

MANAGEMENT 414
SANS +S™
TRAINING PROGRAM
FOR THE CISSP®
CERTIFICATION EXAM

414.3

Applications and Systems Development, and Cryptography

Copyright © 2006, The SANS Institute. All rights reserved. The entire contents of this publication are the property of the SANS Institute. User may not copy, reproduce, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, without the express written consent of the SANS Institute. Without limiting the foregoing, user may not reproduce, distribute, re-publish, display, modify, or create derivative works based upon all or any portion of this publication for purposes of teaching any computer or electronic security courses to any third party without the express written consent of the SANS Institute.

Preface

The cardinal rule for SANS training is that after you take a course you should be able to apply what you learned directly the day you get back into the workplace. My journey into writing about the 10 Domains started when Stephen Northcutt asked that I lead the development of adding the ISC2 10 Domains of Knowledge into SANS Security Essentials. That is SANS most popular training course and when taught bootcamp style it does an amazing job of helping students become capable of using hands-on techie tools. However, there had been a split in the community whether Security Essentials, which favored technical and pragmatic material, or the ISC2 10 Domains, which favors theory, should be the baseline standard for an information security professional. We were discussing this hotly debated issue in the SANS faculty speaker room over lunch one day and it suddenly dawned on us, why not add the 10 Domains into Security Essentials? Tony Cole, CISSP, was assigned to evaluate Security Essentials and determined that about 60% of the 10 Domains material was already covered in Security Essentials. Clement Dupuis and I were the leads on the project. This was a very successful edit and a number of students have passed their CISSP exams after going through SANS Security Essentials with the ISC2 10 Domains. However, when we added the additional material there was no longer time to cover the application of the material to the workplace; in addition, there are some students who prefer the more formal 10 domain structure.

To best meet the needs of the students, SANS authorized the creation of Management 414, SANS CISSP® 10 Domains +S™, which covers the 10 Domains of Knowledge in a formal 10 domain structure. In the meantime, Clement Dupuis, Stephen Northcutt, Marcus Sachs, Bill Stearns and Joshua Wright are removing some of the 10 Domains material from SANS Security Essentials and returning it back to the original vision for that track, to fully cover the essentials of technical security. Moreover, SANS has insisted that the course teach the application of the 10 Domains in the workplace -- something no other 10 Domains course, including ISC2's does. This course meets the SANS promise: what you learn in the course you will be able to apply in the work place. One of the most important things I have learned from Alan Paller, Director of Research, in the years I have been involved with SANS is the importance of community consensus. In order to provide the highest quality training we have recruited experts to review the material and come to consensus on the course content and the application of the information. With the help of Zoe Dias, SANS Faculty Director, we enlisted a total of 68 reviewers from 10 countries. All but two of the reviewers are active CISSP's. The main author for the course, Eric Cole, has been a CISSP for almost 10 years.

SANS enthusiastically applauds the expert work of our technical reviewers/editors:

Monica Anklam, CISSP No. 31995, USA
Alex Arndt, CISSP No. 52343, Canada
Hank Askin, CISSP No. 40792, USA
Anjali Atanacio, CISSP No. 27039, USA
Jason Bevis, CISSP No. 35285, USA
Ron Black, CISSP No. 24245, USA
Anton Bojanec, CISSP No. 24560, Slovenia
Olufremi Bolanle, CISSP No. 51582, Nigeria
Jeff Bontsas, CISSP No. 39135, USA
Derek Browne, CISSP No. 26099, Canada
Sherry Callahan, CISSP No. 21760, USA
Ed Capizzi, CISSP No. 35909, USA
Jim Cate, CISSP No. 37031, USA
Patrick Chan, CISSP No. 40222, Canada
Jerry Chen, CISSP No. 47413, Canada
Daniel Cline, CISSP No. 31366, USA
Chris Cook, CISSP No. 38254, UK

Edwin Covert, CISSP No. 3597, USA
Phil Curran, CISSP No. 31708, USA
Edgar Danielyan, CISSP No. 42834,
UK/Armenia
David Dann, CISSP No. 51571, USA
Gary Delaney, CISSP No. 37636, USA
Sandeep Dhameja, CISSP No. 33585, USA
Joe Dial, CISSP No. 25358, USA
Heinz Durr, CISSP No. 42160, Switzerland
Darin Dutcher, CISSP No. 41299, USA
Rene Evers, CISSP No. 29057, USA
Chris Farrow, CISSP No. 45570, USA
Kenneth Fox, CISSP No. 42293, USA
Roger Fradenburgh, CISSP No. 28099, USA
Brian Freedman, CISSP No. 49504, USA
Donald Glass, CISSP No. 42244, USA
Mark Heinrich, CISSP No. 36190, USA

Lorna Hutcheson, USA
Lawrence Johnson, CISSP No. 25456, USA
Chaiw Kok Kee, CISSP No. 31589, Malaysia
Darrin Lau, CISSP No. 29948, USA
Eliot Leibowitz, CISSP No. 43782, USA
Steven Leong, CISSP No. 30313, Singapore
Chip Meadows, CISSP No. 10070, USA
Sean Mitchell, CISSP No. 36817, USA
Michael Morrell, CISSP No. 36227, USA
Pamela Nottage, CISSP No. 3758, USA
Sanjay Pandit, CISSP No. 44786, USA
John Pao, CISSP No. 29876, USA
Ariya Parsamanesh, CISSP No. 36074, AUS
Stephen Patton, CISSP No. 49746, USA
Robert Pfau, CISSP No. 21572, USA
Gabriel Proulx, CISSP No. 34018, Canada

Jim Purcell, CISSP No. 34519, USA
Andrew Salzman, CISSP No. 25162, USA
Amarottam Shrestha, CISSP No. 41671, AUS
Michael Solomon, CISSP No. 26517, USA
Robert Sorensen, CISSP No. 48304, USA
George Starcher, CISSP No. 34689, USA
Bruce Swartz, CISSP No. 46522, USA
David Taylor, CISSP No. 55890, USA
Brad Towers, CISSP No. 27957, USA
Jill Treu, CISSP No. 43196, USA
Tim Weil, CISSP No. 44250, USA
Deborah Weinstein, CISSP No. 44411, USA
Melody Wilson, CISSP No. 4130, USA
Steven Winterfield, CISSP No. 38096, USA
Kelli Wolfe, USA
Wayde York, CISSP No. 30404, USA

I have had the privilege of the best seat in the house and have really enjoyed working with the CISSP team. I sincerely hope that you benefit greatly from the information in these books and am very interested in your feedback. Please feel free to send me suggestions, corrections or questions to mgt414@sans.org.

Eric Cole, Senior Instructor and Research Fellow
The SANS Institute

4. Applications Security

10 Domains of Knowledge

This section covers Domain 4, the Applications and Systems Development domain.

Applications Security

- Security is most effective when planned and implemented throughout the entire life cycle.
- The goal is to ensure data and software integrity, confidentiality, and availability.
- Current applications and operating systems are vulnerable because adequate controls are not in place.

The Application System Development Process

When designing software, it is most effective to include the concept of application security throughout the entire development cycle—from the initial design layout to the final quality assurance testing—before going to market. In the software development process, you follow the same security goals as you do when designing a security infrastructure: confidentiality, integrity, and availability. This means the goal in the software development process is to ensure data and software integrity (Can the data be changed by anyone?), information confidentiality (Is the data viewable by anyone?), and information availability (Is the data available when you need it?).

Security Issues During Development

When developing software, several security issues must be addressed. It is common practice for application security to be an afterthought. In other words, security requirements are not integrated into the software until late in the development process. Security requirements should be collected and presented to the design and development team as part of the overall initial requirements gathering process. Adding security after the project is underway typically is ineffective. Retrofitting software with security features can lead to incomplete security controls, clumsy interfaces, poor performance, incompatibility with other features, and higher project costs.

Not all software developers have the experience in information security to adequately understand the implications of their software designs. Conversely, not all security professionals are software developers. It is ideal if a symbiotic relationship can be forged between these two groups, so that they work together to integrate security correctly and effectively.

Although the software developers and project managers might have their estimation on how long the software development process will take, in many cases management has a different timeline. Because of fierce competition in trying to get market share and recognition, many organizations push systems out too quickly without either having all of the security features installed and working or not fully testing the software to make sure the security features were implemented correctly.

Domain 4 Overview

- Application controls
- The software life cycle development model
- The software process capability maturity model
- Object-oriented systems
- Artificial intelligence systems
- Database systems and security issues
- Data warehousing and data mining

Overview

As specified in the CBK study guide from ISC2, "Applications and systems development security refers to the controls that are included within systems and applications software and the steps used in their development. Applications refer to agents, applets, software, database, data warehouses, and knowledge-based systems. The applications may be used in distributed or centralized environments."

As a CISSP candidate, you should be fully conversant with the security and controls that are used in the system development process. This includes the whole system life cycle, the type of controls that will be used within applications, how the changes are tracked through change controls, and the concepts related to data warehousing, data mining, knowledge-based systems, and different interfaces that are used. Configuration management plays a key role in all aspects of system design.

The last two bullets on the slide are becoming more important today with the introduction of dynamic web sites that make use of a database to store the dynamic content presented to visitors. Interfaces are often quickly developed and programmers do not realize that information might be exposed through the simple use of a web browser.

Application Controls

- Application controls that work over input, processing, and output
- Based on potential risks
- Selected from available controls
 - Administrative policy and procedure
 - Preventive, detective, and corrective measures

Application Controls

Any controls that are deployed are based on a risk analysis. You have to understand what the risk is, what the impact of the risk might be, and if it makes sense to implement the control in terms of cost.

Input Controls

Input controls are concerned with the validity and completeness of the information. Under this type of control, you find some of the following:

- **Limit or range tests:** Ensure a maximum amount is not exceeded.
- **Logical checks:** Are the dates valid, and is it the proper account type?
- **Self-checking digits:** Digits that have a math formula for validation, such as SSN (social security numbers) or credit card numbers
- **Control totals:**

Following are control types:

- **Transaction counts:** The total number of transactions performed
- **\$ total:** The total amount of a transaction
- **Cross footing:** The total should match with the value of items ordered times quantity.
- **Hash totals:** The sum of account numbers
- **Error detection and error correction:** Errors should be detected or corrected, and then logged.
- **Rejection and resubmission:** Invalid transactions are rejected and resubmission is always validated.

Output Controls

The output controls allow you to verify the accuracy of totals and completeness of the data. In this category of controls you find the following:

- **Reconciliation:** The act of ensuring two entities match
- **Physical handling procedures:** What type of physical security is used on printed documents?
- **Authorization controls:** Who has the authority to approve a specific transaction?

Processing Controls

The processing controls ensure that only valid transactions are performed, that limits imposed are not violated, and end results are verified. This is supplemented by audit trail mechanisms that might enable the detection of fraudulent transactions.

Application Controls Scope

- Distributed Environment
- Local/Non-Distributed

- Open Source
- Closed Source

- Coupling
- Cohesion

Distributed Versus Local

Application controls, as mentioned previously, are in accordance with the level of risks and the value of what is protected. A control is easier to implement if it is a standalone application that is not distributed and does not require network access or remote access from multiple locations.

Distributed environments are a greater challenge to security professionals because they require better identification, authentication, and access control mechanisms. A good example is the client server model that is commonly used today. Central servers are accessed from multiple users at multiple locations. In a distributed environment, the data is centralized or not centralized depending on the type of application used.

The notion of coupling and cohesion are often introduced when speaking about a distributed environment. The coupling mainly involves the dependencies of an application or system on other entities or modules. If coupling is low (the application does not have to depend or interact with other entities), you can claim high cohesion. If coupling is high (an application does depend on other modules or entities), the cohesion is low.

Distributed Environment

- Characterized by multi-location applications sharing information through a network.
- Pose a complex problem with regard to system management (and security management).
- Integrity controls must be in place.

Distributed Environment

A distributed application allows the sharing of information through a network. This is sometimes implemented through the use of agents on the different systems that access a central application. It is important not to mix an agent with a proxy. An *agent* is a process or program that performs a task on behalf of a subject in another environment. A *proxy* is different. It will perform a task on behalf of a subject; however, it hides the identity of the subject requesting the task performed.

As you can imagine, such a system needs strong integrity controls in place to avoid conflict between concurrent processes that might corrupt or damage the data. Most multi-user systems have these integrity mechanisms built in. The system must support configuration management for multiple systems, it must have the capability to establish and maintain connections in a secure way, it must have mechanisms in place to detect and prevent abuse through the network connections, data transferred must be protected, and it must have good backup and recovery measures (otherwise, multiple sites or users can be simultaneously affected).

Distributed System Requirements

- Portability
- Interoperability
- Transparency
- Extensibility
- Robustness and security
- Accommodation of standards
- Meet user's functional requirements
- Examples: Client/server systems, Distributed Data Processing (DDP)

Features and Requirements of a Distributed System

The norm in today's computing infrastructure is to hide the details of and implementation from the end-user. Portability ensures that an application easily adapts to different platforms with little effort. Applications that connect to different platforms should connect to these platforms in a unified manner, regardless the type of platform — Unix, Linux, Solaris, or Mac. With a known standard, it is possible for users not to know the type of system to which they are connecting. Standards also ensure that the data is reusable if an application reaches its end of life. A good example of this is information stored in a database that uses the SQL language versus another database that uses a proprietary format, which cannot be easily exported.

Our task as security professionals is to juggle security, users' functionality requirements, and the friendliness of applications that are developed. Too much security can affect the usability of an application; however, the other side of this is that if an application is shared between different environments or users, you can give access to users only on the basis of need-to-know, and you cannot allow for violators of the security policy that is in place.

Distributed Data Processing (DDP)

- Implemented in two ways:
 - Large central computer accessed by remote I/O devices (terminals, PCs, and printers)
 - A separate system at each location with its own copy of data where data is shared between locations
- In common:
 - Usually involves multiple locations where one or more hardware, software, data, policy, and IS functions can be distributed

Distributed Data Processing

In the days when our computers had 64K of memory and hard drives were non-existent for home PC, most of the data processing that took place was centrally performed on large mainframe computers. Security was easier to implement because you had to have physical access to an access terminal to do the processing on the data stored on the mainframe.

Today the challenge is different. There are partners, employees at home, and a sales force on the road — all of them want to access the data, regardless of physical or geographical location, which often requires a copy of the data. A Distributed Data Processing model is often used when you have an agency with different departments that are geographically disperse. Sharing common data among the locations is necessary, but so might the need to keep some of the data local. The shared data is kept at each of the geographical locations and synchronization occurs at regular intervals.

Some of the key advantages of DDP are:

- Better availability
- Economy through resource sharing
- Increased user involvement and control
- Distance and location independence
- Privacy and security
- Vendor independence

Some of the disadvantages of DDP are:

- Competent staff needed to diagnose failures
- More components also means more dependence on communication methods
- Possibility for incompatibility of data
- More complex systems management
- Difficulty controlling information
- Duplication of effort

Client-Server Requirements

- Fault tolerance
 - Hardware- and software-based solutions
 - Disk duplexing
 - Shadow database: Shadow of primary data base is placed on a separate computer.
 - Fail-over: Database operations continue on a second server if the first server fails.

Tolerance Requirements and Solutions

A client-server implementation is a hybrid approach that combines the advantages of distributed and centralized processing.

A client-server implementation might concentrate some of the resources on a server, which could become a single point of failure in the architecture. It is necessary to implement fail-over or load balancing mechanisms for uninterrupted processing. A large site often makes use of a cluster of servers that can share the load and allow for updates and maintenance without interrupting the service to the clients, customers, or visitors.

Data redundancy is achieved through the use of disk mirroring technologies that allow each single bit to be written to more than one drive or media at once. Other mechanisms based on RAID technologies allow data to be written to two or more drives. If one of the drives fails, some of the RAID implementations can reconstruct the data using a parity check.

Client-Server Requirements (2)

- Client-Server database system should support:
 - Shadow database: A shadow of primary database placed on a separate computer.
 - Fail-over: Database operations continue on a second server if the first server fails.

Tolerance Requirements and Solutions (continued)

Leading Database Management Systems (DBMS) also have their own redundancy mechanisms. Some make use of the built-in replication features whereby each transaction, input, or modification on one database is replicated to a backup copy of the database.

Database shadowing involves simultaneously working with one or more copies of a database. The master is the database that is normally accessed for all transactions or data retrieval. Each of the database copies are the shadows. Each change made to the primary database is replicated to the secondary copies of the database. This type of system allows for backups while the system is operational; there is no need to interrupt or shut down the database. Not all DBMSes support database shadowing.

The two most common mechanisms for protecting databases today are fail-over and load sharing, such as a cluster. Fail-over is a reliable mechanism, but one of the shortcomings is that one of the computers is not used for processing; it simply waits for the primary computer to become unavailable. Unused CPU cycles are not the best investment choice. A load sharing mechanism allows you to use one or more computers simultaneously, so that you can take advantage of the processing power on both computers. If one of the computers suffers from a failure, it is not visible to the end-user, the administrator is warned, and the problem can be fixed while maintaining availability through the other computer.

Peer-to-Peer

- Applications interact on an equal basis.
- Any platform can act as a client, a server, or both.
- It is well suited for situations in which there is a need to access or integrate data on multiple platforms and there is need for a common interface.

Peer-to-Peer

Peer-to-Peer is when applications have equal capabilities and responsibilities, and there is no main controller. Exchange is permitted between applications or systems even though there is no central enforcement that prevents such an exchange. In these cases, the control is delegated to each of the peers and security is difficult to maintain.

Today the term is often used to describe file-sharing programs, such as Kazaa, Gnutella, and others. They are large systems that offer a variety of similar data through a simple and common interface. Of course there are also other security issues with peer-to-peer networks, as demonstrated by the MyDoom virus's damage on Kazaa and other file-sharing networks.

Here is an extract from *PC Magazine*:

This week, as the MyDoom virus wreaks havoc on personal and corporate e-mail systems across the Internet, those words take on a new level of urgency. Apparently, MyDoom was originally let loose through Kazaa. And although the virus propagates predominantly via e-mail messages, it continues to worm its way through the most popular of the peer-to-peer, file-sharing services.

Each time the virus infects a Kazaa user's machine, it copies itself to the Kazaa download folder, assuming one of the following names: winamp5, icq2004-final, activation crack, strip-girl-2.0bdcom_patches, rootkitXP, office crack, or nuke2004. This folder, of course, is shared with the many millions of other people on the Kazaa network, and anyone who searches on those names—or something similar—might be fooled into downloading the virus.

DDP: Advantages

- Local control over data is enhanced.
- Data is readily available to users.
- It supports decentralized organizational structure.
- Productivity can be increased due to local data entry.
- It provides online editing and error-correcting features for data entry.
- Each user can control and schedule his own work.
- The physical distance between the user and the mainframe is transparent.
- Responsiveness to local conditions and needs.
- Minimization of the effect of system down time.
- Smaller investment in hardware for each site than for a central site.
- Decreased telecommunication costs.
- Increased capacity for telecommunications.
- Provides alternate processing locations in case one site's computer is down.

Advantages

DDP (distributed data processing) provides a whole range of benefits, such as mobility, lower cost of infrastructure by which users can work from home, system down time that does not render users idle, and significant savings resulting from communication costs.

As mentioned previously, this can be a likely scenario in which you have multiple offices that are geographically distributed but need to exchange data amongst them. Each of them maintains a local copy of the data and will synchronize at regular intervals with the main site if required. The scheduling of the updates can allow you to maintain a network link that has lower capacity and lower cost versus having the sites do the processing centrally.

DDP: Disadvantages

- Overall costs can be higher due to multiple locations.
- There is a possibility of outsiders breaking into the system.
- There is a possibility of invasion of computer viruses.
- There is a possibility of security breaches due to several network entry points and nodes.
- Security administration is more difficult.
- Data compatibility can be a problem.
- IS professionals might not be properly involved in system design and operation.

Disadvantages

DDP also has some downfalls. It is not always clear how to ensure and maintain all these entry points at all times. You can use a VPN or other secure links, but you must first ensure that the remote host is not already compromised and cannot be used as a gateway into the corporate infrastructure. This can be a security challenge.

One of the challenges I have seen with distributed processing is when you have multiple small sites that might not have competent network and system administrators to validate and protect the site' against unauthorized entry. This type of risk has to be evaluated and the likelihood of it occurring properly assessed. A big issue associated with this is the synchronization of data across multiple sites.

Centralized Versus Decentralized

Centralized: All IS work is performed in one place.

Decentralized: IS work is performed by each department or functional unit.

Centralized Versus Decentralized

A centralized approach might seem like the best approach to gain uniformity and consistency throughout an enterprise. However, there are cases when the use of a centralized approach might hinder security instead of improve it. A good example is a CA that issues digital certificates to end-users. The registration process will take place to request a certificate from the central CA. It might be difficult for a CA located in California to know if a user in New York is a valid company employee and that he has a need to get the certificate he requested. A good solution would be to have local registration authority in reach of the company locations where users will apply locally for a certificate and an onsite manager will approve the request. This can greatly speed up the time required to approve the request and ensure that only users with a need get the certificates.

Centralized Versus Decentralized Advantages

Centralized:

- Central control
- Technical competence, support, and service
- Technical R&D
- Centralized databases
- Professional work environment
- Career path development
- Comparative cost advantages

Decentralized:

- Local control of information resources
- Low cost of HW/SW
- End-user development of applications
- End-user control over system operations

Centralized Versus Decentralized (Advantages)

The previous gives you an overview of the advantages related to a centralized versus decentralized environment.

As you can see from a corporate perspective, the use of a centralized environment is simpler to manage, and better control can be implemented on all of the data and system operations. A centralized model tends to frustrate remote location administrators and users. The central administrators do not always fully understand the requirements of each of the locations. They often impose delay for changes to take place and they take away the flexibility of remote administrators. Once again, it is security versus usability. In a perfect world there is some centralized control, yet there are still decentralized functions in place.

Centralized Versus Decentralized: Disadvantages

Centralized:

- Loss of motivation by local employees and management
- Policies incompatible with local operating environments

Decentralized:

- Lack of career path development
- Higher cost of operation
- Loss of control of information resources
- Difficulty in implementing standards uniformly

Centralized Versus Decentralized (Disadvantages)

In a centralized model, there is often a lack of motivation from the employees and the management. This is driven by the fact that they are not managing "their" data and resources, but other people's resources. They often feel that the remote locations do not communicate their needs, and they often find out at the last minute that special requirements must be implemented. The overall central control might attempt to implement policies that might not be adequate for the specific environment at the remote sites. This is often a cause of frustration on both sides.

On the other end, being fully decentralized also presents some challenges. A small site with a single administrator might find out quickly that the administrator wants new challenges and will ask how his career is progressing, which is a valid question to which there is not a good answer. The administrator in this situation does not have a higher position to strive toward.

Decentralized environments always have difficulties in terms of uniformity. A kingdom within a kingdom develops. Administrators at remote sites who are technically competent might challenge the decisions and simply ignore directives that are issued. Although the central authority might perceive that proper control is deployed, in reality, this might not be the case.

Centralized Versus Decentralized Versus Distributed

- Centralized: Organized, controlled, and performed from one location
- Decentralized: Has multiple, independent locations with essentially no communications among them
- Distributed: Implies communications and coordination among multiple locations

We previously discussed the centralized and decentralized environment. We now introduce the concept of a distributed environment.

According to <http://www.whatis.com>, a distributed environment is defined as follows:

Computing is said to be 'distributed' when the computer programming and data that computers work on are spread out over more than one computer, usually over a network. Computing prior to low-cost computer power on the desktop, was organized in centralized 'glass houses' (so-called because the computers were often shown to visitors through picture windows). Although these centers still exist, large and small enterprises over time are moving (distributing) applications and data to where they can operate most efficiently in the enterprise, to some mix of desktop workstations, local area network servers, regional servers, Web servers, and other servers. A popular trend has been client/server computing which is simply the view that a client computer can provide certain capabilities for a user and request others from other computers that provide services for the clients. (The Web's Hypertext Transfer Protocol protocol is an example of this idea.) The Distributed Computing Environment (DCE) is a particular industry standard for implementing a distributed computing environment. Today, major software makers are fostering an object-oriented view of distributed computing. As a distributed publishing environment with Java and other products that help companies create distributed applications, the World Wide Web is accelerating the trend toward distributed computing and the view that, as Sun Microsystem's slogan says, "The network is the computer."

Modes of Operation

- Dedicated mode
- System high mode
- Compartment mode
- Multi-level secure mode (MLS)

Mode of Operations

Mode of operations is defined in the National Information Systems Security (Infosec) glossary as follows:

Description of the conditions under which an IS operates based on the sensitivity of information processed and the clearance levels, formal access approvals, and need-to-know of its users. Four modes of operation are authorized for processing or transmitting information:

- a. Dedicated mode*
- b. System high mode*
- c. Compartmented/partitioned mode*
- d. Multilevel mode*

Dedicated Mode

- The system is dedicated to and controlled by the processing at a single classification level of information.
- You need the level of the system to access the data.
- It is simple to create and can be done with most operating systems.

Dedicated mode is defined in the: National Information Systems Security (Infosec) glossary (<http://www.iwar.org.uk/infocon/terms/4009.pdf>) as:

IS security mode of operation wherein each user, with direct or indirect access to the system, its peripherals, remote terminals, or remote hosts, has all of the following:

- a. Valid security clearance for all information within the system.*
- b. Formal access approval and signed nondisclosure agreements for all the information stored and/or processed (including all compartments, sub-compartments, and/or special access programs).*
- c. Valid need-to-know for all information contained within the IS.*

When in the dedicated security mode, a system is specifically and exclusively dedicated to and controlled for the processing of one particular type or classification of information, either for full-time operation or for a specified period of time.

System High Mode

- System operates at multiple classification levels.
- You need the highest level of the system to access the data.
- Not all users have "need-to-know" access to the data.
- The system is simple to create and can be done with most operating systems.

System High Mode

System High mode is defined in the: National Information Systems Security (Infosec) glossary (<http://www.iwar.org.uk/infocon/terms/4009.pdf>) as:

INFOSEC mode of operation wherein each user, with direct or indirect access to the IS, its peripherals, remote terminals, or remote hosts, has all of the following:

- a. Valid security clearance for all information within an IS.*
- b. Formal access approval and signed nondisclosure agreements for all the information stored and/or processed (including all compartments, sub-compartments and/or special access programs).*
- c. Valid need-to-know for some of the information contained within the IS.*

Compartment Mode

- System operates at multiple classification levels.
- Each user with access has a formal approval and signed NDA.
- "Need-to-know" only for information they have access to.
- Hard to implement and most operating systems cannot implement it.

Compartment Mode

Compartment mode is defined in the National Information Systems Security (Infosec) glossary (<http://www.iwar.org.uk/infocon/terms/4009.pdf>) as:

INFOSEC mode of operation wherein each user with direct or indirect access to a system, its peripherals, remote terminals, or remote hosts has all of the following:

- a. Valid security clearance for the most restricted information processed in the system.*
- b. Formal access approval and signed nondisclosure agreements for that information that a user is to have access to.*
- c. Valid need-to-know for information that a user is to have access to.*

Multi-Level Secure Mode (MLS)

- Not all users have clearance for all information in the IS.
- System allows users to access information for which they have approval.
- "Need-to-know" only for information they have access to.
- MAC plays a critical role.

Multi-Level Secure Mode (MLS)

Multi-Level Secure mode is defined in the: National Information Systems Security (Infosec) glossary (<http://www.iwar.org.uk/infocon/terms/4009.pdf>) as:

INFOSEC mode of operation wherein all the following statements are satisfied concerning the users who have direct or indirect access to the system, its peripherals, remote terminals, or remote hosts:

- a. Some users do not have a valid security clearance for all the information processed in the IS.*
- b. All users have the proper security clearance and appropriate formal access approval for that information to which they have access.*
- c. All users have valid need-to-know only for information to which they have access.*

General Security Principles

- Authorization
 - All personnel should have proper authorization.
- Risk reduction
 - Code reviews
- Separation of duties
 - Development staff should not implement systems.
 - Development staff should not support production.
 - Development staff should not manage security functions.
- Accountability
 - No access directly to database
 - Production data managed by users, not support staff
 - All access to production data should be logged.
- Least privilege
 - Access given to necessary data only
- Layered defense
 - Multiple controls

A development environment is a nice example of separation of duties. A development environment should have clearly defined borders between the developers, the quality-assurance department, and the code or applications used on production systems. All code being developed must go through quality assurance (QA) before it's transferred to the librarian for release in production. All code going to production will come from the library because this ensures that you will be using a known good copy, and that your copy can be recovered from the library if a disaster occurs. It should never be possible for developers to work directly on code or data that is live production data. Any changes must go through a formal change control process and must be validated through the QA before being used in production.

As you know, it is sometimes necessary to implement hardware devices that will compensate for some of the weaknesses within operating systems and applications. Having a layered approach increases your security posture and requires more work for the adversary to get to your most important data. Do not rely on a single mechanism when you have the ability to use multiple mechanisms together to better enforce your security policy.

Application Controls: What, Where, and How

- | | |
|-------------------------------------|--------------------|
| • Preventive | Form of controls |
| • Detective | - Administrative |
| • Corrective | - Physical |
| • Apply controls to: | - Technical (most) |
| - Input | |
| - Processing | |
| - Data | |
| - Interprocessing
communications | |
| - Interfaces | |
| - Access control | |
| - Output | |

You can deploy different levels of controls across your organization. Each level has a different purpose and focus. It is important to remember that you should always use defense-in-depth measures and deploy multiple levels of controls.

Security Controls

- Input controls
- Output controls
- Transaction controls
- Process isolation
 - Failure of one component has minimal effect on rest of system.
- Hardware segmentation
 - Fences off memory used by the system
- Reference monitor
 - System component that mediates all access to objects
 - Mediates all access
 - Cannot be disabled
 - Thorough
- Security kernel
 - Implements the reference monitor
- Modes of operation
 - Dedicated
 - System high (not everyone has a need to know)
 - Multi-level
 - MAC

Security Controls

Security controls can take many forms and can be implemented at different layers of your systems. It can be at the hardware level, the operating-system level, or at the application layer.

I/O Controls

Input controls are concerned about the data being submitted. Input controls ensure that data is in the proper format, that special characters are not used, and it's used only by someone who is authorized. Input controls can be applied to data entered, data counted, and data edited. The output controls ensure only the proper people have access to the output, they can restrict access to the printed output storage area, and they can work as a document control system by which users must sign for sensitive output documents.

Other Controls

Operating systems provide protection between processes. Processes will run at different levels within a system: A process at one level should never affect a process at another level. Whenever processes at different layers want to communicate, they do so through a controlled interface using messages. A process that is sitting at one level and does not have any communication with other processes is a form of data hiding. Such protection mechanisms are in place to avoid having one process that could affect the entire system.

System architecture also provides some protection against common faults, such as bad memory. Most systems that are built have more than one memory bank. If one bank fails, it is still possible to continue processing by using the other memory bank. These memory banks are physically hard wired to the CPU, and addressing is based on the physical memory address versus the memory address mapping that takes place for applications.

The reference monitor concept is an important concept to understand. It is implemented by the security kernel, and it ensures that the reference monitor validates every single access request. The reference monitor itself is extremely compact to ensure that it can be audited through a thorough analysis that guarantees no tampering has occurred.

Security Control: Change Control

- Changes controlled to ensure:
 - Approved
 - Incorporated properly
 - No original functionality affected adversely
- Ensure security policy can still be implemented and security mechanisms are not negatively affected.
- Changed systems might require certification or accreditation.
- Occurs within configuration management framework
- Process
 - Make formal request.
 - Analyze: Review security implications.
 - Record change requests.
 - Submit for approval.
 - Develop the change.

Change Control

Change control is one of the weakest areas of all organization. In some cases, the change control process is so weak that it would be impossible to reconstruct the environment if a disaster occurred. Changes have to be done in a controlled manner; they have to be documented and submitted for approval. After they are analyzed and approved, they have to be implemented. One step that is not mentioned in the slide that could save you trouble is to have a strong backout plan. The mere fact that you have tested the change in your test environment does not guarantee that it will work in production. A backout plan allows you to return to a known good state and resume operation from that point. This has to be extremely well planned because there are updates, such as patches, which cannot be reversed unless you undo the changes manually, and you know what changes were performed.

The Development Process

The basic steps are:

- Project initiation
- Design analysis
- System design specifications
- Programming and testing
- Installation and implementation
- Operation and maintenance
- Destruction

The Development Process

The development process that different methodologies follow are all similar. In the upcoming slides, we give you more details about each of these steps and tell you what is being addressed in each step, from the birth of a project up to its retirement or destruction.

Security Perspective

Establish need (Project Initiation):

- Perform security risk assessment to define:
 - Sensitivity of information
 - Criticality of system
 - Security risks
 - Level of protection needed
 - Regulatory/legal/privacy issues

Determine what solution should do (Functional Design):

- Determine an acceptable level of security risk.
- Identify security requirements and controls.

Design (Design Specification):

- Design security controls.
- Review design.

Build (Software Development):

- Document security issues and controls.
- Conduct code walk-throughs.

Test (Software Development):

- Review tests.
- Certify system.

Field (Installation/Implementation)

- Accredited.
- Properly configure system.
- Begin configuration management of fielded releases.

Maintain (Operations and Maintenance):

- Constantly assess security posture.

Retire (Destruction)

The first step in a development project is to initiate, plan, or investigate the needs that have to be addressed by the proposed development effort. A quick risk assessment is performed and a high-level description of the system is built. At this point, there is not a whole lot of details because you are still at the conceptual design level.

Functional Design

After you get into the functional design, you attempt to create a list of required characteristics from the system you are about to build. In this phase, identify what are the acceptable risks and what security controls must be put in place. This portion includes the functional specification, such as details about authentication, authorization, access control, confidentiality, audit, integrity, and availability. The test plan is defined here as well. Again, this is still a high-level document that does not contain all the details. This is the type of information that the marketing department (selling) or people involved in the product concepts could used.

Design

In the design process, we get into more details: An entity relationship diagram is built, data flow is defined, database schema, security controls, and other internal and external details of how the new product will be implemented. After the design is completed, it is submitted to other teams for review. It is always good to get a third-party's opinion because it might detect shortcomings or problems before the coding process starts.

Project Initiation

- Information security team should be involved in the initial discussions.
- Perform a risk assessment:
 - Define sensitivity of information.
 - Define criticality of system.
 - Define security risks.
 - Define level of protection needed.

Project Initiation

The first step in a development project is to initiate, plan, or investigate the needs that have to be addressed by the proposed development effort. A risk assessment is performed and a high-level description of the system is built. At this point, there are not many details because you are still at the Conceptual Design level.

Security features must be addressed at the onset of a new project. By doing so, you will have a system that provides an adequate level of security, and it will cost you much less than adding it at a later stage in the project.

Design Analysis

- Define functional and system design requirements.
- Determine acceptable risk levels:
 - Level of loss
 - Percentage of loss
 - Permissible variance

Functional Design

After you get into the functional design, you will attempt to create a list of required characteristics from the system you are about to build. In this phase, you identify what the acceptable risks are and what the security controls are that must be put into place. This portion includes the functional specification, such as details about authentication, authorization, access control, confidentiality, audit, integrity, and availability. This is also where the test plan is defined. Again, this is still a high-level document that does not contain all the details. The marketing department (selling) or people involved in the product concepts can use this information.

System Design

In the design process, we get into more details: An entity relationship diagram is built, data flow is defined, database schema, security controls, and other internal and external details of how the new product will be implemented. After the design is complete, it is submitted to other teams for review. It is always good to get a third-party's opinion because it might detect shortcoming or problems before the coding process starts.

Design Analysis Security

- Identify security requirements and controls:
 - What are the exposure points?
 - What are the controls to mitigate these exposures?
- Determine whether the application meets the security requirements.

Design Analysis Security

In this phase, the security requirement is identified. You can derive the need yourself; however, in some cases, it might be imposed by standards or code of practice for a specific industry. Of course, the controls being implemented would be in direct relation to the value and sensitivity of the data being protected. The bottom line is to always perform a cost-benefit analysis where the cost of the control must be compared to the value of the data to come out with a business decision, whether to implement the controls. If the cost outweighs the benefits, it will most likely be rejected and management will *not* implement it.

System Design Specifications

- Design:
 - Functional components
 - Program controls
 - Security mechanisms
 - Test plan
- Design methodology

Design Specification

In this phase, you look at the type of information to be processed and what type of processing will take place. You define the tasks and functions that the application must perform on your behalf. In this phase, you also define the controls that are being used, specify what type of access subjects will have to objects, and see if there is a need to protect the data either in transit or in storage. The previous phase defined the required functionality, so this phase defines the mechanisms to implement the required functionality. For example, if authentication was one of the functional areas that was identified, it would be defined here how the authentication mechanisms will be implemented and how they can be tested. Finally, a last look ensures that the product meets the specifications laid out in the project initiation phase.

Programming and Testing

- The actual programming effort
- Programmers follow the design specification.
- Code should be tested for possible problems:
 - Buffer overflow
 - Prevent covert channels
 - Proper data type checking
- Proper code reviews should be performed.

Building, Programming, Coding

This is an important step. In many badly designed projects, this is the first phase that is performed. Surprisingly, lots of people are still coding today without having completed a proper evaluation of their needs, functionality requirements, and design. As far as many developers are concerned, this is where they would like to start. However, to have a product that meets the requirements, it is a must to complete the other phases.

This phase is where the actual software development takes place. Modules are developed and tested as they are produced. Secure coding practice should be used in this phase. Your programmers might not all be security specialists, so they need assistance in identifying some of the security issues and controls that should be in place. After all the modules are coded and the product is complete, it is a common practice among large government environments or a large enterprise to perform a code walkthrough where each block of code is examined and a debate takes place among experts to see if it is the best way to implement it. A code walkthrough requires plenty of effort and, sometimes, small companies do not perform it because of cost restrictions.

Programming and Testing: Separation of Duties

It is important to implement separation of duties:

- Separate development, testing, and production teams.
- Developers should *not* have access to the production code.

Three Distinct Environments

As previously mentioned, the notion of separation of duties is of utmost importance in software development. Three distinct and separate environments will be maintained at all times and no direct interaction is allowed between the three. Software developers hand over their work to the Quality Assurance (QA) and Testing department, and after the quality is validated, it can be used in production. Developers should *never* have access to live production code or data; if there is a need to use data similar to the production data, it will always be data that has been screened and sanitized before it's submitted to the developers.

Installation and Implementation

- Focus on use and operation of system or application.
- Develop user manuals and maintenance manuals.
- Certify system and application.
- Management accepts system.

Installation and Implementation

This is where the system will be deployed in production. Training takes place to ensure that support personnel have the required knowledge to properly configure and manage the application or system. After the entire system is complete, tests that were developed in the functional design phase are performed to ensure that the product does what it is supposed to do, it does the functions securely, it does it with accuracy, and that it can do it reliably under all circumstances. All the security features are specifically tested to ensure that the system meets its security requirements. Certification takes place at this point.

After the system is fully configured and ready for processing, the accreditation process takes place. This is where management looks at the system, the way it provides services, the way it is physically secure, the way it is maintained, and grants authority for processing if all criteria are satisfied.

Operation and Maintenance

- Configure the new system.
- Perform periodic vulnerability assessments.
- Monitor system activity.
- Review event logs.
- Record any changes and perform new risk analysis.
 - Change control review.

O&M

As the system goes into wide usage, bugs might be identified, or there might be changed conditions that require updates to some of the controls. It is important to constantly evaluate what the risks are, see how they are addressed, and, if needed, implement supplementary features or controls to properly protect the system.

Destruction

- Used when a system or application will be replaced.
- Disposal mechanisms vary depending on sensitivity issues.
- Information can also be archived, transferred, or destroyed.
- Deals with how sensitive data is destroyed.

Destruction

How we deal with the replacement and destruction of old applications is just as important as the development process itself. Simply deleting applications and data from hard drives is not good enough. With the right tools, deleted data can be extracted from hard drives, even if the drive has been formatted. The more sensitive or confidential the data, the stronger destruction methods should be used. There are many effective ways to delete electronically stored information, including degaussing and multiple overwriting. Paper-based information should be shredded.

You must look at the data stored on the system: Is there a need to salvage the data for reuse on another system that is more recent? Is there a need for data conversion? Proper steps in this phase help avoid embarrassing situations, such as a leading bank that had its systems recycled and sold over auction on the Internet that still contained customers' sensitive financial information.

Sample Operational Controls

- Personnel background checks
- Separation of duties
- Risk reduction
- Accountability
- Least privilege
- Layered defense
- Change control

Background Check

Personnel security starts even before you hire someone on staff. It is a must that you get in touch with previous employers of potential employees. References are always fine, but potential employees rarely list the name of someone that might criticize them. Sometimes, a simple call to a previous supervisor or human resources representative might reveal interesting facts.

Operation Controls: Risk Reduction

- Risk reduction:
 - All code should be reviewed before implementation (change management).
- Accountability:
 - Production data should be managed through programs:
 - No access should be permitted directly to database.
 - All access to production data should be logged.

Risk Reduction

All code should be reviewed before implementation. Changes to code should follow a documented change management procedure. Having developers make arbitrary changes to code is a detrimental thing. Not only do these changes potentially change the scope of the program itself, but they also lead to the introduction of unwarranted vulnerabilities. One classic example is the addition of a new feature. Perhaps the programmer had good intentions when she thought that adding this new feature would enhance the product, but she did not consider that this new feature was not in the original software-design specification, management did not authorize the addition of this new feature, and this new feature introduces a new vulnerability that can easily be exploited.

Accountability

When a change is needed, it should be documented and logged. There are many horror stories of changes made to systems and software, and there is no way of determining when the changes were made or who made them. All changes and modifications should be logged; the more critical the system, the more important it is that log changes are maintained.

Operation Controls Concepts

- Least privilege:
 - Access control
 - Necessary data fields only
- Layered defense:
 - Use access controls in addition to system access.

Least Privilege

The least privilege principle is always important, whether you are configuring a firewall, distributing cards for physical access, or developing an application. A user or subject should always get only the minimum level of privilege required to complete a task. This is extremely important in the case of applications that have vulnerabilities; if the application is running with little or no system privilege, the attacker can do little damage. However, today there are still many applications that do require administrator privileges and a vulnerability within such an application can render your system at very high risk.

Layered Defense

In a layered defense, also called defense-in-depth, you do not rely on just one security control to provide all the necessary security. It is better to have many layers of security in place. This defense provides some assurance that if one mechanism fails, another mechanism is still in place to provide some security protection.

Change Control

- Changes must be authorized, tested, and most importantly, recorded.
- Changed system might require re-certification or accreditation.
- Evaluate change control during system audits.

Change Control

Changes should be implemented in a structured manner, documented, and approved. A proper recovery scenario should also be in place in case trouble occurs while you're implementing the changes. A good backout plan is always optimal. In systems that have been accredited, any changes might require retesting of the specific system. Change control is an important part of a company's survival. Without proper change control, it might be impossible to recover after a disaster occurs.

Software Life Cycle Development Process

- Waterfall model
- Spiral model
- Top-down development
- Bottom-up development
- Hybrid development
- Rapid prototyping model
- Object-oriented development

Software Development Life Cycle Process

In this section, we briefly discuss the various software-development models. According to Russell Kay in an article he wrote for *ComputerWorld* magazine

(http://www.computerworld.com/developmenttopics/development/story/Q.10801,71151_OO.html):

"[The] System Development Life Cycle (SDLC) is the overall process of developing information systems through a multi-step process from investigation of initial requirements through analysis, design, implementation and maintenance. There are many different models and methodologies, but each generally consists of a series of defined steps or stages."

Because the software-development process is complex, it requires teams of software architects, programmers, testers, and users. They all need to work as a team to create programs, sometimes with millions of code lines. To manage this complex process, a number of SDLC models have been created. In this section, we briefly cover the waterfall, spiral, top-down, bottom-up, hybrid, rapid prototyping, and object-oriented models. Our goal is not to teach you the intricate details of the software-development process; it's to highlight the various methodologies to give you an idea of the various options. Many other SDLC models are available, including build and fix, incremental, and synchronize and stabilize. •

Waterfall Model

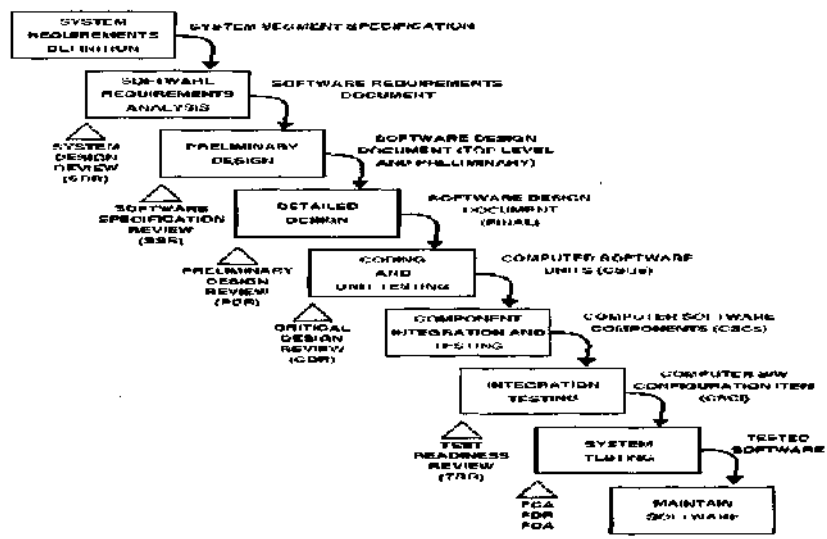
- Phases occur in succession, like water falling down a waterfall.
- After each phase is completed, it is closed and not revisited.
- There is no customer involvement.
- There is no going back.

"Waterfall" Also Called "Traditional Method"

The waterfall methodology divides a software-development project into well-defined sequential stages with specific milestones at each of the stages. After all the phases are complete, the product is delivered. This methodology is known to be the most direct toward the objectives with the shortest development time and cost possible. Some drawbacks exist in this method as well. For example, there is little flexibility in changing the scope of a project because you can only revert back one stage and no more. If there are system shortcomings, they might not be discovered until the product is finally released for use in production.

Included in the waterfall model are verifications and validations. The verifications ensure that the product being developed meets the specifications. The validation process ensures that the product solves a real-world problem or its operational mission.

Waterfall Model



Waterfall Model

In the waterfall model, the software-development phases occur in succession, like water falling over a waterfall. In the classic waterfall model, after a phase is completed, it is closed and cannot be revisited. A variation in this model allows developers to go back one phase for rework, verification, and validation. It also assumes that one phase starts when the previous phase is completed. This is seldom true in real-world development projects.

Spiral Model

- Phases occur in order, but in an ever-widening spiral of larger and larger activity.
 - Phases are repeated over and over.
- Risk is a driving factor behind the spiral model.

Spiral Model

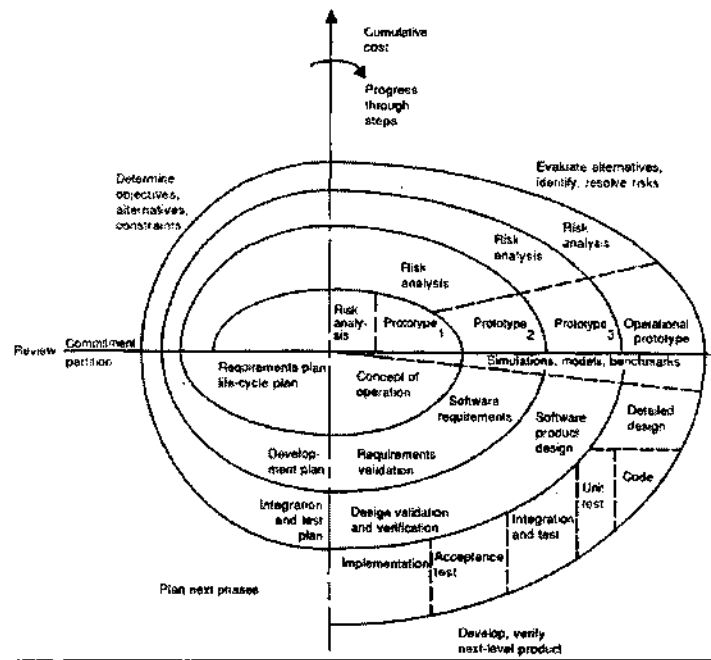
The spiral model is driven by risk. It guides all the people taking part in a software-development project of a large system. The model has two main features: a cyclic approach and a set of anchor-point milestones. The cyclic approach is the spiral; you start at the center of the spiral where the room for definition is not wide, and as you move outward on the spiral, the degree of definition and implementation increases. The set of anchor-point milestones ensure that all stakeholders in the project are committed to feasible and mutually satisfactory system solutions.

In the spiral model, risks are situations or possible events that can cause a project to fail to meet its goals. The risks are classified according to their impact and their likelihood. The impact scale is from trivial to fatal, and the likelihood scale goes from certain to improbable. Within this methodology, a risk-management plan identifies and enumerates the risks and prioritizes them in order of importance, as measured by a combination of the impact and likelihood of each. For each risk, the plan also states a mitigation strategy to deal with the risk. For example, the risk that technology is unready can be mitigated by an appropriate prototype implementation in an early spiral cycle.

A process model such as the Spiral Model answers two main questions:

- What should be done next?
 - For how long should it continue?

Under the spiral model, the answers to these questions are driven by risk considerations and vary from project to project, and sometimes, from one spiral cycle to the next. Each choice of answers generates a different process model. At the start of a cycle, all the project's critical stakeholders must participate concurrently in reviewing risks and choosing the project's process model accordingly. (Risk considerations also apply toward ensuring that progress is not impeded by stakeholders' overparticipation.)



Spiral Model

In the spiral model, the development phases occur in order, but in an ever-widening spiral of larger and larger activities. The model states that each cycle of the spiral involves the same series of steps for each part of the project.

Signoff inside the model, no design review, and customer involvement helps lower the risk of this model compared to the waterfall model.

Top-Down Development

- Like building a house
- Starts at the top: The highest, broadest, most encompassing view of the whole system
- Decomposes into subsystems, so that the system fans out like an inverted tree or an organizational chart

Top-Down Development

In the top-down development model, the project starts at the top—the highest, broadest, most encompassing view of the entire system. Then, the project is broken into smaller subsystems. The project flow fans out like an inverted tree or an organizational chart.

Bottom-Up Development

- Starts with fundamental, primitive pieces that accomplish one small part of a larger whole
- Great for smaller systems

Bottom-Up Development

An approach to design in which you start your development with the low level details, such as small utilities, and later on you decide how you will bundle all these utilities to create higher-level components and finally the entire system by assembling them all together. This type of design allows for the reuse of some of the low-level components throughout the entire system.

Hybrid Development

- Also called "inside out"
- Blend of top-down and bottom-up
- You work from both ends and arrive at the middle.

Hybrid

Both bottom-up and top-down development approaches have their advantages and disadvantages. A third development approach exists, which is a mix of both the top-down and bottom-up styles. Some efforts are deployed to develop small utility components while high-level issues are also addressed simultaneously.

Rapid Prototyping Model

- Involves quickly building a dummy system that presents what the system might do or how it might appear, but not doing it in reality
- Usually shows GUI with input and output, but no processing or functionality
- XP - extreme Programming

Rapid Prototyping

In the rapid prototyping model, the goal is to quickly develop a dummy system that represents what the system might do or what the user interface will look like. In this model, you typically don't use real data—just dummy data. The intent is to validate the design, throw the prototype away, and start over with the true development project.

In today's world of graphical user interface (GUI), this is a common development approach. The prototype is often developed in days (as opposed to weeks) using easy programming tools, such as Visual Studio or another leading RAD tool. The client gets a prototype that can be evaluated for look and feel. Unfortunately, often the looks are evaluated first, and its functionality is looked at second.

Object-Oriented (OO) Development

- Uses OO methodologies, such as classes, objects, attributes, and methods
- Supports modeling of the "real world"
- An object-oriented system can be thought of as a group of independent objects that can be requested to perform certain operations or exhibit specific behaviors.

Object-Oriented (OO) Jargon

Object-oriented methodology is based on Object-Oriented Paradigm. Our world is a collection of collaborating objects and agents. The software developed with OO methodologies are organized according to the structure of our world.

Class

A class is a blueprint or prototype that defines the variables (data) and methods (code) common to all objects of a certain kind. The class serves as a user-defined type. A class thus defines a data type that behaves like the built-in types of a programming language.

Object

An object is a concrete instance of a class. The data within an object is referred to as variables or attributes.

Methods

Methods are functions within an object that can perform functions or manipulate the object variable and perform an operation relevant to the object.

Messages

Objects communicate with one another by sending messages from a statement in one object to a method in another object. They are often requests for an object to perform some operation on the data stored within the object.

Object-Oriented Systems

- An object can be viewed as a "black box" whose internal details are hidden from outside observation and cannot normally be modified.
- Key terms
 - **Message**
 - The communication to an object to carry out some operation
 - **Method**
 - The code that defines the actions an object performs in response to a message
 - **Behavior**
 - Results exhibited by an object upon receipt of a message
 - **Class**
 - A collection of the common methods of a set of objects that defines the behavior of those objects

The Black Box

An object is a "black box" that receives and sends messages.

A black box actually contains *code* and *data*. In traditional programming languages, such as C, the code and data portion are kept apart. The code units are called *functions*, although the data units are called *structures*. In OO, both sit within an object; in object-oriented programming, there is *never* a need for a user to look inside an object. All communications are done via messages. The object getting a message is called the *receiver*. Message format defines the interface to the object. Every function that an object can do is addressable through its message interface.

When you provide access to an object strictly through messages while keeping all the details secret, that is called *information hiding* (also known as encapsulation).

"One of the fundamental principles of object technology is that the internals of an object are private to that object and may not be accessed or even examined from outside the object."

—David Taylor, Author, *Business Engineering with Object Technology* (Wiley, 1995).

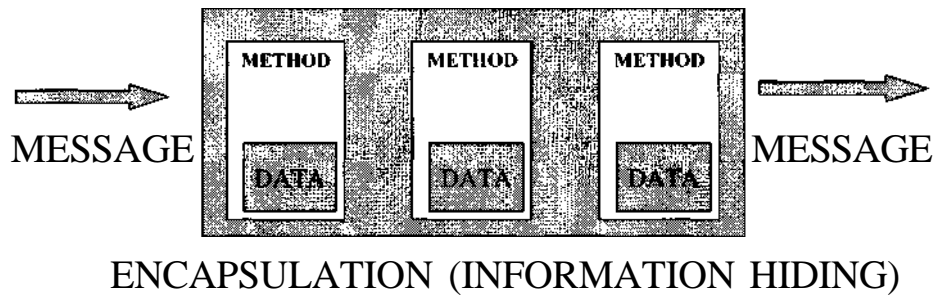
Object-Oriented Systems (2)

- **Definitions**
 - Delegation
 - The forwarding of a request by an object to another object or *delegate*
 - Necessitated by the fact that the object receiving the request does not have a method to service the request
 - Polymorphism
 - An object that is able to respond to some common set of operations in a different way"
 - Ability to use the same syntax for objects of different types, for instance, "+" for addition of reals and integers
 - Binding
 - Dynamic—Association of a method with a message during run time
 - Static—Association of a method with a message during compile time
 - Polyinstantiation
 - Development of a detailed version of an object from another object using different values in the new object
 - In database information security, this term is concerned with the same primary key for different relations at different classification levels being stored in the same database.

Parametric polymorphism allows the same object code for a function to handle arguments of many types.

Polymorphism is similar to a program performing a different algorithm for multiplying floating point or integer numbers. The program responds to the "multiply" function in different ways.

Object-Oriented Systems (3)



Encapsulation

Encapsulation is a simple and effective building tool. You package information within an object in such a way that users cannot access this information. The code and data within the object is protected. The only way of interacting with the information is by interfacing with the services that the object provides through proper messages.

One of the key advantages is that the programmer can update the code within the object and the application will continue to operate properly by sending the proper message to the object. However, there might be some performance improvement for whatever was fixed by the programmer.

Some advantages of encapsulation are:

- Managing complexity
- Managing change
- Protecting data

Object-Oriented Systems (4)

- Object: function + data
- Concepts
 - Class
 - Instance
 - Methods
 - Inheritance
 - Encapsulation
 - Polymorphism
- Advantages
 - Reusability
 - Reduces development risks
 - Model of the real world
- Security aspects
 - Security controls for program library
 - Communications between objects
 - Access control by class: public versus private

Overview and Review

Here is an overview of what we have seen so far in object-oriented development. A few terms are new, but do not worry, we describe them here.

Inheritance

<http://www.whatis.com> defines inheritance as the following:

Inheritance is the concept that when a class of object is defined, any subclass that is defined can inherit the definitions of one or more general classes. For the programmer, this means that an object in a subclass need not carry! its own definition of data and methods that are generic to the class(es) of which it is apart. This not only speeds up program development, it ensures an inherent validity to the defined subclass object (what works and is consistent about the class also works for the subclass).

Polymorphism

Inheritance is an easy concept to understand. Polymorphism, on the other hand, is much more difficult. *Polymorphism* is about an object's ability to provide context when methods or operators are called on the object. Polymorphism (from the Greek word meaning "having multiple forms") is the characteristic of being able to assign a different meaning to a particular symbol or "operator" in different contexts.

Object-Oriented Systems (5)

- Enforces security controls
 - Abstraction: Unnecessary details are suppressed.
 - Data hiding
 - The object performs the "what" without revealing the "how."
 - Presenting what the user needs, not actual data
- Enforces good design principles
 - Tight cohesion: Can perform a single task with little or no help from others
 - Loose coupling: Little interconnection

Abstraction

Abstraction refers to the act of representing essential features without including the background details or explanations. It is the art of concentrating on the essential and ignoring the nonessential. Classes that use the concept of data abstraction are known as *abstract data types*. Abstract data types are achieved by making certain variables and methods in a class private.

Cohesion and Coupling

Cohesion and *coupling* are directly related. Objects are self contained and can perform a single task with little or no help from other objects (low coupling). Because of the low level of interaction necessary with other objects, it is said that they have a high or tight cohesion level. An object that has a need to interact with multiple objects to perform a task has a high coupling level and inversely a low cohesion level. Having interactions is sometime necessary and not *always* a bad design decision; however, too many dependencies can make a program or application less reliable.

Object-Oriented Systems (6)

- Relative to the software development life cycle phases, object-orientation is applied in different phases.
- Object-oriented requirements analysis (OORA):
 - Defines classes of objects and their interactions

OORA

We have seen the phases of a normal development project and its different phases. When using OO, different phases are used.

Object-Oriented Requirements Analysis (OORA): This is where classes of objects and the interaction between them are defined.

Object-Oriented Analysis (OOA): In terms of object-oriented concepts, understanding, and modeling a particular problem within a problem domain.

Domain Analysis (DA): Seeks to identify the classes and objects that are common to all applications within a given domain.

Object-Oriented Design (OOD): The object is the basic unit of modularity; objects are instantiations of a class.

Object-Oriented Programming (OOP): Emphasizes the employment of objects and methods rather than types or transformations, as in other programming approaches.

Object-Oriented Systems (7)

- Object-oriented analysis (OOA)
 - Understanding and modeling a particular problem within a problem domain in terms of object-oriented concepts

OOA

Object-Oriented Analysis (OOA): In the context of OOP, this is the understanding and modeling a particular problem within a problem domain. Object modeling is the central technique; it is a language-independent notation that allows the specification of classes, their data or attributes (private) and methods (public), inheritance, and other more general relationships between classes.

OO requires more work upfront in defining the classes that are part of the system and all the requirements. However, it does pay big dividends in the long term to invest some time on this portion of development.

Object-Oriented Systems (8)

- Domain analysis (DA)
 - According to Booch, "Whereas OOA typically focuses upon one specific problem at a time, domain analysis seeks to identify the classes and objects that are common to all applications within a given domain."

Domain Analysis

In Object-Oriented Analysis (OOA), the purpose is to determine what are the software or system functions common to a domain. A domain could be a specific field, such as air-traffic control, healthcare, or another type of environment. By doing so, you generate general classes that can be reused in a wide variety of problems' instances. Domain analysis is usually better accomplished with experts in the specific domain (see the following Booch note).

Booch note: "It is truly amazing to see what a little bit of domain knowledge can do to assist a developer in making intelligent design decisions."

Object-Oriented Systems (9)

- Object-oriented design (OOD)
 - An object is the basic unit of modularity.
 - Objects are instantiations of a class.
- Object-oriented programming (OOP)
 - It emphasizes the employment of objects and methods as a programming approach.

OOD and OOP

By now, you should be familiar with the concept of Object-Oriented Design (OOD) and Object-Oriented Programming (OOP). It is a different approach to programming, where classes mapping to real-world problems are defined and objects are instantiations of these classes. The use of an object can speed programming development and facilitate some of the development functions. This approach avoids reinventing the wheel for every software-development project.

Object-Oriented Systems (10)

- Objects can be made available to users through object request brokers (ORBS).
 - ORBS act as the locators and distributors of objects across networks.
 - ORBS are considered middleware because they reside between two other entities.
 - ORBS can also provide security features, or the objects can call security services.

ORB

The ORB concept was published years ago. When the concept first came out, it was extremely promising. The primary aim was to give every user the same amount of processing power, regardless of the type of PC sitting on his desk. After you had an application that was ORB aware, the application would locate the computer across the network that had the most CPU cycles available and the processing would be redirected to that machine. However, because of some complexity in its implementation, it took a long time to be widely deployed and used. Today, this technology has lots of implementations out, making it the most popular.

Object-Oriented Systems (11)

- Common Object Request Broker Architecture (CORBA)
 - Developed by the object management group (OMG)
 - Defines an industry standard that enables programs written in different languages and using different platforms and operating systems to interface and communicate
 - For example, CORBA can enable Java code to access and use code written in C++.

CORBA

On its web site (<http://www.omg.org/>), the Object Management Group (OMG) describes CORBA as follows:

CORBA is the acronym for Common Object Request Broker Architecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol HOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.

CORBA is useful in many situations. Because of the easy way that CORBA integrates machines from so many vendors, with sizes ranging from mainframes through minis and desktops to handhelds and embedded systems, it is the middleware of choice for large (and even not-so-large) enterprises. One of its most important and most frequent uses is in servers that must handle large number of clients at high hit rates with high reliability. CORBA works behind the scenes in the computer rooms of many of the world's largest web sites; ones that you probably use every day. Specializations for scalability and fault-tolerance support these systems. But it's not used just for large applications; specialized versions of CORBA run real-time systems and small embedded systems.

Object-Oriented Systems (12)

- The common object model (COM):
 - Supports the exchange of objects among programs
 - Formerly known as object linking and embedding (OLE)
- The distributed common object model (DCOM):
 - Defines the standard for sharing objects in a networked environment
- Examples of object-oriented systems are Simula 67, C++, and Smalltalk.

COM

COM, an open-software architecture from DEC and Microsoft, allows interoperation between ObjectBroker, which is a CORBA implementation, and Object Linking and Embedding (OLE), which is available on Windows platforms. When an object is linked, simply clicking the object starts the appropriate application to look at, modify, or change the object. When an object is embedded within another document, you can modify the object within the document without the need to call the other application (in some cases). OLE works only on local systems and does not work across networks.

DCOM

The following paragraph is the DCOM description given on the Microsoft web site:

The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. Previously called "Network OLE," DCOM is designed for use across multiple network transports, including Internet protocols such as HTTP. DCOM is based on the Open Software Foundation's DCE-RPC spec and will work with both Java applets and ActiveX® components through its use of the Component Object Model (COM).

DCOM is an extension of the COM model previously described. At one point, someone thought that it would be neat to be able to share objects with remote systems instead of just with the local system. DCOM has been submitted to the IETF as a draft standard. DCOM runs primarily on the Windows platform and a few others through third-party software.

Artificial Intelligence Systems

- Systems attempt to mimic the workings of the human mind.
- Two types of artificial intelligence systems are covered in this section
 - Expert systems
 - Neural networks

This is an alternative approach for using software and/or hardware to solve problems.

Systems attempt to mimic the workings of the human mind.

Two types of artificial intelligence systems are covered in this section:

- Expert systems
- Neural networks

Artificial Intelligence Systems (2)

- Expert system
 - Exhibits reasoning similar to that of a human expert to solve a problem
 - Computer programs are usually defined as:
 - Algorithm + data structures = program
 - In an expert system, the relationship is:
 - Inference engine + knowledge base = expert system

Accomplishes reasoning by building a knowledge base of the domain to be addressed in the form of rules and an inferencing mechanism to determine if the system input has satisfied the rules.

Attempts to duplicate how a human solves a problem.

Usually a knowledge engineer will interview the experts and try to extract their thinking process in solving a problem, making a diagnosis, or designing a system.

Artificial Intelligence Systems (3)

- Knowledge base
 - Contains facts and the rules concerning the domain of the problem
 - Form of *if-then* statements
 - Inference engine compares information it has acquired in memory to the *if* portion of the rules in the knowledge base to see if there is a match.
 - If there is a match, the rule is ready to "fire" and is placed in a list for execution.

Certain rules may have a higher priority or salience, and the system will fire these rules before others that have a lower salience.

A knowledge base of rules is assembled that will react to real world inputs to see if what is occurring matches one or more rules. Then, the rules will "fire."

Artificial Intelligence Systems (4)

- Expert system operating modes
 - Forward chaining
 - Expert system acquires information and comes to a conclusion based on that information.
 - Backward chaining mode
 - Expert system back tracks to determine if a given hypothesis is valid.
- Knowledge acquisition in expert systems involves:
 - Interviewing experts in the domain field
 - Obtaining data from other expert sources

As with human reasoning, there is a degree of uncertainty in the conclusions of the expert systems. Uncertainty can be handled through Bayesian networks, certainty factors, and fuzzy logic.

Bayesian networks are based on bayes theorem:

$$P\{h|e\} [\text{equal}] p\{e|h\} * p(h) / p(e)$$

This gives the probability of an event (h) given that an event (e) has occurred.

Certainty factors are easy to develop and use. These factors are the probability that a belief is true. For example, a probability of 85 percent can be assigned to object a occurring under certain conditions.

Artificial Intelligence Systems (5)

- Verification and validation of an expert system:
 - Concerned with inconsistencies inherent in conflicting rules
 - Redundant rules
 - Circular chains of rules
 - Unreferenced or unallowable data values

"Cleaning up" the rules.

Verification and validation of an expert system:

- Concerned with inconsistencies inherent in conflicting rules
- Redundant rules
- Circular chains of rules
- Unreferenced or unallowable data values

Artificial Intelligence Systems (6)

- Neural networks
 - Analog of the biological neuron system
 - Inputs i_i to the neuron analog are modified by weights w_i , and then summed in unit q .
 - If the weighted sum exceeds a threshold, unit q will produce an output, z .

The value of a neural network is its ability to dynamically adjust its weights to associate the given input vectors with corresponding output vectors.

See the next slide.

Artificial Intelligence Systems (7)

The inputs i can be from image sensors, tactile sensors, and so on, and they are multiplied by weights to reflect their importance and relevance to the output. An input that is more important is given a higher weight than one that is less important. Then, all the weighted inputs are fed into a summing junction that will "fire" and produce an output if the sum of the inputs exceeds a certain threshold. If the desired output is not achieved, an algorithm automatically adjusts the weights and the system "learns" until it achieves the desired output.

Artificial Intelligence Systems (8)

- Networks with more than one level of summing nodes are called multi-layer networks.
- The "training" period for the neural network:
 - Input vectors are repeatedly presented.
 - Weights are dynamically adjusted according to the learning paradigm.
 - Delta learning rule is an example of a learning rule.

Networks with more than one level of summing nodes are called multi-layer networks.

The "training" period for the neural network:

- Input vectors are repeatedly presented.
- Weights are dynamically adjusted according to the learning paradigm.
- Delta learning rule is an example of a learning rule.

Artificial Intelligence Systems (9)

– In the delta rule, the change in weight, δ_{ij} , is calculated by:

• $\Delta_{ij} = r * i_i * (t_j - z_j)$, where:

- r is the learning rate
- i_i is the input vector
- T_j is the target output vector
- Z_j is the actual output of node j

For example, if a specific output vector is required for a specific input where the relationship between input and output is non-linear, the neural network is trained by applying a set of input vectors. Using the delta rule, the neural network iteratively adjust the weights until it produces the correct output vector for each given input vector. The neural network is then said to have learned to provide the correct response for each input vector.

Agents (Bots)

- Agents (bots) perform a service on a regular schedule without intervention.
- Push technology gathers information and presents it to you.
- Other agents:
 - Personalize information on a web site depending on who you are and how you use the site
 - Tell you when the site has been updated
 - Gather, organize, and interpret data
- Security considerations to know:
 - Data and programs are pushed.
 - Who provides them?

Bots

Bots are autonomous software that operates as an agent on behalf of a user or program. The bots simulate human activity and pretend that normal access is done by a user. The most common are spiders or crawlers. These critters access a web site, follow each of the hyperlinks on the site, and create an index of the site for use within a search engine.

Other bots can be used for functions, such as searching for the best price on multiple web sites or simulating chat conversations between users or programs.

Some of latest vulnerabilities in browsers should cause concern, especially regarding a tool that can randomly fetch pages. You never know what you will get, and sometimes you might get more than you asked for from a bot.

Applets

Small application program:

- Functions without sending user requests back to the server
- Sent along with a web page to the user
- Written in Java
- Performs interactive animations, calculations, and simple tasks

Applets

Applets are small Java programs downloaded by users who visit web pages. These applets provide more functionality and a richer experience to users. They are common on dynamic web sites or sites that have animated or interactive functions. Applets are restricted from accessing the local file system or the network. Due to some of the difficulties in delivering applets to a variety of different browsers, a lot of developers have switched to server-side Java programs instead. These programs are called Servlets. The next slide includes more detail about the security mechanisms attached to Java.

Java

Object-oriented

Platform-independent

- Generates byte code
- Byte code interpreted into machine code by Java Virtual Machine

Not the same as Javascript

- Sandboxing protects against malicious applets.
 - Applet runs in segregated area.
 - Attempted actions monitored and compared to security policy.
 - Java has not been able to ensure that all code stays in the sandbox.
- Browser settings control actions of applets.
- Applets can be signed.
- JVM runs checks on each object to ensure integrity.

Java

The Java programming language has some interesting features and security mechanisms. One of the concepts is the sandbox, in which an application that runs your browser is executed in what is similar to a virtual machine. It does not have the capability to perform functions outside of this box. Java is also a cross-platform programming language. Applets are created and then compiled into byte code that is not specific to a processor. When the applet is downloaded, the Java Virtual Machine (JVM) converts the byte code into machine language code that is understood by the specific platform on which the code is executed.

Untrusted Java Applets

Applets that are downloaded from the Internet are untrusted and prevented from reading or writing files on the client file system. These applets are also prevented from making network connections except to the originating host on which they were downloaded. In addition, applets loaded over the Internet are prevented from starting other programs on the client. Applets loaded over the Internet are also not allowed to load libraries or to define native method calls. If an applet can define native method calls, the applet gets direct access to the underlying computer.

Trusted Java Applets

Trusted applets occur in two ways. To get a trusted applet, you can install one on the local hard disk in a directory on the CLASSPATH used by the program that runs the applet. This is typically a Java-enabled browser, but it can be the applet viewer or another Java program that loads applets. Another trusted applet is one that is signed by an identity marked as trusted in an identity database.

Protecting Yourself

If there is sensitive data on your computer, consider disabling Java and JavaScript and avoiding plug-ins that are not from well known companies. This is even more important if you visit a web site that is dubious or not well known, which might be the case when doing security research. Use a "sacrificial lamb" machine for this type of surfing.

ActiveX

- Object-oriented programming technologies and tools
- ActiveX control:
 - A self-sufficient program that can be run anywhere in the ActiveX network
 - Equivalent to a Java applet
 - Can be created with several languages
- Security relies on identifying the source of ActiveX controls with certificates.
- ActiveX control downloaded to hard drive, not sandbox.
- Configure browser settings:
 - Users might not understand prompts.
- Configure firewalls to block ActiveX controls.

ActiveX

When a user connects to a web page that has an embedded control, the browser's authenticode technology will verify the signature with the Certificate Authority (CA) that has signed the control to verify it has not been modified. It then downloads the control. Internet Explorer's default is to *not* allow untrusted or unsigned controls to execute. This setting, however, can be easily changed by the end user and it is even possible for an end user to accept all controls without prompt, regardless of the trust level. In other words, the user does not have to deal with annoying security warning messages if he doesn't want to deal with them.

In resume, Java executes code in a sandbox that cannot access the file system or the network. ActiveX relies on the use of a digital signature that can be disabled by the end user. Both technologies contain identified security issues, making it necessary to educate users about the dangers of visiting sites that are unknown or changing their browser setting.

Many companies make use of ActiveX to distribute valid patches or updates to users. These companies educate users or direct them about how to allow an applet to run. Users that are not properly educated will not see the difference between an in-house malign control and a malicious one from the Internet.

Service Level Agreement (SLA)

- The service level agreement guarantees the quality of a service to a subscriber by a network service provider.
- Defined service levels provide a basis for measuring the delivered services and are useful in anticipating, identifying, and correcting problems.
- Service level agreement metrics:
 - Turn-around times
 - Average response times
 - Number of online users
 - System utilization rates
 - System up times
 - Volume of transactions
 - Production problems

SLA

A service level agreement (SLA) is extremely important in that it addresses one of three key areas of security — availability. Even your service providers need a good SLA. How can you guarantee quality of service to your internal users or clients if you do not know what your providers guarantee? If you have an ISP that can give you only 98 percent availability, it is impossible for you to promise a 99 percent availability level to your clients and users. Ensure that the SLA has clear metrics and verify if it is monitored by the provider or if you must monitor it yourself in the event you want claim damage or infringement of the SLA's promise.

Software Prototyping and CASE Tools

- **Software Prototyping:** The development of a working model with test data or real data using an iterative approach supported by user and developer interaction.
- **CASE Tools:** Computer-aided software engineering tools can be used to develop application systems faster and to increase programmers' and analysts' productivity.

Prototyping

Prototyping allows you to prove concepts for the development of software, systems, or applications. Being a prototype, there is no right or wrong answer. If the whole development process becomes too costly or unworkable, the effort can be dropped. A prototype is fluid in terms of scope, and changes can be introduced as required. Such a development effort allows for close interaction between the user community and the developers.

Computer Aided Software Engineering

A CASE tool is a computer-based product aimed at supporting one or more activities within any aspect of the software development process. Case tools might support only one particular part of this process (such as compilers, editors, or UI generators).

The latest generation of CASE tools introduces new capabilities based on I-CASE tools. Such tools use rapid application development (RAD) techniques to rapidly develop applications, which is done at lower cost and with better quality. This RAD approach allows the application to be tested in between development phases because the CASE tools can create a prototype at little cost. The sooner mistakes are discovered in the development cycle, the less costly they are.

Software Capability Maturity Model (CMM)

- A software process maturity framework is used to help organizations improve their software development process. The software process maturity serves as an indicator of the likely range of cost, schedule, and quality results achieved by development projects in a software organization.
- The CMM has five levels, and each level has key process area descriptions.

CMM

The Software Engineering Institute of Carnegie Mellon University at <http://www.sei.cmu.edu/cmm/cmm.sum.html> states: "The Capability Maturity Model for Software describes the principles and practices that underlay software process maturity, and it is intended to help software organizations improve the maturity of their software processes in terms of an evolutionary path from ad hoc, chaotic processes to mature, and disciplined software processes."

Software Capability Maturity Model (CMM) (2)

- The CMM is organized into five maturity levels:
 - **Level 1 Initial:** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
 - **Level 2 Repeatable:** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
 - **Level 3 Defined:** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
 - **Level 4 Managed:** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
 - **Level 5 Optimizing:** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

The CMM is organized into five maturity levels:

- **Initial:** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
- **Repeatable:** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- **Defined:** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
- **Managed:** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
- **Optimizing:** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Databases: CIA

- Concurrency
 - Updates by more than one person at the same time
 - Solved by locking
- Semantic integrity
 - Ensures that data types, logical values, uniqueness constraints, and operations are enforced
 - Enforcer: DBMS
- Referential integrity
 - Prevents users from entering inconsistent data
 - Enforcer: DBMS
- Commit
 - Executes changes that were just made
 - 2-phase commit: Vote first before committing (distributed databases).
- Rollback if commit is unsuccessful
 - Database returns to its previous state.
 - Checkpoints: If a system fails, there is a return to the point before failure.

Database systems also have serious security issues. Large database systems have hundreds if not thousands of users who access information that is centrally stored in a Database Management System (DBMS). Such a system must have mechanisms in place to ensure that two users do not attempt to access the same data at the same time. This is usually controlled through the use of locks that are imposed on rows or fields in the database. As soon as a user accesses a specific record, another person cannot access the record. In some cases, a deadlock is put in place. A *deadlock* is a lock that was not properly removed and although no one will make use of the specific record or entry, the system will not allow use of it.

DBMSes are also adept at enforcing semantic integrity. They keep track of what type of data is entered and only valid types are accepted. A field is defined as a date field, a numeric field of x character, or a constraint (such as a unique customer ID). If a field data type is not proper, an error message is generated and resubmission has to take place with the corrected data.

Referential integrity in a database has a few rules. A database table usually has a unique primary key for each record. This key can be referenced by other tables in the database; this is called a *foreign key*. All foreign keys must point to an existing primary key or there can be a serious integrity problem.

A commit is executed when the changes you make to a record are submitted to the database. As long as the commit is not completed, the information is temporarily stored and not saved.

On rare occasions, problems arise when doing entries in a database. If for some reason the system no longer responds or exhibits strange behavior, it is possible to complete a rollback. A *rollback* is when you return to a previous known good state. Some of the leading DBMS systems allow you to take regular snapshots or checkpoints and revert back to a specific checkpoint if a problem arises.

Database Systems

- Database
 - Collection of related data about an organization intended for sharing by multiple users
- Database management system (DBMS)
 - Stores data and provides operations on the database, such as create, delete, update, and search
 - Provides security and integrity controls
- Types of data models
 - Hierarchical
 - Mesh
 - Object-oriented
 - Relational

Database

- Collection of related data about an organization intended for sharing by multiple users

Database management system (DBMS)

- Stores data and provides operations on the database such as create, delete, update and search
- Provides security and integrity controls

Database Systems (2)

- Security issues
 - Aggregation
 - User has a right to only certain data items in a larger collection of data items.
 - Obtains knowledge that he/she does not have a right to about the larger collection
 - Inference
 - User deduces information of higher sensitivity from lower sensitivity information.
 - Inference controls
 - Enforced during query processing
 - Content-dependent access rules

Security issues:

Aggregation

- User has a right to only certain data items in a larger collection of data items
- Obtains knowledge that he/she does not have a right to about the larger collection

Inference

- User deduces information of higher sensitivity from lower sensitivity information

Inference controls

- Enforced during query processing
- Content-dependent access rules

Database Systems (3)

- Data warehouse
 - Storage facility where data from heterogeneous databases are brought together for users to make queries against.
 - Purpose is information retrieval and data analysis.
 - Redundant and inconsistent data are removed from databases (normalizing).
 - Metadata: Data about data in the data warehouse.
- Data mining
 - By detecting abnormal patterns, can be used for:
 - Intrusion detection
 - Fraud detection
 - Auditing the database

Contains "massaged" or correlated data.

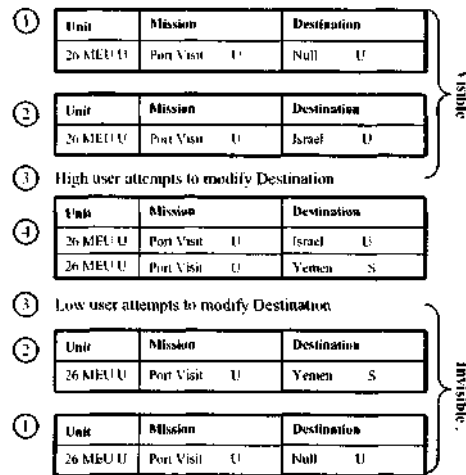
The idea is to conduct data mining and attempt to discern relationships among variables in the database that are not obvious.

Extracted metadata is not usually stored in the warehouse, but it is stored in another system with high levels of protection called a data mart.

Information obtained from metadata should be sent back for incorporation into the data warehouse to be available for future queries and metadata analyses.

Polyinstantiation

- Solution for multi-level databases
- Cover story:
 - Two versions of the same object, so lower-level subject will not know true information.
- Prevents "downward signaling channel" when:
 - High user attempts to modify lower level data (visible).
 - Low user attempts to modify higher level data (invisible).
- The challenge of polyinstantiation is maintaining integrity.



Polyinstantiation

Polyinstantiation is used to generate two versions of the same object to be presented to subjects according to their security level. This prevents low-level users from getting access to information or inferring information from what they know and what they are capable of getting from the systems.

A simple example of this is a company's reservation system for conference rooms. If human resources books all conference rooms on a specific date, and there is a pending rumor of a large layoff due to acquisition by a larger firm, someone can easily infer that the big layoffs will occur on this date. Some people might take advantage of the situation to sell stock, buy stock, or leak the information to the press. A polyinstantiation system prevents an inference attack by presenting different names for each of the conference rooms booked. As a result, one cannot tell that human resources booked all the rooms.

Such a system must have a strong mechanism in place to ensure proper updates to all the versions of the data to maintain data integrity.

Summary

- Software life cycle development model
- Importance of development methodologies
- Object-oriented systems
- Artificial intelligence systems
- Database systems

This domain covered a lot of different critical topics related to designing, building, deploying, and maintaining software. Although several sections focused on the material from a programmatic or development standpoint, there is an implicit focus on security when project development is discussed.

5. Cryptography

10 Domains of Knowledge

This section covers Domain 5, the Cryptography domain.

Overview

- Basic cryptographic concepts
- Application of public and private key algorithms
- Key distribution and management
- Methods of attack
- Digital signatures

Why Use Cryptography?

Cryptography is vitally important to information security. One of the main goals of cryptography is to help fend off eavesdroppers. Communicating over any kind of medium has the inherent risk of an unauthorized third party listening in, and we want to minimize that risk. In its most basic form, cryptography garbles text in such a way that anyone who intercepts the message cannot understand it. Although there are many ways to perform cryptography, most follow similar methodologies.

Note

David Kahn's *The Code Breakers* is an excellent source that can provide a better appreciation for the field of cryptography. Starting with the Pharaohs, this book chronicles hidden writing throughout history. Cryptography has always existed; perhaps one of history's most famous cryptographers was Julius Caesar, who invented the Caesar cipher. He used a basic substitution similar to the encryption schemes that are used on the back of some children's cereal boxes. But cryptography has come a long way since then. An indispensable reference for modern ciphers and techniques is *Applied Cryptography: Protocols, Algorithms and Source Code in C* by Bruce Schneier. At over 750 pages, it is a comprehensive guide to modern algorithms that is written by someone who understands that the quality of a cryptosystem requires more than just a good algorithm.

Cryptography

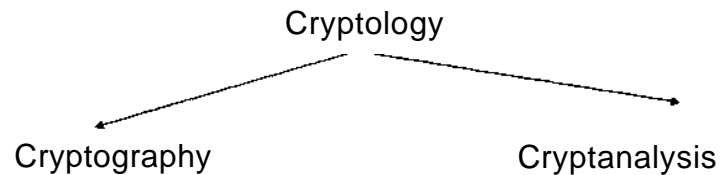
- Cryptography means "hidden writing/"
- Encryption is coding a message in such a way that its meaning is concealed.
- Decryption is the process of transforming an encrypted message into its original form.
- Plaintext is a message in its original form.
- Ciphertext is a message in its encrypted form.

Nearly every cryptographic algorithm performs two distinct operations: encryption and decryption. *Encryption* is the practice of coding a message in such a way that its meaning is concealed. How the message is transformed depends on a mathematical formula called an *encryption algorithm* or a *cipher*. After a message has been transformed with a cipher, the resulting message is called *ciphertext*. Because ciphertext contains the message in its encrypted form and not its native form, it is unintelligible. For the ciphertext recipient to read the message, he must *decrypt* it. *Decryption* is the process of transforming an encrypted message back to its original plaintext or cleartext form.

Who creates these encryption algorithms? Computer scientists called *cryptographers*, who are well trained in several different fields of mathematics and who usually work in groups, take many years to invent and refine ciphers. But with so much depending on *cryptography*, individuals called *cryptanalysts* dedicate their lives to breaking ciphers. Some cryptanalysts work for the military and for governments; others are simply interested in the study of ciphers and want to find weaknesses in ciphers to ensure that they cannot be broken by others. The generic term for the study of both cryptography and cryptanalysis is called *cryptology*.

Cryptography and Cryptology

- **Cryptography:** Art and science of hiding the meaning of a communication from unintended recipients. The word cryptography comes from the Greek words, kryptos (hidden) and graphein (to write).
- **Cryptology:** Encompasses cryptography and cryptanalysis



Codes: A cryptographic transformation that operates at the level of words or phrases

Cryptanalysis: Act of obtaining the plaintext or key from ciphertext that is used to obtain valuable information and to pass on altered or fake messages to deceive the original intended recipient

Cryptographic algorithm: A a step-by-step procedure used to encipher plaintext and decipher ciphertext

Block cipher: Obtained by segregating plaintext into blocks of n characters or bits and applying the identical encryption algorithm and key to each block

Cipher: A cryptographic transformation that operates on characters or bits

Ciphertext or cryptogram: An unintelligible message

Clustering: Situation in which a plaintext message generates identical ciphertext messages using the same transformation algorithm, but with different cryptovars or keys

Plaintext: A message in cleartext readable form

Cryptosystem

- **Cryptosystem:** A set of transformations from a message space to a ciphertext space
 - $E(m, k) = c$
 - $D(c, k) = d[e(m, k), k] = m$
 - Where m =plaintext, c =ciphertext, e =the encryption transformation, d =the decryption transformation, and k is the key (cryptovariable)
- **End-to-end encryption**
 - Encrypted information that is sent from the point of origin to the final destination

Link encryption

Each entity has keys in common with its two neighboring nodes in the transmission chain.

A node receives the encrypted message from its predecessor (the neighboring node), decrypts it, and then re-encrypts it with another key that is common to the successor.

Encrypted message is sent on to the successor node where the process is repeated until the final destination is reached.

This mode does not provides protection if the nodes along the transmission path can be compromised.

Exclusive OR

Exclusive OR: Boolean operation that essentially performs binary addition without carry

INPUTS		OUTPUT
A	B	T
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive or is easily implemented in hardware and can be executed at hardware speeds.

The inverse of the function can be obtained by performing another exclusive or on the output. For example, if input a is the digital data stream to be enciphered and input b is the key stream, the output t is the enciphered data. To retrieve the original digital data stream, the output t can be exclusive or'ed with the keystream at b again.

The exclusive or function is a basic function that is used in many cryptographic algorithms. When the two inputs, a & b, are the same, the output of the exclusive or function is a logic 0. When the inputs are different, the output is a logic 1.

One-Time Pad

One-time pad:

- The key has the same length as the message.
- The key is used only once and never used again.
- Ideally, the key's components are truly random and have no periodicity or predictability, thus making the ciphertext unbreakable.

The one-time pad was invented in 1917 by Major Joseph Mauborgne of the United States army signal corps and Gilbert Vernam of AT&T.

If used properly, that is, the key is truly random, it must satisfy the following: The key is the same length as the data to be enciphered, the key is used only once, and the one-time pad is unbreakable. For large amounts of data, it is sometimes difficult or impossible to satisfy the first two constraints.

Work Function

Work function (factor):

- Difficulty in recovering the plaintext from the ciphertext as measured by cost and/or time
- A system's security is directly proportional to the value of the work function.

The work function only needs to be large enough to suffice for the intended application.

If the message to be protected loses its value after a short time period, the work function only needs to be large enough to ensure that the decryption would be highly infeasible in that period of time.

Crypto History

- The history of Cryptography is long and interesting.
- The next few slides discuss the highlights.

Cryptography dates back to the ancient Egyptians, who began using secret writing called *hieroglyphics* as early as 3000 B.C. The term comes from the ancient Greek word *hieroglyphica*, meaning *sacred carvings*. The Egyptians used this writing to hide messages from unintended recipients.

By the fifth century B.C., the Spartans used a *scytale*, or a wooden staff of prescribed thickness with a strip of cloth or parchment wrapped around it. The cleartext message was written along the length of the rod so that when the cloth was unwrapped, it appeared to contain a stream of random letters. The recipient would then wrap the cloth on an identical scytale, thereby decrypting the message.

Crypto History: Secret Writing

Egyptian hieroglyphics: 3000 B.C.

- Hieroglyphics is derived from the Greek word *hieroglyphica*, which means sacred carvings.
- Hieroglyphics evolved into hieratic, which was a stylized script that was easier to use.

Even though the popularity of modern computers has brought crypto to a new level of automation, the concepts behind cryptography are not new. In 3000 B.C., the Egyptians referred to crypto as secret writing because they were able to use various techniques to keep their communication secret among a small group of people. This was the beginning of secret communications. The technique of using special symbols seems trivial today, but it was difficult to crack at the time.

Crypto History: Spartan Scytale

- 400 B.C.: Military cryptography
- A strip of papyrus or parchment was wrapped around a wooden rod.
- Message to be encoded was written lengthwise down (or up) on the rod on the wrapped material.
- Material was unwrapped and carried to the recipient.
- Material was rewound on a rod of the same diameter, to read.

In 400 B.C., the Spartans developed a fairly sophisticated method of encryption based on the current form of communication. With this technique, the two parties that wanted to communicate were required to use wooden rods of the same diameter. The sender would take wrap a long piece of cloth around the wooden post and write his message vertically across the pole. He would then remove the cloth from the rod and carry it with him. If anyone looked at the cloth, it contained random type characters. It was only when the cloth was wrapped around a wooden rod of the same diameter that the message was revealed.

Crypto History: Caesar Cipher

- 50 B.C.: Julius Caesar
- Substitution cipher
- Letters of the alphabet are substituted for other letters of the same alphabet.
- Mono-alphabetic substitution
- Involved shifting the alphabet three letters and substituting those letters
- Sometimes known as c3 substitution cipher

ATTACK AT DAWN



DW WDFN DW GCZQ

Based on what we have learned already, it should be of no surprise that Julius Caesar used encryption. For his time, his technique was extremely advanced. The Caesar cipher used a simple rotation to encrypt a message. The Caesar cipher used a rot-3, or rotation 3, in which each letter of the alphabet was rotated three letters forward. In our language, the letter A became D. To decrypt the message, you would rotate the letter back three places in the alphabet.

UNIX ROT 13

Unix systems use a substitution cipher called ROT 13:

- It shifts the alphabet by 13 places.
- Another shift of 13 places brings the alphabet back to its original position, thus decoding the message.

Unix systems use a substitution cipher called ROT 13.

- It shifts the alphabet by 13 places.
- Another shift of 13 places brings the alphabet back to its original position, thus decoding the message.

Polyalphabetic Cipher

Polyalphabetic cipher:

- It is accomplished through the use of multiple substitution ciphers.
- Blaise De Vigenere, a French diplomat born in 1523, consolidated the cryptographic works of Alberti, Trithemius, and Porta to develop the polyalphabetic cipher.
- Because multiple alphabets are used, this approach counters frequency analysis.

This cipher can be attacked by discovery of the periods when the substitution repeats.

The difficulty posed by this cipher is that the same letter in the plaintext does not transform to the same ciphertext letter (different alphabets).

For example, if the plaintext was "hello," the first letter l might transform into the letter g and the second letter l might transform into the letter r.

This polyalphabetic substitution was performed in the German Enigma machine.

Crypto History: Battista Cipher Disk

- In Italy, around the year 1460, Leon Battista Alberti developed cipher disks for encryption.
- Two concentric disks
- Each disk had an alphabet around its periphery.
- By rotating one disk with respect to the other, a letter in one alphabet could be transformed to a letter in another alphabet.

As we go through history, the closer we come to modern times, the more advanced the techniques become. You have to remember that although these techniques seem simple with the use of computers, they would be considered fairly difficult to develop and crack. Leon Battista took the concept of the Caesar cipher to the next level. Rather than developing a scheme in which each letter was rotated three spaces in the alphabet, why not develop a scheme in which you change the key and rotate a letter any number of spaces? Battista did this by creating two disks, one slightly smaller than the other. He then attached these in the center. Each disk included the letters of the alphabet; by rotating the disks, a certain number of places would give a different value for x in the rotation cipher.

Crypto History: Cryptanalysis

Because of their expertise in mathematics, statistics, and linguistics (around the seventh century), the Arabs invented cryptanalysis.

A cryptologist is someone who works in the area of encryption. These people specialize in encryption and the mathematics surrounding it. As with any technique, the only way to determine the robustness is to crack it. Figuring out weaknesses in an algorithm would allow you to build more robust techniques. As obvious as this methodology is, the Arabs — with their extensive knowledge in mathematics — originally came up with it. Although this concept was developed a long time ago, it still serves as a basis for strong crypto today.

Crypto History: Jefferson Disks

- In 1790, Thomas Jefferson developed an encryption device.
- He created a stack of 26 disks that could be rotated individually.
- A message was assembled by rotating each disk to the proper letter under an alignment bar that ran the length of the disk stack.
- The alignment bar was rotated through a specific angle, and the letters under the bar were the encrypted message.

Thomas Jefferson also did extensive work in the area of encryption. He took the ideas Battista and others developed and expanded them into more complex structures. Battista used a two-disk mechanism to encrypt and decrypt information. Thomas Jefferson developed a 26-disk mechanism that was based on the unique letters in the English alphabet.

Crypto History: Jefferson Disks (2)

This diagram shows a picture of the machine that Thomas Jefferson created. It essentially contains 26 disks, each with a unique marking. Aligning the disks in a certain way would allow someone to encrypt a message and decrypt the message. The alignment of the 26 disks served as the key. Even if two people had the same device but not the same key, they would not be able to read an encrypted message.

Crypto History: Stafford

- U.S. Navy cryptologist Lawrence Stafford
- Stafford headed the team that broke the Japanese purple machine naval codes during World War II.
- In the spring of 1942, Commander Joseph J. Rochefort led the group that intercepted and deciphered Japanese codes telling of Japan's plans to invade Midway Island.
- This intelligence led to a convincing defeat of the attacking Japanese navy task force at Midway and turned the tide of the war in the Pacific.

In addition to the fighting, another war takes place during most World Wars. This is the war where one side tries to communicate and the other side tries to break the communication. If one side can figure out what the other side is planning, it has the best advantage of winning the war.

During World War II, the United States was able to break the Japanese encryption; however, even though the U.S. could read messages, the Japanese still used code words within their messages. One set of messages that the United States was able to read talked about a Japanese attack on Island X. Even though the U.S. could read the message, they did not know what Island X stood for. Lawrence Stafford thought that they were referring to the Island of Midway. He then constructed a message with weak encryption that stated that Midway was having water problems. Later that day, the United States intercepted and read a message from the Japanese that Island X was having water problems. This confirmed that the Japanese were planning to attack Midway.

Crypto History: Enigma

The German enigma rotor machine has the following properties:

- Poly-alphabetic substitution cipher machine
- Used by the Germans in World War II
- Developed by Dutchman Hugo Koch in 1919
- Produced for the commercial market in 1923 by Arthur Scherbius
- Working with the French from 1928 to 1938, Pole Marian Rejewski solved the wiring of the German 3 rotor enigma.
- In 1938, the Germans changed the number of rotors to six.
- The Poles and the French constructed a prototype machine called "the bombe" for use in breaking the enigma cipher. Bletchley park in England took over the work of breaking the enigma cipher.

Crypto History: Hebern Machines

- Rotor systems are also referred to as Hebern machines.
- Other rotor machines include:
 - The Japanese red and purple machines
 - American sigaba (big machine)
 - Sigaba ciphers were never broken.

Any system based on a rotor mechanism, such as Battista and Jefferson, are often referred to as Hebern machines.

Other rotor machines include the following:

- The Japanese red and purple machines
- American sigaba (big machine)
- Sigaba ciphers were never broken.

As with any encryption scheme, the method's robustness is determined by how well it is implemented. Some rotor systems are easily breakable, and others claim that they have never been broken.

Crypto History: Vernam Cipher

- One-time pad
- It is implemented through a key that consists of a random set of non-repeating characters.
- Each key letter is added as modulo 26 to a letter of the plaintext.
- The key is used only once, and then it is never used again.
- The length of the key character stream is equal to the length of the message.

Some consider the Vernam cipher to be unbreakable. This is not entirely true because all encryption is breakable from a brute-force perspective. It might take 800 years to crack, but it is still breakable. With the Vernam cipher, each message is encrypted with a different key, which is referred to as a one-time pad. When people say that the Vernam cipher is unbreakable, they are actually saying that the usefulness of the message has expired by the time you crack the key.

Book or Running Key Cipher

Book or running key cipher:

- Uses text from a source, such as a book, to encrypt the plaintext
- Key, known to the sender and the intended receiver, may be the page and line number of text in the book.
- Text is matched character for character with the plaintext, and modulo 26 addition is performed to effect the encryption.

This type of cipher eliminates periodicity, but it is attacked by exploiting the redundancy in the key.

For example, the sender might tell the receiver to go to the book *All Quiet on the Western Front*, and go to page 101. Then, the receiver would be instructed to go to the second paragraph and, starting with the first word in that paragraph, use the letters as the key.

Import and Export Issues

- COCOM (Coordinating Committee for Multilateral Export Controls)
 - 17 members
 - 1991 allowed export of encryption
 - Prevent crypto from being exported to dangerous countries
- Wassenaar Arrangement
 - 1995, 28 countries followed up to COCOM
 - Symmetric crypto free for export
 - Export of other crypto still requires a license
- European Union Controls
 - Regulated by the Council Regulation (EC) No. 1334/2000
 - Focused on export of encryption
- United States Controls
 - No import restrictions
 - Signed the Wassenaar Arrangement but has stricter export controls
 - Looser export controls occurred on July 2000
 - Retail crypto
 - Crypto source code

COCOM (Coordinating Committee for Multilateral Export Controls)

- 17 members
- 1991 allowed export of encryption
- Prevent crypto from being exported to dangerous countries

Wassenaar Arrangement

- 1995, 28 countries followed up to COCOM
- Symmetric crypto free for export
- Export of other crypto still requires a license

European Union Controls

- Regulated by the Council Regulation (EC) No. 1334/2000
- Focused on export of encryption

United States Controls

- No import restrictions
- Signed the Wassenaar Arrangement but has stricter export controls
- Looser export controls occurred on July 2000
 - Retail crypto
 - Crypto source code

Goals of Cryptography

- A cryptosystem achieves the following goals:

Confidentiality

Data Integrity

Authentication

Non-repudiation

- *"Cryptography is about communications in the presence of adversaries."* (Rivest, 1990)

Like many of us, Alice does not care *how* the cryptography works, as long as it works. She must send a message to Bob with the same level of integrity that it would have if she walked up and handed it to him. In addition to being unreadable by adversaries (*confidentiality*), we might have the following requirements:

- **Authentication:** If Alice walks up to Bob and hands him a message, he positively knows that the message is from Alice. Alice might require the cryptosystem to provide an equivalent service for her; Bob must hand deliver messages to her.
- **Integrity:** It should be possible to prove that the message has not been tampered with—that this message is exactly the same as the one that Alice sent to Bob.
- **Non-repudiation:** The system should be able to prove that Alice, and only Alice, sent the message, and that it has not been falsified or subsequently altered. In essence, this is a requirement that both authentication and integrity are provable.

The technology to do this is available, but for this system to work in practice, the non-technical issues are also important. Alice and every system user must be trained in its use and limitations and have access to the keys, and also keep them protected and current. Processes must be as foolproof as they are practical. Think about *social engineering*, human error, and operator efficiency, accuracy, and understanding.

Basic Terms

We already defined the four most common cryptographic terms: *encryption*, *decryption*, *plaintext*, and *ciphertext*. Encryption and decryption must use the same algorithm. Those operations are also managed by the choice of the *key*. The key changes periodically (for example, daily, hourly, or per-message), so that the same algorithm can be used for years on end while communications remain secure.

The following discussion mentions weaknesses with crypto schemes. Although this is not a focus of the section, there are several ways to defeat even a good crypto algorithm, including exhaustively searching every possible key or exploiting a mathematical weakness in the algorithm.

General Encryption Techniques

- Goal: Garble the original message, so that its meaning is concealed.
- Basic techniques:
 - Substitution
 - Permutation
 - Hybrid
- Single-key systems use these techniques.

Essential Operations

The main goal of encryption is to garble text, so that someone cannot understand it. Two basic methods of encrypting or garbling text are *substitution* and *permutation*. A third approach is actually a hybrid, or a mixture of both. There are also two basic types of key encryption systems: one-key and two-key systems. The first methods we discuss are for one-key systems; later you see that two-key systems are much more complex. In this section, you learn that, despite being based on high school mathematics, one-key systems are effective.

Arbitrary Substitution

- Uses a one-to-one substitution of characters.
- Replace x with y.
- For example:
 - A B C D E
 - W K M P D
 - So CAB becomes MWK
- Easy to break using character frequency analysis

Substitution

Substitution involves exchanging one character (or byte) for another. Simple substitution schemes use mapping; therefore, one character would be substituted with another character to encrypt a message, with decryption being the inverse action." The mapping function is the key; that is, anyone who knows how the characters were mapped to encrypt the message can decrypt the message.

Consider a simple example. Suppose we define the following map (only a portion of the alphabet is shown):

Plaintext: A B C D E . . .

Ciphertext: W K M P D . . .

To encrypt the word "CAB," Alice would substitute characters and send the string "MWK." In turn, Bob, would reverse the substitution to recover the plaintext.

For substitution to work, there must be a unique one-to-one mapping from plaintext character to ciphertext character. A many-to-one or one-to-many mapping would make decryption difficult or impossible. For example, if W replaced both A and C, you would still be able to encrypt the message; therefore, CAB would become WWK. But if we tried to decrypt it now, we would not know whether the W should be an A or a C because they are both mapped to the same letter.

Rotation Substitution

- Rotation substitution uses a one-to-one substitution of characters, so it's also easy to break.
- "Rotate" the alphabet by N characters.
- Easy to remember. For example:
 - A B C D E
 - D E F G H
 - So CAB becomes FDE
- Caesar Cipher was "ROT-3."
- Usenet uses "ROT-13" (symmetric!).

An alternate substitution method that does not require mapping is *rotation*. In this type of substitution, you shift every character a set number of spaces. For example, if you shift A three spaces, it becomes D, B becomes E, and so on. The Caesar Cipher, invented by Julius Caesar to encode messages to his generals, is a famous rotation cipher. If Alice were using this "ROT-3" scheme, she would encrypt her message as "FDE." In its day (roughly 50-60 B.C.), the Caesar Cipher was considered good enough to fool almost anyone because few people could read, even fewer could write, and couriers would rather kill a snooper than let him capture a message. Caesar was no fool; however, he did not use just one encryption tool. He also transliterated Latin into Greek and used other forms of subterfuge.

Though many people believe the Caesar cipher is the earliest cipher, cryptography actually goes back nearly 2000 years earlier to ancient Egypt and China. For more information, see the Crypto Timeline at http://emmy.nmsu.edu/crypto/public_HTML/Timeline.HTML.

Although character rotation is a trivial scheme, rotation ciphers came back into vogue in the early 1980s, primarily in the form of ROT-13. Shortly after USENET newsgroups and electronic mailing lists became popular, subscribers realized that they did not always want to see a message's contents. Some messages contained jokes that might offend some subscribers, while other messages might contain riddles or puzzles complete with answers that the recipients might not have wanted to see before reading the riddle or puzzle.

The answer was to encrypt (or obscure) jokes and answers using ROT-13. ROT-13 was never meant to be a strong cipher; it is trivially breakable. The point was for the reader to make a deliberate effort to decipher the message. No one could later claim accidental discovery, and no one could ruin a puzzle by accidentally glimpsing at the solution. ROT-13 eventually became part of newsreader software and a common function of the Unix operating system. ROT-13 had another nice feature; because there are 26 letters in the English alphabet, ROT-13 is a symmetric operation. The same implementation both encodes plaintext and decodes ciphertext.

Permutation

- It keeps the same letters, but it changes the position within the text.
- Changes the order from xyz to zxy.
- For example:
 - Change 1 2 3 4 5 to 3 5 2 1 4
 - So order becomes drroe
- Easy to break.
- Substitution and permutation can be combined.

Permutation

Permutation, also called *transposition*, shuffles the order in which characters (or bytes) appear rather than substituting one for another. Suppose that Alice and Bob chose the key word "SCUBA" to determine the character permutation order. Alphabetizing the letters in the key word, you obtain the string "ABCSU." Because "A " is the first letter, it is assigned the number 1, and "U" is assigned the number 5; the string "43521" then determines the way in which we move around letters. Alice takes her message, breaks it into blocks of five characters (because that is the length of the key word), and then moves the characters within each block accordingly. Assume Alice wanted to send the following message:

DINNER TONIGHT OK

Alice would encrypt it into the following ciphertext:

NNEID NOITR OTKHG

Unfortunately, permutation is also relatively easy to break. Remember, however, that while a few thousand or million combinations is nothing for a computer, it can defeat an adversary using pencil and paper. Today's computer-based methods still use substitution and permutation, but in a combination that's applied many times. Let's take a look at the mechanics of current encryption methods.

Ways to Encrypt Data

Two general ways to encrypt information:

- Break the data into **blocks** and encrypt each block.
- Encrypt the entire **stream** on a bit-by-bit basis.

Ways to Encrypt Data

There are two ways to manipulate the data while encrypting and decrypting: breaking up the data into blocks and encrypting each block or encrypting a stream bit-by-bit (or byte-by-byte). Hence, crypto schemes are generally classified as either stream ciphers or block ciphers, depending on how much information they manage at once and how the key is generated.

Types of Cryptosystems

Three general types:

-Secret key

- Symmetric
- Single or one-key encryption

-Public key

- Asymmetric
- Dual or two-key encryption

-Hash

- One-way transformation
- No key encryption

Types of Cryptosystems

Today's cryptosystems contain three general types of crypto algorithms: *secret key* or *symmetric*, *public key* or *asymmetric*, and the *hash*.

Each algorithm is used because it provides a different function from other algorithms. These schemes are usually distinguished by the number of keys employed. The remainder of this section discusses these different types of algorithms.

Symmetric-Key Cryptosystems

"Secret-Key" or "Private-Key" Encryption:

- Fast! Single key for encryption and decryption
- Requires secure key distribution channel (scalability)
 - Pre-shared secret
 - Asymmetric encryption
 - Diffie-Hellman key exchange
- Not technical, non-repudiation

Examples:

- DES
- Triple-DES
- RC4
- RC6
- IDEA
- AES

Secret Key Cryptography

Secret key cryptography (SKC) uses a single key for both encryption and decryption; this key is the shared secret between sender and receiver. Because SKC uses only one key for both encryption and decryption, it is also called *symmetric cryptography*. The primary application of SKC is privacy, where only the parties with the key can encrypt and decrypt messages for each other.

Given an adequate SKC algorithm, the basic attack is brute force. Until 1998, this has mostly been a joke and the product of a few Internet research efforts to harness loosely coupled parallel attacks. Now anyone with a six-figure budget can build a specialized DES cracker. Those who are willing to attack systems and steal their computing power might not even need money! The RingZero and DDoS attacks of February 2000 beg the question, "If an encrypted message were worth \$20 million, and you could assign a thousand Trojanized zombie systems to work on the problem, how long would the symmetric key length need to be?" In 1997, a 40-bit RSA challenge key fell in 3.5 hours using 250 computers. Keep Moore's law in mind: Computing power doubles every 18 months. Therefore, 40 bits is inadequate for today's threat model.

That being said, the bigger issue with secret keys is managing the key creation and exchange to avoid key compromise. Also, the greater the number of parties that share the secret key, the greater the key's exposure. The bottom line is that symmetric-key cryptosystems and asymmetric-key systems are often combined because the former are so much faster than the latter, but they lack the latter's key management and digital signatures.

Many SKC schemes are in common use today and all are believed to be mathematically strong. If a cryptanalyst cannot defeat the ciphers by finding a weakness in the mathematical algorithms, the remaining approach is a brute-force attack to guess all possible keys. As explained in a paper by Matt Blaze, Whitfield Diffie, Ron Rivest, Bruce Schneier, and others in the cryptographic community, key size does matter in SKC. The paper, "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security" (<http://www.counterpane.com/keylength.HTML>), describes brute-force attacks that are within the cost and computing means of many attackers and the key lengths necessary to keep such attackers at bay.

Examples of SKC schemes in common use today are the Advanced Encryption Standard (AES), Blowfish, the Data Encryption Standard (DES), Triple DES, and the International Data Encryption Algorithm (IDEA).

Asymmetric-Key Cryptosystems

"Public-key" Encryption

- Slow! Public/private key pair
- Trusted channel
 - Public keys widely distributed within digital certificates.
- Technical non-repudiation via digital signatures
- Private key cannot be derived from the public key.
- Thus, the need to exchange secret keys is eliminated.
- Message that is encrypted by one of the keys can be decrypted with the other key.
- The private key is kept private.

Examples:

- RSA
- El Gamal
- ECC
- Merkle-Hellman
- Knapsack
- Chor Rivest
- Knapsack
- LUC

Public Key Cryptography

The management problems associated with symmetric keys are so overwhelming that they virtually preclude their use by themselves in commerce. However, we can use public key computation to develop a shared message key. In addition, algorithms such as Diffie-Hellman can be used to exchange a secret key. Again, the general idea is to exchange keys securely — perhaps only once — to secure a given session, such as a visit to a Web page to execute a credit card transaction.

Public key cryptography (PKC) methods have two keys: one that is used for encryption and the other for decryption. Because two keys are used, thereby making the encryption and decryption process different, PKC is also called *asymmetric cryptography*. PKC has many applications, but the primary ones today are key exchange (for SKC), authentication, and non-repudiation.

Stanford University professor Martin Hellman and graduate student Whitfield Diffie first described modern PKC publicly in 1976. Their paper described a two-key cryptosystem through which two parties could engage in a secure communication over a non-secure communications channel without sharing a secret key. PKC's mathematical trick depends on the existence of so-called *trapdoor functions*, or mathematical functions that are easy to calculate, whereas their inverse is difficult to calculate. Following are two simple examples:

- *Multiplication vs. factorization:* Multiplication is easy; given the two numbers 9 and 16, it takes almost no time to calculate the product of 144. Factoring is more difficult; it takes longer to find all of the pairs of integer factors of 144 and then determine the *correct* pair that was actually used.
- *Exponentiation vs. logarithms:* It is easy to calculate, for example, the number 3 to the 6th power to find the value 729. However, given the number 729, it is much more difficult to find the set of integer pairs, x and y , so that $\log_x y = 729$ and then, again, determine which pair was actually used.

Hash Functions

hash function
plaintext ————— • ciphertext

- No key
 - Plaintext (and length of plaintext) not recoverable from the ciphertext
 - Examples: HMAC, MD2, MD4, MD5, RIPEMD-160, and SHA
 - Also called *message digests* or *one-way encryption*
- Primary use: message integrity

Hash Functions

Remember that there are three types of cryptography algorithms: secret key, public key, and hash functions. Unlike secret key and public key algorithms, *hash functions*, also called *message digests* or *one-way encryption*, have no key. Instead, a fixed-length hash value is computed based on the plaintext that makes it impossible for either the contents or *length* of the plaintext to be recovered.

The primary application of hash functions in cryptography is message integrity. The hash value provides a digital fingerprint of a message's contents, which ensures that the message has not been altered by an intruder, virus, or other means. Hash algorithms are effective because of the extremely low probability that two different plaintext messages will yield the same hash value.

Several well-known hash functions are in use today:

Hashed Message Authentication Code (HMAC): Combines authentication via a shared secret with hashing

- Message Digest 2 (MD2): Byte-oriented; produces a 128-bit hash value from an arbitrary-length message; designed for smart cards

MD4: Similar to MD2; designed specifically for fast processing in software

- MD5: Similar to MD4 but slower because the data is manipulated more; developed after potential weaknesses were reported in MD4

Secure Hash Algorithm (SHA): Modeled after MD4 and proposed by NIST for the Secure Hash Standard (SHS); produces a 160-bit hash value

Kerberos

- Secret-key protocol and distributed service for third-party authentication
- Kerberos KDC is the trusted intermediary.
- Confidentiality: DES (CBC mode), ...
- Integrity: Cryptographic hash algorithms
- Authentication: Login password (local)
- Non-repudiation: Knowledge of a password

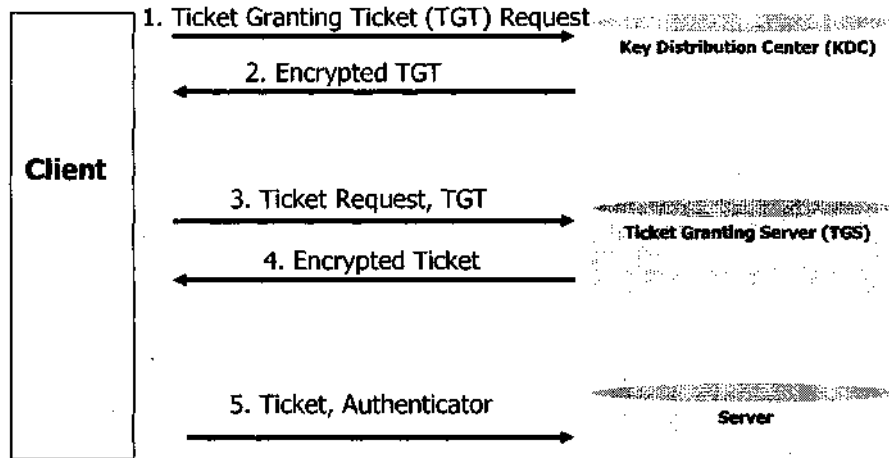
Case Study: Kerberos

Kerberos is a trusted third-party scheme that allows a user to log on to a system once and use any available services without re-authenticating to many different servers. Developed by Massachusetts Institute of Technology's (MIT) Project Athena, Kerberos is named for the three-headed dog from Greek mythology that guards the entrance of Hades. Kerberos version 4 was released in 1987 and is still used in select locations; however, version 5, described in IETF's RFC 1510 and released in 1990, is far more common. Meanwhile, Microsoft incorporated Kerberos in Windows 2000, although that version contains some proprietary extensions to version 5. Although there are significant differences between the three versions, they have many conceptual similarities. Despite its complete approach to providing a cryptographic infrastructure for communications, Kerberos is still far from ubiquitous—primarily because Kerberized applications used to be rare. Now that Windows 2000 has built-in support for Kerberos, the scheme has become much more popular.

Kerberos provides authentication based on secret key technology (DES for V4, DES, Triple DES, or other schemes for V5). Network users have conventional passwords that are effectively their secret keys. In addition, every *service* on the network (for example, Telnet, IMAP, and so on) has its own secret key, called a *service key*. In Kerberos parlance, we call these *services application services* to differentiate from services offered by the KDC. Rather than users authenticating to services, users and services (also called *principals*) authenticate *to each other*. Servers can detect and thwart imposters, and users can be assured that they are not talking to spoofed servers.

When authenticating or encrypting data, both the client and the server must share a secret, randomly generated session key. As discussed previously, the difficulty with secret key cryptography is in the key exchange. Principals could use their own secret keys to secure the key exchange. However, every principal would then have to know every other principal's secret key; this introduces another, much bigger, key exchange problem.

How Kerberos Works



When a participating entity wishes to communicate in a trusted manner with another participating entity over the insecure network, the KDC issues a ticket that becomes the basis for the establishment of trust between the two participating entities. Kerberos principals use encrypted timestamps to prove to the KDC that they possess the correct secret keys. Hence, the clocks of all participating entities on the network must be synchronized.

The KDC comprises two distinct services: the *authentication service* (AS) and the *ticket-granting service* (TGS). In Kerberos terms, *application services* refer to services such as Telnet and IMAP so they are not confused with the AS and TGS. Following are the steps for establishing an authenticated session between a client and server:

1. The Kerberos client software establishes a connection with the AS. The AS authenticates the client and provides the client with a secret key for this login session (the TGS session key) and a ticket-granting ticket (TGT), which gives the client permission to talk to the TGS. The ticket has a finite lifetime (for example, ten hours) to reduce the chances of it being stolen while it is still valid. Because the TGT expires, the authentication process must be repeated periodically if the login session has a long duration.
2. The client now requests from the TGS a *service ticket* for the application service it wants to contact. To make this request, the client must supply the TGS with the TGS session key and TGT it obtained in step 1. The TGS responds with two copies of a randomly generated session key: one encrypted with the application service's secret key (this is the *service ticket*) and one encrypted with the client's key.
3. The client can now prove its identity to the application service by supplying the service ticket. The application server responds with encrypted information to authenticate itself to the client. At this point, the client can initiate the intended service requests (for example, Telnet, FTP, HTTP, and so on).

DES: Data Encryption Standard

- Deviced in 1972 as a derivation of the Lucifer algorithm developed by Horst Feistel at IBM
- Released on March 17, 1975
- Symmetric key cryptosystem
- Used for commercial and non-classified purposes; *de facto* standard
- DES describes the data encryption algorithm (DEA)
- Rather fast encryption algorithm
- Symmetric-key, 64-bit block cipher
- 56-bit key size -> Small 2^{56} keyspace
- Today, DES is not considered secure.

Data Encryption Standard (DES)

DES is the most commonly used encryption algorithm in the world. On March 17, 1975, the United States government proposed its adoption as a national standard for use with unclassified computer data. Based on IBM's Lucifer cipher, DES is specified in Federal Information Processing Standard (FIPS) 42. The American National Standards Institute (ANSI) adopted DES as a standard (ANSI X3.92) in 1981, calling it the Data Encryption Algorithm (DEA).

Due to the internal bit-oriented operations in the design of DES, software implementations are slow and hardware implementations are faster. The National Institute of Standards and Technology (NIST) standardized four different DES operation modes for use in the United States: electronic codebook (ECB) mode, cipher block chaining (CBC) mode, output feedback (OFB) mode, and cipher feedback (CFB) mode.

DES Modes

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)

To make it more difficult to predict the plaintext message given the ciphertext, DES operates in four different modes, so that each block is encrypted with a different key. ECB is standard or native mode DES. This is where the plaintext message is broken into 64-bit blocks, and each block is encrypted with the key. If two blocks of the plaintext are identical, the corresponding ciphertext is also identical. The other modes of DES are meant to stop this from occurring.

With CBC, the system starts off with an IV, or initialization vector, which is a random number meant to seed the encryption. This IV is combined with the key and used to encrypt the first block of text. The encrypted ciphertext of the first block is combined with the key to encrypt the second block of text, and the process continues in this fashion.

Both CFB and OFC are stream ciphers. However, the key difference is that with CFB, errors will propagate; with OFC they will not.

ECB and CBC

- Electronic code book (ECB)
 - "Native" mode of DES
 - Block cipher
 - Applied to 64-bit blocks of plaintext and produces corresponding 64-bit blocks of ciphertext
- Cipher block chaining (CBC)
 - Operates with plaintext blocks of 64 bits
 - Randomly generated 64-bit initialization vector is xored with the first block of plaintext.
 - Used to disguise the first part of the message that may be predictable
 - Result encrypted using the DES key

Electronic code book (ECB)

- "Native" mode of DES
- Block cipher
- Applied to 64-bit blocks of plaintext, and produces corresponding 64-bit blocks of ciphertext

Cipher block chaining (CBC)

- Operates with plaintext blocks of 64 bits
- Randomly generated 64-bit initialization vector is xored with the first block of plaintext.
 - Used to disguise the first part of the message that may be predictable
- Result encrypted using the DES key

CFB and OFB

- Cipher feedback mode (CFB)
 - Stream cipher
 - Ciphertext is used as feedback into the key generation source to develop the next key stream.
 - Ciphertext generated by performing an xor of the plaintext with the key stream.
 - Ciphertext has the same number of bits as the plaintext.
 - In this mode, errors will propagate.
- Output feedback mode (OFB)
 - Stream cipher that generates the ciphertext key by xoring the plaintext with a key stream.
 - Feedback is used to generate the key stream; therefore, the key stream varies.
 - Initialization vector is required in OFB.

Cipher feedback mode (CFB)

Stream cipher

- Ciphertext is used as feedback into the key generation source to develop the next key stream.
- Ciphertext generated by performing an xor of the plaintext with the key stream.
- Ciphertext has the same number of bits as the plaintext.
- In this mode, errors will propagate.

Output feedback mode (OFB)

- Stream cipher that generates the ciphertext key by xoring the plaintext with a key stream.
- Feedback is used to generate the key stream; therefore, the key stream varies.
- Initialization vector is required in OFB.

DES Weaknesses

- DES is considered non-secure for sensitive encryption. It is crackable in a short period of time.
- See *Cracking DES* by O'Reilly Press.
- Multiple encryptions and key size increase security.
- Double DES is vulnerable to the meet-in-the-middle attack and only has an effective key length of 57 bits.
- Triple DES is preferred.

Strength of DES

From the beginning, concerns were raised about the strength of DES because of the rather small key length of 56 bits (a 64-bit ciphertext block minus 8 bits for parity); this resulted in a keyspace containing only 256 possible different keys. The effectiveness of attacks based on brute force searches depends on keyspace size. Because of DES's relatively small keyspace, brute force attacks are feasible. DES was first (publicly) cracked in the RSA Challenge, a program that offers monetary rewards for breaking ciphers and solving computationally intensive mathematical problems. The DES challenge took only five months for the public to solve, and subsequent attempts are taking less and less time.

Consequently, DES is no longer considered secure because of its key size. In fact, anyone can build a DES cracking engine these days. All the information you need—including sample code—is available in a book called *Cracking DES*. But with the global ecommerce infrastructure build-out proceeding at a furious pace because of all the new e-business initiatives that are sprouting up all over the world, the need for a fast, symmetric block cipher is extremely urgent. If DES can no longer be considered secure, what can we do in the interim?

Again, DES was already widely deployed in both hardware and software products, and it had withstood unbridled cryptanalysis for decades. It did not take long to realize what a great advantage it would be to somehow increase DES's key size and use the existing implementations until a new standard was built. One way to effectively increase the key length is to perform the encryption more than once. That is, encrypt the cleartext, and then encrypt the resulting ciphertext, and so on. However, this only works if the cipher algorithm is not a *group*.

We say that the function E is a group if $E(K_2, E(K_1, M)) = E(K_3, M)$. In other words, encrypting once with key K and then again with key K_2 is equivalent to encrypting once with K_3 . Thus, if a cipher algorithm is a group, encrypting multiple times is no stronger than encrypting once.

DES

- In 1992, it was proven that DES is not a group. This means that multiple DES encryptions are not equivalent to a single encryption. THIS IS A GOOD THING.
- If something is a group, then
 - $E(K_2, E(K_1, M)) = E(K_3, M)$.
- Because DES is not a group, multiple encryptions increase the security.

Whether an algorithm is a group is an important statistical consideration. If it is a group, applying the algorithm multiple times is a waste of time. In 1992, it was proven that DES is not, in fact, a group—thus, encrypting multiple times with DES is not equivalent to encrypting once. That is good news ' because it means that encrypting more than once with DES could increase the security of the ciphertext.

Meet-in-the-Middle Attack (Double DES)

So, double DES gives only effective key length of 57 bits, which is 1 more than DES.

However, encrypting twice with DES (Double DES) does not significantly increase the effective key size. If a cryptanalyst can obtain both a cleartext message (M) and its corresponding ciphertext (C), she can perform a *meet-in-the-middle attack*.

We already mentioned that brute force attacks on DES are feasible; this means that we can attempt to decrypt a message with every possible key until we find the one that gives us sensible cleartext. For a meet-in-the-middle attack, we first encrypt the cleartext M with every possible key K1:

$$C' = E(K1, M)$$

giving us 2^{56} values of intermediate ciphertext C. We then decrypt the ciphertext C with every possible key K2:

$$C' = D(K2, C)$$

Again, this gives us 2^{56} values of C. The values of K1 and K2 that yield the same C in the above equations are the two keys that are used for the double DES encryption. The number of operations, and therefore the resulting key length, is only $2^{56} + 2^{56} = 2^{57}$.

Triple DES

Triple DES:

- Encrypts a message three times
- Can be accomplished in several ways
- For example, the message can be encrypted with key 1, decrypted with key 2 (essentially another encryption) and encrypted again with key 1:
 - $[e\{d[e(m,k_1)], k_2\}, k_1]$
 - A triple DES encryption in this manner is denoted as des - ede2.

Triple DES

To thwart meet-in-the-middle attacks, Triple DES adds a third round of encryption. Thus, when performing the two steps of the meet-in-the-middle attack, a cryptanalyst ends up with two sets of ciphertext that are not comparable; they are separated by another encryption step. Triple DES is well-known, widely implemented, and has been intensely scrutinized by the global community of cryptologists. Furthermore, it uses the same tried and true DES algorithm, and all existing DES implementations can be used to perform Triple DES. See the ANSI X9.52 standard for additional information on Triple DES encryption.

Support for Triple DES is built into popular Web clients such as Microsoft Internet Explorer and Netscape Communicator. Using Triple DES from the end user perspective is often as simple as reconfiguring the Web browser's SSL version 3 support to prefer Triple DES over DES. SSL version 2 should be disabled altogether; otherwise the user could unwittingly communicate with a site using only DES, and those communications could be intercepted.

AES — Advanced Encryption Standard

- AES initiative was announced in January 1997 by the national institute of standards and technology (NIST).
- AES is a new encryption algorithm that is being designed to be effective well into the twenty-first century.
- AES is a block cipher that has replaced DES.
- Candidate encryption algorithm submissions were solicited.
- On August 29, 1998, a group of 15 AES candidates were announced.
- In 1999, NIST announced five finalist candidates.
- NIST closed round 2 of public analyses of these algorithms on May 15, 2000.
- On October 2, 2000, NIST announced the selection of the Rijndael block cipher as the proposed AES algorithm.
- AES is the new federal information processing standard (FIPS) publication 197.
- AES specifies a cryptographic algorithm for use by the U.S. Government to protect sensitive but unclassified information.
- AES has an iterated block cipher with a variable block length and key length that can be independently chosen as 128, 192, or 256 bits.

Advanced Encryption Standard (AES)

On January 2, 1997, NIST (National Institute of Standards and Technology) announced the initiation of an effort to develop the Advanced Encryption Standard (AES). On September 12, 1997, a formal call for algorithms was made that stipulated that AES must specify unclassified, publicly disclosed encryption algorithm(s) that were available royalty-free, worldwide. In addition, the algorithm(s) would implement symmetric key cryptography as a block cipher and (at a minimum) support a block size of 128 bits and key sizes of 128, 192, and 256 bits. The evaluation criteria were divided into three major categories: *security*, *cost*, and *algorithm and implementation characteristics*.

Table 1: AES Finalists

<u>Cipher</u>	<u>Developers</u>
MARS	IBM
RC6™	RSA Laboratories
Rijndael	Joan Daemen, Vincent Rijmen
Serpent	Ross Anderson, Eli Biham, Lars Knudsen
Twofish	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson

NIST selected the five AES finalists shown in Table 1 on August 9, 1999. In October of 2000, Rijndael (pronounced "Rain Doll" or "Rhine Dahl") was announced as the winner and approved as the official AES cipher. The two Belgian researchers who developed Rijndael are Dr. Joan Daemen (YO-ahn DAH-mun) of Proton World International and Dr. Vincent Rijmen (RYE-mun) of Katholieke Universiteit Leuven.

On December 26, 2001, NIST announced the approval of FIPS (Federal Information Processing Standards) 197, which describes AES as an official government standard by the U.S. Secretary of Commerce. FIPS 197 became effective on May 26, 2002. See the official AES Web site for more information: <http://www.nist.gov/aes/>.

The AES has supplanted the inadequate 56-bit DES, which is only to be used in legacy systems. AES has three key sizes: 128-bit, 192-bit, and 256-bit. NIST and the Canadian Communications Security Establishment (CSE) tested the algorithm.

AES Basic Functions

AES algorithm employs four basic transformations:

- *AddRoundKey*: XOR Round Key with State
- *SubBytes*: Substitute bytes in State *s* to form State *s'* on a byte-for-byte basis using S-box.
- *ShiftRows*: Left circular shift of rows 1-3 in State *s* by 1, 2, and 3 bytes, respectively
- *MixColumns*: Apply mathematical transformation to each column in State *s* to form State *s'*

AES Basic Functions

The AES algorithm employs four basic transformations:

- *AddRoundKey()*: Takes the appropriate round key and performs a bit-by-bit XOR with the current state
- *SubBytesQ*: Using a substitution box (S-box) defined in the specification, substitutes each 8-bit quantity in the State array to a different 8-bit value
- *ShiftRowsQ*: Circularly shifts the contents of state array rows 1, 2, and 3 by 1, 2, and 3 bytes, respectively, to the left. A left circular shift of one byte, for example, means that the bytes in columns 1, 2, and 3 move to positions 0, 1, and 2, respectively, and the value in byte position 0 moves to position 3.
- *MixColumnsQ*: Another byte value substitution, but performed on a column (32-bit) basis in this case; that is, rather than performing an S-box substitution on a per byte basis, this transformation applies a polynomial transformation on four bytes at a time.

IDEA

The idea cipher:

- The international data encryption algorithm (idea) cipher
 - Secure, secret, key block encryption algorithm
 - Operates on 64-bit plaintext blocks and uses a 128 bit key
 - Performs eight rounds and operates on 16-bit sub-blocks using algebraic calculations that are amenable to hardware implementation
- The international data encryption algorithm (IDEA) cipher
 - With its 128 bit key, an idea cipher is much more difficult to crack than des.
 - Idea operates in the modes described for des.

The idea cipher was developed by James Massey and Xuejia Lai.

It evolved in 1992 from earlier algorithms called the proposed encryption standard and the improved proposed encryption standard.

RC5

RC5:

- Family of cryptographic algorithms invented by Ronald Rivest in 1994
- Block cipher of variable block length
- Typical block sizes are 32, 64, or 128 bits.
- Key size can range from 0 to 2048 bits.

Encrypts through:

Integer addition

Application of a bit-wise exclusive

Variable rotations

Key size and number of rounds are also variable.

Number of rounds can range from 0 to 255.

Patented by RSA data security in 1997.

Asymmetric Encryption

An Example of an Intractable Problem...

Difficulty of factoring a large integer into its two prime factors.

- A "difficult" problem.
- Years of intense public scrutiny suggests intractability.
- No mathematical proof so far.

Example: RSA

- Based on difficulty of factoring a large integer into its prime factors
- ~1000 times slower than DES
- Considered "secure"
- *De facto* standard
- Patent expired in 2000

Prime Factorization of Large Integers

Factoring integers does not seem too difficult. It does not take much thought to figure out that 15 can be factored into 1×15 and 3×5 . So why is it on our list of intractable problems?

The operative word here is *large*. The larger the integer, the more difficult it is to factor. In fact, there is no known recipe for factoring other than trial and error: keep multiplying primes together until you arrive at the number. Remember that, even though most researchers in complexity theory believe factoring large integers is a hard problem, there is no unequivocal proof to that effect. Only the years of public scrutiny of this problem lead us to conclude the problem cannot be solved in polynomial time.

Perhaps the most popular public-key algorithm today RSA — takes advantage of the intractability of the integer factorization problem.

Asymmetric Encryption (2)

Another Intractable Problem...

Difficulty of solving the discrete logarithm problem—for finite fields.

- A "difficult" problem.
- Years of intense public scrutiny suggests intractability.
- No mathematical proof so far.
- The discrete logarithm problem is as difficult as the problem of factoring a large integer into its prime factors.

Examples

- El Gamal encryption and signature schemes
- Diffie-Hellman key agreement scheme
- Schnorr signature scheme
- NIST's Digital Signature Algorithm (DSA)

The Discrete Logarithm Problem for Finite Fields

Another intractable problem is the *discrete logarithm problem for finite fields*. The discrete logarithm is based on a statement of the form $ax \bmod n = b$, where a , b , n , and x are integers and a and n are known. The *mod* operator simply means that we take the remainder of the first number (a^*) when divided by the second number (n)—finding b when we know that x is easy, but not the other way around.

For example, it is easy to calculate $8^3 \bmod 7$; because $8^3 = 512$ and the next lowest multiple of 7 is 511, the remainder must be $512 - 511 = 1$. However, it takes trial and error to discover that $8^x \bmod 7 = 1$ is satisfied only by $x = 3$. This problem is the discrete logarithm. Just as with prime factorization, the problem *really* becomes difficult when x is a hundred- or thousand-bit number.

Again, the notion that discrete logarithms are intractable is the consensus of computational complexity researchers, and there is no unequivocal proof that this problem cannot be solved easily. The years of public scrutiny of this problem lead us to conclude that it is a difficult problem that cannot be solved in polynomial time. But how does it compare with the previous intractable problem we considered: The factorization of large integers into two primes? Evidence shows that the discrete logarithm problem is just as difficult.

Therefore, we should be able to use the discrete logarithm problem in building a cipher. In fact, several ciphers that are in use today are built upon the intractability of the discrete logarithm problem over finite fields: the ElGamal encryption and signature schemes, the Diffie-Hellman key agreement scheme, the Schnorr signature scheme, and the Digital Signature Algorithm (DSA) by the U.S. Department of Commerce's National Institute of Standards and Technology (NIST).

Asymmetric Encryption (3)

Yet Another Intractable Problem...

Difficulty of solving the discrete logarithm problem—as applied to elliptic curves.

Examples

- A "difficult" problem.
 - Years of intense public scrutiny suggests intractability.
 - No mathematical proof so far.
 - In general, elliptic curve cryptosystems (ECC) offer higher speed, lower power consumption, and tighter code.
- Elliptic curve El Gamal encryption and signature schemes
 - Elliptic curve Diffie-Hellman key agreement scheme
 - Elliptic curve Schnorr signature scheme
 - Elliptic Curve Digital Signature Algorithm (ECDSA)

The Discrete Logarithm Problem for Elliptic Curves

The ciphers mentioned in the previous section use the discrete logarithm problem, but only for certain sets of numbers that belong to what are known as *finite fields*. It turns out this problem also makes for a good cipher algorithm when it is applied to *elliptic curves*.

This class of problem is considered every bit as intractable as the previous two. In addition, it lends some additional useful features to our algorithm: high security levels even at low key lengths, high speed processing, and low power and storage requirements. These characteristics are useful in crypto-enabling the many new devices that are rapidly appearing in the marketplace — that is, mobile telephones, information appliances, smart cards, and even the venerable ATM.

RSA

RSA:

- Rivest, Shamir and Addleman
- Supports secrecy and authentication
- Based on difficulty of finding the prime factors of large integers
- Keys of 1024 bits or greater

RSA

The RSA algorithm has been widely implemented all over the world in all kinds of cryptography-enabled applications. It can be used to support both encryption and digital signature schemes. A central part of the Secure Sockets Layer (SSL), it is also included in major Web clients such as Microsoft Internet Explorer and Netscape Communicator.

Although there have been several claims to having cracked the RSA algorithm, they have all turned out to be false. Vulnerabilities have been found in certain RSA implementations, however. Poor implementations of the RSA algorithm can be compromised, but as in the case of other cryptographic algorithms, it does not mean that the algorithm itself has been cracked.

The working mechanism of most public-key (asymmetric) cryptographic algorithms is generally openly published and widely known. The security of the cryptosystem comes from the secrecy and size of the private key and not from the secrecy of the algorithm itself. As for other cryptographic algorithms, it is important to ensure that the key size is not so small that brute force attacks become feasible as a result of the small size of the resulting key space.

RSA vs. DES (Asymmetric vs. Symmetric)

- Fastest implementation of RSA can encrypt kilobits/second.
- Fastest implementation of DES can encrypt megabits/second.
- It is often proposed that RSA can be used for secure exchange of DES keys.
- This 1000-fold difference in speed is likely to remain independent of technology advances.
- In software, DES is approximately 100 times faster than RSA.

RSA versus DES (Asymmetric Versus Symmetric)

Symmetric cryptography is generally much faster than asymmetric cryptography. Although the fastest hardware RSA implementation can encrypt on the order of kilobits per second, hardware DES is on the order of megabits per second. (DES was designed to run slowly in software, so in software it is only about 100 times faster.) The major drawback to symmetric cryptography is that, because both the sender and receiver use the same key, the key must be exchanged via a secure mechanism before the two parties can communicate. Therefore, RSA is often used for the initial exchange of a symmetric session key. After the session key has been securely transmitted, Triple DES or some other symmetric cipher is used for the remainder of the session. Thus, we take advantage a symmetric cipher's speed without the worry of a shared key getting misplaced or stolen.

Elliptic Curve Cryptosystem

Elliptic curve:

- Analogous to discrete logarithm problem
- Recent technology
- Faster
- Amenable to hardware implementation
- Larger cryptographic strength per bit than other approaches
- Can independently implement digital signatures and encryption

Like RSA, ECCs are capable of supporting both encryption and digital signature schemes. In addition, the ECC has some interesting characteristics: high security even at relatively small key lengths (that is, a higher strength per bit), high-speed implementations, low processing power requirements, and low storage requirements. These properties make ECCs particularly attractive for use in resource-constrained computing environments, such as mobile telephones, PDAs, information appliances, smart cards, and ATMs. We expect to see increasing deployments of ECC-enabled applications in our ecommerce-enabled environments.

Cryptanalysis Attacks

When developing a cipher, it helps to be aware of the types of attacks cryptanalysts employ when trying to break ciphers. Each type of attack has to do with what pieces of information a cryptanalyst has in her possession. We assume that the cryptanalyst always has full knowledge of the cipher algorithm.

Cryptographic Technologies

Comparison of asymmetric and symmetric algorithm key sizes for equivalent protections

ASYMMETRIC KEY SIZE	SYMMETRIC KEY SIZE
512 Bits	64 Bits
1792 Bits	112 Bits
2304 Bits	128 Bits

This space intentionally left blank.

Digital Signatures

- Electronic analog of a handwritten signature
- Must provide that:
 - A receiver must be able to validate the sender's signature.
 - Signature must not be forgeable.
 - Sender of a signed message must not be able to repudiate it later.

According to NIST digital signature standard (DSS), [National Institute of Standards and Technology, nist fips pub 186, U.S. Department of Commerce, May 1994]:

"Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory. In addition, the recipient of signed data can use a digital signature in proving to a third party that the signature was in fact generated by the signatory."

Digital Signatures (2)

- Digital signature characteristics
 - Is not constant
 - Is a function of the document which it signs
 - Is a function of the entire document that it signs, such that changing one bit would change the signature

A digital signature should be a unique function of the document it represents such that a change in the document would result in a change in the digital signature. Thus, if a digital signature of a message is calculated at the sending end and attached to the message, the receiver can apply the same digital signature algorithm to the message when it is received. Then, the receiver can compare his or her calculated digital signature with the signature that was sent with the message. If the two are identical, the message was not changed in transit.

Digital Signatures (3)

- Digital signature implementations
 - True: Signed messages are forwarded directly from signer to recipient.
 - Arbitrated: Human or automated witness validates a signature and transmits the message on behalf of the sender.

Digital signature implementations

- True: Signed messages are forwarded directly from signer to recipient.
- Arbitrated: Human or automated witness validates a signature and transmits the message on behalf of the sender.

Hash Functions and Message Digest

Hash function collision avoidance:

- It must not be computationally feasible to find a message that hashes to the same digest as a given message.
 - $H(m1) \neq h(m2)$
- It should not be possible to find any two strings that hash to the same message digest (birthday attack).

Birthday attack comes from the mathematical question that asks:

"How many people must be in a room to have a greater than 50% chance that two or more people will have the same birthday?" It can be any day of the year, not a specific date that has to be found.

The answer is 23.

Thus, if any two messages can generate the same hash function (message digest), this is a birthday attack because one of the messages can substitute for the other and the change not detected after transmission.

Public Key Electronic Signatures

- Apply originator's private key to data or message digest of data.
- Send result with message.
- Public key of signer opens data.

After the message digest is calculated, it is encrypted with the sender's private key. The encrypted message digest is then attached to the original file and is sent to the receiver. The receiver then decrypts the message digest by using the sender's public key. If this public key opens the message digest and it is the true public key of the sender, verification of the sender is then accomplished. Verification occurs because the sender's public key is the only key that can decrypt the message digest encrypted with the sender's private key.

Then, the receiver can compute the message digest of the received file by using the identical hash function as the sender. If this message digest is identical to the message digest that was sent as part of the signature, the message has not been modified.

Crypto Attacks

- Brute force - generic
 - Try every combination of keys & passwords.
- Man-in-the-middle - generic
 - Attacker intercepts messages between two parties.
 - Intercepts messages before passing them on to intended receiver
- Known plaintext - symmetric
 - Portions of plaintext and corresponding portions of ciphertext are known.
- Ciphertext only - generic
 - Portion of ciphertext is known.
- Chosen plaintext - symmetric
 - Plaintext inserted into device with unknown secret key and corresponding ciphertext is generated.
- Adaptive chosen plaintext - symmetric
 - Chosen plaintext attack with iterations of input is based on knowledge of output.

Cipher Attacks

A cryptanalyst with access to both the ciphertext and the plaintext of a message can mount a *known-plaintext attack*. The goal is to find the key used to encrypt the ciphertext or an alternate algorithm to decrypt *any* message with a key the cryptanalyst knows.

Similar to the known-plaintext attack is the *chosen-plaintext attack*. For this attack, the cryptanalyst is able to choose what plaintext gets encrypted and see the resulting ciphertext. At times, being able to choose what gets encrypted can reveal information about the key.

An *adaptive-chosen-plaintext attack* is a special case of the chosen-plaintext attack. After choosing the plaintext that gets encrypted, the cryptanalyst can also choose other blocks to be encrypted. This attack allows even more analysis based on the results of each encryption step.

The previous attacks all require the cryptanalyst to have plaintext and ciphertext versions of a message. Attacks can be guarded against by keeping the plaintext secret and deleting it when it is no longer needed. You must also guard against mechanisms that allow an attacker to encrypt arbitrary messages using your secret key. Even if the attacker does not know the key, he could use an adaptive-chosen-plaintext attack by encrypting his own crafted messages.

More Crypto Attacks

- Chosen ciphertext - asymmetric
 - With a portion of ciphertext, attempt to obtain corresponding plaintext.
- Adaptive chosen ciphertext- asymmetric
 - Chosen ciphertext attack with iterations dependent upon previous results
- Chosen key attack - asymmetric

A *ciphertext-only attack* requires only encrypted messages; no plaintext is available. The goal is to recover one or more plaintext messages or the key used to encrypt the messages.

In a *chosen-ciphertext attack*, the cryptanalyst can choose the ciphertext to be decrypted. Thus, the cryptanalyst has ciphertext and plaintext for messages that he chooses. This attack is primarily used against public-key ciphers.

In a *chosen-key* attack, the cryptanalyst knows something about specific relationships between the keys. Contrary to what the name suggests, the cryptanalyst does not choose the key; that would not leave much to reveal!

Cryptographic Attacks (Cryptanalysis)

- **Analytic**
 - Using algorithms and mathematics to deduce key or reduce key space to be searched
- **Statistical**
 - Using statistical characteristics of language or weaknesses in keys
- **Differential**
 - Analyze resultant differences as related plaintexts are encrypted using a cryptographic key

Analytic

Using algorithms and mathematics to deduce key or reduce key space to be searched

Statistical

Using statistical characteristics of language or weaknesses in keys

Differential

Analyze resultant differences as related plaintexts are encrypted using a cryptographic key

Cryptographic Attacks (Cryptanalysis) (2)

- Linear
 - Linear analysis of pairs of plaintext and ciphertext
- Differential linear
 - Applying differential analysis with linear analysis

Linear

Linear analysis of pairs of plaintext and ciphertext

Differential linear

Applying differential analysis with linear analysis

Birthday Attack

- Probability of two different messages using the same hash function producing a common message digest
 - When 23 people are put together, the odds are greater than half that two or more people share a birthday.
- Our attack relates to that probability and hash collisions.

The Birthday Attack

Cryptanalysts can sometimes use a phenomenon known as *the birthday paradox* to attack hash signatures. People in large groups often find that at least two of them share the same birthday. They are usually astonished at the coincidence, thinking that the odds must be slim that two people could be born on the same day of the year. It is true that it would be rather unusual to find a person with your exact birthday unless the groups were large. The odds of finding someone born on a particular day are 1 in 365 (assuming that all days of the year are equally likely birthdays and that nobody is born on February 29).

Without specifying who, simply specifying that any two people have the same birthday improves the odds considerably. For a group as small as 23 people, the odds are greater than 50% that two or more of them will share a birthday. If each of the 23 people compares birthdays with another, you would have 253 comparisons. The odds, then, that *none* of the 23 have the same birthday are $(364/365)^{253} = 0.4995$. Thus, the odds that two of them share a birthday are $1 - 0.4995 = 0.5005$.

Just as pairs of people in a group might have the same birthday, pairs of messages might have the same hash signature. Of course, there are many more possibilities for hash signatures than birthdays, but the same logic applies. If an attacker can find any two messages that generate the same hash value — that is, a *collision* — she could substitute one message for the other at will. For example, perhaps she has a list of password hashes but not the cleartext. If she can hash enough of her own generated cleartext to cause a collision, she has a password that works just as well as the real thing.

The entire attack is a statistical probability problem. Using the birthday attack against MD5 is improbable because of processing power limitations with our PCs. A machine processing a billion messages per second would take 586 years on average to identify the inputs. We hope that our users would have changed their passwords by then!

Public Key Infrastructure

Specifies a basis for interoperation
between components from different
vendors

PKI, or public key infrastructure, is the glue that holds crypto together. As we discussed the different types of crypto and how they are used, one issue that continues to surface is that of key exchange. How do you exchange keys in a trusted manner, so that someone cannot launch a man-in-the-middle attack against you? PKI is the solution for exchanging and validating keys across a wide geographical area.

Public Key Infrastructure: Components

Divided into five components:

- Certification authorities (CA) that issue and revoke certificates
- Organizational registration authorities(ORA) that vouch for the binding between public keys, certificate holder identities, and other attributes
- Certificate holders that are issued certificates and can sign digital documents
- Clients that validate digital signatures and their certification paths from a known public key of a trusted CA
- Repositories that store and make certificates and certificate revocation lists (CRL's) available

PKI is composed of five main components. On some systems, these components can be combined or expanded, but nonetheless, all the functionality must be addressed. The PKI infrastructure must have some CA or certification authorities that everyone trusts. If PKI is meant to create a trusted path for communicating keys, there must be some central authority that everyone trusts. This is a critical pillar of PKI; this trusted authority is responsible for issuing and revoking keys.

After you have a CA that everyone trusts, you still need a way to bind users to a given key so they can be trusted. There are many different ways to do this, but one of the most common is with an organizational registration authorities(ORA) that vouch for the binding between public keys and certificate holder identities and other attributes.

After the infrastructure is in place, you need people who have digital certificates called *certificate holders*. These certificate holders can sign and authenticate documents across a network. It is beneficial to have an infrastructure to support PKI if someone has certificates.

When you have people who sign documents to prove that a piece of information came from them and is authentic, you need to have clients who can validate the certificates. Finally, all these certificates do not do any good if there is not some central repository in which to store them.

Public Key Infrastructure (PKI)

- Integration of digital signatures, certificates, and other services required for e-commerce
- Includes:
 - Digital certificates
 - Certificate authority
 - Registration authorities
 - Policies and procedures
 - Certificate revocation
 - Non-repudiation support
 - Cross certification

The PKI is actually a combination of individual components. You cannot point to an individual component and say that is PKI; instead, you would point to a range of systems and say that all those components together implement PKI. Following are some of the key components of PKI:

- Digital certificates: Something users can present to someone to prove that he is who he says he is.
- Certificate authority: A central system that everyone trusts to validate certificates from unknown entities.
- Registration authorities: An authority with which everyone can register and that validates the information.
- Policies and procedures: Rules for exchanging and validating keys.
- Certificate revocation: Being able to revoke a certificate when someone leaves an organization or in the event that he is no longer trusted.
- Non-repudiation support: The feature that allows someone prove in a court of law that someone actually sent or agreed to something.
- Time stamping: Being able to prove the time and date on which a certain event occurred.
- Cross certification: People being able to have people who trust two different CAs to trust each other.

Escrowed Encryption

NIST FIPS Pub 185, "escrowed encryption standard/' U.S. Department of Commerce, Feb. 1994

- Strives to achieve individual privacy
- Strives to provide for legal monitoring of the encrypted transmissions

Escrow of encryption keys is a hot topic that, over the last decade, constantly reappears based on political events that occur. The entire concept of key escrow is to have some central authority, usually the government, which contains a copy of everyone's key so the CA can decrypt any message it wants. As you can imagine, this would have to be carefully controlled so it would not get out of hand. The most common way it is done is by mandating the use of a certain encryption algorithm that has built-in keys that can be used to decrypt any message.

Escrowed Encryption: Details

- Divide the key into two parts.
- Escrow two portions of the key with two separate "trusted" organizations.
- After obtaining a court order, law enforcement officials can retrieve the two pieces of the key from the organizations and decrypt the message.
- Escrowed encryption standard is embodied in the U.S. Government's clipper chip:
 - Implemented in tamper-proof hardware
 - The skipjack secret key algorithm performs the encryption (algorithm now unclassified).

One of the big issues that arises with key escrow is who do you trust with the key. One party who has the ability to read everyone's private message is probably too great a risk; therefore, the concept of separation of duties comes into play. This is where you break the key into many parts—usually two—and two different groups work together to read the messages. Even doing this still requires that those two organizations are properly trusted and do not abuse their privileges.

Escrowed Encryption: Fair Cryptosystems

- In 1992, Sylvio Micali introduced the concept of fair cryptosystems.
- The private key of a public/private key pair is divided into multiple parts and distributed to different trustees.
- One valuable characteristic of Micali's approach is that each portion of the secret key is verified as correct without having to reconstruct the entire key.
- Accomplished by giving each trustee a piece of each public and private key.

In 1992, Sylvio Micali introduced the concept of fair cryptosystems, which takes the concepts of key escrow and asymmetric key encryption and combines them. Remember, with key escrow systems, a trusted third party who could potentially decrypt every message stores the keys. Also, asymmetric encryption is based on two-key encryption where there is a public and a private key. With this scheme, the private key of a public/private key pair is divided into multiple parts and distributed to different trustees. One valuable characteristics of Micali's approach is that each portion of the secret key verified as correct without having to reconstruct the entire key.

Key Management Issues

- Key management
 - Protects keys against:
 - Modification
 - Unauthorized disclosure
- Procedures and protocols (manual and automated)
 - Key generation, distribution, storage, entry, use, recovery, destruction, and archiving:
 - Key notarization
 - PKI: Support for binding a key and its owner

As with any solution, there are benefits to using key management and also weaknesses. With PKI, there are critical components that are required for the system to work properly. For example, the CA that everyone trusts must properly validate everyone's certificate. If an attacker can convince the CA that he is someone else and get improperly validated, the system starts to collapse. Also, with any type of authentication, such as passwords, the security begins to breakdown if users share their information with others and do not properly protect the information. PKI is no different; anyone who uses it must be properly trained and must follow clear guidelines.

Key Control

Issues and Measures

- Key recovery
- Key storage
- Key retirement/destruction
- Key change
- Key generation
- Key theft
- Frequency of key use

Following are some of the key issues associated with properly controlling and securing PKI:

- Key recovery: How you recover a key if someone loses it.
- Key storage: Where you store your key. On the one hand, you want to have a backup, but you also want to make sure that the backup is secure.
- Key retirement/destruction: Keys are not going to last forever, but if you do not carefully discard it, someone else can get a hold of it.
- Key change: Based on certain events, you might feel, that your key is no longer secure and wish to change it.
- Key generation: When you decide to change your key, what is the process to generate a new secure key?
- Key theft: How do you determine or figure out whether your key has been stolen?
- Frequency of key use: Based on how often you use your key and where you use it. Is there a higher frequency that it might be compromised?

Secure Sockets Layer (SSL)/ Transaction Layer Security (TLS)

- Developed by Netscape in 1994
- Authenticates the server to the client
- Provides for optional client-to-server authentication
- Supports public key and private key and hash function
- Https
- *De facto* standards
- Does not provide the end-to-end capabilities
- Between the application and TCP layer

The notion of a server-side certificate is a compelling one. The software to handle the encryption is bundled into the browser, so it is as simple as clicking on a link marked "https" instead of "http." As previously mentioned, this is a bit of a panacea; most Web servers do not have "valid" certificates, and the credit card information depends on the Web server's security, which is not a comforting thought. Finally, the transactions are probably cleared between the company and the bank in the clear or with DES encryption, at best. Still, the solid key makes the consumer happy and willing to put his credit card information into the Web form, which might be the only thing that matters.

Netscape Communications Corporation introduced the **Secure Sockets Layer (SSL)** protocol in 1994. Because support for SSL has been available in Netscape clients/browsers since the initial release of Navigator v1.0, it quickly became the *de facto* standard. Today, SSL v3 support is built into virtually all popular Web clients and browsers. SSL is used in more applications than just Web, including Usenet news. The **IETF** is currently developing the open-standard **Transport Layer Security (TLS)** protocol, which is based on SSL v3.0.

Secure Shell (SSH-2)

Set of protocols that are primarily used for remote access over a network by establishing an encrypted tunnel between an SSH client and an SSH server.

Many of the common protocols today, such as Telnet and FTP, are not considered secure. One of the main reasons is that all of the data — including the passwords — are sent over the wire in plaintext. Anyone intercepting the communication can read and capture someone's password. SSH is a secure alternative where not only the passwords, but all the data is sent over the wire encrypted.

Pretty Good Privacy (PGP)

- Public key cryptography (DH/DSS, RSA) to encrypt/sign e-mail and files
- Developed by Philip Zimmermann (1991)
- Confidentiality: CAST, IDEA, or Triple-DES
- Integrity: MD5
- Authentication: Knowledge of private-key
- Non-repudiation: Digital signature

Case Study: PGP

Pretty Good Privacy (PGP) is currently one of the most widely used public-key cryptography products. Developed by Philip Zimmermann in 1991, PGP was the subject of controversy and debate for a long time because it made strong cryptography available to virtually anyone; it was therefore in possible conflict with the laws governing the export of crypto products at the time. PGP is available as a plug-in for many e-mail systems, including Claris E-mailer, Microsoft Outlook/Outlook Express, and Qualcomm Eudora.

PGP can be used to sign or encrypt e-mail messages with a mere click of the mouse. Depending on the version, PGP's crypto algorithms include the following:

- SHA-1 or MD5 for message hashing
- DES, CAST, Triple DES, or IDEA for encryption
- RSA or DSS/Diffie-Hellman for key exchange and digital signatures

The PGP distributed web-of-trust model is based on the concept of trusted introducers or CAs. In this model, each party can independently decide whom it will trust to introduce other parties to it. A PGP user can sign the PGP public key of any other PGP user; by doing so, the signer asserts that the public key belongs to that person and the person understands how to use PGP. What does this mean to a person who is presented with a signed PGP public key? The recipient can examine the public key with which he has been presented to determine exactly who has signed it. Based on the number of people who have signed the key and the quality of these signers, the recipient can then decide whether to trust that particular key. The quality of a signer is a subjective decision that the key recipient makes on a case-by-case basis.

When PGP is first installed on a computer, the user creates a public/private key pair. The public key can be advertised and widely circulated, whereas the private key is protected by a passphrase the user enters each time the private key is used.

IPSec

- Standard that provides encryption, access control, non-repudiation, and authentication of messages over IP
- Designed to be functionally compatible with IPv6
- Two main protocols of IPSec are:
 - Authentication header (AH)
 - The encapsulating security payload (ESP)
- AH provides integrity, authentication, and non-repudiation.
- ESP primarily provides encryption, but it can also provide limited authentication.

IPSec is a Layer 3 method for providing tunnels. As illustrated here, it is an IETF standard that enables encrypted communication between users and devices. The goal is enabling a lot of different types of devices to understand one another. One of the first applications of IPSec is remote Access VPNs.

IPSec is transparent to the network infrastructure and scalable from small applications to large networks.

As an open standard, IPSec is available to everyone, so vendors can ensure interoperability. As of now, different levels of implementation are available among the different vendors, but ideally, the same technology must be available to everyone to assure future interoperability in multi-vendor networks, including the Internet.

At Cisco, IPSec functionality is available in Cisco IOS Software releases 11.3T and later. Initially, Cisco targeted gateway devices for IPSec, including routers and access servers.

IPSec Security Association (SA)

- The heart of IPSec
- An SA is required for communication between two entities.
- Provides a one-way (simplex) connection
- Comprised of a security parameter index (SPI), destination IP address, and the identity of the security protocol (AH or ESP)
- SPI is a 32-bit number that is used to distinguish among various SAs that terminate at the receiving station.
- Because an SA is simplex, two SAs are required for bi-directional communication between entities.

There are two basic modes you can use when encrypting information. The first and most commonly used is the Tunnel Mode. This mode is applied to an IP tunnel between gateway devices and can also be used on remote clients talking to gateways. In Tunnel Mode, the original packet is encrypted. The IPSec header is then added (as we just discussed), along with a second IP header that corresponds to the gateway to which you want to talk. The flow goes like this: Information goes to the first gateway, which encodes the payload, puts a new header on, and sends it to the second gateway. The second gateway strips the new header, decrypts the payload, checks the packet for integrity, and forwards it to the destination.

Transport Mode happens between two hosts. As diagrammed here, the packet header is removed, the payload is encrypted, an IPSec header is added, the first header is reattached, and the packet is forwarded.

IPSec: AH and ESP

- If the AH protocol is used and bi-directional communication is required, two SAs must be established.
- If both the AH and ESP protocols are to be employed bi-directionally, four SAs are needed.
- In a VPN implementation, IPSec can operate in either the transport or tunnel mode.
- Security associations can be combined into "bundles" to provide authentication, confidentiality, and layered communication.
- SA bundles can be developed using transport adjacency or iterated tunneling.

IPSec adds two headers to an IP packet.

First, it provides an authentication header, which provides knowledge that a packet originated from a trusted source. It also guarantees that if a packet is changed, you know it. This is not encryption; it simply ensures that information is not intercepted and that its content has not changed.

The second header is the encapsulated security payload. This does the same thing as the authentication header and also allows you to encrypt the payload.

Steganography (Stego)

- Steganography is abbreviated as stego, but is not to be confused with stenography.
- It involves concealing the fact that you are sending "sensitive" information.
- Data hiding
- Relatively new field
- Can hide in a variety of formats:
 - Images: bmp, gif, jpg
 - Word documents
 - Text documents
 - Machine-generated images: Fractals

Steganography is fairly new, but it is an active and growing field. It involves hiding data within a file, such as an image or sound file, so the meaning of the message and the fact that a message is being sent is concealed. Numerous methods allow data to be embedded in a wide range of file types. Data can be hidden in images, such as bitmap, GIF, or jpeg files, in Microsoft Word documents,, or even in computer-generated pictures such as fractals.

Crypto versus Stego

- Cryptography (Crypto) provides confidentiality, but not secrecy.
- It is fairly easy to detect that someone is sending an encrypted message; however, it is difficult for someone to read it.
- With stego, you do not even know that someone is sending a message; you are hiding the true intent.
- Ideally, you would combine the two together.

We now compare cryptography to steganography. With crypto, an unauthorized party cannot read the message because it has been mathematically combined with an encryption key. However, by analyzing the traffic, the party can tell that the data has been encrypted. With stego, someone cannot tell that a secret message is being sent because the message is hidden.

How Steganography Works

- Stego requires a host (to carry the data) and the hidden message.
- The host (usually a file) can be generated on the fly or by using existing data.
- The message can be hidden in certain parts of an existing file or can cause a new file to be generated.

Because stego is data hiding, you must be able to hide information within another file. Therefore, stego requires an overt file and a covert file. The overt file is the open file that is exchanged between two parties; the covert or secret message is the data that is hidden within the overt file.

General Types of Stego

- There are many ways to hide information; take a lesson in creativity.
- General methods:
 - Injection
 - Substitution
 - Generate new file

To hide data within a file, the secret message can be embedded or injected within another file. This technique is called injection. It increases the size of the file and is often easier to detect.

Alternatively, certain information in the host file might be able to be replaced, which will not increase the size of the file. By overwriting pieces of the original message, this so-called substitution technique could lower the quality of the resulting file.

Finally, a newer technique involves using the secret message to generate a new text or image file.

Injection

- Most file types have ways of putting information in a file that will be "ignored/"
- For example, think of hidden form elements in HTML.
- Word documents also have hidden information.
 - Create a large document and remove data; notice that file size is large.

With injection, data is put within a host file in such a way that when the file is actually read by a given program, the program ignores the data. Most programs, such as a Web browser or word processor, have ways of putting data "hidden" within a file. This hidden data exists in the file, but is ignored when the program displays it to the user.

Substitution

- Data in a file can be replaced or substituted with hidden text.
- Depending on the type of file and/or the amount of data, it could result in degradation of the file.
- It usually replaces insignificant data in the host file.

Another technique for hiding data in a file is substitution. Data in a host file can be replaced or substituted with the message to be hidden. If there is a small amount of text to be hidden, little change will occur to the host file. If a large amount of text is to be hidden, the host file could be degraded significantly. To make this technique undetectable to the human observer, substitution usually replaces insignificant data in the host file.

Generate New File

- The hidden data can also be used to generate a new file.
- No host file is needed.
- For example, the input text can be used to generate fractals or "human-like" text.

A third technique eliminates the need for a host file. The secret message itself can be used to generate a new file. For example, a file that consists of complex fractals can be generated based on using the input file as a set of parameters for the fractal generation routine. This means that each unique input file would generate a unique output file that could be analyzed to retrieve the original message. This technique is often used for computer generated images or sounds.

Summary

- Cryptography is critical to Defense-in-Depth.
- All three types of cryptography are used together:
 - Symmetric
 - Asymmetric
 - Hash

This domain was a fairly technical domain that covered the details of cryptography. We saw that cryptography is nothing new; it has been used throughout history and will continue to be used in the future. Cryptography does not solve all of our problems, but it will go along way to providing a robust Defense-in-Depth security infrastructure when used in conjunction with other security devices.