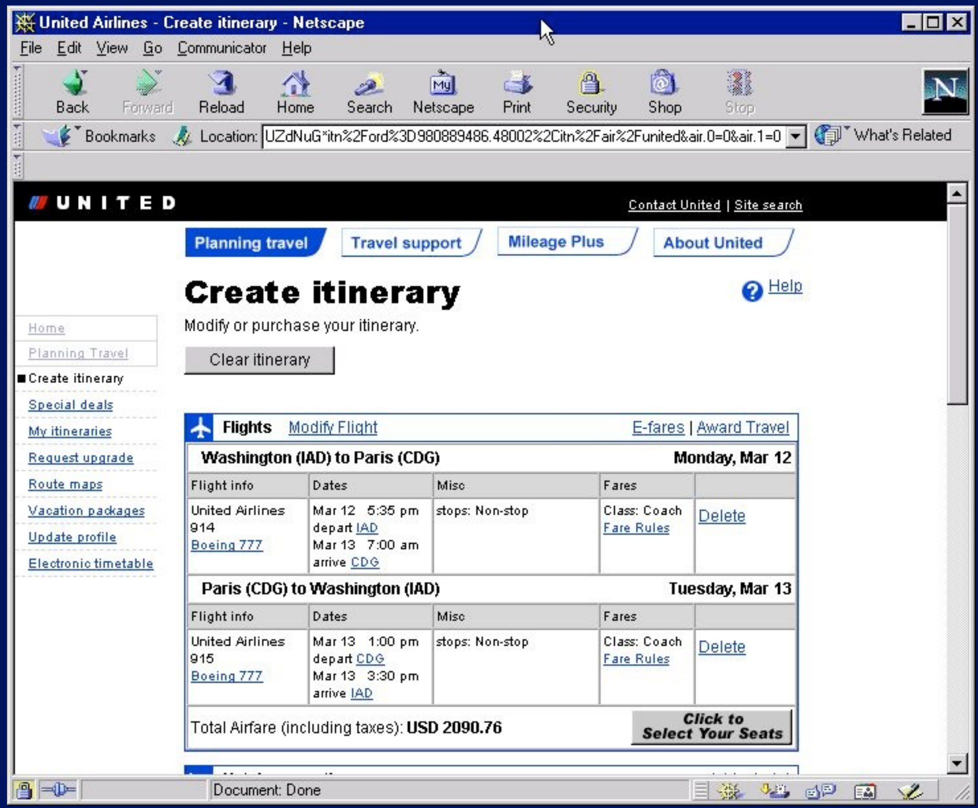
Example:
GSM Network Architecture



Circuit-switched technology

Attack pattern examples

- Exploit race condition
- Provide unexpected input
- Bypass input validation



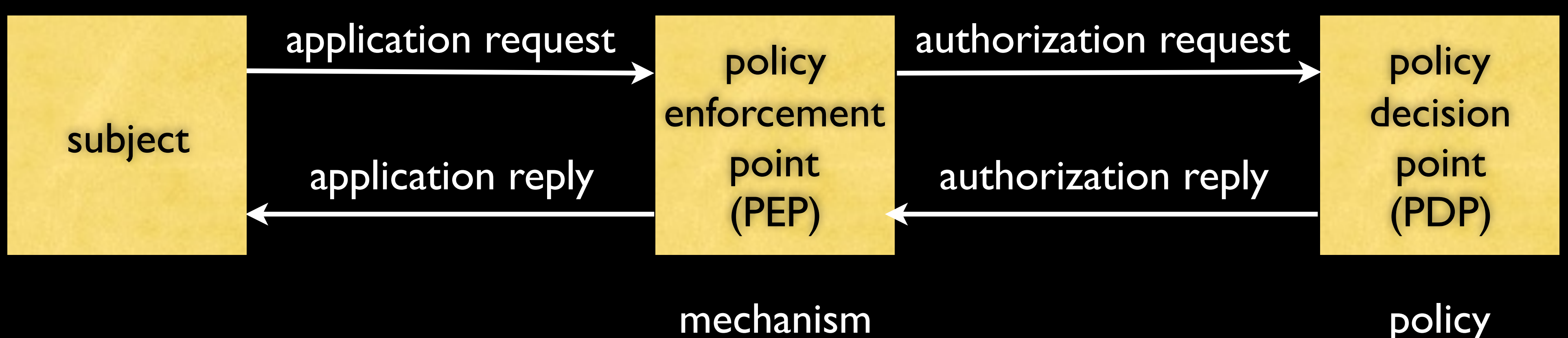
Principles

1. Least Privilege
2. Fail-Safe Defaults
3. Economy of Mechanism
4. Complete Mediation
5. Open Design
6. Separation of Duty
7. Least Common Mechanism
8. Psychological Acceptability
9. Defense in depth
10. Question assumptions

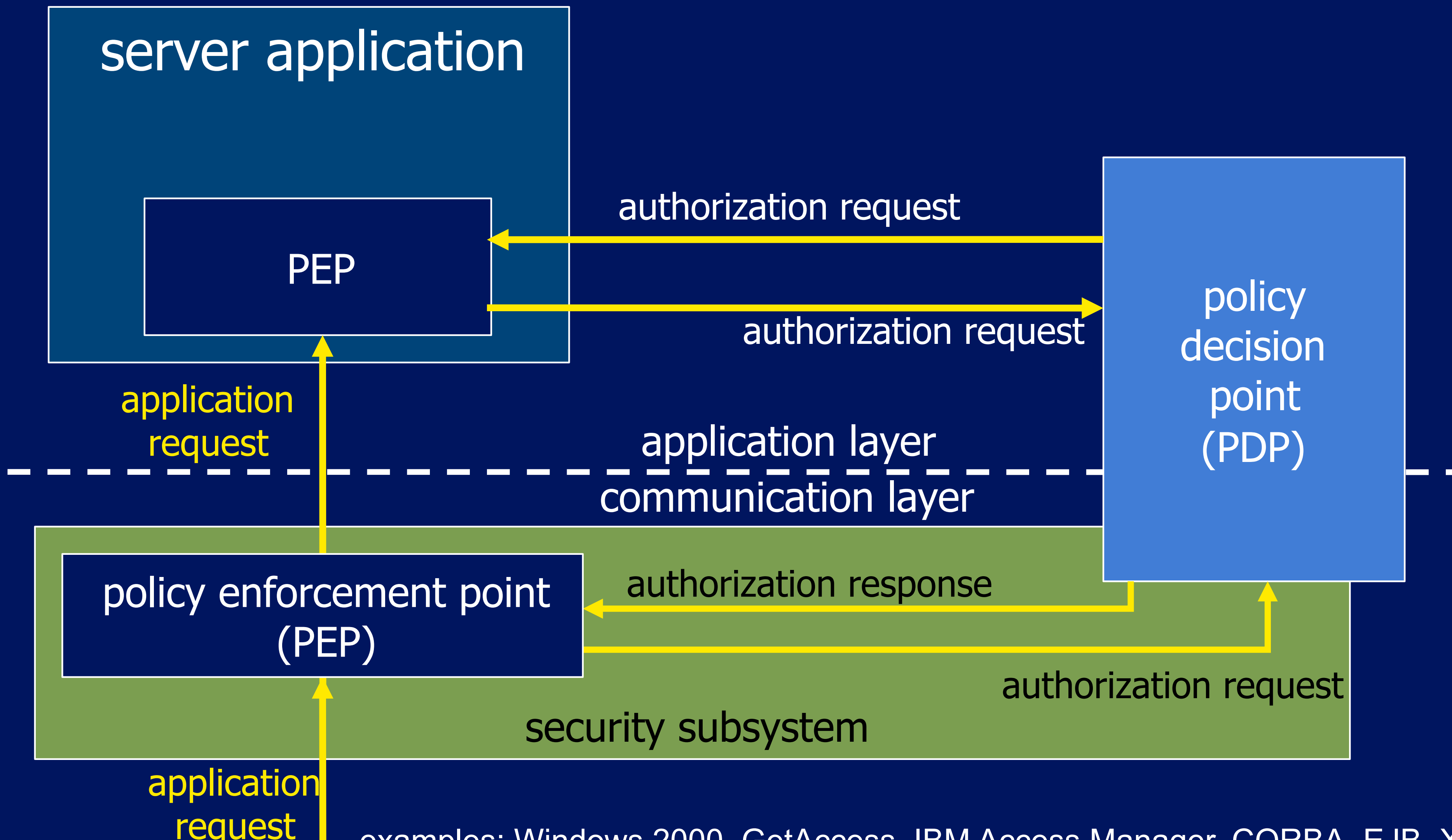
Security Architectures: Policies and Mechanisms

Policies and Mechanisms

- Policies describe what is allowed
- Mechanisms control how policies are enforced



how enterprise authorization systems work



examples: Windows 2000, GetAccess, IBM Access Manager, CORBA, EJB, XACML