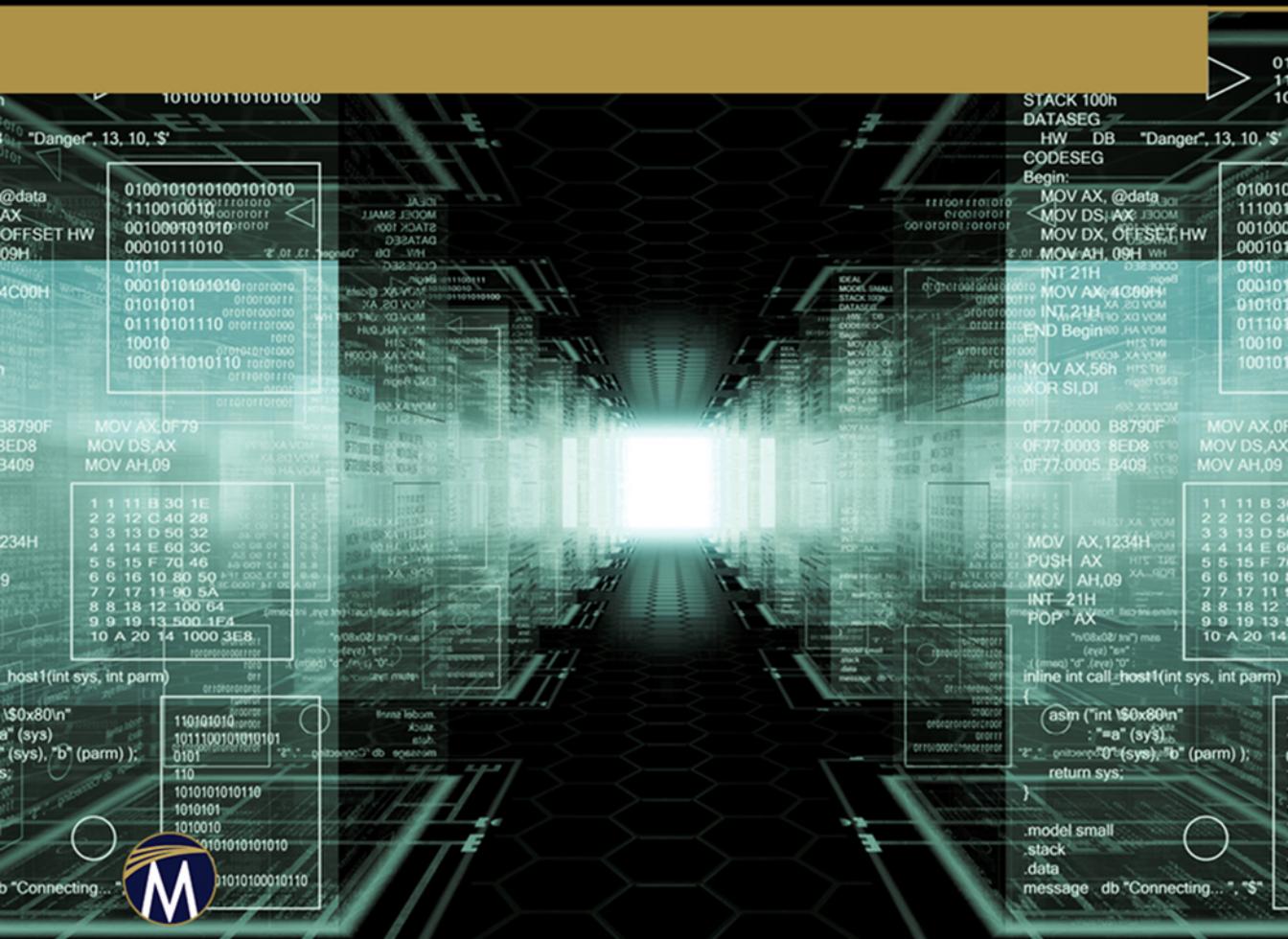


CRYPTOGRAPHY AND NETWORK SECURITY

An Introduction



R. Achary

CRYPTOGRAPHY AND NETWORK SECURITY

LICENSE, DISCLAIMER OF LIABILITY, AND LIMITED WARRANTY

By purchasing or using this book (the “Work”), you agree that this license grants permission to use the contents contained herein, but does not give you the right of ownership to any of the textual content in the book or ownership to any of the information, files, or products contained in it. *This license does not permit uploading of the Work onto the Internet or on a network (of any kind) without the written consent of the Publisher.* Duplication or dissemination of any text, code, simulations, images, etc. contained herein is limited to and subject to licensing terms for the respective products, and permission must be obtained from the Publisher or the owner of the content, etc., in order to reproduce or network any portion of the textual material (in any media) that is contained in the Work.

MERCURY LEARNING AND INFORMATION (“MLI” or “the Publisher”) and anyone involved in the creation, writing, production, accompanying algorithms, code, or computer programs (“the software”), and any accompanying Web site or software of the Work, cannot and do not warrant the performance or results that might be obtained by using the contents of the Work. The author, developers, and the Publisher have used their best efforts to ensure the accuracy and functionality of the textual material and/or programs contained in this package; we, however, make no warranty of any kind, express or implied, regarding the performance of these contents or programs. The Work is sold “as is” without warranty (except for defective materials used in manufacturing the book or due to faulty workmanship).

The author, developers, and the publisher of any accompanying content, and anyone involved in the composition, production, and manufacturing of this work will not be liable for damages of any kind arising out of the use of (or the inability to use) the algorithms, source code, computer programs, or textual material contained in this publication. This includes, but is not limited to, loss of revenue or profit, or other incidental, physical, or consequential damages arising out of the use of this Work.

The sole remedy in the event of a claim of any kind is expressly limited to replacement of the book and only at the discretion of the Publisher. The use of “implied warranty” and certain “exclusions” vary from state to state, and might not apply to the purchaser of this product.

CRYPTOGRAPHY AND NETWORK SECURITY

An Introduction

R. ACHARY



MERCURY LEARNING AND INFORMATION
Dulles, Virginia
Boston, Massachusetts
New Delhi

Copyright ©2021 by MERCURY LEARNING AND INFORMATION LLC. All rights reserved.

This publication, portions of it, or any accompanying software may not be reproduced in any way, stored in a retrieval system of any type, or transmitted by any means, media, electronic display or mechanical display, including, but not limited to, photocopy, recording, Internet postings, or scanning, without prior permission in writing from the publisher.

Publisher: David Pallai
MERCURY LEARNING AND INFORMATION
22841 Quicksilver Drive
Dulles, VA 20166
info@merclearning.com
www.merclearning.com
1-800-232-0223

R. Achary. *Cryptography and Network Security*.

ISBN: 978-1-68392-691-7

The publisher recognizes and respects all marks used by companies, manufacturers, and developers as a means to distinguish their products. All brand names and product names mentioned in this book are trademarks or service marks of their respective companies. Any omission or misuse (of any kind) of service marks or trademarks, etc. is not an attempt to infringe on the property of others.

Library of Congress Control Number: 2021940313

212223321 Printed on acid-free paper in the United States of America.

Our titles are available for adoption, license, or bulk purchase by institutions, corporations, etc. For additional information, please contact the Customer Service Dept. at 800-232-0223(toll free).

All of our titles are available in digital format at academiccourseware.com and other digital vendors. The sole obligation of MERCURY LEARNING AND INFORMATION to the purchaser is to replace the book, based on defective materials or faulty workmanship, but not based on the operation or functionality of the product.

CONTENTS

<i>Preface</i>	xiii	
Part 1:	Cryptography	1
Chapter 1:	Introduction	3
1.1	Introduction	3
1.2	Security Needs	3
1.3	Security Attacks	5
1.4	Types of Attacks	8
1.5	Security Services	12
1.6	Security Mechanisms	13
Summary		15
Review Questions		16
Multiple Choice Questions		16
Chapter 2:	Conventional Encryption and Message Confidentiality	19
2.1	Introduction	19
2.2	Conventional Encryption Principles	19
2.3	Cryptography	21
2.4	Cryptanalysis	22
2.5	Location of Encryption Devices	26
Summary		28
Review Questions		28
Multiple Choice Questions		29

Chapter 3:	Symmetric Key Encryption	31
3.1	Introduction	31
3.2	General Idea of Symmetric Key Encryption	31
3.3	Symmetric Key Encryption	33
3.4	Classifications of Traditional Ciphers	34
3.5	Stream and Block Ciphers	63
	Summary	78
	Review Questions	79
	Multiple Choice Questions	81
Chapter 4:	Modern Block Ciphers	83
4.1	Introduction	83
4.2	General Idea of Modern Block Ciphers	83
4.3	Types of Modern Block Ciphers	85
4.4	Components of Modern Block Ciphers	88
4.5	Circular Shifting	94
4.6	Product Ciphers	96
4.7	Feistel and Non-feistel Ciphers	99
4.8	Attacks on Block Ciphers	103
	Summary	105
	Review Questions	105
	Multiple Choice Questions	106
Chapter 5:	Data Encryption Standard (DES)	109
5.1	Introduction	109
5.2	DES Structure	109
5.3	Round Key Generation: To Generate 16 Sub-Keys $(K_1, K_2 \dots K_{16})$	113
5.4	Encoding Each 64 Bit Block of Plaintext	117
5.5	DES Cryptanalysis	125
5.6	Triple Data Encryption Standard (TDES)	126
5.7	International Data Encryption Algorithm (IDEA)	129
5.8	Blowfish Algorithm	141

Summary	147
Review Questions	148
Multiple Choice Questions	149
Chapter 6: Advanced Encryption Standard (AES)	151
6.1 Introduction	151
6.2 Data Representation	152
6.3 Structure of AES	155
6.4 AES Key Expansion	164
6.5 AES Decryption	176
6.6 AES Encryption and Decryption Example	180
6.7 Advantages of AES Algorithm	184
6.8 Comparison Between AES and DES Algorithms	184
Summary	185
Review Questions	185
Multiple Choice Questions	186
Chapter 7: Asymmetric Key Encryption	189
7.1 Introduction	189
7.2 Public Key Cryptography	190
7.3 RSA Algorithm	192
7.4 Rabin Cryptosystem	200
7.5 Diffie-Hellman Key Exchange	203
7.6 ElGamal Public Key Cryptosystem	206
7.7 Elliptical Curve Cryptosystem (ECC)	209
7.8 ElGamal Encryption using Elliptic Curve Cryptography	219
7.9 Elliptic Cryptography and Diffie-Hellman Key Exchange	220
Summary	223
Review Questions	223
Multiple Choice Questions	224

Part 2:	Security Systems	227
Chapter 8:	Message Integrity and Message Authentication	229
8.1	Introduction	229
8.2	Message Integrity	230
8.3	Hash Function	232
8.4	Random Oracle Model	234
8.5	Message Authentication	238
Summary		246
Review Questions		246
Multiple Choice Questions		247
Chapter 9:	Hash Functions	251
9.1	Introduction	251
9.2	Constructions of Hash Functions	251
9.3	Message Digest	254
9.4	Secure Hash Algorithm (SHA)	261
Summary		273
Review Questions		273
Multiple Choice Questions		275
Chapter 10:	Digital Signature	277
10.1	Introduction	277
10.2	Digital Signature	278
10.3	Digital Envelope	281
10.4	Digital Signature Standard (DSS)	284
Summary		291
Review Questions		292
Multiple Choice Questions		293
Chapter 11:	Entity Authentication	295
11.1	Introduction	295
11.2	Entity Authentication	296
11.3	Password Authentication	296
11.4	Challenge-Response Identification	307
11.5	Biometric Security Systems	316
Summary		324

Review Questions	325
Multiple Choice Questions	326
Chapter 12: Authentication and Key Management Applications	329
12.1 Introduction	329
12.2 Public Key Distribution	331
12.3 Public Key Infrastructure (PKI)	340
12.4 Kerberos	342
12.5 Pluggable Authentication Module (PAM)	347
12.6 Key Distribution Center (KDC)	347
Summary	353
Review Questions	354
Multiple Choice Questions	355
Part 3: Network Security Applications	357
Chapter 13: Network Security and Protocols	359
13.1 Introduction	359
13.2 Internet	359
13.3 Vulnerability and Attacks in Networks	360
13.4 Network Communication Architecture and Protocols	361
Summary	379
Review Questions	380
Multiple Choice Questions	381
Chapter 14: Network Attacks and Security Threats	383
14.1 Introduction	383
14.2 Different Types of Network Attacks and Security Threats	384
Summary	416
Review Questions	417
Multiple Choice Questions	417
Chapter 15: Application Layer Security	421
15.1 Introduction	421
15.2 E-mail Security	422

15.3	E-Mail Security Service	426
15.4	E-Mail Security Mechanisms	431
	Summary	447
	Review Questions	448
	Multiple Choice Questions	449
Chapter 16:	Transport Layer Security	451
16.1	Introduction	451
16.2	Secure Socket Layer (SSL)	452
16.3	Transport Layer Security (TLS)	461
16.4	Hypertext Transfer Protocol Secure (HTTPS) – HTTP over SSL	465
16.5	SSH – Secure Shell	466
	Summary	475
	Review Questions	476
	Multiple Choice Questions	476
Chapter 17:	Network Layer Security	479
17.1	Introduction	479
17.2	Need of Network Layer Security	480
17.3	Internet Protocol Security (IPsec)	481
17.4	IPsec Core Protocol	483
17.5	Operations of Transport and Tunnel Mode	485
17.6	Security Association and Encryption Strength	497
17.7	Key Distribution	499
	Summary	509
	Review Questions	510
	Multiple Choice Questions	511
Chapter 18:	Data Link Layer Security	513
18.1	Introduction	513
18.2	ARP Spoofing	513
18.3	MAC Flooding	515
18.4	Port Stealing	518
18.5	DHCP Attacks	519
18.6	CAM Overflow Attacks	521

18.7	VLAN Hopping	525
18.8	Spanning Tree Protocol (STP) Manipulation Attacks	526
	Summary	529
	Review Questions	530
	Multiple Choice Questions	530
Chapter 19:	Intruders, Viruses, Worms, and Trojan Horses	533
19.1	Introduction	533
19.2	Viruses	533
19.3	Worms	542
19.4	Trojan Horses	546
19.5	Adware	550
19.6	Malicious Software	551
19.7	Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)	554
19.8	Honeypots	564
	Summary	570
	Review Questions	570
	Multiple Choice Questions	572
Chapter 20:	Firewalls and Virtual Private Networks (VPN)	575
20.1	Introduction	575
20.2	Packet Filtering Routers and Bastion Hosts	579
20.3	Proxy Filter (Server)	587
20.4	Network Address Translation (NAT)	589
20.5	Port Address Translation (PAT)	594
20.6	Virtual Private Network (VPN)	597
20.7	Tunneling Protocols	607
	Summary	619
	Review Questions	620
	Multiple Choice Questions	621
<i>Appendix A: Answers for Multiple Choice Questions</i>		623
<i>Appendix B: References</i>		627
<i>Index</i>		631

PREFACE

Cryptography is an indispensable tool used to protect data and information in a computing and data communication system. It is used worldwide on a daily basis. In many cases cryptography is an integral part of standard protocols, most commonly the HTTPS, Transport Layer Security (TLS), and IPsec protocols, making it relatively easy to incorporate strong encryption into a wide range of applications.

Cryptography is also highly vulnerable due to program errors. When developing a cryptosystem, a simple testing of a program will usually uncover a security vulnerability. This book takes the readers through all of the important design and implementation details of various cryptographic algorithms and network security protocols to ensure network security. Throughout the book, the cryptography and network security concepts are explained with diagrams and examples which apply the concepts covered in each chapter.

INTENDED AUDIENCE

The book is intended for both academic and professional audiences. It is an introduction to the field of cryptography and network security. Readers are required to have a basic understanding of number systems and computer network and Internet protocols.

The book attempts to cover aspects of cryptography, information security, and network security from a computer science point of view.

PLAN OF THE BOOK

The book is divided into four parts:

Part I – Cryptography: Presents the conceptual framework of cryptography with a detailed description of conventional encryption and message confidentiality, as well as symmetric and asymmetric key encryption. There are seven chapters in this section:

Chapters 1 and 2: These two chapters define security needs, different types of attacks, and conventional cryptographic principles.

Chapter 3: It describes traditional symmetric key cryptosystems and different terms, concepts, and algorithms used in symmetric key cryptography.

Chapter 4: Describes modern block ciphers, different types of block ciphers, and an introduction to stream ciphers.

Chapters 5 and 6: Describe modern symmetric key block cipher algorithms such as the Data Encryption Standard (DES), triple DES (TDES), International Data Encryption Algorithm (IDEA), Blowfish, and Advanced Encryption Standard (AES) algorithms, as well as their basic structures and implementation for data encryption and decryption.

Chapter 7: Discusses asymmetric key cryptosystems, including the popular RSA, Rabin, ElGamal and Elliptic curve cryptosystems.

Part II – Security Systems: Contains five chapters related to information security, which deals with message integrity and authentication, digital signature, and key management.

Chapter 8: Describes hash functions and several ideas related to message integrity, message authentication, generation of message digest, and guaranteeing the integrity of the message.

Chapter 9: Describes the construction of hash functions and message digest using MD5 and other types of secure hash algorithms (SHA).

Chapter 10: Describes digital signature and contains several digital signature schemes (DSS), digital signature algorithms (DSA), and elliptic curve DSA (ECDSA).

Chapter 11: Describes the different entity authentication techniques, including password authentication, challenge-response authentication, and biometrics security systems.

Chapter 12: Describes the different key management approaches, including public key distribution, public key infrastructure (PKI), and Kerberos.

Part III – Network Security Applications: This section deals with network security based on the TCP/IP protocol layer structure, including application layer security, transport layer security, network layer security, and data link layer security.

Chapters 13 and 14: Explain different types of network attacks and security threats, as well as network security mechanisms and protocols.

Chapter 15: Describes the general structure of e-mail application programs, as well as application layer security services using PGP and s-MIME.

Chapter 16: Describes security services at the transport layer, including transport layer security protocols such as Transport Layer Security (TLS) and Secure Socket Layer (SSL).

Chapter 17: This chapter explains network layer security, including IPSec architecture, IPSec security services, and Security Association (SA).

Chapter 18: Describes security methods for data link layer security challenges including ARP spoofing, MAC flooding, port stealing, and DHCP Attacks.

Part IV – System Security: Presents systems security, which includes intruders, intrusion detection systems, and firewalls.

Chapter 19: Describes intruders and viruses. This includes explaining intrusions and intrusion detection systems, viruses, worms, malware, adware, Trojans, DoS, DDoS, and Network Security Models (NSM).

Chapter 20: This chapter explains firewall structure, firewall design processes, and virtual private networks.

A detailed summary appears at the end of every chapter. In addition, the book includes conceptual and objective questions, an extensive glossary, and acronyms.

R. Achary
June 2021

PART

1

CRYPTOGRAPHY

INTRODUCTION

1.1 INTRODUCTION

This book mainly covers information security in wired networks. The chapter gives an outline of concepts of information security, including sound approaches to the structure of appropriate security controls. Once this background is established, it is easy to follow rest of the chapters. It is difficult to cover all the fundamentals in detail in this chapter. For a clear understanding of these concepts, they are explained with diagrams and examples.

1.2 SECURITY NEEDS

Computer security is increasingly important these days, as the number of security breaches are increasing. Many corporations are worried about protecting their data from disclosure, modification, or destruction from unauthorized individuals and disgruntled employees. In case of data loss the cost of recovering the data is steadily rising along with the rise in the number of incidents themselves. Computer security rests on *confidentiality*, *integrity*, and *availability* (see Figure 1.1).

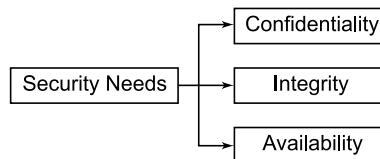


FIGURE 1.1 Security needs.

1.2.1 Confidentiality

Confidentiality means the need for keeping information securely against malicious actions. This is essential for the user of computers in sensitive fields such as government organizations and industrial establishments. In industry the safeguarding of sensitive information or resources from competitors is crucial to the operation of the organization. The concept of the “need to know” principle adopted as part of the proprietary design of database management systems secures the data and information from malicious attacks. Access control mechanisms support confidentiality. Cryptography is one such mechanism used for preserving confidentiality.

1.2.2 Integrity

Integrity refers to the trustworthiness of data or information. Data needs to undergo changes constantly. Integrity usually means preventing improper or unauthorized changes to data. For example, in a bank when a customer makes a transaction, the balance of his account needs to be changed. These changes need to be done only by authorized users and through authorized mechanisms. Integrity includes data integrity and originality integrity.

Data Integrity: It is related to the content of information. The integrity of information is assured when it is timely, accurate, complete, and consistent. However, computers are unable to provide or protect all of these qualities. Therefore, in the computer security field, integrity is often discussed more narrowly as having two faces: *data integrity* and *system integrity*.

- Data integrity is a requirement that information and programs are changed only in a specified and authorized manner.
- System integrity is a requirement that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Originality Integrity: It is related to the source of data.

Integrity mechanisms fall into two categories:

1. **Prevention mechanism:** A prevention mechanism seeks to maintain the integrity of the data by blocking any unauthorized attempts to modify the information or any attempts to modify the information in unauthorized ways. This can be provided by adopting adequate authentication and access control mechanisms.
2. **Detection mechanism:** A detection mechanism simply reports the violation of integrity. It analyzes the system events to detect problems.

1.2.3 Availability

Availability refers to the ability to use the information as and when desired. The information created and stored by an organization needs to be available to the authorized users, because an unavailable system is at least *as bad as no system at all*. The attempt to block the availability of information is called a *Denial of Service* (DoS) attack. It is very difficult to detect DoS.

1.3 SECURITY ATTACKS

A threat is a potential violation of security. The actions toward the violation are called *attacks*. The three security mechanisms, *confidentiality*, *integrity*, and *availability*, counter the threats to the security of a system. Different approaches to categorizing attacks are depicted in Figure 1.2.

Security Attacks	Security Threats
Snooping	Threat to Confidentiality
Traffic Analysis	
Modification	
Masquerading	Threat to Integrity
Replaying	
Repudiation	
Denial of Services (DoS)	Threat to Availability

FIGURE 1.2 Security attacks in relation to security threats.

1.3.1 Threat to Confidentiality

The two types of attacks which threaten confidentiality are:

1. **Snooping:** Snooping is the unauthorized interception of information. Some unauthorized entity is listening to communications or browsing through files or information. For example, an unauthorized user intercepts a file transferred through the Internet. It may contain confidential information. Information can be protected from snooping by encrypting it.
2. **Traffic Analysis:** An interceptor may not be able to tamper with the encrypted data. They can obtain some other type of information by

monitoring online traffic. This may help them to guess the nature of transactions.

1.3.2 Threat to Integrity

The general types of attacks that threaten integrity are *modification*, *masquerading*, *replaying*, and *repudiation*.

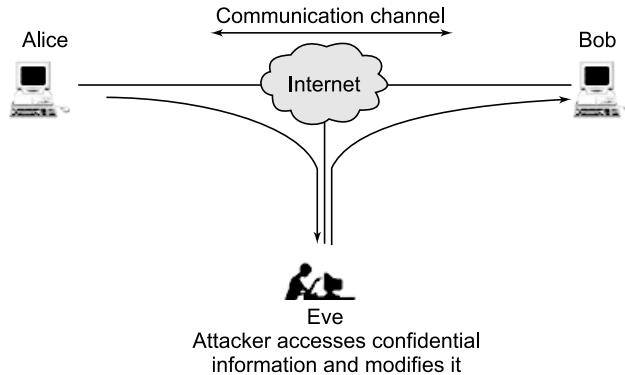


FIGURE 1.3 Modification of message.

Modification: It is the unauthorized change of information. The attacker modifies the information to make it beneficial to himself. In Figure 1.3, the attacker Eve gathers confidential information from Alice and modifies it before forwarding it to Bob.

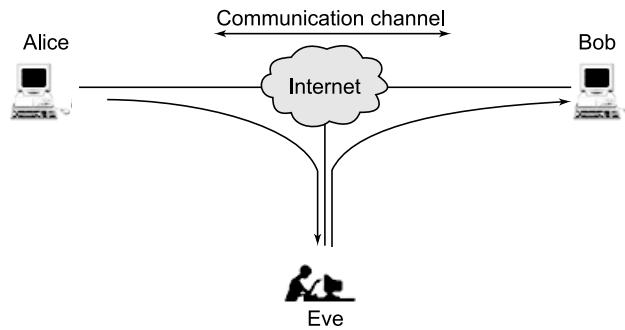


FIGURE 1.4 Masquerade.

Masquerading: Masquerading is an impersonation of one entity by another. For example, an attacker might steal the data of the customers. If a user tries to receive an actual file, the attacker arranges a different file to be given to the user in place of original file or message.

In Figure 1.4, Eve pretends to be Alice and engages into a message exchange with Bob. Eve thereby attempts to illegitimately access a system or perform a malicious action.

Replaying: The attacker obtains a copy of a message sent by a user and later tries to replay it. For example, Alice sends a letter to her bank to ask for payment to Bob who has done a job for her. The attacker intercepts the message and sends it again to receive payment from the bank.

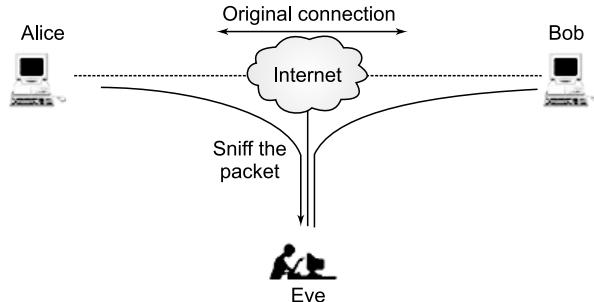


FIGURE 1.5 Replaying attack.

In Figure 1.5, Eve captures a previously observed valid message from Alice and retransmits this message to the original recipient Bob. This could be used as a part of masquerade attack.

Repudiation: This type of attack is performed by one of the two parties in the communication. The message sender later denies that they have sent the message. Also sometimes the receiver might deny that they have received the message.

1.3.3 Threat to Availability

The common threat to availability is *denial of service* (DoS).

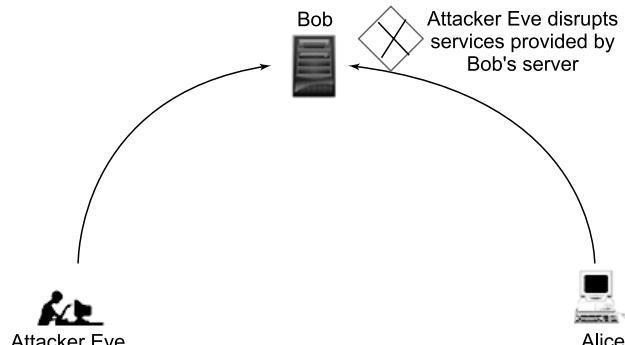


FIGURE 1.6 Denial of service attack.

Denial of Service: It is a very common type of attack. The attacker prevents a server from providing services. The denial may occur at the source by preventing the server from obtaining the resources needed to perform its function. At the destination it is done by blocking the communication from the server as in Figure 1.6. It may be done along the intermediate path by discarding messages from either the client or the server or both. The DoS results in an infinite delay.

1.4 TYPES OF ATTACKS

Classes of attacks might include passive monitoring of communications, active network attacks, close-in attacks, exploitation by insiders, and attacks through the service provider. The different types of attacks are detailed as follows:

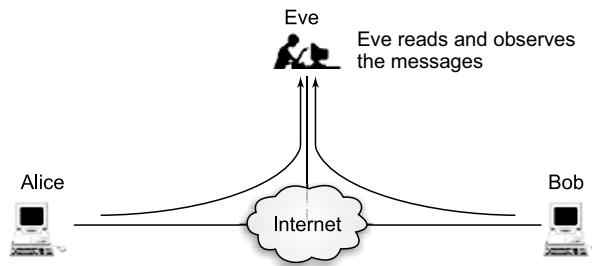


FIGURE 1.7 Passive attack.

Passive Attack: A passive attack monitors unencrypted traffic and looks for sensitive information that can be used in other types of attacks. Passive attacks include traffic analysis, monitoring of unprotected communications, decrypting weakly encrypted traffic, and capturing authentication information such as passwords. Passive interception of network operations enables adversaries to see upcoming actions. Passive attacks result in the disclosure of information of data files to an attacker without the consent or knowledge of the user. In Figure 1.7, Eve attempts to access a sensitive information exchange between Alice and Bob. Additionally Eve can also perform traffic analysis in order to retrieve sensitive information from Alice and Bob.

Active Attack: In an active attack, the attacker tries to bypass or break into secured systems. This can be done through stealth, viruses, worms, or Trojan horses. Active attacks include attempts to circumvent or break

protection features, introduce malicious code, and steal or modify information. Active attacks result in the disclosure or dissemination of data files, DoS, or modification of data. In an active attack, Eve modifies the message exchanged between Alice and Bob. Instead Eve can also create new messages and send them to Alice or Bob. Active attacks can be categorized as masquerades, replay attacks, and DoS attacks as in Figure 1.8.

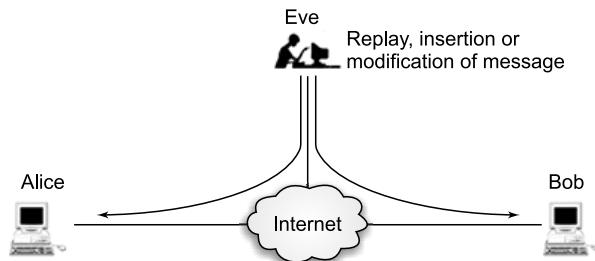


FIGURE 1.8 Active attack.

Distributed Attack: In a distributed attack, the attacker alters the communication by adding malicious code like a Trojan horse or backdoor program. This code acts as useful software that will be distributed later to the networks and computers of many other companies and users. Distributed attacks focus on the malicious modification of software at the factory or during distribution of information.

Insider Attack: An insider attack involves someone in the organization, such as a disgruntled employee, attacking the network. Insider attacks can be malicious or non-malicious. Malicious insiders intentionally eavesdrop, steal, or damage information; use information in a fraudulent manner; or deny access to other authorized users. Non-malicious attacks typically result from carelessness, lack of knowledge, and intentional circumvention of security procedures in performing a task.

Close-in Attack: A close-in attack involves someone attempting to get physically close to network components, data, and systems. Their purpose is to learn more about a network. Close-in attacks consist of regular individuals attaining close physical proximity to networks, systems, or facilities for the purpose of modifying, gathering, or denying access to information. Close physical proximity is achieved through secret entry into the network, open access, or both.

One popular form of close-in attack is social engineering. In a social engineering attack, the attacker compromises the network or system through

social interaction with a person, through an e-mail message or phone. Various tricks can be played by the individual to learn information about the security of company. The information that the victim unwillingly reveals to the hacker would most likely be used in a subsequent attack to gain unauthorized access to a system or network.

Phishing Attack: A phishing attack aims to acquire sensitive information about the user, such as user name, password, and full address from various websites of which the user is a member. In a phishing attack the hacker creates a fake Website that looks exactly like a popular site such as the State Bank of India (SBI) or PayPal. The phishing part of the attack is that the hacker then sends an e-mail message trying to trick the user into clicking a link that leads to the fake site. When the user attempts to log on with their account information, the hacker records the username and password and then tries that information on the real site.

The different steps involved in phishing attack are explained in Figure 1.9. It is quite easy for the attacker to send an e-mail posing as a legitimate online entity. It can be difficult to spot whether an e-mail is from a phishing attack or a legitimate e-mail. E-mails are meant to phish user information so they can use it to gain access to their online accounts and usually ask for sensitive information like their password and user name.

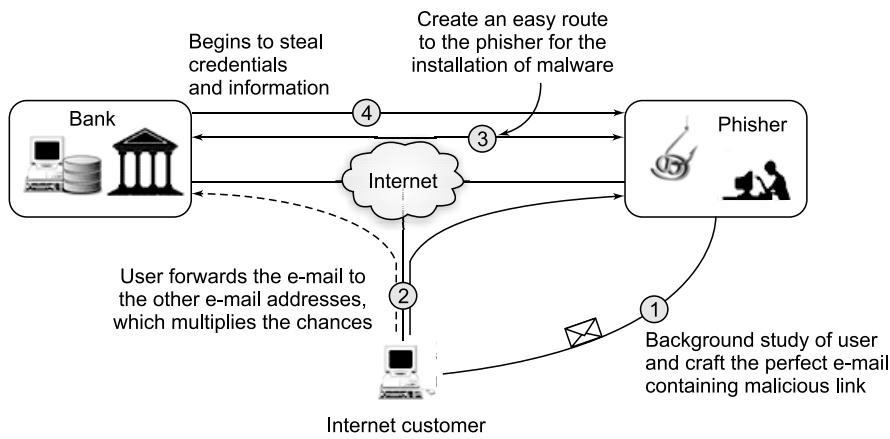


FIGURE 1.9 Phishing attack.

Hijack Attack: In a hijack attack, a hacker takes over a session between a user and another individual and disconnects the other individual from the communication. The user in the other end of the network believes that he is still talking to the original party and may send private information to the hacker accidentally.

Spoof Attack: In a spoof attack, the hacker modifies the source address of the packets he or she is sending so that they appear to be coming from someone else. This may be an attempt to bypass your firewall rules.

Buffer Overflow: A buffer overflow attack is when the attacker sends more data to an application than is expected. In this attack, the attacker gains administrative access to the system in a command prompt or shell.

Exploit Attack: In an exploit attack, the attacker recognizes the security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability.

Password Attack: In a password attack an attacker tries to crack the passwords stored in a network account, database, or password-protected file. There are three major types of password attacks: a *dictionary attack*, a *brute-force attack*, and a *hybrid attack*. A dictionary attack uses a word list file, which is a list of potential passwords. A brute-force attack is when the attacker tries every possible combination of characters. Hybrid attacks combine these two types of attack methods. Currently, even with increased complexity of targets, hybrid attacks have become increasingly popular.

1.4.1 Comparison between Passive and Active Attacks

Attacks are categorized as passive and active attacks.

TABLE 1.1 Comparison between Passive and Active Attacks.

Passive/Active	Intention	Attacks	Threat
Passive attacks	<ul style="list-style-type: none"> • The attackers goal is just to obtain information • Not to modify data • It is difficult to detect • Information can be protected by encryption 	<ul style="list-style-type: none"> • Snooping • Traffic analysis 	Confidentiality
Active attacks	<ul style="list-style-type: none"> • It changes the data or harms the system • Easier to detect than to prevent 	<ul style="list-style-type: none"> • Modification • Masquerading • Replaying • Repudiation 	Integrity
		<ul style="list-style-type: none"> • Denial of Service 	Availability

1.5 SECURITY SERVICES

The different security services and mechanisms implemented are briefly discussed in this section. The five common security services are as shown in Figure 1.10.

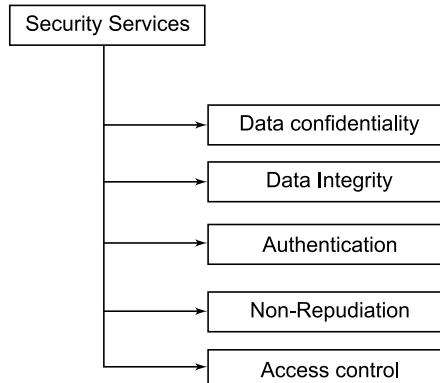


FIGURE 1.10 Security services.

Data Confidentiality: Data confidentiality is a process of the protection of transmitted information from passive attacks. These passive attacks include snooping and traffic analysis. The main aspect of confidentiality is several levels of protection of the information under transmission.

Data integrity: It is designed to protect the data from modification, insertion, deletion, and replaying by malicious user. Data integrity can be applied to a stream of messages, a single message, or selected fields within the message. In a connection-oriented integrity service, a stream of messages assures that messages are received as sent. There is no duplication, insertion, modification, reordering, or replays. This service also covers the destruction of data. Thus the connection-oriented integrity service addresses both message stream modification and DoS, whereas the connectionless integrity service generally provides protection against message modification only.

Authentication: Authentication service assures that the communication is authentic. In connection-oriented communication, it provides authentication of the sender or receiver (peer entity authentication). This authentication process occurs during connection establishment. In connectionless communication it authenticates the source of the data (it is considered as data origin authentication).

Non-repudiation: Non-repudiation is the term used to describe the inability of a person to deny or repudiate the origin of a message or receipt of

a message. Non-repudiation is most commonly used in the verification and trust of signature. For example: if a user has manually or digitally signed a document, it is difficult for a user to claim they didn't sign it. Non-repudiation is provided by digital signature, usually digital certificate, often based on the X.509 v3 standard, and a public key infrastructure (PKI).

Access Control: The objective of access control is to limit and control the access to data. Access control can be achieved when the individuals trying to gain access must be first identified and/or authenticated. Only authorized users are allowed to access the resources.

1.6 SECURITY MECHANISMS

The different security mechanisms used to provide security services are as given in Figure 1.11.

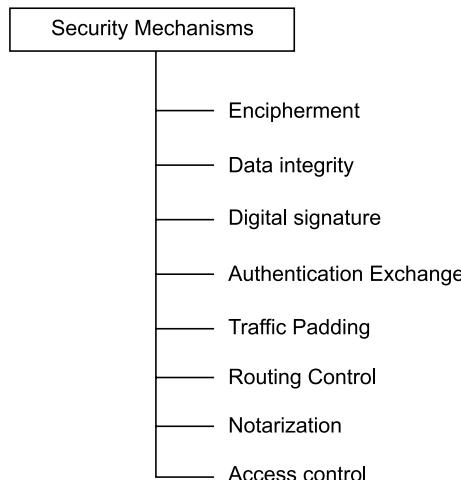


FIGURE 1.11 Security mechanisms.

Encipherment: Encipherment means conversion of data into an unintelligible form by means of a reversible translation. The translation can be done based on an algorithm. It provides confidentiality. Cryptographic techniques are used for enciphering.

Data Integrity: Data integrity refers to the accuracy and the validity of the data transmitted over an unsecure channel. Data integrity can be compromised in a number of ways. Each time data is replicated or transferred, it should remain intact and unaltered between updates. Error checking

methods and validation procedures are typically adopted to ensure the integrity of data that is transferred or reproduced without the intention of alteration.

Data integrity may be compromised through:

- Human error, whether malicious or unintentional.
- Transfer error including unintended alteration or data compromising during transfer from one device to another.
- Bugs, viruses/malware, and other cyber threats.
- Compromised hardware, such as a device or disk crash.

Only some of these compromises may be adequately prevented through data security mechanisms. Other data integrity best practices include input validation to preclude the entering of invalid data, error detection/data validation to identify errors in data transmission, and data security measures such as data loss prevention access control and encryption.

Digital Signature: A digital signature is a concept that authenticates both the origin and contents of a message in a manner that is provable to an unbiased third party.

Authentication Exchange: This is the message exchange by two entities to prove their identities to each other.

Traffic Padding: The process of inserting some bogus characters into the data traffic to prevent the adversary's attempt to use the traffic analysis.

Routing control: It is a process of continuously changing different available routes between the sender and the receiver. This process helps to prevent malicious users from eavesdropping on the communication between two entities.

Notarization: The process of selecting a trusted third party for authentication and control of the communication between two entities.

Access control: Access control is a set of controls to restrict access to certain resources. The different access control mechanisms are:

1. **Discretionary Access Control (DAC):** This access control model is based on user discretion. Here the owner of the data resource can give access rights on the resource to other users based on his discretion. Access control lists (ACL) are a typical example of DAC.
2. **Mandatory Access Control (MAC):** In the MAC model the users/owners do not enjoy the privilege of deciding who can access their files. Here the operating system is the decision maker. In this model, all subjects (users) and objects (resources) are classified and assigned with a security label. The security labels of the subject and object along with

the security policy determine if the subject can access the object. The rules for how the subject accesses objects can be made by the security officers and enforced by the operating systems and supported by security technologies.

3. **Role Based Access Control (RBAC):** In this model the access to a resource is governed based on the roles that the subject holds within an organization. RBAC is also known as *nondiscretionary access control*, because the user inherits privileges that are tied to his role. The user does not have control over the role that he will be assigned.

1.6.1 Comparison between Security Services and Security Mechanisms

The comparison between security services and security mechanisms is shown in Table 1.2.

TABLE 1.2 Comparison between Security Services and Security Mechanisms.

Security service	Security mechanism
Data confidentiality	Encryption and routing control
Data integrity	Encryption, digital signature, and integrity
Authentication	Encryption, digital signature, and authentication exchange
Non-repudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

SUMMARY

- Information security has gained enormous importance in the last few years. The internet has become one of the main channels for conducting business. An ever-increasing number of people are becoming more comfortable with making monetary transactions online.
- There are a number of different means for hackers to perpetuate their agenda, including spam, malware, and phishing.
- The principles of any security mechanism are confidentiality, authentication, integrity, non-repudiation, access control, and availability. These are designed to detect, prevent, and recover from a malicious attacks.
- A security attack is an action toward the violation of security. Different types of attacks are passive attacks, active attacks, distributed attacks, insider attacks, etc.

- Different services and mechanisms implemented are data confidentiality, data integrity, authentication, non-repudiation, and access control.

REVIEW QUESTIONS

1. Explain the goals of security.
2. What are the significant features of network security?
3. Define information security. Discuss the three aspects of security that are to be considered in the design of a secure information system.
4. Define security attacks. Distinguish between passive and active attacks. Give examples.
5. Explain the threat to integrity.
6. Describe security services.
7. Explain the categories of security mechanisms.
8. Briefly explain the denial of service attack.
9. What is the need for information security?
10. Distinguish between direct attacks and indirect attacks.
11. Differentiate between attack and threat.
12. How does a threat to information security differ from an attack? Explain the groups of threats to information security.

MULTIPLE CHOICE QUESTIONS

1. A _____ is anything that can cause harm.
(a) Vulnerability (b) Phish (c) Threat (d) Spoof
2. In the right setting, a thief will steal your information by simply watching what you type.
(a) Snagging (b) Spying
(c) Social engineering (d) Shoulder surfing

3. A _____ is a small program embedded inside of a GIF image.
(a) Web bug (b) Cookie (c) Spyware application (d) Spam
4. A hacker contacts you by phone or e-mail and attempts to acquire your password.
(a) Spoofing (b) Phishing (c) Spamming (d) Bugging
5. This power protection device includes a battery that provides a few minutes of power.
(a) Surge suppressor (b) Line conditioner
(c) Generator (d) UPS
6. The phrase _____ describes viruses, worms, Trojan horse attack applets, and attack scripts.
(a) Malware (b) Spam (c) Phish (d) Virus
7. A hacker that changes or forges information in an electronic resource is engaging in _____.
(a) Denial of service (b) Sniffing (c) Terrorism (d) Data diddling
8. Hackers often gain entry to a network by pretending to be at a legitimate computer.
(a) Spoofing (b) Forging (c) IP spoofing (d) ID theft
9. The _____ of a threat measures its potential impact on a system.
(a) Vulnerabilities (b) Countermeasures (c) Degree of harm
(d) Susceptibility
10. The power level drops below 120V.
(a) Brownout (b) Spike (c) Blackout (d) Surge
11. In computer security, _____ means that computer system assets can be modified by only authorized parties.
(a) Confidentiality (b) Integrity (c) Availability (d) Authenticity
12. In computer security, _____ means that the information in a computer system is only accessible for reading by authorized parties.
(a) Confidentiality (b) Integrity (c) Availability (d) Authenticity

- 13.** The types of threats on the security of a computer system or network are:
- (i) Interruption (ii) Interception (iii) Modification
(iv) Creation (v) Fabrication
(a) (i), (ii), (iii), and (iv) only (b) (ii), (iii), (iv), and (v) only
(c) (i), (ii), (iii), and (v) only (d) All (i), (ii), (iii), (iv), and (v)
- 14.** This is the hiding of a secret message within an ordinary message and the extraction of it at its destination.
- (a) Secret key algorithm (b) Message Queuing
(c) Spyware (d) Steganography
- 15.** Considering commerce and marketing, which of the following present the most significant obstacle to developing IT security?
- (a) There is no direct return on investment in building security systems.
(b) Security systems are detrimental to usability and can make IT systems less functional, and therefore less attractive to the consumer.
(c) There is pressure to reduce the time it takes to get a new IT product or system onto the market, so security systems are sacrificed in order to reduce the time to market.
(d) All of the above.
- 16.** Threats to IT systems can be classified in many ways; in this chapter three different threat categories are listed. According to this classification, which of the following would be classified as a “failure”?
- (a) Security systems were not adequate to protect the system against attack from a hacking group, and sensitive data was lost.
(b) There is a programming error in the software which causes the system to perform badly.
(c) The IT system has failed due to a random unexpected event, such as a tsunami which destroys key electronic equipment.
(d) All of the above.

CONVENTIONAL ENCRYPTION AND MESSAGE CONFIDENTIALITY

2.1 INTRODUCTION

Due to the rapid growth of digital communication and electronic data transfer, information security has become a crucial issue in industry, business, and administration. Modern cryptography provides essential techniques for securing information and protecting data.

The word cryptography comes from the Greek words *κρυπτό* (hidden or secret) and *γραφή* (writing). Oddly enough, cryptography is the art of secret writing. More generally, people think of cryptography as the art of mangling information into apparent unintelligibility in a manner allowing a secret method of un-mangling. The basic service provided by cryptography is the ability to send information between participants in a way that prevents others from reading it. This chapter covers the key concept of cryptography and different cryptanalysis techniques.

2.2 CONVENTIONAL ENCRYPTION PRINCIPLES

Cryptography is the art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible and then retransforming that message back to its original form. This kind of cryptography can provide other services, such as:

- **Integrity Checking:** Reassuring the recipient of a message that the message has not been altered since it was generated by a legitimate source.
- **Authentication:** Verifying someone's (or something's) identity.

But back to the traditional use of cryptography, a message in its original form is known as *plaintext*. The distorted information is known as *ciphertext*. The process for producing ciphertext from plaintext is known as *encryption*. The reverse of encryption is called *decryption*.

Encryption is a mechanism used to protect valuable information such as online transactions, documents, and images from unauthorized people accessing or changing it. This can be performed by using a mathematical formula called a *cipher* and a *key* to convert readable data (plaintext) into a form that others cannot understand (ciphertext). Decryption is the translation of encrypted text or data into original text. It is also called *deciphering*.

A conventional encryption scheme has five elements.

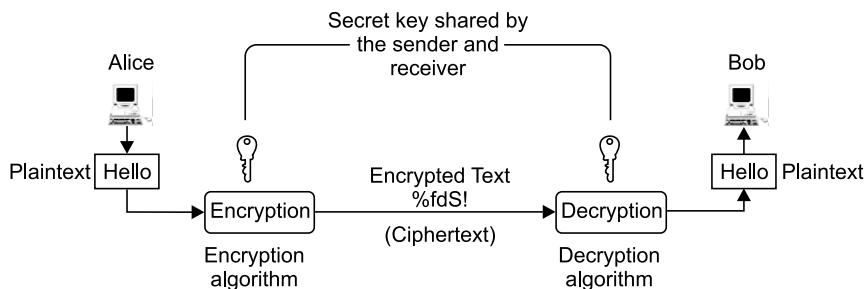


FIGURE 2.1 Conventional encryption.

Plaintext: This is the original message before encryption and after decryption.

Encryption Algorithm: It performs various substitutions and transformations on the plaintext during encryption.

Secret Key: The secret key is an input to the algorithm along with the plaintext. The exact substitution and transformation performed by the algorithm depends on the key selected.

Ciphertext: The message produced after encryption as an output is ciphertext. It is a scrambled message. For different secret keys the ciphertext is different.

Decryption Algorithm: It is a reverse process of the encryption operation. It takes ciphertext and the secret key as an input and produces the original plaintext.

In an encryption process, the security level mainly depends on the secrecy of the key used and not the secrecy of the algorithm. Generally we need not to keep the algorithm secret; we need to keep only the key secret.

2.3 CRYPTOGRAPHY

In cryptography, the transformed message will be secure and resistant to attacks. Cryptography is the process of encryption and decryption of a message using secret keys. Cryptography is classified along three independent dimensions. The three different methods of classifications are based on the following:

1. The types of operations used for transforming plain text to ciphertext.

Substitution: In which each element in the plaintext is mapped into another element.

Transposition: In which elements in the plaintext are rearranged.

In the previous two operations, the fundamental requirement is that no information be lost.

2. The number of keys used.

If both sender and receiver use the same key, the system is referred to as symmetric key, encryption, single key, secret key, or conventional encryption. If the sender and receiver each use a different key, the system is referred to as asymmetric, two key, or public key encryption.

Symmetric Key Encryption: In this type of crypto system a single key is used for both the encryption and decryption process.

Asymmetric Key Encryption: In this type of crypto system there are two keys instead of one, one *public key* and one *private key*. The sender encrypts the message using the receiver's public key. To decrypt the message the receiver uses his private key.

3. The way in which the plaintext is processed.

Block Cipher: A block cipher processes the input one block of elements at a time, producing an output block for each input block.

Stream Cipher: A stream cipher processes the input elements continuously, producing output one element at a time as it goes along.

2.4 CRYPTANALYSIS

The process of breaking a crypto system to discover the plaintext or key is called *cryptanalysis*. A study of cryptanalysis will help us to know how vulnerable our cryptosystem is. Using this, better secret codes can be created to protect our transactions. The five different types of cryptanalysis attacks are listed in Figure 2.2.

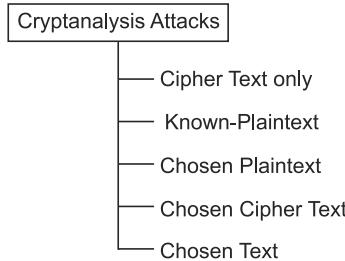


FIGURE 2.2 Cryptanalysis attacks.

2.4.1 Ciphertext Only Attack (COA)

In a ciphertext only attack, the attacker Eve accesses only the ciphertext. This ciphertext is used to find the corresponding plaintext and the key. It is assumed that the attacker knows the cryptographic algorithm. She can intercept the ciphertext for cryptanalysis. The attacker will try to find the key or decrypt one or more pieces of ciphertext.

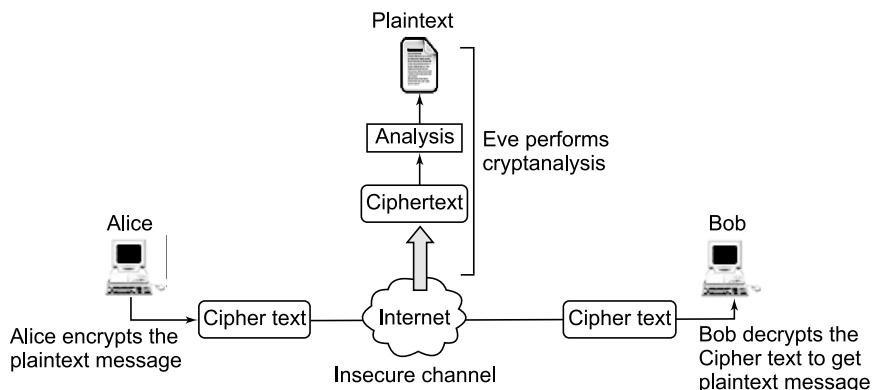


FIGURE 2.3 Ciphertext only attack.

Different methods used in ciphertext only attacks are:

1. Brute-force attack
2. Statistical attack
3. Pattern attack

TABLE 2.1 Frequency of Characters and Character Groups in the English Language.

Monogram Frequency				Bigram Frequency				Trigram Frequency			
A :	8.55	K :	0.81	TH :	2.71	EN :	1.13	THE:	1.81	ERE :	0.31
U :	2.68	B :	1.60	NG :	0.89	HE :	2.33	HES:	0.24	AND :	0.73
L :	4.21	V :	1.06	AT :	1.12	AL :	0.88	TIO:	0.31	VER :	0.24
C :	3.16	M :	2.53	IN :	2.03	ED :	1.08	ING:	0.72	TER :	0.30
W :	1.83	D :	3.87	IT :	0.88	ER :	1.78	HIS:	0.24	ENT:	0.42
N :	7.17	X :	0.19	ND :	1.07	AS :	0.87	EST:	0.28	OFT :	0.22
E :	12.10	O :	7.47	AN :	1.61	TO :	1.07	ION:	0.42	ERS :	0.28
Y :	1.72	F :	2.18	IS :	0.86	RE :	1.41	ITH:	0.21	HER :	0.36
P :	2.07	Z :	0.11	OR :	1.06	HA :	0.83	ATI:	0.26	FTH :	0.21
G :	2.09	Q :	0.10	ES :	1.32	EA :	1.00	FOR:	0.34	HAT :	0.26
H :	4.96	R :	6.33	ET :	0.76	ON :	1.32	STH:	0.21	THA :	0.33
I :	7.33	S :	6.73	TI :	0.99	SE :	0.73	ATE:	0.25	OTH :	0.21
J :	0.22	T :	8.94	ST :	1.25	AR :	0.98	NTH:	0.33	ALL :	0.25
				OU :	0.72	NT :	1.17	RES:	0.21	INT :	0.32
				TE :	0.98	OF :	0.71	ETH:	0.24	ONT :	0.20
Quadgram Frequency				Quintgram Frequency				Common English Words			
TIO:	0.31	OTH:	0.16	OFTH:	0.18	ANDT:	0.07	THE:	6.42	ON :	0.78
THE:	0.12	NTH:	0.27	CTIO:	0.05	ATIO:	0.17	ARE:	0.47	OF :	2.76
THE:	0.16	RTH:	0.12	NDTH:	0.07	WHIC:	0.05	WIT:	0.75	THI:	0.42
THE:	0.24	DTH:	0.15	INTH:	0.16	ONTH:	0.07	AND:	2.75	HE :	0.75
THE:	0.11	THA:	0.21	THES:	0.05	THER:	0.09	I :	0.41	TO :	2.67
ING:	0.15	FRO:	0.10	EDTH:	0.06	AFTE:	0.05	IT :	0.74	BUT:	0.40
OFTH:	0.19	ETHE:	0.15	INGTH:	0.09	THEIR:	0.06	A :	2.43	AS :	0.71
THIS:	0.10	FTHE:	0.19	EOFTH:	0.05	TOTHE:	0.08	HAVE:	0.39	IN :	2.31
SAND:	0.14	TING:	0.10	TIONA:	0.06	ABOUT:	0.04	AT :	0.58	AN :	0.37
THES:	0.18	STHE:	0.14	NGTHE:	0.08	ORTHE:	0.06	IS :	1.12	HIS:	0.55
THEI:	0.10	WITH:	0.18	ERTHE:	0.04	OTHER:	0.07	HAS :	0.35	FOR:	1.01
HERE:	0.13	NGTH:	0.10	FORTH:	0.06	IONAL:	0.04	BY :	0.51	NOT:	0.34
INTH:	0.17	THEC:	0.13	ATTHE:	0.07	INGTO:	0.06	THAT:	0.92	BE :	0.48
IONS:	0.10	ATIO:	0.17	FIRST:	0.04	TIONS:	0.07	THEY:	0.33	WAS:	0.88
MENT:	0.12	ANDT:	0.10	THECO:	0.05	WOULD:	0.04	FROM:	0.47	OR :	0.30

Brute-force Attack: In a brute-force attack the attacker tries to decrypt the message using all possible keys, until the actual plaintext is obtained.

Statistical Attack: In this the attacker tries to find the most frequently used characters, pair of characters, or triplet of characters in the ciphertext. A statistical cryptanalysis exploits weakness in cryptosystems, such as the inability to produce random numbers or floating point error. It computes the frequency of such characters in ciphertext. Cryptanalysis is based on the fact that in any given stretch of ciphertext, certain letters and combinations of letters occur with varying frequencies; moreover, there is a character distribution of letters that is roughly the same for all samples of ciphertext.

For Example: For a given section of ciphertext, E, T, A, and O are the most common, while Z, Q, and X are rare. Likewise TH, ER, ON, and AN are the most common pairs of letters, and SS, EE, TT, and FF are the most common repeats. The frequency of different characters and character groups in the English language is given in Table 2.1. It is clear that the most frequently used character is “e.”

Pattern Attack: In this case, it is known that all clear messages contain some pattern. This may be English language words, or encodings of some non-compact alphabets such as ASCII letters or XML start and end tags, or some combination of all of these.

2.4.2 Known Plaintext Attacks (KPA)

In a known plaintext attack an attacker has seen the plaintext and the corresponding ciphertext. The attacker can make conclusions about the encrypting key and will have validation if the encrypting key is discovered. In Figure 2.4, Alice (A) wants to send an encrypted file to Bob (B); later Alice makes the content of this file public. The attacker will keep this plaintext as a sample and use the intercepted ciphertext for mapping. Once the attacker becomes successful, he uses it to break the next message from Alice to Bob, assuming that Alice has not changed her key.

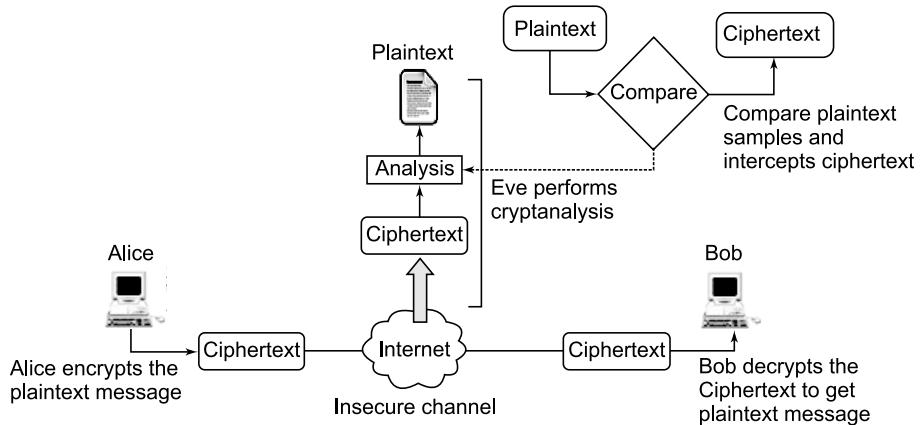


FIGURE 2.4 Known plaintext attacks.

2.4.3 Chosen Plaintext Attack (CPA)

In a chosen plaintext attack, the attacker chooses the plaintext to be encrypted. This can occur when the user steps away from the computer and the attacker sends a message and captures the resulting ciphertext. The attacker can select plaintext that will produce clues to the encryption key used.

This is possible when the attacker has access to Alice's computer. He can choose some plaintext and intercept the created ciphertext as shown in Figure 2.5. The attacker does not have the key because the key is normally embedded in the software used by Alice.

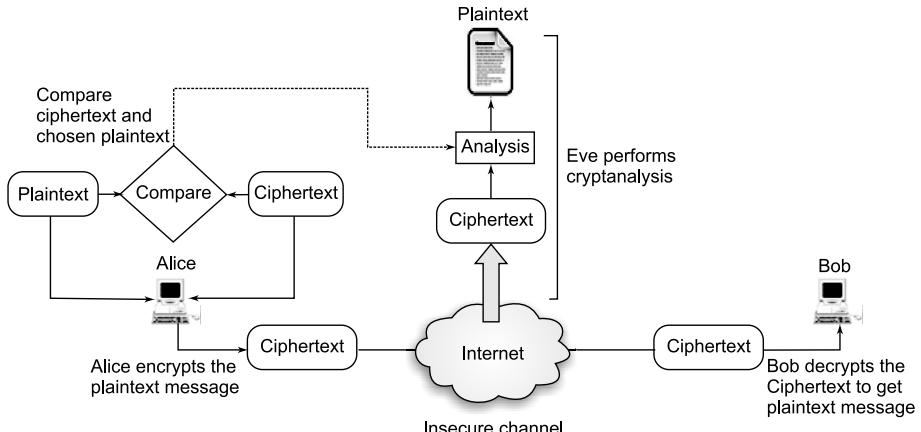
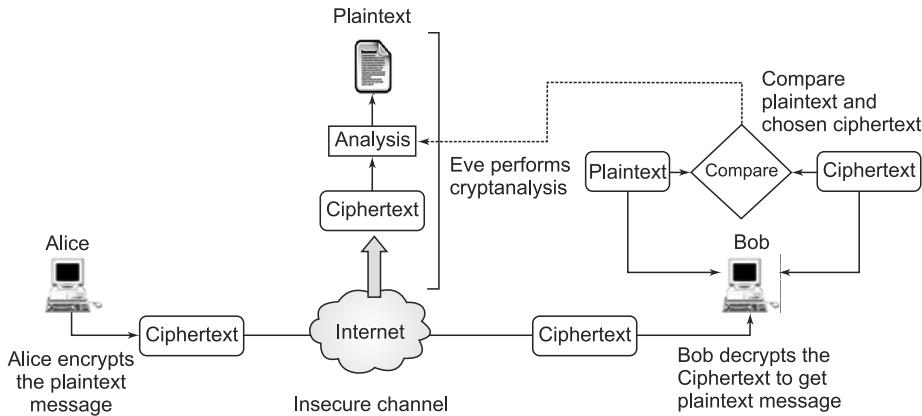


FIGURE 2.5 Chosen plaintext attack.

2.4.4 Chosen Ciphertext Attack (CCA)

The attacker has the ability to select any ciphertext and study the plaintext produced by decrypting it. The attacker has the capability to make the victim decrypt a selected ciphertext and send him the result. Analyzing the chosen ciphertext and corresponding received plaintext, the attacker links them together to guess the sent key which was used by the victim.



2.4.5 Chosen-Text Attack

The attacker assumes the encryption algorithm to decrypt the intercepted ciphertext. A set of secret keys are used to generate the plaintext.

2.5 LOCATION OF ENCRYPTION DEVICES

Encryption is a powerful and commonly used method to protect the information in a network. When using encryption we need to decide what to encrypt and where the encryption mechanism should be located. There are two fundamental approaches, as shown in Figure 2.7:

1. Link Encryption
2. End-to-End Encryption

In the link encryption method each vulnerable communication link is equipped on both ends with an encryption device. Thus all traffic over

every communication link is secured. This mechanism includes a number of encryption devices in a large network. It provides high-level security. The security level in the link is very high.

One of the main limitations of this approach is the message is vulnerable at each switch. The message must be decrypted each time it enters a packet switch. This is essential, because the switch must read the address in the packet header to route the packet. Also, if it is a public packet switching network, the user has no control over the security of these nodes.

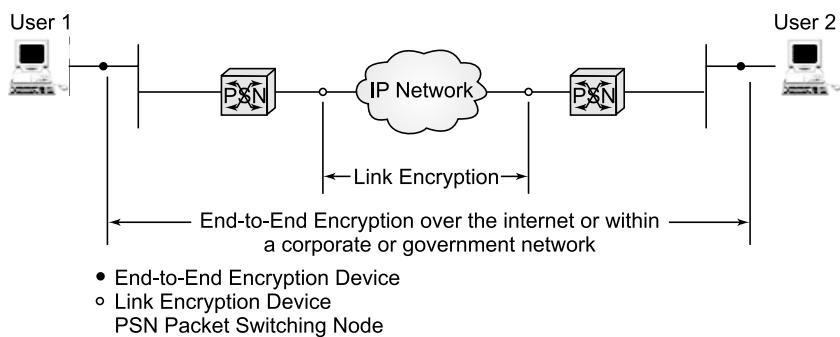


FIGURE 2.7 Location of encryption devices in a packet switched network.

In end-to-end encryption, the encryption process is carried out at the end systems by the sender at the point of origin and only decrypted by the intended receiver. The end terminals encrypt the message and it is then transmitted across the network to the destination terminal. The receiver at the destination decrypts the message using the shared key. This approach would seem to secure the transmission against attacks on the network links or switches. This method also has some limitations. Consider that the terminal device encrypts the entire packet. The packet contains a header and part of the user data. The receiving terminal will not receive this encrypted packet. This is because to read and forward this encrypted message, the intermediate nodes will require the address of the next node. If the packet with both header and data are encrypted, it cannot route the packets. Therefore, the terminal device may only encrypt the user data portion of the packet and must leave the header in the clear so that it can be read by the next node in the network. To attain a higher level of security, both link and end-to-end encryption are needed.

SUMMARY

- Cryptography is a technique of transforming plaintext into ciphertext by encrypting plaintext messages and also obtaining original plaintext by decryption. This kind of cryptography can also provide other services such as integrity checking and authentication.
- In cryptography, the encryption process and security level mainly depend on the secrecy of the key used and not on the secrecy of the algorithm.
- The type of transformation from plaintext to ciphertext is based on either the substitution or transformation method.
- Based on the number of keys used, cryptography is classified as symmetric key or asymmetric key encryption.
- Based on the way in which the plaintext is processed, it is classified as a block cipher or stream cipher method.
- The process of breaking a crypto system to discover the plaintext or key is called cryptanalysis.
- The encryption is made safer by using a particular encryption technique and the location of the encryption mechanism. The two approaches are link encryption and end-to-end encryption.

REVIEW QUESTIONS

1. What are the principle procedures of a conventional encryption technique?
2. Explain a standard data encryption algorithm in detail.
3. What is cryptanalysis?
4. What is plaintext? What is ciphertext? Explain the procedure for the conversion of plaintext into ciphertext.
5. What is the difference between a substitution cipher and a transposition cipher?
6. What are the two basic ways of transforming plaintext into ciphertext?
7. Define ciphertext.

MULTIPLE CHOICE QUESTIONS

1. In cryptography, what is a cipher?
 - (a) Algorithm for performing encryption and decryption
 - (b) Encrypted message
 - (c) Both (a) and (b)
 - (d) None of the above
2. In asymmetric key cryptography, the private key is kept by:

(a) Sender	(b) Receiver
(c) Sender and receiver	(d) All the connected devices to the network
3. Cryptanalysis is used:
 - (a) To find some insecurity in a cryptographic scheme
 - (b) To increase the speed
 - (c) To encrypt the data
 - (d) None of the above
4. The encrypted messages are called:
 - (a) Plaintext
 - (b) Clear text
 - (c) Ciphertext
 - (d) Encryption text
5. A substitution cipher performs the following:
 - (a) Characters are replaced by other characters
 - (b) Rows are replaced by columns
 - (c) Columns are replaced by rows
 - (d) None of the above
6. A combination of an encryption algorithm and a decryption algorithm is called a:
 - (a) Cipher
 - (b) Secret
 - (c) Key
 - (d) None of the above
7. In a(n) _____ cipher, the same key is used by both the sender and receiver.
 - (a) Symmetric key
 - (b) Asymmetric key
 - (c) Both (a) and (b)
 - (d) None of the above

- 8.** In a(n) _____ cipher, the key is called the secret key.

 - (a) Symmetric key
 - (b) Asymmetric key
 - (c) Both (a) and (b)
 - (d) None of the above
- 9.** In the link encryption method each vulnerable communication link is equipped on both ends with a(n):

 - (a) Encryption device
 - (b) Communication link
 - (c) User terminal
 - (d) Packet switched network
- 10.** The _____ of a threat measures its potential impact on a system.

 - (a) Vulnerabilities
 - (b) Countermeasures
 - (c) Degree of harm
 - (d) Susceptibility

CHAPTER 3

SYMMETRIC KEY ENCRYPTION

3.1 INTRODUCTION

The concept of securing messages through cryptography has a long history. Indeed, Julius Caesar is credited with creating one of the earliest cryptographic systems to send military messages to his generals.

Historically, encryption systems used what is known as symmetric key cryptography. It is also referred to as conventional encryption or single key encryption. Using symmetric cryptography, it is safe to send encrypted messages without any fear of interception, however, there always remains the difficulty of how to securely transfer the key to the recipients of a message so that they can decrypt the message. This chapter explains the different techniques of symmetric key cryptography or symmetric key encryption. These concepts can be used to better understand modern ciphers.

3.2 GENERAL IDEA OF SYMMETRIC KEY ENCRYPTION

A cryptographic algorithm, or cipher, is the mathematical function used for the encryption or decryption of a message. Cryptography is the process of dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of a message.

The basic component of cryptography is a cryptosystem. It has five tuples (E, D, P, K, and C),

Where:

P – is the set of plaintext

K – is the set of keys

C – is the set of ciphertext

Where: $E : P \times K \rightarrow C$ is the set of encryption functions
 $D : C \times K \rightarrow P$ is the set of decryption functions

For example, the Caesar cipher is a widely known algorithm of symmetric key encryption in which letters are shifted using the Caesar cipher with a key of 3. With this key the character “a” becomes “D,” “b” becomes “E” and so forth, ending with “z” becoming “C.”

Example 3.1: Encrypt the plaintext “secure” using the Caesar cipher.

This cipher is a cryptosystem with:

$$\begin{aligned} K &= \{i | i \text{ an integer such that } 0 \leq i \leq 25\} \\ E &= \{E_k | k \in K \text{ and for all } p \in P\} \quad E_k(p) = (p + k) \bmod 26 \end{aligned}$$

i.e., Plaintext \rightarrow (Encryption) \rightarrow Ciphertext

Table 3.1 represents each letter by its position in the alphabet (with A in position 0 and so on):

TABLE 3.1 Letters and their Positional Values.

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

For example, the plaintext “secure” is encrypted as:

Encryption: Key $K = 3$

Plaintext	Position Value (P)	Encryption C = (P+K)	Ciphertext (C)
S	18	21	V
E	04	07	H
C	02	05	F
U	20	23	X
R	17	20	U
E	07	07	H

Plaintext is “secure” \rightarrow Ciphertext is “VHFXUH”

Decryption: For decryption:

$$\begin{aligned} D &= \{D_k | k \in K \text{ and for all } c \in C\} \\ D_k(c) &= (26 + c - K) \bmod 26 \end{aligned}$$

For each D_k simply invert the corresponding E_k
i.e., Ciphertext → (Decryption) → Plaintext

Cipher	Position Value (C)	Decryption P = (C-K)	Plaintext (P)
V	21	$21 - 3 = 18$	s
H	07	$7 - 3 = 4$	e
F	05	$5 - 3 = 2$	c
X	23	$23 - 3 = 20$	u
U	20	$20 - 3 = 17$	r
H	07	$7 - 3 = 4$	e

Ciphertext is “VHFXUH” → Plaintext is “secure”

3.3 SYMMETRIC KEY ENCRYPTION

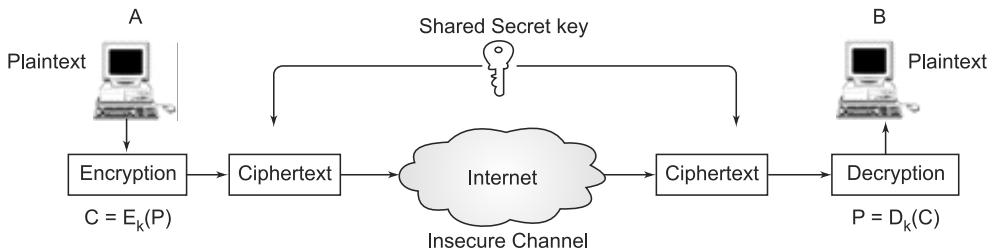


FIGURE 3.1 General idea of symmetric key encryption.

In a symmetric key encryption, a single key is used for both the encryption and decryption process. If P is the plaintext, C is the ciphertext, and K is the key as in Figure 3.1.

The encryption algorithm is:

$$C = E_k(P)$$

This creates the ciphertext C from the plaintext P . The decryption algorithm $P = D_k(C)$ creates the plaintext P from the ciphertext C .

In the late 1940s, Claude Shannon laid out some good design criteria for symmetric ciphers. First he considered *Kerckhoff's* principle. It states that the security of a cryptosystem should depend solely on the secrecy of the key. Another way of putting it is that a method of secretly coding and transmitting information should be secure even if everyone knows how it works. Based on *Kerckhoff's* law one should assume that the adversary knows

the cryptographic algorithm. The resistance of the cipher to attack must be based only on the secrecy of the key. In other words if the algorithm is available for an attacker, it should be difficult to break the security by guessing the key. Second, Shannon emphasized that a good cipher should incorporate both confusion and diffusion. By confusion we mean that a cipher should hide local patterns in language from an attacker. By diffusion we mean that the cipher should mix around different parts of the plaintext, so that nothing is left in its original position.

3.4 CLASSIFICATIONS OF TRADITIONAL CIPHERS

Symmetric key ciphers are classified as *substitution ciphers* and *transposition ciphers*. The encryption techniques for substitution ciphers and transposition ciphers are shown in Figure 3.2:

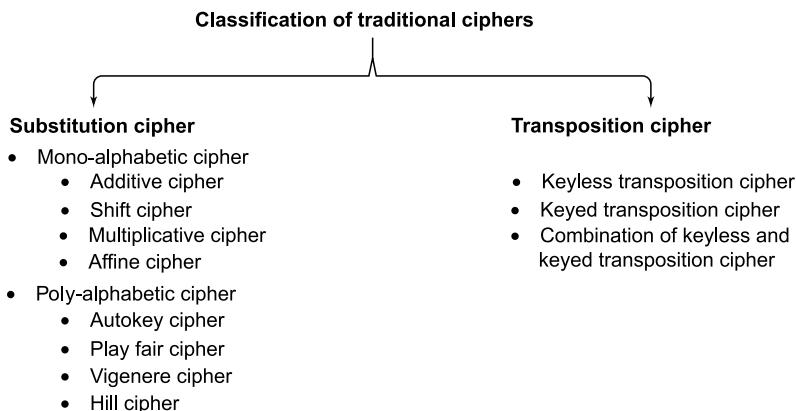


FIGURE 3.2 Classifications of traditional ciphers.

3.4.1 Substitution Ciphers

A substitution cipher replaces a character in the plaintext with another. The Caesar cipher is a general example for a substitution cipher. In a Caesar cipher if the key is 3, then the letter “s” is replaced with “V,” and “e” with “H.” If the character is a numerical digit 0 to 9, then 3 is replaced with 6 and 2 with 5. To perform mathematical operations on the plaintext and ciphertext, we assign numerical values to each alphabetic character (upper or lowercase) as shown in Figure 3.3.

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Numerical value	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

FIGURE 3.3 Representation of the numerical values of plaintext and ciphertext.

Substitution ciphers can be categorized as:

1. Mono-alphabetic ciphers
2. Poly-alphabetic ciphers

3.4.1.1 Mono-Alphabetic Ciphers

A mono-alphabetic cipher is a substitution cipher. It takes a letter of an alphabet and substitutes it with another letter; this is the way a ciphertext is generated. The way of conversion is fixed. A character of a plaintext will always be replaced by the same ciphertext character in the entire ciphertext.

Example 3.2: Encrypt the plaintext message “secure” by mono-alphabetic cipher using the key K = 3.

A plaintext and its corresponding ciphertext is shown as follows:

Plaintext	Encryption → (P + K)mod26	Ciphertext
s → 18	(18 + 3) mod26	21 → V
e → 4	(4 + 3) mod26	7 → H
c → 2	(2 + 3) mod26	5 → F
u → 20	(20 + 3) mod26	23 → X
r → 17	(17 + 3) mod26	20 → U
e → 4	(4 + 3) mod26	7 → H

Plaintext: *secure* → Ciphertext: *VHFXUH*

In the previous example the ciphertext obtained is mono-alphabetic, because both plaintext “e”’s are encrypted as “H.”

The different types of mono-alphabetic ciphers are:

1. Additive ciphers
2. Shift ciphers
3. Multiplicative ciphers
4. Affine ciphers

Additive Cipher

An additive cipher takes letters in a plaintext message and shifts the letter through the alphabet by some integer amount (key). An additive cipher is also known as Caesar cipher or shift cipher. The additive cipher operation is depicted as in Figure 3.4.

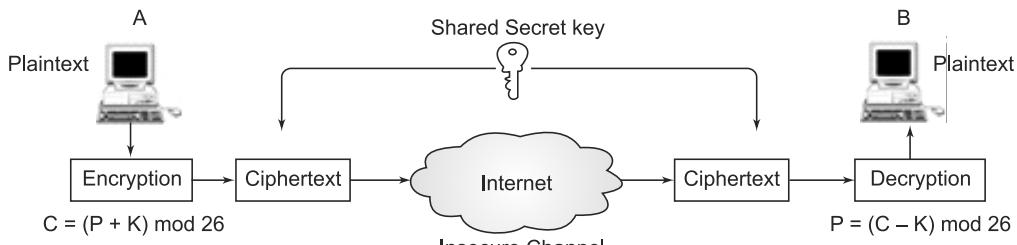


FIGURE 3.4 Additive cipher.

In an additive cipher, the encryption algorithm adds the key to the plaintext character

$$C = (P + K) \text{ mod } 26$$

where the plaintext character is assigned an integer in Z_{26} . Any number greater than the index Z_{26} loops back around, otherwise known as the mod 26 secret key used to message communication between the sender A and the receiver B, which is also an integer Z_{26} as in Figure 3.3.

In the previous process, we can easily prove that the encryption and decryption are inverse of each other, that is, the plaintext received by B(P_B) after the decryption of the ciphertext is the same and the one sent by A.

$$P_B = (C - K) \text{ mod } 26 = (P + K - K) \text{ mod } 26 = P$$

Example 3.3: Using the additive cipher with key 12, encrypt the message “secure” and decrypt the encrypted message using the same key.

Encryption

Plaintext	Encryption	Ciphertext
s → 18	$(18 + 12) \text{ mod } 26$	04 → E
e → 04	$(04 + 12) \text{ mod } 26$	16 → Q
c → 02	$(02 + 12) \text{ mod } 26$	14 → O
u → 20	$(20 + 12) \text{ mod } 26$	06 → G

Plaintext	Encryption	Ciphertext
r → 17	$(17 + 12) \text{ mod } 26$	03 → D
e → 04	$(04 + 12) \text{ mod } 26$	16 → Q

After encryption the result is “EQOGDQ.” As one of the characteristics of mono-alphabetic ciphers, the plaintext character “e” is encrypted with the same character “Q” in both places.

Plaintext: *secure* → Ciphertext: *EQOGDQ*

Decryption

Ciphertext	Decryption	Plaintext
E → 04	$(04 - 12) \text{ mod } 26$	18 → s
Q → 16	$(16 - 12) \text{ mod } 26$	04 → e
O → 14	$(14 - 12) \text{ mod } 26$	02 → c
G → 06	$(06 - 12) \text{ mod } 26$	20 → u
D → 03	$(03 - 12) \text{ mod } 26$	17 → r
Q → 16	$(16 - 12) \text{ mod } 26$	04 → e

Ciphertext: *EQOGDQ* → Plaintext: *secure*

Shift Cipher

A shift cipher works by using the model operative to encrypt and decrypt the messages. The shift cipher has a key K , which is an integer from 0 to 25. If the key is K , each letter in the plaintext is replaced with the K^{th} letter following the corresponding number (shift right). Decryption for the given K , performs the shift left operation.

Example 3.4: Using a shift cipher with key 11, encrypt the plaintext message “CRYPTOGRAPHY” and decrypt the ciphertext using the same key.

Plaintext: CRYPTOGRAPHY
Key: $K = 11$ Ciphertext: NCJAVZRCLASJ

Multiplicative Cipher

A multiplicative cipher uses arithmetic operations such as multiplication and division for the encryption and decryption process as shown in Figure 3.5. For the encryption operation multiply the key (K) by plaintext, and for decryption divide the ciphertext by the same key.

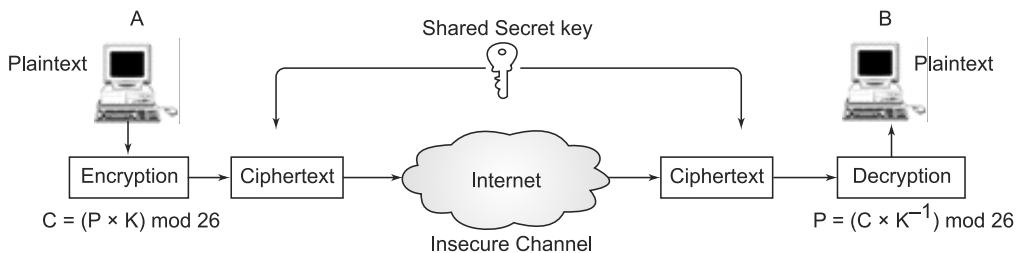


FIGURE 3.5 Multiplicative cipher.

Encryption steps

1. Translate the message (plaintext) into the corresponding number (e.g., $c = 2$).
2. Multiply the number by key (K), (let $K = 11$, $c = 2$; $c * K = 2 * 11 = 22$).
3. Reduce result to mod 26 and translate back into a letter (ciphertext) $22 \rightarrow W$.

Decryption steps

1. Translate the ciphertext into its corresponding number (e.g., $Q = 16$).
2. Multiply the number by the multiplicative inverse of K , K^{-1} ($K^{-1} = 19$; $16 * 19 = 304$).
3. Reduce mod26 and translate it back to plaintext ($304 = 18 \text{ mod } 26$; $18 \rightarrow s$).

Example 3.5: Encrypt the message “*secure*” using a multiplicative cipher with a key = 11. Also decrypt the ciphertext.

Encryption

Plaintext	Encryption	Ciphertext
$s \rightarrow 18$	$(18 \times 11) \text{ mod } 26$	198
$e \rightarrow 04$	$(04 \times 11) \text{ mod } 26$	44
$c \rightarrow 02$	$(02 \times 11) \text{ mod } 26$	22
$u \rightarrow 20$	$(20 \times 11) \text{ mod } 26$	220
$r \rightarrow 17$	$(17 \times 11) \text{ mod } 26$	187
$e \rightarrow 04$	$(04 \times 11) \text{ mod } 26$	44
Plaintext:	<i>secure</i>	<i>Ciphertext:</i> <i>QSWMF</i>

Decryption: The possible multiplicative inverses of mod 26 are:

1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, and 25.

TABLE 3.2 Multipliers and their Inverse.

Number	1	3	5	7	9	11	15	17	19	21	23	25
Multiplicative Inverse	1	9	21	15	3	19	7	23	11	5	17	25

Here are the possible multipliers and their inverse:

$1 \times 1 = 1; \quad 1 \text{ mod } 26$	$11 \times 19 = 209; \quad 1 \text{ mod } 26$
$3 \times 9 = 27; \quad 1 \text{ mod } 26$	$17 \times 23 = 391; \quad 1 \text{ mod } 26$
$5 \times 21 = 105; \quad 1 \text{ mod } 26$	$25 \times 25 = 625; \quad 1 \text{ mod } 26$
$7 \times 15 = 105; \quad 1 \text{ mod } 26$	

For 2, there is no number mod 26 that 2 can be multiplied by so that the number will result in 1 (or 1 mod 26). Hence, 2 does not have an inverse.

Ciphertext	Decryption	Plaintext
$Q \rightarrow 16$	$16 \times 11^{-1} \text{ mod } 26 = 16 \times 19 \text{ mod } 26$	304
$S \rightarrow 18$	$18 \times 11^{-1} \text{ mod } 26 = 18 \times 19 \text{ mod } 26$	342
$W \rightarrow 22$	$22 \times 11^{-1} \text{ mod } 26 = 22 \times 19 \text{ mod } 26$	418
$M \rightarrow 12$	$12 \times 11^{-1} \text{ mod } 26 = 12 \times 19 \text{ mod } 26$	228
$F \rightarrow 05$	$05 \times 11^{-1} \text{ mod } 26 = 05 \times 19 \text{ mod } 26$	95
$S \rightarrow 18$	$18 \times 11^{-1} \text{ mod } 26 = 18 \times 19 \text{ mod } 26$	342

Affine Cipher

An affine cipher is obtained by combining both additive and multiplicative ciphers. It uses a combination of both ciphers with a pair of keys K_1 and K_2 . The key K_1 is used for multiplication and key K_2 is used for addition, as shown in Figure 3.6.

Using the pair of keys K_1 and K_2 , the affine cipher performs encryption or decryption operations as:

$$\text{Encryption operation} - C = (P \times K_1 + K_2) \text{ mod } 26$$

$$\text{Decryption operation} - P = (C - K_2 + K_1^{-1}) \text{ mod } 26$$

This also shows the relationship between the plaintext P and the ciphertext C.

Encryption steps

1. Convert the plaintext to numbers.
2. Multiply by the multiplicative encryption key (K_1) and reduce mod 26.
3. Add the additive encryption key (K_2) and reduce mod 26.
4. Convert numbers to the equivalent ciphertext.

Decryption steps

1. Convert ciphertext to numbers.
2. Subtract the additive inverse using key (K_2), and reduce mod 26.
3. Multiply by the multiplicative inverse using the key (K_1^{-1}), and reduce mod 26.

$$\{(K_1) \times (K_1^{-1}) = 1 \pmod{26}\}$$

4. Convert numbers to plaintext.

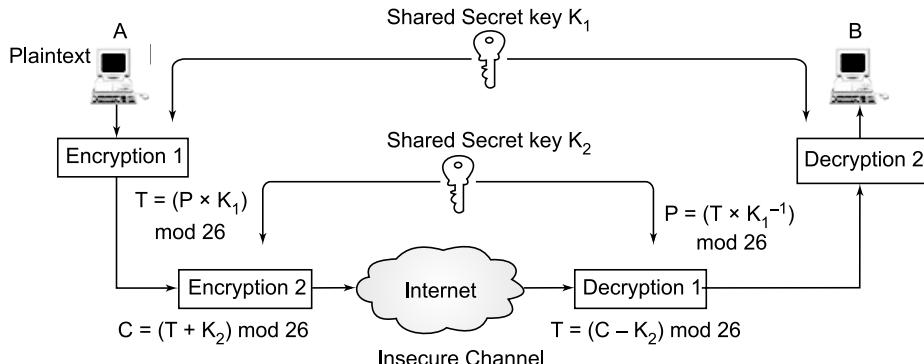


FIGURE 3.6 Affine cipher.

Example 3.6: Use an affine cipher to show the encryption and decryption of the plaintext “secure” with the key (5, 3).

Encryption

Plaintext	Encryption	Ciphertext
$s \rightarrow 18$	$(18 \times 5 + 3) \pmod{26}$	93
$e \rightarrow 04$	$(04 \times 5 + 3) \pmod{26}$	23

Plaintext	Encryption		Ciphertext
c → 02	(02 × 5 + 3) mod 26	13	13 → N
u → 20	(20 × 5 + 3) mod 26	103	25 → Z
r → 17	(17 × 5 + 3) mod 26	88	10 → K
e → 04	(04 × 5 + 3) mod 26	23	23 → X

Plaintext: “**secure**” → Ciphertext: “**PXNZKX**”

Decryption

Ciphertext	Decryption			Plaintext
P → 15	$(15 - 3) \times 5^{-1} \text{ mod } 26$	12 × 21 mod 26	252	18 → s
X → 23	$(23 - 3) \times 5^{-1} \text{ mod } 26$	20 × 21 mod 26	420	4 → e
N → 13	$(13 - 3) \times 5^{-1} \text{ mod } 26$	10 × 21 mod 26	210	2 → c
Z → 25	$(25 - 3) \times 5^{-1} \text{ mod } 26$	22 × 21 mod 26	462	20 → u
K → 10	$(10 - 3) \times 5^{-1} \text{ mod } 26$	7 × 21 mod 26	147	17 → r
X → 23	$(23 - 3) \times 5^{-1} \text{ mod } 26$	20 × 21 mod 26	420	4 → e

Ciphertext: “**PXNZKX**” → Plaintext: “**secure**”

3.4.1.2 Poly-Alphabetic Cipher

In a poly-alphabetic cipher each existence of a character in plaintext uses a different substitution mapping (called multiple alphabets) on various portions of the plaintext. In the simplest case, the different alphabets are used sequentially and then repeated, so the position of each plaintext character in the source string determines which mapping is applied to it. Under different alphabets the same plaintext character is thus encrypted to different ciphertext characters precluding simple frequency analysis as per mono-alphabetic substitution.

For example, “r” is encrypted with “A” in the beginning of the text and “Y” at the middle. A poly-alphabetic cipher is obtained by using a stream of sub-keys. Each sub-key depends somehow on the position of the plaintext character which uses that sub-key for encryption. That is, we require more keys or a key stream as: $k_i = (k_1, k_2, \dots)$, where ($i = 1, 2, 3, \dots$). The key k_i is used to encipher the i^{th} character in the plaintext to create the i^{th} character in the ciphertext.

The different types of poly-alphabetic ciphers are:

1. Autokey cipher
2. Playfair cipher
3. Vigenère cipher
4. Hill cipher

Autokey Cipher

The autokey cipher is a poly-alphabetic substitution cipher that uses the plaintext itself as a key, along with a keyword placed at the beginning of the plaintext. For the autokey cipher, the key is a value $K \in Z_{26}$, and the key stream generator sets $l_1 = k$ and generates subsequent key stream elements by shifting the plaintext character by one position, that is $l_i = x_{i-1}$, where the plaintext is x_1, x_2, \dots . The ciphertext obtained by adding plaintext and key stream element mod26.

Since the sub-keys are automatically generated from the plaintext, the ciphertext character changes for the same plaintext character.

Let the plaintext be: $P = P_1 P_2 P_3 \dots$

Sub-keys: $K = K_1, K_2, K_3 \dots$

Ciphertext: $C = C_1 C_2 C_3 \dots$

The encryption and decryption process is represented as:

Encryption $\rightarrow C_i = (P_i + K_i) \text{ mod } 26$

Decryption $\rightarrow P_i = (C_i - K_i) \text{ mod } 26$

Example 3.7: Consider that A and B agreed to exchange a message “**send password**” using an auto-key cipher encryption method with $K_1 = 5$.

The complete encryption process is explained in the following steps:

1. Replace each character in the plaintext by its integer value.
2. The first sub-key is added to generate the first ciphertext character.
3. The rest of the keys are generated as the plaintext characters are used.

Plaintext (P)	S	E	N	D	P	A	S	S	W	O	R	D
P 's value	18	04	13	03	15	00	18	18	22	14	17	03
Key streams	05	18	04	13	03	15	00	18	18	22	14	12
C 's value	23	22	17	16	18	15	18	10	14	10	05	20
Ciphertext	X	W	R	Q	S	P	S	K	O	K	F	U

Plaintext (P): “**send password**” \rightarrow Ciphertext (C): “**XWRQSPSKOKFU**”

Playfair Cipher

The Playfair cipher is a manual symmetric encryption cipher invented in 1854 by Charles Wheatstone, however its name and popularity came from the endorsement of Lord Playfair. The Playfair cipher encrypts pairs of letters, instead of single letters as is the case with simpler substitution ciphers such as the Caesar cipher. To generate the secret key, a matrix of the order of 5×5 using 25 alphabetic characters is constructed. The matrix is constructed by selecting a keyword that does not contain any letter more than once. For example, consider the word “*keyword*,” which does not contain any letter more than once. Now write the letter of that word (keyword) in the first squares of the 5×5 matrix.

K	E	Y	W	O
R	D			

FIGURE 3.7 Five-by-five matrix for the generation of a secret key.

Then fill the remaining space with the rest of the letters of the alphabet; in order to reduce the alphabet to fit, you can either omit “Q” or replace “I” with “J.” The encryption and decryption rules are as follows:

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

FIGURE 3.8 Secret key matrix.

For Encryption

1. Group the message into pairs of letters. In the plaintext if both letters are same, add “x” between them. Regroup the remaining letters. If there is only one letter remaining, append “x” to the last letter.
2. If the letters appear in the same row of the table, replace them with the letters to their immediate right respectively, wrapping around to the left side of the row if necessary.

3. If the letters appear in the same column of the table, replace them with the letters immediately below, wrapping around to the top if necessary.
4. If the letters are in different rows and columns, replace them with the letters in the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important. The first letter of the pair should be replaced first.

For Decryption

1. Ignore rule 1.
2. In rules 2 and 3 shift up and left/right instead of down and right.
3. Rule 4 remains the same.
4. Drop any extra x's that don't make sense in the final message and locate any missing Q's or any I's that should be J's.

In each case the encryption algorithm takes a pair of characters from the plaintext and creates pairs of sub-keys by considering the previous rules. The key stream depends on the position of the character in the plaintext. This key stream is considered as the ciphertext.

$$\begin{aligned}
 \text{Plaintext: } P_i &= P_1 P_2 P_3 P_4 \dots \\
 \text{Two character pair} &= P_1 P_2, P_3 P_4, \dots \\
 K &= \{(K_1, K_2), (K_3, K_4), \dots\} \\
 \text{Encryption} \rightarrow C_i &= K_i \\
 \text{Ciphertext} \rightarrow C_1 C_2, C_3 C_4, \dots &= C_1 C_2 C_3 C_4, \dots \\
 \text{Decryption} \quad P_i &= K_i
 \end{aligned}$$

Example 3.8: Encrypt the message “password” using the Playfair cipher.

Use the secret key as in Figure 3.8.

Group the message in to two characters: “pa, ss, wo, rd”

We need to insert an x between the two s's, and x at the end. It gives; “pa, sx, sw, or, dx”

The ciphertext is obtained by following rules 2, 3, and 4.

Plaintext	Ciphertext
Pa	NB
Sx	PZ
Sw	PO
Or	KC
Dx	BU

FIGURE 3.9 Ciphertext in playfair cipher.

Plaintext: “***password***” → Ciphertext: “**NBPZPOOKCBU**”

Example 3.9: Encrypt the plaintext “send password” using the Playfair cipher with the secret key in Figure 3.8.

Plaintext: “***send password***”

Group the characters: “*se, nd, pa, ss, wo, rd*”

Add x: “*se, nd, pa, sx, sw, or, dx*”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

Secret key matrix

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

for plaintext “*se*”
the ciphertext is “*MO*”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

for plaintext “*nd*” the ciphertext is “*MA*”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

for plaintext “*pa*” the
ciphertext is “*NB*”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

For plaintext “sx” the ciphertext is “PZ”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

For plaintext “sw” the ciphertext is “PO”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

For plaintext “or” the ciphertext is “KC”

K	E	Y	W	O
R	D	A	B	C
F	G	H	I	J
L	M	N	P	S
T	U	V	X	Z

For plaintext “dx” the ciphertext is “BU”

Finally, the ciphertext for the complete plaintext message is: “MO MA
NB PZ PO KC BU”

Plaintext: “**send password**” Ciphertext: **MOMANBPZPOKCBU**

Vigenère Cipher

The Vigenère cipher was proposed by Blaise de Vigenère, a French mathematician in the sixteenth century. It is a poly-alphabetic substitution-based encryption technique. It uses a key stream of length m , where we have $1 \leq m \leq 26$. The following points are to be considered during the selection of the key stream.

In a Vigenère cipher, to make brute force decryption impractical, the key should have at least 15 or 16 characters. Also, it may be best if all letters of the key are distinct.

Duplicate the keys as many times as necessary so that the length of the (duplicated) key matches the length of the plaintext. The key stream does not depend on the plaintext characters; it depends only on the position of the characters in the plaintext.

The cipher can be described as follows:

The plaintext is - $P = P_1, P_2, P_3, \dots$

If K_1, K_2, \dots, K_m is the initial secret key agreed to by both sender and the receiver, the generated key stream using this initial secret key is $K = [(K_1, K_2, \dots, K_m)(K_1, K_2, \dots, K_m) \dots]$

The ciphertext obtained out of this is $C = C_1, C_2, C_3, \dots$

Encryption $C_i = (P_i + K_i) \bmod 26$

Decryption $P_i = (C_i - K_i) \bmod 26$

Example 3.10: Encrypt the message “**send password**” using the five-character keyword “**HELLO**.” The initial key stream is:

Plaintext: “**send password**”

Key duplicated: “**hello**”

Plaintext (P)	s	e	n	d	p	a	s	s	w	o	r	d
P 's value	18	04	13	03	15	00	18	18	22	14	17	03
Key (duplicated) K	h	e	l	l	o	h	e	l	l	o	h	e
K 's value	07	04	11	11	14	07	04	11	11	14	07	04
$C_i = (P_i + K_i) \bmod 26$	25	08	24	14	03	07	22	03	07	08	24	07
Ciphertext(C)	Z	I	Y	O	D	H	W	D	H	I	Y	H

Plaintext: “**send password**” Ciphertext: **ZIYODHWDHIYH**

Decryption

To decrypt the message encrypted with the Vigenère keyword method, first write the key repeatedly. Write the ciphertext beneath it. The decryption is performed by $P_i = (C_i - K_i) \bmod 26$.

Key (duplicated) K	h	e	l	l	o	h	e	l	l	o	h	e
K 's value	07	04	11	11	14	07	04	11	11	14	07	04
Ciphertext (C)	Z	I	Y	O	D	H	W	D	H	I	Y	H
C 's value	25	08	24	14	03	07	22	03	07	08	24	07
$P_i = (C_i - K_i) \bmod 26$	18	04	13	03	15	00	18	18	22	14	17	03
Ciphertext(C)	s	e	n	d	p	a	s	s	w	o	r	d

Ciphertext: **ZIYODHWDHIYH** Plaintext: “**send password**”

Example 3.11: Encrypt the message “top secret message” using the key “wonderland.”

Plaintext: ***topsecretmessage*** Key (duplicated): ***wonderland***

Plaintext (P)	t	o	p	s	e	c	r	e	t	m	e	s	s	a	g	e
P 's value	19	14	15	18	4	2	17	4	19	12	4	18	18	0	6	4
Key (duplicated) K	w	o	n	d	e	r	l	a	n	d	w	o	n	d	e	r
K 's value	22	14	13	3	4	17	11	0	13	3	22	14	13	3	4	17
$C_i = (P_i + K_i) \bmod 26$	15	02	02	21	8	19	02	04	06	15	0	06	05	03	10	21
Ciphertext(C)	P	C	C	V	I	T	C	E	G	P	A	G	F	D	K	V

Plaintext: ***top secret message*** Ciphertext: ***PCCVITCEGPAGFDKV***

Hill Cipher

A Hill cipher is a poly-alphabetic cipher based on linear algebra. It uses more advanced mathematics as well as an encryption and decryption scheme. A Hill cipher solves the frequency distribution problem as in a Caesar cipher. In both the encryption and decryption process, a numerical value is assigned to each letter of the plaintext characters.

For the encryption process the given plaintext message is divided into blocks of size “ n ” (where n is an integer). Therefore, the Hill cipher is also considered as a block cipher. These blocks are written as a column vector, and it is multiplied by any invertible matrix of size $(n \times n)$. The encryption matrix must be invertible because its inverse will be used to decrypt the ciphertexts created by the Hill cipher. The invertibility of the encryption matrix shows that its determinant value must not be zero.

During encryption the plaintext blocks are encrypted one at a time in such a way that each character in the block contributes to the encryption of other characters in the block. This is performed by using a square matrix of size $n \times n$ as a key where n is the size of the plaintext block. The elements of a key matrix K are represented as K_{ij} in Figure 3.9(a).

$$K = \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix}$$

FIGURE 3.9 (a). The key used in a Hill cipher.

For the encryption of one block of plaintext:

The n -characters in the plaintext block are $P_1 P_2 \dots P_n$

The resulting ciphertext are: $C_1 C_2 \dots C_n$, which is obtained as follows:

Where,

$$\begin{bmatrix} C_1 \\ C_2 \\ \cdot \\ \cdot \\ C_n \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \cdot \\ \cdot \\ P_n \end{bmatrix} \times \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix}$$

$$C_1 = P_1 K_{11} + P_2 K_{21} + \dots + P_n K_{n1}$$

$$C_2 = P_1 K_{12} + P_2 K_{22} + \dots + P_n K_{n2}$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$C_n = P_1 K_{1n} + P_2 K_{2n} + \dots + P_n K_{nn}$$

The resulting ciphertext characters $C_i = C_1 C_2 \dots C_n$ depend on all plaintext characters $P_i = P_1 P_2 \dots P_n$ in the block. To obtain the cipher the message sender and receiver have to carefully select the key, because not all square matrices have multiplicative inverses in Z_{26} . If a matrix does not have a multiplicative inverse, the receiver cannot decrypt the message.

Hill cipher encryption algorithm steps:

The steps used to encrypt the message are as follows:

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix}$$

Step 1: Choose an $n \times n$ encryption matrix, say $K =$

The conditions for the selection of this $n \times n$ matrix are:

- (i) The determinant of the matrix K should not be zero; $|K| \neq 0$
- (ii) The determinant of the matrix K , ($|K|$) is also to be relatively prime to 26.

Step 2: Split the plaintext message into a block of size “ n ” (ignoring spaces). Convert the letters into numerical values and align them as column

vectors. If the length of the plaintext is not evenly divisible by “ n ,” add a previously decided character to the end of the string until the plaintext is evenly divisible by “ n .”

Step 3: Multiply each of these column vectors by the encryption matrix and take modulo 26 of the result.

Step 4: Convert each of the matrices obtained in step 3 to their alphabetical vectors and combine them to produce the ciphertext.

Hill cipher decryption algorithm steps:

Now the receiver has the ciphertext message and the encryption key. Using this he can decrypt the ciphertext to produce the plaintext. The decryption algorithm is essentially the same as the encryption algorithm, except that we use K^{-1} in place of K .

Since $C = KP$ and K is invertible, we can calculate $P = K^{-1}C$.

We will call $D = K^{-1}$

So, the decryption matrix is $DC = P$. This inverse is the inverse modulo 26.

Step 1: Find $D = K^{-1} \pmod{26}$. This is the decryption key.

Step 2: Convert the ciphertext into the matrix C .

Step 3: Calculate $DC = P$.

Step 4: Convert the matrix P to its equivalent plaintext message. Insert the appropriate spaces and punctuation symbols.

Hill cipher invertible matrix selection:

The procedure for the selection of a 3×3 invertible matrix K to encrypt data and another 3×3 invertible matrix A to decrypt data is as follows:

Matrix K might be selected as $K = \begin{bmatrix} 1 & 2 & 3 \\ -3 & 2 & 1 \\ 2 & -1 & 3 \end{bmatrix}$

Calculate the determinant of K

1. $\det \begin{vmatrix} 2 & 1 \\ -1 & 3 \end{vmatrix} = +1 \times (2 \times 3 - (-1 \times 1)) = 7$

$$2. \det \begin{vmatrix} -3 & 1 \\ 2 & 3 \end{vmatrix} = -2 \times ((-3 \times 3) - (2 \times 1)) = 22$$

$$3. \det \begin{vmatrix} -3 & 2 \\ 2 & -1 \end{vmatrix} = +3 \times ((-3 \times -1) - (2 \times 2)) = -3$$

$$\therefore \det K = 7 + 22 - 3 = 26$$

$$26 \bmod 26 = 0$$

In the inverse table (Table 3.2) there is no entry for 0, so there is no modular inverse. For this reason matrix K cannot be selected; the message could be encrypted by using matrix K , but there would be no way to decrypt them.

$$\text{If the matrix } K \text{ is modified as } K = \begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix}$$

Its determinant is found as:

$$1. \det \begin{vmatrix} 3 & 1 \\ -1 & 2 \end{vmatrix} = +1 \times ((3 \times 2) - (-1 \times 1)) = 7$$

$$2. \det \begin{vmatrix} -2 & 1 \\ 3 & 2 \end{vmatrix} = -2 \times ((-2 \times 2) - (3 \times 1)) = 14$$

$$3. \det \begin{vmatrix} -2 & 3 \\ 3 & -1 \end{vmatrix} = -2 \times ((-2 \times 1) - (3 \times 3)) = 14$$

$$\therefore \det K = 7 + 14 + 14 = 35$$

$$35 \bmod 26 = 9$$

In the inverse table (Table 3.2), the modular inverse of 9 is 3.

To calculate A (where $A = 3 \text{ adj } k$) we have to multiply $\text{adj. } K$ by the modular inverse of $\det |K|$ for mod (m) and then calculate mod (m) for the entries in the resulting matrix.

$\text{adj. } K$ is calculated as:

$$C_{11} = +\det \begin{vmatrix} 3 & 1 \\ -1 & 2 \end{vmatrix} = ((3 \times 2) - (-1 \times 1)) = 7$$

$$C_{12} = -\det \begin{vmatrix} -2 & 1 \\ 3 & 2 \end{vmatrix} = ((-2 \times 2) - (3 \times 1)) = 7$$

$$C_{13} = + \det \begin{vmatrix} -2 & 3 \\ 3 & -1 \end{vmatrix} = ((-2 \times -1) - (3 \times 3)) = -7$$

$$C_{21} = - \det \begin{vmatrix} 2 & -2 \\ -1 & 2 \end{vmatrix} = -((2 \times 2) - (-1 \times -2)) = -2$$

$$C_{22} = + \det \begin{vmatrix} 1 & -2 \\ 3 & 2 \end{vmatrix} = ((1 \times 2) - (-2 \times 3)) = 8$$

$$C_{23} = - \det \begin{vmatrix} 1 & 2 \\ 3 & -1 \end{vmatrix} = -((1 \times 1) - (3 \times 2)) = 7$$

$$C_{31} = + \det \begin{vmatrix} 2 & -2 \\ 3 & 1 \end{vmatrix} = ((2 \times 1) - (3 \times -2)) = 8$$

$$C_{32} = - \det \begin{vmatrix} 1 & -2 \\ -2 & 1 \end{vmatrix} = -((1 \times 1) - (-2 \times -2)) = 3$$

$$C_{33} = + \det \begin{vmatrix} 1 & 2 \\ -2 & 3 \end{vmatrix} = ((1 \times 3) - (-2 \times 2)) = 7$$

The resulting new matrix is $C = \begin{bmatrix} 7 & 7 & -7 \\ -2 & 8 & 7 \\ 8 & 3 & 7 \end{bmatrix}$

adj.K is the transpose of matrix $C = C^T = \begin{bmatrix} 7 & -2 & 8 \\ 7 & 8 & 3 \\ -7 & 7 & 7 \end{bmatrix}$

\therefore The modular inverse of K is: $A = 3 \times \text{adj } K$

The modular inverse of matrix K is: $A = 3 \times \text{adj } K$

$$A = 3 \times \begin{bmatrix} 7 & -2 & 8 \\ 7 & 8 & 3 \\ -7 & 7 & 7 \end{bmatrix} = \begin{bmatrix} 21 & -6 & 24 \\ 21 & 24 & 9 \\ -21 & 21 & 21 \end{bmatrix}$$

$$A = \begin{bmatrix} 21 & -6 & 24 \\ 21 & 24 & 9 \\ -21 & 21 & 21 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix}$$

\therefore The modular inverse of matrix K is: $K^{-1} = \begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix}$

This is also the key matrix to be used to decrypt a message encrypted by the Hill cipher using the matrix:

$$\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix}$$

To check that $\begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix}$ is a modular inverse of $\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix}$

$$\begin{aligned} &= \begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 21 + 42 - 10 & 20 + 48 - 42 & 24 + 18 - 42 \\ -42 + 63 + 5 & -40 + 72 + 21 & -48 + 27 + 21 \\ 63 - 21 + 10 & 60 - 24 + 42 & 72 - 9 + 42 \end{bmatrix} \\ &= \begin{bmatrix} 53 & 26 & 0 \\ 26 & 53 & 0 \\ 52 & 78 & 105 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I \begin{bmatrix} 53 & 26 & 0 \\ 26 & 53 & 0 \\ 52 & 78 & 105 \end{bmatrix} \text{ mod } 26 = I \end{aligned}$$

Example 3.12: Encrypt the message “**send password**” using a Hill cipher.

The encryption matrix is $K = \begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix}$.

Encryption

To encrypt, divide the plaintext into blocks the size of three characters and assign their values.

The plaintext is “**send password**”

$$\begin{vmatrix} S \\ E \\ N \end{vmatrix} = \begin{vmatrix} 18 \\ 4 \\ 13 \end{vmatrix}, \quad \begin{vmatrix} D \\ P \\ A \end{vmatrix} = \begin{vmatrix} 3 \\ 15 \\ 0 \end{vmatrix}, \quad \begin{vmatrix} S \\ W \end{vmatrix} = \begin{vmatrix} 18 \\ 18 \\ 22 \end{vmatrix}, \quad \begin{vmatrix} O \\ R \\ D \end{vmatrix} = \begin{vmatrix} 14 \\ 17 \\ 3 \end{vmatrix}$$

$$\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix} \times \begin{vmatrix} 18 \\ 4 \\ 13 \end{vmatrix} = \begin{bmatrix} 18 + 8 - 26 \\ -36 + 12 + 13 \\ 54 - 4 + 26 \end{bmatrix} = \begin{vmatrix} 0 \\ -11 \\ 76 \end{vmatrix} \bmod 26 = \begin{vmatrix} 0 \\ 15 \\ 24 \end{vmatrix} = \begin{vmatrix} A \\ P \\ Y \end{vmatrix}$$

$$\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix} \times \begin{vmatrix} 03 \\ 15 \\ 0 \end{vmatrix} = \begin{bmatrix} 3 + 30 + 0 \\ -6 + 45 + 0 \\ 9 - 15 + 0 \end{bmatrix} = \begin{vmatrix} 33 \\ 39 \\ -6 \end{vmatrix} \bmod 26 = \begin{vmatrix} 7 \\ 13 \\ 20 \end{vmatrix} = \begin{vmatrix} H \\ N \\ U \end{vmatrix}$$

$$\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix} \times \begin{vmatrix} 18 \\ 18 \\ 22 \end{vmatrix} = \begin{bmatrix} 18 + 36 - 44 \\ -36 + 54 + 22 \\ 54 - 18 + 44 \end{bmatrix} = \begin{vmatrix} 10 \\ 40 \\ 80 \end{vmatrix} \bmod 26 = \begin{vmatrix} 10 \\ 14 \\ 2 \end{vmatrix} = \begin{vmatrix} K \\ O \\ C \end{vmatrix}$$

$$\begin{bmatrix} 1 & 2 & -2 \\ -2 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix} \times \begin{vmatrix} 14 \\ 17 \\ 3 \end{vmatrix} = \begin{bmatrix} 14 + 34 - 6 \\ -28 + 51 + 3 \\ 42 - 17 + 6 \end{bmatrix} = \begin{vmatrix} 42 \\ 26 \\ 31 \end{vmatrix} \bmod 26 = \begin{vmatrix} 16 \\ 0 \\ 5 \end{vmatrix} = \begin{vmatrix} Q \\ A \\ F \end{vmatrix}$$

The ciphertext is “**ALYHNUKOSQAF**”

Decryption

To decrypt the ciphertext “**ALYHNUKOSQAF**” divide it into a block size of three characters and assign its values.

$$\begin{vmatrix} A \\ P \\ Y \end{vmatrix} = \begin{vmatrix} 0 \\ 15 \\ 24 \end{vmatrix}, \quad \begin{vmatrix} H \\ N \\ U \end{vmatrix} = \begin{vmatrix} 7 \\ 13 \\ 20 \end{vmatrix}, \quad \begin{vmatrix} K \\ O \\ C \end{vmatrix} = \begin{vmatrix} 10 \\ 14 \\ 2 \end{vmatrix}, \quad \begin{vmatrix} Q \\ A \\ F \end{vmatrix} = \begin{vmatrix} 16 \\ 0 \\ 5 \end{vmatrix}$$

Now multiply each of the numerical vectors by the decryption matrix

$$(K^{-1}) \begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix}$$

which is the modular inverse of Matrix K (the calculation is explained in the previous example).

$$\begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix} \times \begin{vmatrix} 0 \\ 15 \\ 24 \end{vmatrix} = \begin{bmatrix} 0 + 300 + 576 \\ 0 + 360 + 216 \\ 0 + 315 + 504 \end{bmatrix} = \begin{bmatrix} 876 \\ 576 \\ 819 \end{bmatrix} \text{ mod } 26 = \begin{vmatrix} 18 \\ 4 \\ 13 \end{vmatrix} = \begin{bmatrix} S \\ E \\ N \end{bmatrix}$$

$$\begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix} \times \begin{vmatrix} 7 \\ 13 \\ 20 \end{vmatrix} = \begin{bmatrix} 147 + 260 + 480 \\ 147 + 312 + 180 \\ 35 + 273 + 420 \end{bmatrix} = \begin{bmatrix} 887 \\ 637 \\ 728 \end{bmatrix} \text{ mod } 26 = \begin{vmatrix} 3 \\ 15 \\ 0 \end{vmatrix} = \begin{bmatrix} D \\ P \\ A \end{bmatrix}$$

$$\begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix} \times \begin{vmatrix} 10 \\ 14 \\ 2 \end{vmatrix} = \begin{bmatrix} 210 + 280 + 48 \\ 210 + 336 + 18 \\ 50 + 294 + 42 \end{bmatrix} = \begin{bmatrix} 538 \\ 564 \\ 386 \end{bmatrix} \text{ mod } 26 = \begin{vmatrix} 18 \\ 18 \\ 22 \end{vmatrix} = \begin{bmatrix} S \\ S \\ W \end{bmatrix}$$

$$\begin{bmatrix} 21 & 20 & 24 \\ 21 & 24 & 9 \\ 5 & 21 & 21 \end{bmatrix} \times \begin{vmatrix} 16 \\ 0 \\ 5 \end{vmatrix} = \begin{bmatrix} 336 + 0 + 120 \\ 336 + 0 + 45 \\ 80 + 0 + 105 \end{bmatrix} = \begin{bmatrix} 456 \\ 381 \\ 185 \end{bmatrix} \text{ mod } 26 = \begin{vmatrix} 14 \\ 17 \\ 3 \end{vmatrix} = \begin{bmatrix} O \\ R \\ D \end{bmatrix}$$

The decrypted message is “***sendpassword***”

Example 3.13: Encrypt the message “send password” with $n = 2$.

Plaintext: “***send password***”

Matrix size $(n \times n) = (2 \times 2)$

Step 1: plaintext – “send password” $n = 2$

The matrix is $K = \begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix}$ determinant of $|K| = [(3 \times 7) - (2 \times 5)] = 21 - 10 = 11$

Since, $11 \neq 0$ this matrix is invertible. 11 is also relatively prime to 26. These two qualities satisfy the requirements listed in step 1 for considering K as a key to perform the encryption.

Step 2: Split the plaintext into blocks of size 2.

Plaintext: “***sendpassword***”

$$\begin{vmatrix} s \\ e \end{vmatrix} = \begin{vmatrix} 18 \\ 04 \end{vmatrix}, \begin{vmatrix} n \\ d \end{vmatrix} = \begin{vmatrix} 13 \\ 03 \end{vmatrix}, \begin{vmatrix} p \\ a \end{vmatrix} = \begin{vmatrix} 15 \\ 00 \end{vmatrix}, \begin{vmatrix} s \\ s \end{vmatrix} = \begin{vmatrix} 18 \\ 18 \end{vmatrix}, \begin{vmatrix} w \\ o \end{vmatrix} = \begin{vmatrix} 22 \\ 14 \end{vmatrix}, \begin{vmatrix} r \\ d \end{vmatrix} = \begin{vmatrix} 17 \\ 03 \end{vmatrix}$$

Step 3: Multiply each column vector by the encryption matrix and take modulo 26 of the result.

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 18 \\ 04 \end{vmatrix} = \begin{vmatrix} (3 \times 18) + (2 \times 04) \\ (5 \times 18) + (7 \times 04) \end{vmatrix} = \begin{vmatrix} 54 + 8 \\ 90 + 28 \end{vmatrix} = \begin{vmatrix} 62 \\ 118 \end{vmatrix} = \begin{vmatrix} 10 \\ 14 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 13 \\ 03 \end{vmatrix} = \begin{vmatrix} (3 \times 13) + (2 \times 03) \\ (5 \times 13) + (7 \times 03) \end{vmatrix} = \begin{vmatrix} 39 + 6 \\ 65 + 21 \end{vmatrix} = \begin{vmatrix} 45 \\ 86 \end{vmatrix} = \begin{vmatrix} 19 \\ 8 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 15 \\ 00 \end{vmatrix} = \begin{vmatrix} (3 \times 15) + (2 \times 00) \\ (5 \times 15) + (7 \times 00) \end{vmatrix} = \begin{vmatrix} 45 + 0 \\ 75 + 0 \end{vmatrix} = \begin{vmatrix} 45 \\ 75 \end{vmatrix} = \begin{vmatrix} 19 \\ 23 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 18 \\ 18 \end{vmatrix} = \begin{vmatrix} (3 \times 18) + (2 \times 18) \\ (5 \times 18) + (7 \times 18) \end{vmatrix} = \begin{vmatrix} 54 + 36 \\ 90 + 126 \end{vmatrix} = \begin{vmatrix} 90 \\ 216 \end{vmatrix} = \begin{vmatrix} 12 \\ 8 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 22 \\ 14 \end{vmatrix} = \begin{vmatrix} (3 \times 22) + (2 \times 14) \\ (5 \times 22) + (7 \times 14) \end{vmatrix} = \begin{vmatrix} 66 + 28 \\ 110 + 98 \end{vmatrix} = \begin{vmatrix} 94 \\ 206 \end{vmatrix} = \begin{vmatrix} 16 \\ 0 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 2 \\ 5 & 7 \end{vmatrix} \times \begin{vmatrix} 17 \\ 03 \end{vmatrix} = \begin{vmatrix} (3 \times 17) + (2 \times 03) \\ (5 \times 17) + (7 \times 03) \end{vmatrix} = \begin{vmatrix} 51 + 6 \\ 85 + 21 \end{vmatrix} = \begin{vmatrix} 57 \\ 106 \end{vmatrix} = \begin{vmatrix} 5 \\ 2 \end{vmatrix} \text{ mod } 26$$

Step 4: Convert each matrix obtained in step 3 to their alphabetical vectors and combine them to produce the ciphertext.

$$\begin{vmatrix} 10 \\ 14 \end{vmatrix} = \begin{vmatrix} K \\ O \end{vmatrix}, \begin{vmatrix} 19 \\ 8 \end{vmatrix} = \begin{vmatrix} T \\ I \end{vmatrix}, \begin{vmatrix} 19 \\ 23 \end{vmatrix} = \begin{vmatrix} T \\ X \end{vmatrix}, \begin{vmatrix} 12 \\ 8 \end{vmatrix} = \begin{vmatrix} M \\ I \end{vmatrix}, \begin{vmatrix} 16 \\ 0 \end{vmatrix} = \begin{vmatrix} Q \\ A \end{vmatrix}, \begin{vmatrix} 5 \\ 2 \end{vmatrix} = \begin{vmatrix} F \\ C \end{vmatrix},$$

The ciphertext is; “**KOTITXMIQAF**C”

Example 3.14: Decrypt the ciphertext “**KOTITXMIQAF**C” with a (2×2) key matrix as used in the encryption process (Example 3.10).

Step 1: Find K^{-1}

$$\begin{aligned} \det \begin{pmatrix} 3 & 2 \\ 5 & 7 \end{pmatrix} &= (3 \times 7) - (2 \times 5) \\ &= 21 - 10 \\ &= 11 \end{aligned}$$

$$11^{-1} \text{ mod } 26 = 19$$

(Multiplicative inverse of 11 = 19)

$$19 \begin{vmatrix} 7 & -2 \\ -5 & 3 \end{vmatrix} = \begin{vmatrix} 133 & -38 \\ -95 & 57 \end{vmatrix} = \begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \text{ mod } 26$$

Step 2: Split the ciphertext into blocks of 2, determine the letters' numerical values and align them as column vectors.

$$\begin{vmatrix} K \\ O \end{vmatrix} = \begin{vmatrix} 10 \\ 14 \end{vmatrix}, \quad \begin{vmatrix} T \\ I \end{vmatrix} = \begin{vmatrix} 19 \\ 8 \end{vmatrix}, \quad \begin{vmatrix} T \\ X \end{vmatrix} = \begin{vmatrix} 19 \\ 23 \end{vmatrix}, \quad \begin{vmatrix} M \\ I \end{vmatrix} = \begin{vmatrix} 12 \\ 8 \end{vmatrix}, \quad \begin{vmatrix} Q \\ A \end{vmatrix} = \begin{vmatrix} 16 \\ 0 \end{vmatrix}, \quad \begin{vmatrix} F \\ C \end{vmatrix} = \begin{vmatrix} 5 \\ 2 \end{vmatrix}$$

Step 3: Multiply each of these column vectors by the decryption matrix calculated in step 1 and take modulo 26 of the result.

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 10 \\ 14 \end{vmatrix} = \begin{vmatrix} (3 \times 10) + (14 \times 14) \\ (9 \times 10) + (5 \times 14) \end{vmatrix} = \begin{vmatrix} 30 + 196 \\ 90 + 70 \end{vmatrix} = \begin{vmatrix} 226 \\ 160 \end{vmatrix} = \begin{vmatrix} 18 \\ 4 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 19 \\ 8 \end{vmatrix} = \begin{vmatrix} (3 \times 19) + (14 \times 8) \\ (9 \times 19) + (5 \times 8) \end{vmatrix} = \begin{vmatrix} 57 + 112 \\ 171 + 40 \end{vmatrix} = \begin{vmatrix} 169 \\ 211 \end{vmatrix} = \begin{vmatrix} 13 \\ 3 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 19 \\ 23 \end{vmatrix} = \begin{vmatrix} (3 \times 19) + (14 \times 23) \\ (9 \times 19) + (5 \times 23) \end{vmatrix} = \begin{vmatrix} 57 + 322 \\ 171 + 115 \end{vmatrix} = \begin{vmatrix} 379 \\ 286 \end{vmatrix} = \begin{vmatrix} 15 \\ 0 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 12 \\ 8 \end{vmatrix} = \begin{vmatrix} (3 \times 12) + (14 \times 8) \\ (9 \times 12) + (5 \times 8) \end{vmatrix} = \begin{vmatrix} 36 + 112 \\ 108 + 40 \end{vmatrix} = \begin{vmatrix} 148 \\ 148 \end{vmatrix} = \begin{vmatrix} 18 \\ 18 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 16 \\ 0 \end{vmatrix} = \begin{vmatrix} (3 \times 16) + (14 \times 0) \\ (9 \times 16) + (5 \times 0) \end{vmatrix} = \begin{vmatrix} 48 + 0 \\ 144 + 0 \end{vmatrix} = \begin{vmatrix} 48 \\ 144 \end{vmatrix} = \begin{vmatrix} 22 \\ 14 \end{vmatrix} \text{ mod } 26$$

$$\begin{vmatrix} 3 & 14 \\ 9 & 5 \end{vmatrix} \times \begin{vmatrix} 5 \\ 2 \end{vmatrix} = \begin{vmatrix} (3 \times 5) + (14 \times 2) \\ (9 \times 5) + (5 \times 2) \end{vmatrix} = \begin{vmatrix} 15 + 28 \\ 45 + 10 \end{vmatrix} = \begin{vmatrix} 43 \\ 55 \end{vmatrix} = \begin{vmatrix} 17 \\ 3 \end{vmatrix} \text{ mod } 26$$

Step 4: Convert each of the matrices obtained in step 3 to their respective alphabetic vectors and combine them to produce the plaintext.

$$\begin{vmatrix} 18 \\ 4 \end{vmatrix} = \begin{vmatrix} s \\ e \end{vmatrix}, \quad \begin{vmatrix} 13 \\ 3 \end{vmatrix} = \begin{vmatrix} n \\ d \end{vmatrix}, \quad \begin{vmatrix} 15 \\ 0 \end{vmatrix} = \begin{vmatrix} p \\ a \end{vmatrix}, \quad \begin{vmatrix} 18 \\ 18 \end{vmatrix} = \begin{vmatrix} s \\ s \end{vmatrix}, \quad \begin{vmatrix} 22 \\ 14 \end{vmatrix} = \begin{vmatrix} w \\ o \end{vmatrix}, \quad \begin{vmatrix} 17 \\ 3 \end{vmatrix} = \begin{vmatrix} r \\ d \end{vmatrix},$$

The plaintext obtained after the decryption process is: “**sendpassword**”

The ciphertext is: “**KOTITXMIQAF**” and the corresponding plaintext is “**sendpassword**.”

3.4.2 Transposition Cipher

A transposition cipher rearranges all the characters in the plaintext to form the ciphertext. For example, a character in the first position of a plaintext

may appear in the tenth position of the ciphertext and a character in the sixth position in the plaintext may appear in the first position of the ciphertext.

Transposition ciphers are classified as:

1. Keyless transposition ciphers
2. Keyed transposition ciphers

3.4.2.1 Keyless Transposition Ciphers

There are two methods in keyless transposition ciphers:

1. The first method is also known as the rail fence cipher. In this method the plaintext is arranged in two lines as a zigzag arrangement as shown in Figure 3.10. Reading this arrangement row by row will give the ciphertext.

For example: To encrypt the message “**the job is done.**” Rearrange the plaintext using the rail fence cipher:

Plaintext: “**the job is done**”

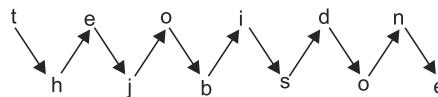


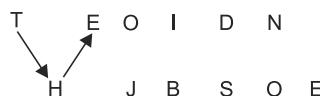
FIGURE 3.10 Rail fence cipher.

The sender creates the ciphertext: “**TEOIDNHJSOEH**”

To decrypt the ciphertext message, the receiver divides the ciphertext into two halves (in some cases the second half has one character less) and regenerates the first row using the first half text and the second row by the second half. Reading the result in a zigzag pattern will give the plaintext message. If the number of rows is not mentioned, then it is assumed it is two.

Decryption: Ciphertext: **TEOIDNHJSOEH**

TEOIDN | HJSOEH



Plaintext: *thejobisdone*

2. The second method is a rail fence cipher using a definite number of rows. In this method the plaintext message is divided into groups of an equal number of characters and it is arranged in zigzag manner as in Figure 3.10. The encryption and decryption process is explained as follows:

Encryption: To encrypt a message using the rail fence cipher, write the message into a known number of zigzag lines across the page, and then read off each row. Write the letters in these lines diagonally down to the right until the number of rows specified is reached. This continues until the end of the plaintext.

Example 3.15: Encrypt the plaintext “*defend the east wall*” using a rail fence cipher with three rows.

The encryption process is as follows:

Divide the plaintext into three equal-length blocks. If the total number of characters in each row are not equal, then add a null character (x) at the end of each row. These nulls act as place holders. Arrange these characters in the rail fence cipher method as shown in Figure 3.11. The ciphertext is created by reading the characters row by row.

D				n			E				T			I		
	E		E		D	H		E	S	w		I	x			
		F			T			a			a				x	

FIGURE 3.11 Rail fence cipher using a definite number of rows.

Ciphertext: “**DNETLEEDHESWLXFTAAX**”

Decryption: The decryption process for the rail fence cipher involves reconstructing the diagonal grid used to encrypt the message. Write the message, but leave a dash in place of the spaces yet to be occupied. Gradually replace all the dashes with the corresponding letters and read off the plaintext from the table. Prepare the rows for making a grid. The number of rows is equal to the number rows used in the encryption process. The first letter of the ciphertext is placed in the top left square and dashes are placed diagonally downward where the letters from the ciphertext will be placed. Continue this process across the row and start the next; when the end is reached it give the plaintext.

Example 3.16: Decrypt the message “**DNETLEEDHESWLXFTAAX**” using three rows.

Step 1:

D				N				E				T			L		
	-		-		-		-		-		-		-		-		-
		-				-			-			-		-			-

Step 2:

D				N				E				T			L		
	E		E		D		H		E		S		W		L		X
		-				-				-			-				-

Step 3:

D				N				E				T			L		
	E		E		D		H		E		S		W		L		X
		F			T				A				A				X

Plaintext: “**DEFENDTHEEASTWALLXX**” “DEFEND THE EAST WALL”

Plaintext: “**defend the east wall**”

In the second method, the text is written into the table row by row and then transmitted column by column.

Example 3.17: To encrypt the plaintext “the job is completed” with key $K = 4$. Split the plaintext message into a block of size 4 ($K = 4$) and arrange it as follows:

t h e j	o b i s
c o m p	l e t e
l e t e	d

The ciphertext is obtained by transforming the characters column by column as follows: “**TOCLDHBOEEIMTJSPE**”

Decryption: The receiver receives the ciphertext and follows the reverse process to obtain the plaintext. To decrypt the message, write the received message column by column and read it row by row as the plaintext. To decrypt the ciphertext in Example 3.17:

The ciphertext is “**TOCLDHBOEEIMTJSPE**”

T	H	E	J
O	B	I	S
C	O	M	P
L	E	T	E
D	*	*	*

If it is known that the key $K = 4$, find the plaintext. There are 17 letters in the ciphertext, which means that there are $17 \text{ DIV } 4 = 4$ full rows and one partial row with $17 \text{ mod } 4 = 1$ letter in it.

After decryption the plaintext message is “**THEJOBISCOMPLETED**”

3.4.2.2 Keyed Transposition Cipher

In a keyed transposition cipher, the plaintext is divided into a group of blocks of a predetermined size, using a key to permute the character in each block separately.

Example 3.18: Encrypt the message “**the boy has the bag**” using a keyed transposition cipher.

Plaintext: “**the boy has the bag**”

Both sender and receiver divide the text into groups of *five* characters and then permute the characters in each group. The given plaintext can be grouped as:

thebo yhast hebag

The permutation key is agreed upon by both sender and receiver. Using the same key, both encryption and decryption is performed. The permutation key is shown in Figure 3.12, where the first character in the ciphertext is the fourth character of the plaintext, the second character of the ciphertext is the fifth character of the plaintext, and so on.

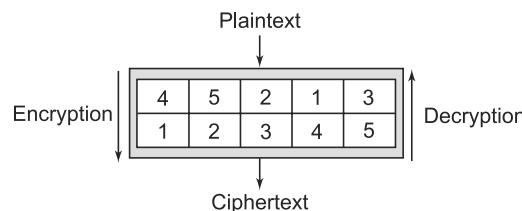


FIGURE 3.12 Permutation key.

After encryption using this permutation key, the characters in the ciphertext are:

bohte sthys agehb

Ciphertext: “**BOHTESTHYAAGEHB**”

To decrypt this at the receiver end, divide the ciphertext into 5 character groups and use the key in reverse order to give the plaintext.

3.4.2.3 Combining Keyless and Keyed Transposition Cipher Approaches

This encryption mechanism is a combination of both the keyed and keyless transposition cipher techniques. It provides better scrambling of the characters. Encryption or decryption follow the three steps mentioned as follows:

Step 1: Divide the plaintext into groups of blocks of predefined size. These texts are arranged into a table row by row as shown in Figure 3.13.

Step 2: The columns in the table are reordered based on the permutation key.

Step 3: The ciphertext is obtained by reading the new table column by column.

Among these three steps step 1 and step 3 are keyless transposition and step 2 is keyed column transposition.

Example 3.19: Encrypt the message “the boy has the bag” using the combined approach.

Encryption: As explained previously in steps 1, 2, and 3.

Decryption: To decrypt the ciphertext and to create the plaintext, the receiver does the same three steps in the reverse order as shown on the right-hand side of Figure 3.13.

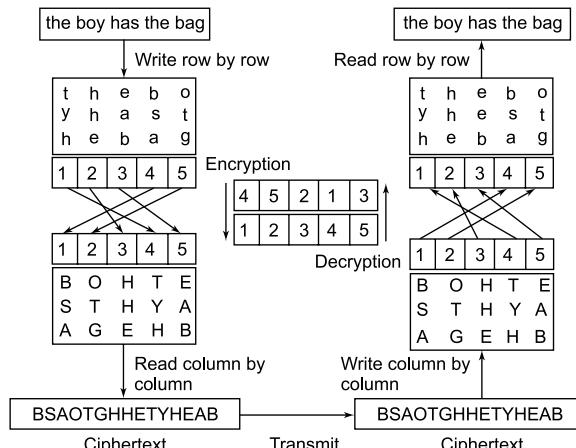


FIGURE 3.13 Combined transposition cipher approach.

Step 1: The receiver writes the ciphertext column by column into the first table.

Step 2: Permute the column using the key to permute.

Step 3: Read the second table row by row.

3.5 STREAM AND BLOCK CIPHERS

3.5.1 Stream Cipher

A stream cipher is a symmetric cipher which encrypts the plaintext message by applying an encryption algorithm with a key stream. Each character of the plaintext message is encrypted one by one (character by character) with the corresponding key stream. Stream ciphers are typically used in cases where speed and simplicity are both requirements.

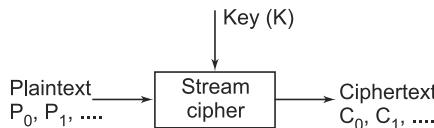


FIGURE 3.14 Stream cipher.

A stream cipher is a symmetric cipher which operates with time varying transformation on individual plaintext digits. It can encrypt plaintext messages of variable lengths. More precisely, in a stream cipher a sequence of plaintext digits are $P = P_1, P_2, P_3, \dots$

A pseudorandom sequence of the key stream is $K = K_1, K_2, K_3, \dots$, and these keys are also known as *running keys*. On encryption the resulting ciphertext stream is $C = C_1, C_2, C_3, \dots$.

Where:

$$\begin{aligned} C_1 &= E_{k1}(P_1) \\ C_2 &= E_{k2}(P_2) \\ C_3 &= E_{k3}(P_3) \end{aligned}$$

Stream ciphers are classified into two types:

1. Synchronous stream cipher
2. Asynchronous stream cipher

In a synchronous stream cipher the key stream depends only on the key provided for encryption as in Figure 3.15 (a), while in asynchronous ones, the key stream depends on the ciphertext obtained in the previous stage. The

ciphertext obtained in the first stage is used as a key for the encryption of the next segment of the plaintext message as in Figure 3.15 (b). The most famous stream cipher is the Vernam cipher. It is also known as the one-time pad.

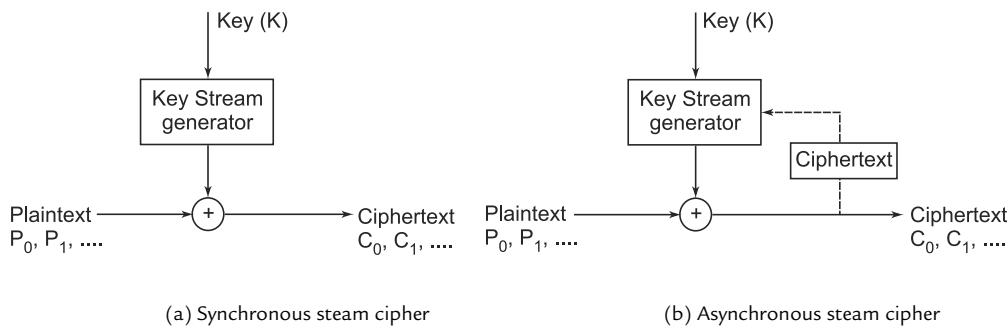


FIGURE 3.15 Stream ciphers.

3.5.1.1 Encryption and Decryption Using Stream Ciphers

Stream ciphers encrypt plaintext bits individually. Each plaintext bit P_i is encrypted by adding a secret key stream bit K_i modulo 2.

$$\text{i.e., } P_i, C_i, K_i \in \{0, 1\}$$

The plaintext, the ciphertext, and the key stream consist of individual bits.

$$\text{Encryption: } C_i = E_{k_i}(P_i) \equiv P_i + K_i \pmod{2} \dots \dots \dots \quad (3.1)$$

$$\text{Decryption: } P_i = D_{k_i}(C_i) \equiv C_i + K_i \pmod{2} \dots \dots \dots \quad (3.2)$$

Since encryption and decryption functions are both simple additions mod 2, this can be depicted as in Figure 3.16.

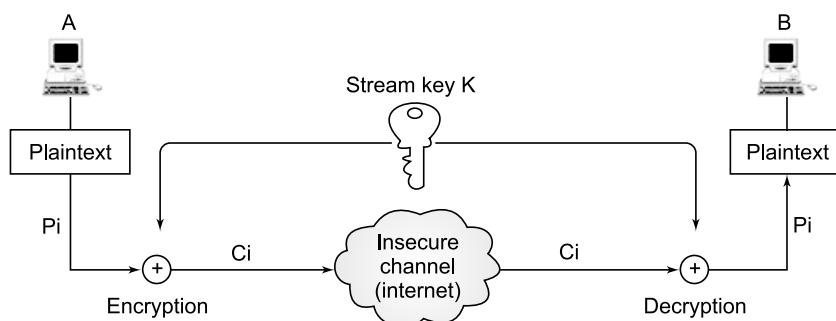


FIGURE 3.16 Encryption and decryption with stream ciphers.

A circle symbol is used in Figure 3.16 to indicate the mod 2 addition operation. This is equivalent to the *XOR* operation, and its truth table is as in Figure 3.17. Equations (3.1) and (3.2) perform the arithmetic mod 2 operation using only the possible values 0 and 1.

P_i	K_i	$C_i = P_i + K_i \text{ mod } 2$
0	0	0
0	1	1
1	0	1
1	1	0

FIGURE 3.17 XOR operation to get ciphertext.

The encryption and decryption are the same functions described as follows. Insert the encryption expression on the decryption function,

$$\begin{aligned}
 D_{k_i}(C_i) &= C_i + k_i \text{ mod } 2 \\
 &= (P_i + K_i) + K_i \text{ mod } 2 \\
 &= P_i + 2K_i \text{ mod } 2 \\
 &\quad [\because 2K_i \text{ mod } 2 \text{ always has the value zero; } 2 = 0 \text{ mod } 2] \\
 &= P_i + 0 \text{ mod } 2 \\
 &= P_i \text{ mod } 2
 \end{aligned}$$

The ciphertext C_i produced using the encryption function is: $C_i = P_i + K_i \text{ mod } 2$.

The decryption function produces the plaintext P_i .

3.5.1.2 Shift Register Based Stream Cipher

A sophisticated way of realizing long pseudorandom sequences is by using linear feedback shift registers (LFSR). LFSRs are the basic components of many running key generators for stream cipher applications. They can be easily implemented in hardware and many, but certainly not all, stream ciphers make use of LFSRs. A prominent example of this type of cipher is the A5/1 cipher, which is standardized for voice encryption in the global system for mobile communication (GSM). An LFSR consists of clocked storage elements (flip-flops) and a feedback path. The number of storage elements gives us the degree of the LFSR. In other words an LFSR with m -flip-flops is said to be of degree m . The feedback network computes the input for the last flip-flop as an XOR sum of certain flip-flops in the shift register. To illustrate the working of a simple

LFSR, consider an LFSR of degree $m = 3$ with FF2, FF1, FF0, and the feedback path as shown in Figure 3.18.

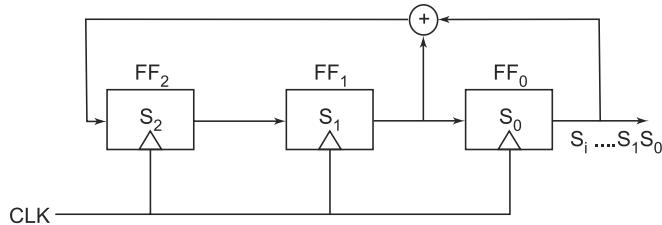


FIGURE 3.18 Linear feedback shift register.

Let the internal state bit be S_i . During every clock pulse it is shifted one bit to the right. The rightmost state bit is also the present output bit. The leftmost state bit is computed in the feedback path, which is the XOR sum of some of the flip-flop values in the previous clock period. This circuit is called a *Linear Feedback Shift Register* due to the linear operation of XOR . Assume the initial state of S_i is $S_2=1$, $S_1=0$, $S_0=0$.

In a different clock period the complete sequence of the state of the LFSR is given in Table 3.2.

The output is indicated in the rightmost column of Table 3.2. From the table it is clear that the sequence state repeats after clock cycle 6. This means the LFSR output has a period of length 7 and has the form: 0010111 0010111 0010111

Considering S_0, S_1 , and S_2 are the initial state bits, the output S_i is computed as:

$$\begin{aligned} S_3 &= S_1 + S_0 \bmod 2 \\ S_4 &= S_2 + S_1 \bmod 2 \\ S_5 &= S_3 + S_2 \bmod 2 \end{aligned}$$

TABLE 3.2 Sequence States of the LFSR.

CLK	FF₂	FF₁	FF₀ = S_i
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1

CLK	FF₂	FF₁	FF₀ = S_i
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

In general, the output bit is computed as $S_{i+3} = S_{i+1} + S_i \bmod 2$, where $i = 0, 1, 2, \dots$.

3.5.1.3 General LFSR with Mathematical Description

The general form of an LFSR of degree “ m ” is shown in Figure 3.19. It contains m flip-flops and m possible feedback locations. All these components are combined by the XOR operations. The feedback coefficients are $f_0 = f_1, f_2, \dots, f_{(m-1)}$. The status of the feedback depends on its value.

If $f_i = 1$ the feedback is active (closed switch).

If $f_i = 0$ the corresponding flip-flop output is not used for the feedback (open switch).

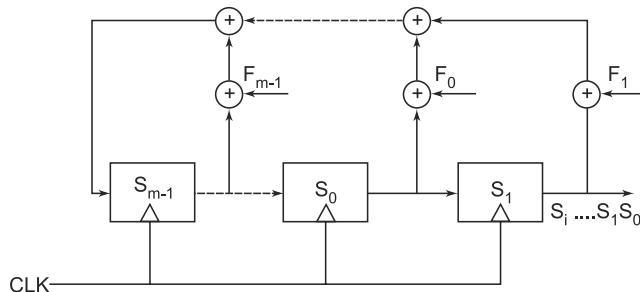


FIGURE 3.19 General form of LFSR of degree m .

The mathematical description for this feedback path is described as follows:

Let us multiply the output of flip-flop i , by its feedback coefficient f_i . If $f_i = 1$ (closed switch) the result is the output value of f_i , or the result is zero if $f_i = 0$ (for open switch). Now consider the LFSR is initially loaded with value $S_0, \dots, S_{(m-1)}$. The next output bit of the LFSR is S_m , which is also the input to the leftmost flip-flop. It can be computed by the XOR sum of the products of the flip-flop output, and the corresponding feedback coefficient is:

$$S_m = S_{(m-1)}f_{(m-1)} + \dots + S_1f_1 + S_0f_0 \bmod 2$$

The next LFSR output sequence is:

$$S_{(m+1)} = S_m f_{(m-1)} + \dots + S_2 f_1 + S_1 f_0 \bmod 2$$

In general, the output sequence is described as:

$$S_{(i+m)} = \sum_{j=0}^{(m-1)} f_j S_{(i+j)} \bmod 2$$

$$S_i, f_j \in \{0, 1\} \quad i = 0, 1, 2, \dots$$

i.e., the output values are given through a combination of some previous output values. Therefore the LFSRs are also referred to as linear recurrences. Due to the finite number of recurring states, the output sequence of an LFSR repeats periodically.

3.5.2 Block Cipher

A block cipher is a symmetric key cipher, operating on fixed length group of bits called *blocks*. In a block cipher the encryption algorithm might take a 128 bit block of plaintext as input and output a corresponding 128 bit ciphertext. The exact transformation is controlled using a second input known as the *secret key*. If the message is longer than 128 bits, it can still be encrypted by using the block cipher method. It now breaks the message into blocks and encrypts each block individually. In this method all the blocks are encrypted with the same key. This may degrade the security level (because each repetition in the plaintext becomes a repetition in the ciphertext also).

A block cipher is a function

$$E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$$

Where E takes two inputs, one being a k -bit string, which is the key for encryption, and the other n -bit string is the plaintext. The resulting ciphertext is an n -bit string. The key length k and the block length n are the parameters associated with the block cipher. The values of k and n vary from one type of block cipher to another.

For each key $K \in \{0, 1\}^k$

$E_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is the function defined by $E_k(P) = E(K, P)$

For any block cipher, using any key K , it is required that the function E_k be a permutation on $\{0, 1\}^n$.

For every $C \in \{0, 1\}^n$ there is exactly one $P \in \{0, 1\}^n$

Such that: $E_k(P) = C$

E_k has an inverse and it is denoted as E_k^{-1} . This function also maps $\{0, 1\}^n$ to $\{0, 1\}^n$ and the decryption process is:

$$E_k^{-1}(E_k(P)) = P \quad \text{and} \quad E_k(E_k^{-1}(C)) = C \text{ for all } P, C \in \{0, 1\}^n.$$

Where $E^{-1} : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

It is defined by $E^{-1}(K, C) = E_k^{-1}(C)$

This is the inverse of block cipher E . In a block cipher both cipher E and its inverse E^{-1} should be easily computable. This means that for a given K and P it can compute $E(K, P)$ and for a given K, C it can compute $E^{-1}(K, C)$. To maintain a higher level of security, a random key K is chosen and kept secret between a pair of users.

3.5.2.1 Block Cipher Modes of Operations

The different possible ways in which block codes can be utilized to implement cryptosystems is explained in the following sections. Each technique uses a block length n , with encrypting maps E_k and decrypting maps D_k for each key K . To overcome security issues different modes of operations are used to make encryption probabilistic. The commonly used block cipher modes are:

1. Electronic codebook (ECB)
2. Cipher block chaining (CBC)
3. Cipher feedback (CFB)
4. Output feedback
5. Counter (CTR) mode encryption

Electronic Codebook Mode (ECB)

The input for the electronic codebook mode is a key of length K bits and n -bit plaintext blocks. These n -bit plaintext blocks are $p = P_0, P_1, \dots, P_t$.

Encryption: The encryption algorithm in ECB, which produces the ciphertext is:

$$C_j = E_k(P_j) \quad \text{where } j = 0, 1, 2, \dots, t$$

The ciphertext obtained for different blocks of plaintext are:

$$\begin{aligned}C_0 &= E_k(P_0) \\C_1 &= E_k(P_1) \\C_2 &= E_k(P_2) \dots\dots\end{aligned}$$

The n -bit cipher blocks are $C = C_0, C_1, \dots, C_t$

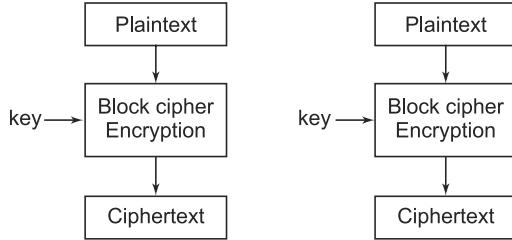


FIGURE 3.20 Electronic codebook (ECB) mode encryption.

Decryption: Decryption is performed by using k -bit key K and n -bit ciphertext blocks $C = C_0, C_1, \dots, C_t$

$$\begin{aligned}\text{Algorithm: } P_j &= D_k(C_j) & P_0 &= D_k(C_0) \\P_1 &= D_k(C_1) \\P_2 &= D_k(C_2) \dots\end{aligned}$$

After decryption the n -bit plaintext blocks are; $P = P_0, P_1, \dots, P_t$

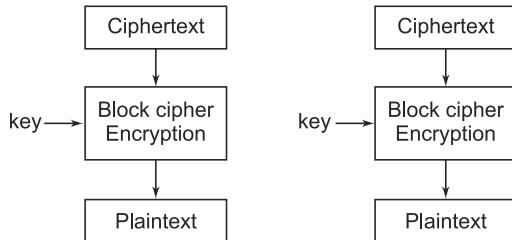


FIGURE 3.21 Electronic codebook (ECB) mode decryption.

Properties of ECB

1. **Identical plaintext:** The same plaintext block always maps to the same ciphertext block.
2. **Chaining dependencies:** Reordering the plaintext blocks induces a reordering of the same ciphertext blocks.

- 3. Error propagation:** An error in a ciphertext block results in a decryption error only in the corresponding plaintext block.

Advantages:

- Encryption or decryption of each block could be parallelized.

Disadvantages:

- Two blocks with identical plaintext produce identical ciphertext.
- A bit error in one block affects the whole block.
- Plaintext patterns are still visible after encryption.

Cipher Block Chaining Mode (CBC)

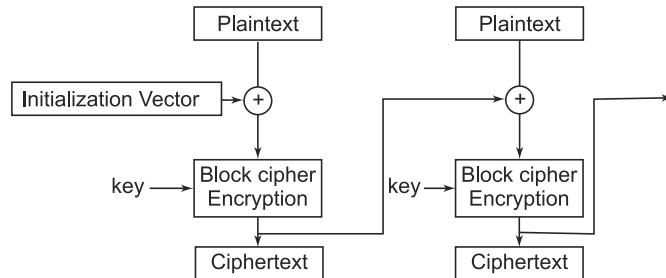


FIGURE 3.22 Cipher block chaining (CBC) mode encryption.

Cipher block chaining mode involves a vector bit sum operation of the plaintext block with the previous ciphertext block prior to encryption. A random initialization vector (IV) is required to initialize CBC mode. This indicates that the security of the cryptosystem is based on the security of the key used and not on the confidentiality of the initialization vector.

Encryption: The input for the encryption process is k -bit key K , n -bit initialization vector (IV), and n -bit plaintext blocks: $P = P_0, P_1, \dots, P_t$

$$\text{Algorithm: } C_j = E_k(C_{j-1} \oplus P_j)$$

The ciphertext obtained for every block of plaintext are:

$$\begin{aligned} C_0 &= E_k(IV \oplus P_0) \\ C_1 &= E_k(C_0 \oplus P_1) \\ C_2 &= E_k(C_1 \oplus P_2) \dots \end{aligned}$$

The output produced is the n -bit ciphertext block:

$$C = C_0, C_1, \dots, C_t$$

Decryption: Decryption is performed by using k -bit key K , and n -bit ciphertext blocks $C = C_0, C_1, \dots, C_t$ as inputs.

$$\text{Algorithm:} \quad P_j = C_{j-1} \oplus D_k(C_j) \quad P_0 = IV \oplus D_k(C_0)$$

$$P_1 = C_0 \oplus D_k(C_1) \quad P_1 = C_0 \oplus D_k(C_1)$$

$$P_2 = C_1 \oplus D_k(C_2) \dots$$

After decryption the n -bit plaintext blocks are:

$$P = P_0, P_1, \dots, P_t$$

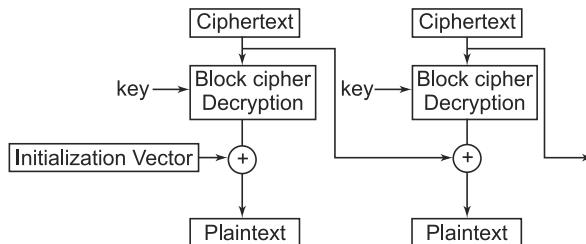


FIGURE 3.23 Cipher block chaining (CBC) mode decryption.

Properties of CBC Mode

- Identical plaintext:** Changing the IV or the first plaintext block results in different ciphertext. The IV need not be secret, but its integrity should be protected.
- Chaining dependencies:** Ciphertext block C_j depends on x_j and all preceding plaintext blocks
- Error Preceding:** A single bit error on C_j may flip the corresponding bit on x_{j+1} but changes x_j significantly.

Advantages

- Decryption could be parallelized.
- Different initial vectors result in different ciphertext blocks.
- Plaintext patterns are blurred.

Disadvantages

- Encryption has to be done sequentially.
- A bit error in one block affects two blocks.

Cipher Feedback Mode (CFB)

The cipher feedback mode allows one to process a block of size $r < n$ at a link. The typical value for $r = 1$, while n may be a size of 64 bits using the data encryption standard (DES) algorithm.

Encryption: The encryption process of the CFB takes the input as k -bit key K , n -bit initialization vector (IV_1), and r -bit plaintext blocks $P = P_0, P_1, \dots, P_t$.

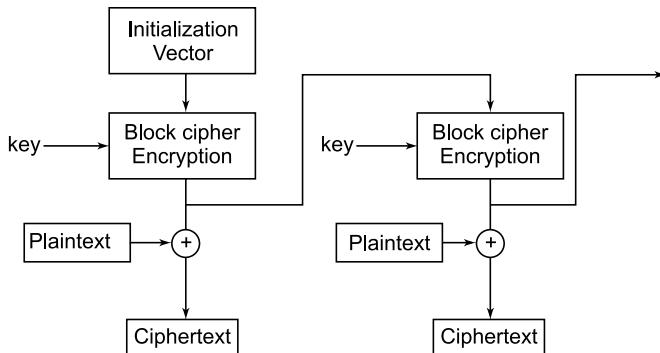


FIGURE 3.24 Cipher feedback (CFB) mode encryption.

$$\text{Algorithm: } C_j = P_k \oplus L_r(E_k(IV_j)) \\ IV_{j+1} = R_{n-r}(IV_j) \parallel C_j$$

Where L_r and R_{n-r} are the operators which take the leftmost r -bits and the rightmost $n-r$ bits and \parallel is the concatenation operator. The vector IV_j should be thought of as a shift register, a block of n -bits of memory, which stores some state of the algorithm. The information IV_{j+1} is a left shift by r bits with the rightmost r bits replaced by C_j .

Description: To obtain the plaintext from the ciphertext, the inputs considered are k -bit key K , n -bit IV_1 , and r -bit ciphertext blocks: $C = C_0, C_1, \dots, C_t$.

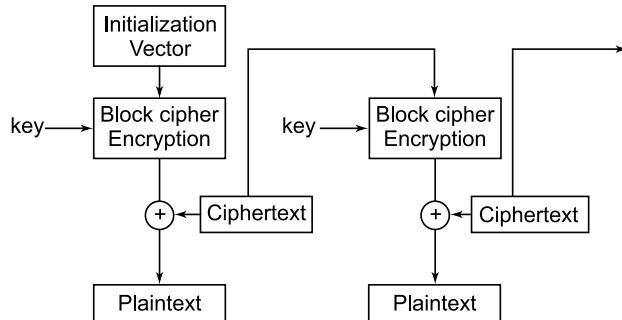


FIGURE 3.25 Cipher feedback (CFB) mode decryption.

Algorithm: Compute IV_1, IV_2, \dots, IV_t as in the encryption algorithm. It can be generated independently of the decrypted plaintext, and then compute:

$$P_j = C_j \oplus L_r(E_k IV_j)$$

From the previous expression it is clear that the decryption of CFB requires only the block cipher E_k and not D_k .

Properties of CFB

- Identical Plaintext:** The same sequence of ciphertext blocks results when the same key and IV are used. Changing the IV changes the ciphertext.
- Chaining Dependencies:** Ciphertext C_j depends on the previous plaintext blocks P_{j-1}, \dots, P_1 as well as P_j , so the ciphertext blocks are not reordered after decryption. Proper decryption of a correct ciphertext block requires the preceding (n/r) ciphertext block to be correct.
- Error propagation:** An error in C_j affects the decryption of the next (n/r) plaintext blocks. The recovered plaintext P_j^1 will differ from P_j at exactly the bits for which C_j was in error. These bit errors will appear in subsequent blocks P_{j+1}^1 at transacted positions.

Advantages:

- No padding is required.
- A bit error affects only one bit.
- Decryption can be parallelized.

Disadvantages:

1. Bit-flipping attacks are easy.
2. Encryption cannot be parallelized.
3. No pre-computation of the key stream.

Output Feedback (OFB)

Output feedback mode has a use similar to cipher feedback mode. This mode is relevant to applications for which error propagation must be avoided. OFB is an example of a synchronous stream cipher in which the key stream is created independently of the plaintext stream.

Encryption: The encryption process uses the input as k -bit key K , n -bit $IV(IV_0)$, and r -bit plaintext block $P = P_0, P_1, \dots, P_t$.

Algorithm: The encryption process in OFB mode is represented as:

$$\begin{aligned} IV_j &= E_k(IV_{j-1}) \\ C_j &= P_j \oplus L_r(IV_{j-1}) \end{aligned}$$

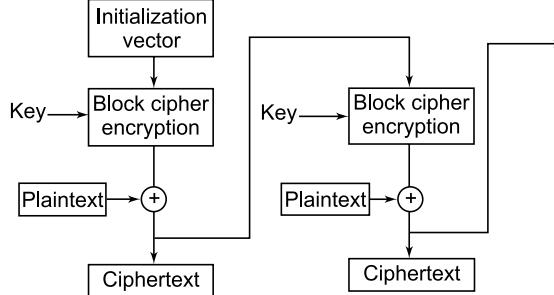
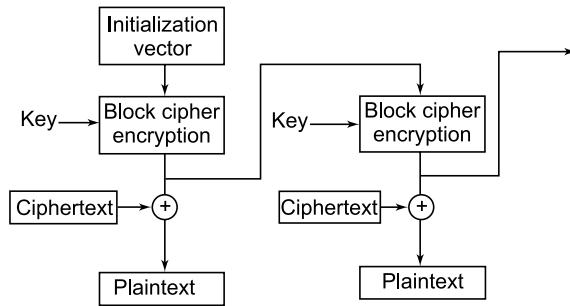


FIGURE 3.26 Output feedback (OFB) mode encryption.

Decryption: The decryption process uses the k -bit key K , n -bit $IV(IV_0)$, and r -bit ciphertext block $C = C_0, C_1, \dots, C_t$.

Algorithm: For decryption compute the IV as IV_1, IV_2, \dots, IV_t in the encryption algorithm:

$$P_j = C_j \oplus L_r(IV_j)$$

**FIGURE 3.27** Output feedback (OFB) mode decryption.

Properties of OFB

- Identical plaintext:** As per CBC and CFB mode, changing the IV results in the same plaintext being encrypted to a different output.
- Chaining dependencies:** The ciphertext output is order dependent, but the key streams IV_1, IV_2, \dots, IV_t are plaintext independent.
- Error propagation:** An error in a ciphertext bit affects only that bit of the plaintext.

Advantages

- The key stream can be pre-computed.
- No padding is required.
- A bit error affects only one bit.

Disadvantages

- Key stream computation cannot be parallelized.
- Reusing of a key for an IV is dangerous.
- Bit-flipping attacks are easy.

Counter Mode (CTR)

Counter mode is quite similar to the OFB mode, except that the difference is in terms of the generation of the key stream. A counter is used and in each case it is incremented by 1. For encryption in the first block, consider an IV ,

apply the encryption function, and obtain the corresponding output. This is *XORed* with the plaintext block. In the next mode the counter is incremented by 1. Again apply the encryption function, obtain the ciphertext and *XOR* that with the next plaintext block.

Encryption: The encryption process takes the input as k -bit key K , plaintext message $p = p_0, p_1, \dots, p_t$, and the initial vector $IV + 1, IV + 2, \dots$

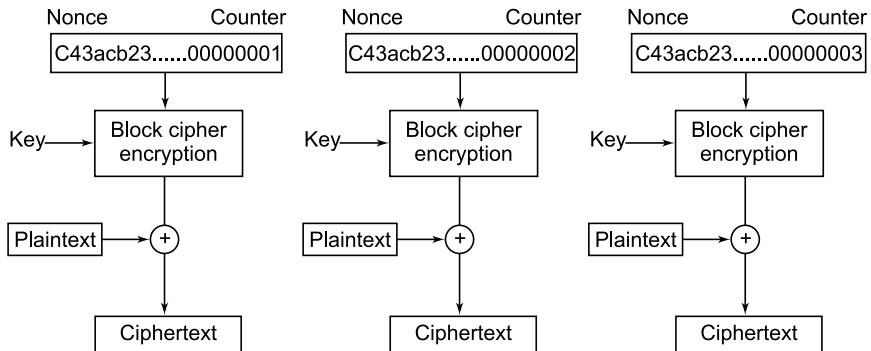


FIGURE 3.28 Counter (CTR) mode encryption.

Algorithm: The encryption process to obtain the ciphertext is:

$$\begin{aligned} C_j &= P_j \oplus E_k (IV + 1) \\ C_0 &= P_0 \oplus E_k (IV) \\ C_1 &= P_1 \oplus E_k (IV + 1) \\ C_2 &= P_2 \oplus E_k (IV + 2) \end{aligned}$$

Decryption: The input for the decryption process is: k -bit key K , ciphertext $C = C_0, C_1, \dots, C_t$, and initiation vector, $IV + 1, IV + 2, \dots$

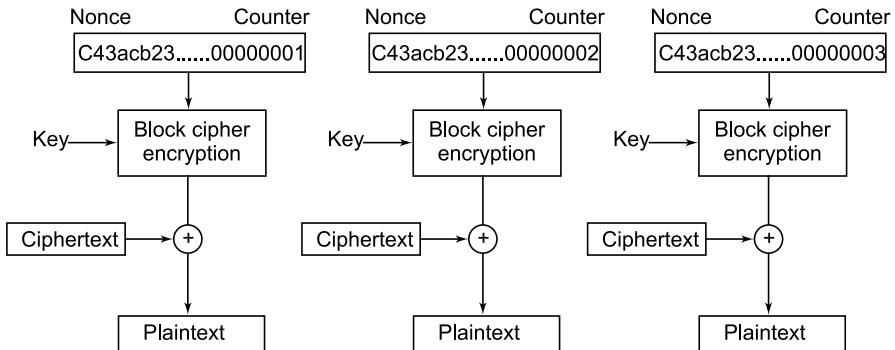


FIGURE 3.29 Counter (CTR) mode decryption.

Algorithm: The decryption process which results in the plaintext is:

$$\begin{aligned} p_j &= C_j \oplus E_k(IV_j + 1) \\ p_0 &= C_0 \oplus E_k(IV) \\ p_1 &= C_1 \oplus E_k(IV + 1) \\ p_2 &= C_2 \oplus E_k(IV + 2) \end{aligned}$$

Advantages

1. Encryption or decryption of each block could be parallelized.
2. No padding is required.
3. The key stream can be pre-computed.

Disadvantages

1. Bit-flipping attacks are easy.
2. Reusing of a key and nonce is dangerous.

SUMMARY

- A cryptographic algorithm or cipher is the mathematical function used to encrypt or decrypt a message. Cryptography is the process of dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of message.
- A Caesar cipher is a widely known algorithm of symmetric key encryption. In symmetric key encryption a simple key is used for both the encryption and decryption process. Symmetric key ciphers are classified as substitution ciphers and transposition ciphers.
- Substitution ciphers replace a character in the plaintext with another. It is categorized as mono-alphabetic ciphers and poly-alphabetic ciphers.
- In mono-alphabetic ciphers a character in the plaintext is always changed to the same character in the ciphertext regardless of its position in the plaintext. Different types of mono-alphabetic ciphers are: additive ciphers, shift ciphers, multiplicative ciphers, and affine ciphers.
- In poly-alphabetic ciphers each occurrence of a character in plaintext may have a different substitute in the ciphertext. The relationship between characters in a plaintext may have a different substitute in the

ciphertext. The relationship between a character in a plaintext to a character in the ciphertext is one-to-many. Different types of poly-alphabetic ciphers are: auto-key ciphers, Playfair ciphers, Vigenère ciphers and Hill ciphers.

- A transposition cipher rearranges all the characters in the plaintext to form the ciphertext. A character in the first position of a plaintext may appear in the tenth position of the ciphertext and a character in the sixth position in the plaintext may appear in the first position of the ciphertext. Transposition ciphers are classified as keyless transposition ciphers and keyed transposition ciphers. A combination of both keyless and keyed transposition ciphers produces better scrambling of the characters.
- In a stream cipher, encryption and decryption are typically done character by character. A stream cipher is a symmetric cipher which operates with time varying transposition on individual plaintext digits. It can encrypt plaintext messages of variable length.
- A block cipher is a symmetric key cipher, operating on fixed length group of bits called blocks. The encryption algorithm might take a 128 bit block of plaintext as input and output a corresponding 128 bit ciphertext. The security level of a block cipher degrades because all these blocks are encrypted with the same key.
- To overcome security issues, different modes of operation are used to make encryption probabilistic. The commonly used block cipher modes are: electronic code block (ECB), cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and counter (CTR) mode encryption.

REVIEW QUESTIONS

1. Discuss the difference between mono-alphabetic and poly-alphabetic ciphers.
2. Explain the block cipher.
3. Explain the classical encryption technique with a symmetrical cipher model.
4. Define diffusion.
5. Explain the Caesar cipher and mono-alphabetic cipher.

6. Why has modular arithmetic been used in cryptography?
7. How are poly-alphabetic ciphers implemented and how are they superior to mono-alphabetic ciphers?
8. Encrypt the message “SECURED NETWORK” using a Hill cipher

with the key
$$\begin{vmatrix} 2 & 1 & 4 \\ 3 & 0 & 3 \\ 1 & 3 & 4 \end{vmatrix}$$

9. Convert the given text “SECURED NETWORK” into ciphertext using the rail fence technique.
10. Describe the linear feedback shift register sequence and finite fields with their application in cryptography.
11. A block cipher operates on blocks of fixed length, often 64 or 128 bits. How does output feedback (OFB) mode make a block cipher into a synchronous stream cipher?
12. Construct a Playfair matrix with the key LARGEST, and encrypt this message: MEET ME AT THE TOGA PARTY
13. Explain about the Hill cipher. Consider the plaintext “paymoremoney” and use the encryption key:

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \text{ Find the ciphertext.}$$

14. What are the different transposition techniques? Explain.
15. Use the Caesar cipher with key =15 to encrypt the message “Hello”.
16. What is differential cryptanalysis?
17. What are transposition ciphers?
18. How do you convert a block cipher into a stream cipher by using the cipher feedback (CFB) mode? Explain.

MULTIPLE CHOICE QUESTIONS

- 8.** In cryptography the order of the letters in a message is rearranged by:

 - (a) Transposition cipher
 - (b) Substitution cipher
 - (c) Both (a) and (b)
 - (d) None of the above
- 9.** Which is the principle of encryption using a key?

 - (a) The key indicates which function is used for encryption. Thereby it is more difficult to decrypt an intercepted message as the function is unknown.
 - (b) The key contains the secret function for encryption including parameters. Only a password can activate the key.
 - (c) All functions are public; only the key is secret. It contains the parameters used for the encryption and decryption.
 - (d) The key prevents the user from having to reinstall the software at each change in technology or in the functions for encryption.
- 10.** In cryptography what is the cipher?

 - (a) Algorithm for performing encryption and decryption
 - (b) Encrypted message
 - (c) Both (a) and (b)
 - (d) None of the above
- 11.** Which is the important disadvantage of symmetric key encryption?

 - (a) More complex and therefore more time consuming calculation.
 - (b) Problem of the secure transmission of the secret key.
 - (c) Less secure encryption function.
 - (d) Practically not viable.

CHAPTER 4

MODERN BLOCK CIPHERS

4.1 INTRODUCTION

The modern block cipher is one of the most widely used types of cryptographic algorithms. It provides confidentiality by using DES and AES algorithms. These algorithms are designed to encrypt or decrypt data blocks of a fixed size. Most practical examples have data blocks of fewer than 64 bits or greater than 128 bits. A modern block cipher encrypts an n -bit block of plaintext or decrypts an n -bit block of ciphertext. The encryption or decryption algorithm uses a K bit key. A block cipher uses a Feistel structure. A Feistel cipher is a symmetric structure used in the construction of block cipher. A Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Feistel constructions are iterative in nature, which makes the cryptosystem in hardware easier.

4.2 GENERAL IDEA OF MODERN BLOCK CIPHERS

Systematic key-based modern block ciphers are a general category of ciphers that are sort of a combination of substitution and transposition ciphers. It performs the encryption of an n -bit block of plaintext or decrypts an n -bit block of ciphertext. In both cases the algorithm uses a key of length K -bits. The decryption algorithm must be the inverse of the encryption process as in Figures 4.1 (a) and (b).

The basic principal of block ciphers is encryption of blocks of data bits. A block is a fixed-length series of bits. If the number of bits in this message

is less than n -bits, padding must be added to make it an n -bit block. Also if the message contains more than n -bits, it should be divided into n -bit blocks, and padding must be added only to the last block if necessary. The common values of n are 64, 128, 256, or 512 bits.

The basic cipher is the result of a pair of functions (E, E^{-1}) , where E (the encryption function) takes a block B and a key K , and generates a new block $B^1 = E(K, B)$, which is the encrypted form of the block; and E^{-1} (the decryption function) takes a key and an encrypted block, and returns the original plaintext block: $B = E^{-1}(K, B^1)$.

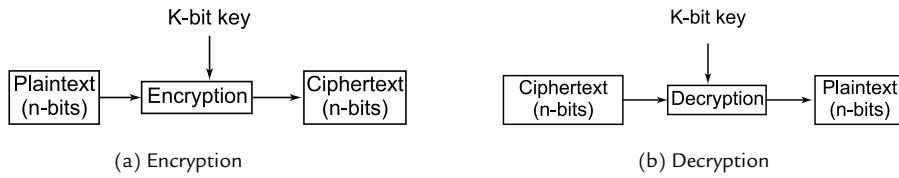


FIGURE 4.1 Modern block cipher.

The ideal block cipher will have the following properties:

1. For most blocks B and keys K , $E(K, B)$ is not very similar to the block B .
2. For most blocks B and C which differ by one bit, $E(K, B)$ and $E(K, C)$ will differ by much more than one bit in the ciphertext.
3. Given E and K , for any block B , it's easy to compute $E(K, B)$ and $E^{-1}(K, B)$.
4. Given E and an encrypted text B , it's hard to compute either the encryption key or the plaintext block.

Example 4.1: A plaintext message bit has 132 characters; how many padding bits must be added to it? The message is encoded by 8 bit ASCII and the block cipher accepts a block length of 64 bits.

Solution: 132 characters encoded using 8 bit ASCII results:

$$132 \times 8 = 1056 \text{ bits}$$

$$\text{Plaintext} = 1056 \text{ bits}$$

The plaintext is divided by 64 bits.

The length of the message and the length of padding is $|m|$ and $|Pad|$.

$$\begin{aligned} &= |m| + |Pad| \\ |Pad| &= 1056 \bmod 64 \rightarrow 32 \bmod 64 \end{aligned}$$

i.e., the last block of the message contains 32 bits of padding (containing 0s).

$$\begin{aligned}\text{Total length of the plaintext} &= 1056 + 32 \\ &= 1088 \text{ bits}\end{aligned}$$

This forms 17 blocks of 64 bits each ($17 \times 64 = 1088$ bits)

Using the encryption algorithm 17 blocks of ciphertext are produced.

4.3 TYPES OF MODERN BLOCK CIPHERS

Modern block ciphers can be designed as substitution ciphers and transposition ciphers.

Substitution Cipher: In a substitution cipher a bit 1 or 0 in the plaintext can be replaced by either 0 or 1. Once a plaintext is encrypted using the substitution cipher, the number of 1s are different in the plaintext and ciphertext. In a substitution cipher, for the number of n -bits of plaintext we can have 2^n possible combinations of ciphertext, because each of the n -bits in the block can have one of the two values 0 or 1.

Example 4.2: In a block cipher where $n = 64$, if there are ten 1s in the ciphertext, how many trial-and-error tests does the attacker need to do to recover the plaintext from the intercepted ciphertext using the substitution cipher method?

Solution: In this encryption process the attacker has no idea how many 1s are in the plaintext. He has to try all possible 2^{64} combinations of 64 bit blocks to find one that makes sense. That is, the attacker could try 1 billion blocks per second, and it would still take hundreds of years on average before he could be successful.

Transposition Cipher: A transposition cipher transposes the bits in the plaintext to create the required ciphertext, that is, the bits are only reordered (transposed). The number of 1s or 0s in the plaintext and in the ciphertext are the same. In this case also the number of n -bit possible plaintexts or ciphertexts is 2^n , because each of the n bits in the blocks can have one of the two values 0 or 1.

Example 4.3: There is a block cipher with $n = 64$ and there are ten 1s in the ciphertext. How many trial-and-error tests does an attacker need to do to recover the plaintext from the intercepted ciphertext if the ciphertext is designed as a transposition cipher?

Solution: Using a transposition cipher the total number of 1s and 0s in the ciphertext and the plaintext are the same. The attacker knows that there are exactly ten 1s in the plaintext.

Using an exhaustive search an attacker can make an attempt to recover the plaintext. He considers those 64 bit blocks that have exactly ten 1s. The possible combinations he has to try by trial-and-error test is:

$$\frac{64!}{[(10!)(54!)])} = 151,473,214,816$$

From the given ciphertext block of length 64 bits, those have exactly ten 1s, the attacker can test all of them in less than 3 minutes if he can do 1 billion tests per second. Compared to the transposition cipher, the substitution cipher is more resistant to exhaustive search attacks. Hence, a modern block cipher needs to be designed as a substitution cipher.

Full-size Key Cipher: In this type the key is long enough to choose every possible mapping from the input to the output. However, this is not used in practice.

Full-size Key Transposition Block Cipher: In a full size key transposition block cipher, the number of bits in the plaintext and ciphertext are the same. For example, if the plaintext has n bits the ciphertext can be obtained by scrambling the n bits into a set of $n!$ possible combinations.

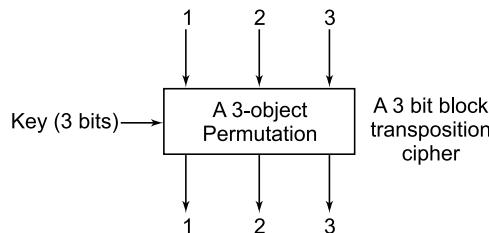


FIGURE 4.2 Three bit transposition cipher.

Example 4.4: Prepare a set of permutations table for a 3 bit block transposition cipher. Take the block size of 3 bits.

Although a 3 bit key can select $2^3 = 8$ different combinations, only the $n!$ combinations are given in the table. If $n = 3$, the key size $\log_2 6 = 3$.

The set of permutations table with $3! = 6$ elements [$3 \times 2 \times 1 = 6$].

The six combinations are:

$$\begin{bmatrix} (1 & 2 & 3) \\ (2 & 1 & 3) \\ (3 & 2 & 1) \\ (1 & 3 & 2) \\ (2 & 3 & 1) \\ (3 & 1 & 2) \end{bmatrix}$$

Full-size Key Substitution Block Cipher: A full-size key substitution block cipher substitutes the plaintext bits. For the substitution process it performs decoding at the input and encoding at the output.

Decoding Operation: In the decoding operation the n bit integer is transformed into a 2^n bit string. For example, if the number of bits $n = 3$, on decoding it is transformed into $2^n = 2^3 = 8$ bits. Out of these 8 bits there is only a single 1 bit and $(2^n - 1)$ 0 bits.

The three bit input plaintext can be an integer between 0 – 7, and it is decoded as an 8 bit string with a single 1. The decoded 8 bit string can be scrambled into $8!$ possible combinations.

It is $8! = 40,320$. The size of the key is $\log_2 40,320 = 16$ bits.

Using 16 bit key we can generate as many as 65,536 different combinations. Out of this only 40,320 combinations are used.

Encoding Operation: An encoding operation is performed in the output, which is the reverse process of a decoding operation. In an encoding operation the cipher has to perform $2^{n!}$ permutations, where n is the key size.

Partial-Size Key Ciphers: The implementation of a full-size key may not make any sense to build the complete key schedule. In the encryption rounds of a block cipher the key size is reduced. For example, in the Data Encryption Standard (DES) algorithm, the key size is effectively reduced from 64 bits to 56 bits. Using a partial size key of 56 bits performs only a 2^{56} mapping out of $2^{n!}$ objects. For a full size key of DES, $n = 64$ bits.

Integer	Binary 2^n Combination	Decoding
0	000	00000001
1	001	00000010
2	010	00000100
3	011	00001000
4	100	00010000
5	101	00100000
6	110	01000000
7	111	10000000

4.3.1 Keyless Cipher

A keyless cipher does not substitute one symbol for another; instead it changes the location of the symbol. In practice a keyless cipher cannot be used independently. It is used in combination with a keyed cipher. Keyless ciphers are classified as *keyless transposition ciphers* and *keyless substitution ciphers*.

Keyless Transposition Cipher: A keyless transposition cipher does not substitute one symbol for another; instead, it changes the location of the symbols. This transposition may be considered as a pre-wired hardware implementation. For an n bit input the fixed key can result $n!$ output permutations.

Keyless Substitution Cipher: In a keyless substitution cipher, the mapping between the input to the output is performed based on a pre-defined mathematical function.

4.4 COMPONENTS OF MODERN BLOCK CIPHERS

A modern block cipher is made of a combination of transposition units for diffusion (called D-Boxes), substitution units (called S-Boxes), and also other units.

4.4.1 D-Boxes (Diffusion Boxes)

A D-box or diffusion box is a transposition cipher. Figure 4.3 shows the transposition of bits from input to the output. The D-boxes are the keyless cipher where the mapping between the input and output are defined.

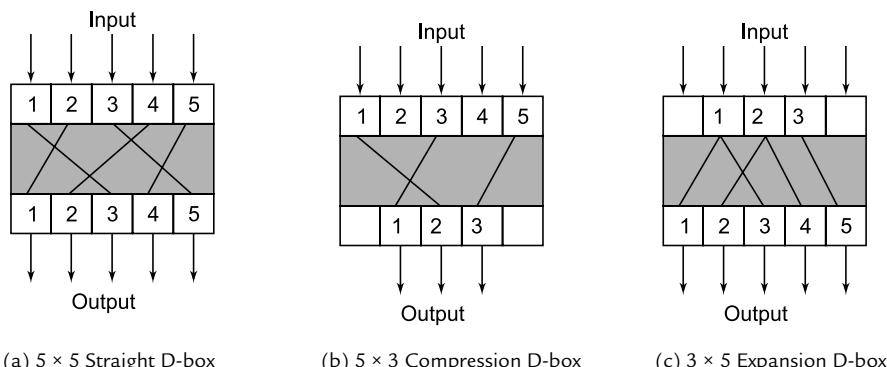


FIGURE 4.3 Three types of D-boxes.

D-box ciphers are classified as:

1. Straight D-boxes
2. Compression D-boxes
3. Expansion D-boxes

Straight D-boxes: They have n -input and n -output permutations. This forms $n!$ possible mappings. For example, a 3×3 D-box has $3 \times 2 \times 1 = 6$ possible mappings as in Figure 4.4.

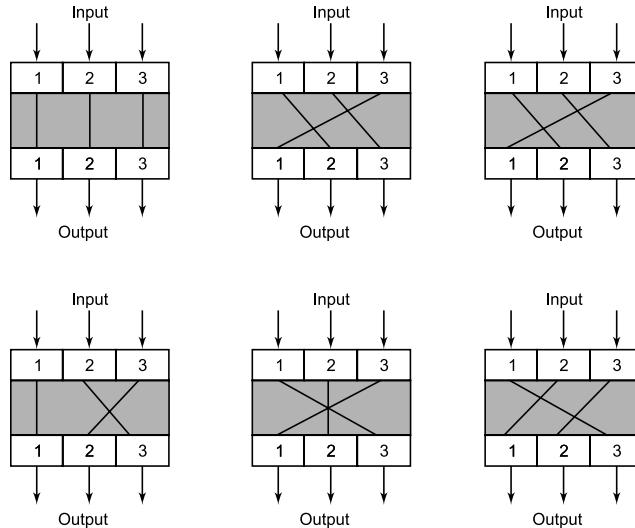


FIGURE 4.4 A 3×3 D-box.

Example 4.5: Design a 5×5 permutation table of a D-box that moves the two middle bits (3 and 4) in the input word to the two ends (1 and 5) in the input word. Relative positions of other bits should not be changed. The resulting D-box with the table [3 1 2 5 4] is shown in Figure 4.5.

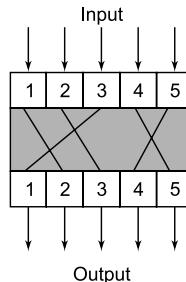


FIGURE 4.5 A 5×5 D-box.

Compression D-Boxes: A compression D-box is a keyless cipher. In a compression D-box the number of output bits is always less than the number of input bits ($m > n$). This can be achieved by blocking some of the input bits so they do not reach the output.

Example 4.6: Consider a compression D-box with $m = 16$ and $n = 12$. Table 4.1 indicates the output of a 16×12 compression D-box, in which input bits 02, 06, 13, and 15 are blocked. The table has m -entries, but the content of each entry is from 1 to n with the same mission rules.

TABLE 4.1 Example of a 16×12 D-box.

01	03	11	07	04	12
16	05	08	14	06	09

Expansion D-boxes: An expansion D-box has more output bits than input bits. If the number of input bits is m and the number of output bits is n , then $m < n$. This is achieved by connecting the same input to more than one output. The expansion D-boxes are also keyless ciphers.

Example 4.7: Table 4.2 indicates the rules for transposing bits in an expansion cipher. Using expansion D-boxes, the table has m entries, but $m - n$ of the entries are repeated. The table is for a 12×16 expansion D-box in which the input bits 2, 6, 9, and 11 are mapped to two outputs.

The number of inputs = 12; the number of outputs = 16.

TABLE 4.2 Example of a 12×16 D-box.

01	09	11	07	06	04	12	02
06	02	05	08	10	09	11	03

Invertibility: In invertibility a straight D-box is used in the encryption cipher. The decryption process is the reverse process of this. The mapping permutation is defined by a straight D-box. Thus, it is also known as a P-box. Figure 4.6 shows the inverse of a permutation table represented as one-dimensional elements.

Step 1: Original Table

3	5	2	1	4
---	---	---	---	---

Step 2: Add indices to the original Table

3	5	2	1	4
1	2	3	4	5

Step 3: Map the contents of original Table with indices

1	2	3	4	5
3	5	2	1	4

Step 4: Rearrange based on the indices

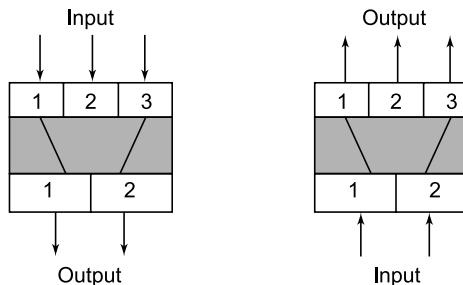
4	3	1	5	2
1	2	3	4	5

Step 5: Inverted table

4	3	1	5	2
---	---	---	---	---

FIGURE 4.6 Inverting a permutation table.

The compression and expansion D-boxes have no inverses, as in Figure 4.7. In case of a compression D-box, we can drop an input during encryption. The decryption algorithm does not have an indication of how to replace the dropped bit.

**FIGURE 4.7** D-box.

In an expansion D-box, an input may be mapped to more than one output during encryption. The decryption algorithm does not have indications which of the several inputs are mapped to an output. In Figure 4.8, it is clear that a compression D-box is not the inverse of an expansion D-box or

vice versa. This means that if we use a compression D-box in an encryption cipher, we cannot use an expansion D-box in the decryption cipher or vice versa.

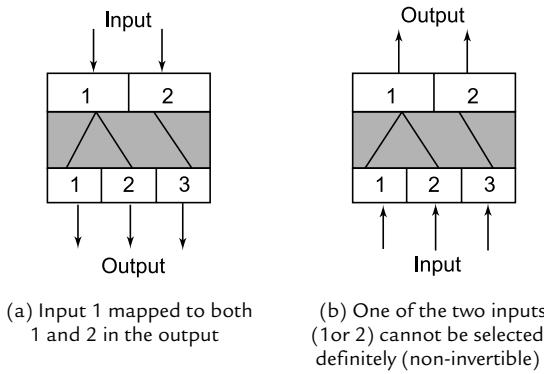


FIGURE 4.8 Expansion D-box (they are non-invertible).

4.4.2 S-Boxes: (Substitution Boxes)

The security of data relies on the substitution process. Substitution is a non-linear transformation which performs a confusion of bits. It provides cryptosystems with the confusion property described by Shannon. He suggested strong ciphers could be built by combining substitution with transposition repeatedly. In a modern encryption algorithm a nonlinear transformation is essential and is provided to be a strong cryptographic primitive against linear and differential cryptanalysis. S-box mapping is shown in Figure 4.9.

S-box mapping is represented with input x and output y . In order to detect the linear cryptanalysis of the S-box, the linear approximation should be examined to determine the probability bias of the S-box.

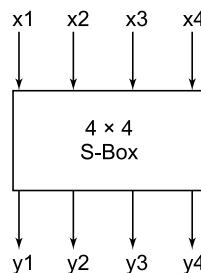


FIGURE 4.9 S-box mapping.

S-box Properties: The properties of S-boxes have been widely used as a base of new encryption strategies such as nonlinearity, differential uniformity, and the strict avalanche criterion. An (x, y) S-box is a map,

$$S : \{0, 1\}^x \rightarrow \{0, 1\}^y$$

It comprises n -variable component Boolean functions indicated as follows:

$$\begin{aligned} y_1 &= f_1(x_1, \dots, x_n) \\ y_2 &= f_2(x_1, \dots, x_n) \\ &\vdots \\ y_n &= f_n(x_1, \dots, x_n) \end{aligned}$$

Each of them needs to satisfy S-box properties. The following are a list of several properties in S-box.

Robustness: Let $F = (f_1, f_2, \dots, f_n)$ be an $n \times n$ S-box, where f_i is a component function of S-box mapping.

$$\begin{aligned} f_i : \{0, 1\}^n &\rightarrow \{0, 1\} \\ R &= \left[1 - \frac{N}{2^n} \right] \left[1 - \frac{L}{2^n} \right] \end{aligned}$$

F must be robust against differential cryptanalysis.

Balancing: $S : \{0, 1\}^n \rightarrow \{0, 1\}^m$ balanced, if $(f) = 2^{n-1}$. The significance of the balance property is that the higher the magnitude of the function imbalance, the higher the probability that linear approximation is obtained.

Strict Avalanche Criterion (SAC): A change in one bit of the input bits of an S-box should produce a change in half of the output bits of the S-box. It is harder to perform an analysis of ciphertext when trying to come up with an attack. A cryptographic function which satisfies the previous condition is said to have satisfied the *Strict Avalanche Criteria*.

Nonlinearity: $S : \{0, 1\}^x \rightarrow \{0, 1\}^y$ is defined as the least value of nonlinearity of all nonzero linear combinations of x Boolean functions $f_i : \{0, 1\} \rightarrow \{0, 1\}$, $i = x-1, \dots, 1, 0$. The nonlinearity of an S-box must be high to resist against linear cryptanalysis.

Differential Uniformity: The smaller the differential uniformity, the better the S-box's resistance against differential cryptanalysis.

Linear Approximation: The lower the linear approximation value, the better the S-box's resistance against linear cryptanalysis.

Algebraic Complexity: Algebraic complexity is important to resist against interpolation attacks and other algebraic attacks.

Fixed (F_p) and Opposite Fixed Points (OF_p): The number of F_p and OF_p should be kept as low as possible to avoid leakage in any statistic cryptanalysis.

Bit independence criterion: Bit independence is a highly desirable property, as with increasing independence between bits, it becomes more difficult to understand and predict the design of the system.

4.4.3 Exclusive OR (XOR)

XOR is an important operation in most of the block ciphers. The XOR operation has the following important properties.

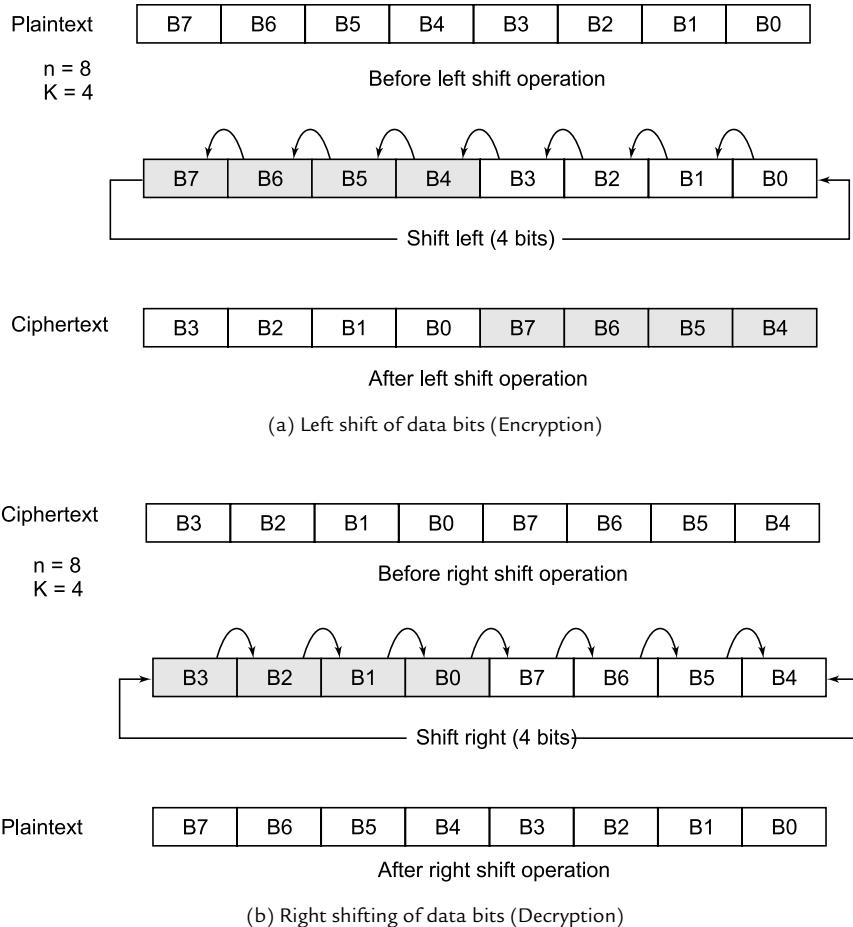
XOR Properties

- | | |
|---------------------------------|---------------------------------------------------------------------------|
| 1. <i>Closure</i> | XOR operation between two n -bit words results in another n -bit word |
| 2. <i>Associativity</i> | $A \oplus (B \oplus C) = (A \oplus B) \oplus C$ |
| 3. <i>Commutative</i> | $A \oplus B = B \oplus A$ |
| 4. <i>Existence of identity</i> | $A \oplus 0 = A$ |
| 5. <i>Existence inverse</i> | $A \oplus A = 0$ |

4.5 CIRCULAR SHIFTING

In a circular shifting operation, bits in an n -bit word are shifted either left or right. There are two types of circular shift operation, shift left and shift right. Encryption of the plaintext message is performed by the shift left operation and decryption by the shift right operation.

- In a shift left operation each bit moves one position left. Depending on the key size (K) the lower order bits are replaced (rightmost bits) by the left most bits as shown in Figure 4.10 (a).
- The shift right moves each bit to the right by one position. Based on the key size, leftmost bits are replaced by the rightmost bits as in Figure 4.10 (b).

**FIGURE 4.10** Circular shifting of 8 bit words.

Swap Operation

A swap operation is similar to a circular shift operation. The key size used for a swap operation is $K = \frac{n}{2}$, where n is the number of bits in the plaintext message. For a swap operation the plaintext message is divided into two equal-length parts, hence it is valid only if n is an even number. The encryption and decryption process using the swap operation is shown in Figure 4.11.

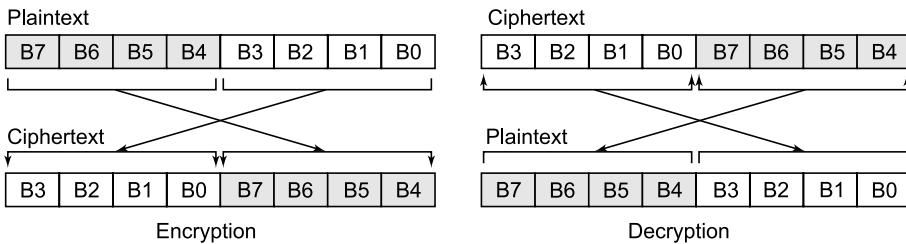


FIGURE 4.11 Swap operations on an 8 bit word.

Split and Combine Operation

In this method of encryption the plaintext message block of length n bits is split into two parts (I and II) to create two equal-length words as shown in Figure 4.12(a). The decryption is performed by combining them as in Figure 4.12(b).

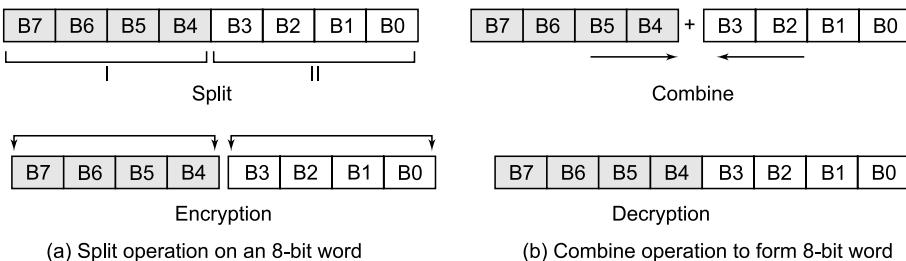


FIGURE 4.12 Split and combine operations on an 8 bit word.

4.6 PRODUCT CIPHERS

A product cipher is a composite of two or more elementary ciphers with a goal of producing a cipher that is more secure compared to ciphers with only individual components. It is a combination of substitution, permutation, and other components in which the block can be partitioned into smaller blocks for substitution and recombined with permutations. An initial block cipher is a block cipher involving the repetition of an internal round function, which may involve a key as another input. Each of the sequential steps is called a round. Shannon introduced the concept of a product cipher to enable the block cipher to have two important properties, *diffusion* and *confusion*.

Diffusion: The purpose of diffusion is to hide the relationship between the ciphertext and the plaintext. This process will frustrate hackers who use ciphertext statistics to find the plaintext. With diffusion, each character in

the ciphertext is dependent on some or all symbols in the plaintext. Using this method, if a single symbol in the plaintext is changed, it also changes several or all symbols in the ciphertext.

Confusion: Confusion is to mask the relationship between the ciphertext and the key. This process will frustrate hackers who try to use the ciphertext to find the key. In this if a single bit in the key is changed, most or all bits in the ciphertext will also be changed.

Rounds: The principle of confusion and diffusion are the most essential concepts in the design of the modern block cipher; they defend against statistical attacks. It can be achieved by using an iterated product cipher. These iterations are called rounds. Each iteration includes S-box operations, D-box operations, and other components. For an N round cipher, N iterations are completed to convert a plaintext into ciphertext (encryption). N iterations are also completed for the decryption process to convert ciphertext into plaintext. A separate key is generated for each round by a key generator. The text generated in the intermediate stage of the encryption and decryption process is called middle text. A two-round product cipher is shown in Figure 4.13. It includes the following three steps in each round:

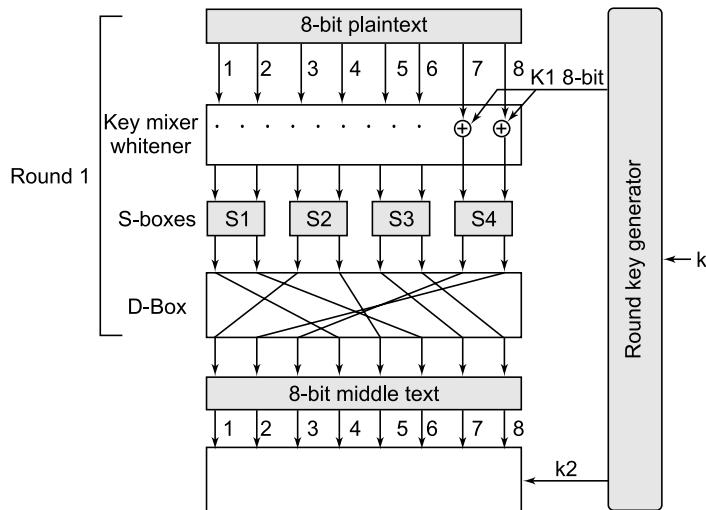


FIGURE 4.13 A two-round product cipher.

Step 1: In this step a whitening or key mixing operation is performed. It is the technique of XORing key materials in the rounds. It is shown that the whitener operations substantially increase the difficulty of a key search attack against the remainder of the cipher. This increases

the difficulty of attacking the cipher by hiding from an attacker the specific input to the rounds' F functions. Using XOR the 8-bit plaintext is mixed with 8-bit key.

Step 2: The output of the whitener is divided into four 2 bit groups. These 2 bit groups will now act as input to the S-boxes. The S-box operation performs the required transformation of these bits.

Step 3: The output of the S-box is taken as input for the straight D-box operation, which transposes these bits. It creates a different set of inputs for the next round.

Diffusion Operation

Figure 4.13 shows diffusion in a two-round product cipher using S-boxes and D-boxes. The operations in different rounds are explained as follows:

In round 1, bit 1 after being XORed with the corresponding bits of round key K_1 changes the two bits (bits 1 and 2) through the S-box 1; bit 1 is then permuted in the P-box and becomes bit 4; bit 2 is permuted and becomes bit 6; that is by the end of round 1, bit 1 has affected bits 4 and 6. In round 2 the same operation is repeated with a different key K_2 and a different substitution and permutation set. In the decryption process each bit in the ciphertext is affected by several bits in the plaintext.

Confusion Operation

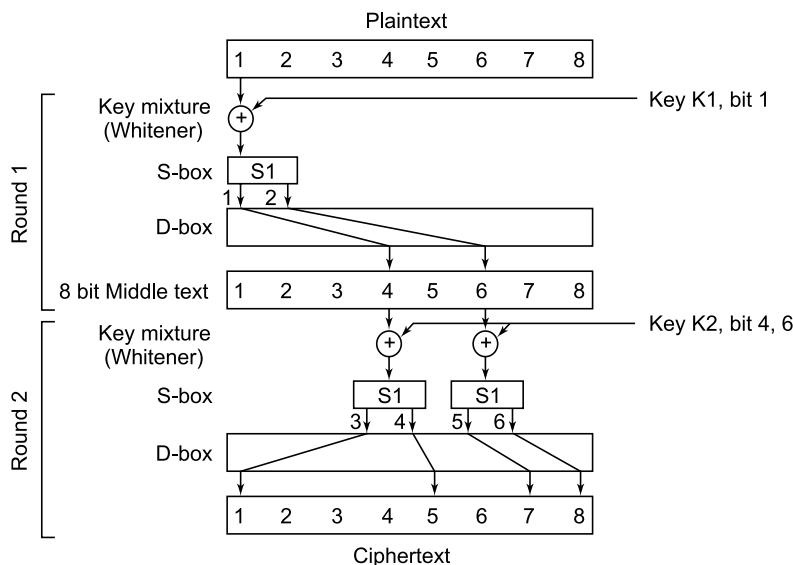


FIGURE 4.14 Confusion property using a product cipher.

The implementation of the confusion property using a product cipher is shown in Figure 4.14. Consider the four 1, 5, 7, and 8 bits in the ciphertext. These bits are affected by bit 1 in K_1 , and bit 4 and 6 in K_2 . This shows that each bit in each round key affects several bits in the ciphertext; it is also hard for an adversary to find the relationship between the ciphertext and the key. In a practical cipher a combination of larger data blocks, more S-boxes, and more rounds are used to improve the effectiveness of diffusion and confusion properties. This enables a better cipher and can better hide the relationship between the ciphertext and the key.

4.7 FEISTEL AND NON-FEISTEL CIPHERS

The modern block cipher uses the concepts of product ciphers. They are divided into two categories. The first category of product cipher uses both invertible and non-invertible components. These ciphers are referred to as Feistel ciphers. Data Encryption Standard (DES) is the best example of a Feistel cipher. Ciphers in the second category use only invertible components. These ciphers are non-Feistel ciphers. Advanced Encryption Standard (AES) is the best example of a non-Feistel cipher.

4.7.1 Feistel Ciphers

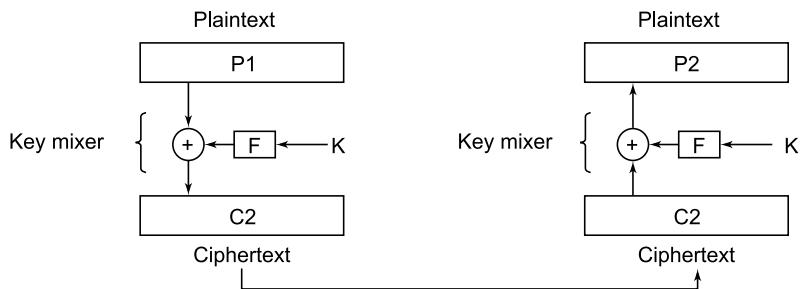


FIGURE 4.15 Feistel cipher.

A Feistel cipher is a block cipher having three components: self-invertible, invertible, and non-invertible. For the encryption and decryption process, a Feistel cipher combines all non-invertible elements in a unit and uses the same unit. The block diagram representation of a Feistel cipher is shown in Figure 4.15.

Encryption

$$\begin{aligned}
 C_1 &= P_1 \oplus F \\
 &= P_1 \oplus f(k) \\
 &= C_1 \oplus f(k) \\
 &= P_1 \oplus f(k) \oplus f(k) \\
 P_2 &= P_1 \oplus (0) \\
 \therefore P_2 &= P_1
 \end{aligned}$$

Decryption

$$\begin{aligned}
 P_2 &= C_2 \oplus F \\
 &= C_2 \oplus f(k) \\
 \mid \therefore C_1 &= C_2
 \end{aligned}$$

In the encryption process F is a non-invertible function ($F = f(K)$). It accepts the key K as another input. The output of this function F , is XORed with the plaintext. This results in the ciphertext. The combination of the XOR operation with the plaintext and $f(K)$ is called a *mixer*. In the Feistel cipher the effect on the non-invertible component in the encryption and decryption algorithms is done by using an XOR operation. Both encryption and decryption are the inverse of each other as shown in the expression. The same key K is used for the encryption and decryption algorithms.

An improved version of the Feistel cipher is shown in Figure 4.16. In this the input to function F is also part of the plaintext during encryption and part of the ciphertext in decryption. The key K will be the second input to the function.

The plaintext and the ciphertext are divided into two equal-length blocks (L_1, R_1 and L_2, R_2) where L and R represent the left and right blocks of the plaintext respectively. The right block of the plaintext is taken as the input for the function F along with the key. The left block is taken as the input for the XOR operation with the output of the function F as another input. The input for the function in both the encryption and the decryption process must be the same. From the encryption and decryption process it is clear that the algorithms are still the inverse of each other.

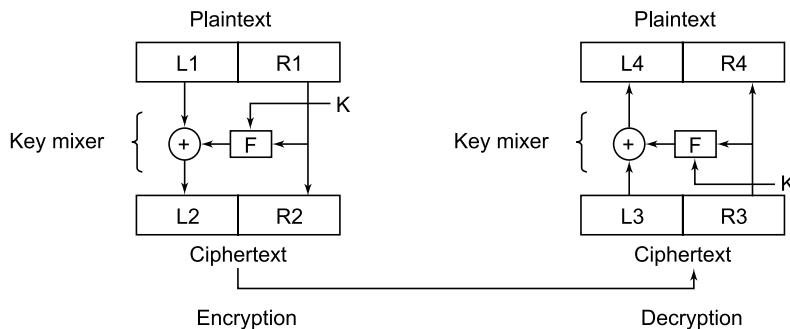


FIGURE 4.16 Modified version of the Feistel cipher design.

In Figure 4.16 $R_3 = R_2$ and $L_3 = L_2$ Also $R_1 = R_2 = R_3 = R_4$

$$\therefore R_4 = R_1$$

$$\begin{aligned} L_4 &= L_3 \oplus F \\ &= L_3 \oplus f(R_3, K) \\ &= L_2 \oplus f(R_2, K) \\ &= L_1 \oplus F \oplus f(R_1, K) \\ &= L_1 \oplus f(R_1, K) \oplus f(R_1, K) \\ \therefore L_4 &= L_1 \end{aligned}$$

The decryption algorithm correctly regenerates the plaintext.

One of the main limitations of this design is the right half of the plaintext remains the same. An intruder can directly find the right half of the ciphertext as the right half of the plaintext block. This limitation is overcome by the improved design of the Feistel cipher. It has more rounds, and a new element *swapper* is added to each round. This allows swapping the left and right halves in each round. The improved version of a practical Feistel cipher with multiple rounds is as shown in Figure 4.17. The function F of a practical Feistel cipher is defined as follows. It operates in N rounds. In the i^{th} round the following operations are executed:

1. The plaintext block is split into two halves L_i and R_i , where $i = 1, 2, 3, \dots, N$ rounds.
2. The round function F_i is applied to the right half, yielding $F_i = f_i(R_i, K_i)$, where K_i is the round key.
3. The XOR of this value and the left half is computed, yielding $L_i \oplus R_i$.
4. The left and right side are swapped, thus yielding the round output:

$$L_i \parallel R_i = R_i \parallel L_i \oplus f_i(R_i, K_i)$$

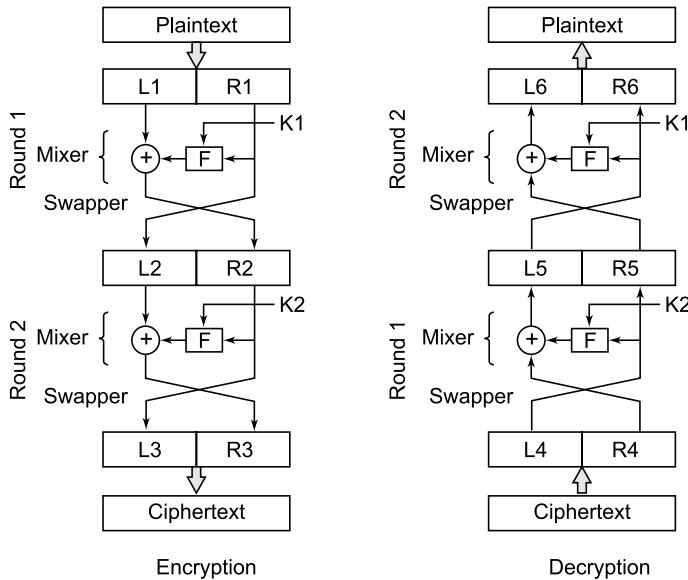


FIGURE 4.17 Feistel cipher with two rounds containing mixer and swapper.

During encryption:

$$L_{i+1} = R_i \quad \therefore L_3 = R_2$$

$$R_{i+1} = L_i \oplus f(R_i, K_i)$$

In the second round: the ciphertext is ($R_{i+1} = L_{i+1}$)

The decryption of a ciphertext (R_i, L_i) is accomplished by

$$R_{i+1} = L_i \quad R_5 = L_4 = L_3 = R_2$$

$$L_{i+1} = R_{i+1} \oplus f(L_i, K_i)$$

$$L_6 = R_5 + f(L_5, K_1)$$

Then (L_i, R_i) is the plaintext.

4.7.2 Non-Feistel Ciphers

The second category of cipher only uses invertible components and is known as a non-Feistel cipher. In a non-Feistel cipher each element of plaintext has a corresponding element in the ciphertext. Advanced Encryption Standard (AES) uses S-boxes with an equal number of inputs and outputs and a straight P-box that is invertible.

4.8 ATTACKS ON BLOCK CIPHERS

The attacks on any traditional ciphers may also be applicable to block ciphers. Modern block ciphers can resist these attacks. For example, a brute force attack to disrupt the key is usually infeasible due to the larger key size. There are several attacks which are specific to block ciphers. The four important types of these attacks are:

- (i) Differential Cryptanalysis
- (ii) Linear Cryptanalysis
- (iii) The Exploitation of Weak Keys
- (iv) Algebraic Attack

(i) Differential Cryptanalysis

Differential cryptanalysis is a method of breaking a certain class of cryptosystem. It is efficient when the cryptanalyst can choose plaintext and obtain the ciphertext (chosen plaintext cryptanalysis). Known plaintext differential cryptanalysis is also possible; however, often the size of the known text pair is very large. In this case the cryptanalyst searches for plaintext, ciphertext pairs whose difference is constant, and investigates the differential behavior of the cryptosystem. It is applicable to the iterated cipher with a weak round function (in a Feistel cipher).

After two rounds of DES, knowing both the input and output, it is trivial to determine the two sub-keys used since the output of both F functions are known. For each S-box there are four possible inputs to produce the known output. Since each sub-key is 48 bits long, but the key is only 56 bits long, finding which of the four possibilities is true for each group of six bits in the sub-keys becomes easier.

Once the number of rounds is increased, the problem becomes much harder. However, it is still true that the output depends on the input and the key. For a limited number of rounds it is inevitable, without the need for any flows in the S-boxes, that there will be some cases where a bit or a combination bits in the output will have the same correlation with the simple combination of some input bits and some key bits. Ideally that correlation should be absolute with respect to the key bits, since there is only one key to solve for, but it can be probabilistic with respect to the input and output bits, since there need to be many pairs of tests. As the number of rounds increases, though the simple correlation disappears. Differential

cryptanalysis represents an approach to finding more subtle correlations. In a very simple sense if the input bit is 1, the output bit will be either 0 or 1. Changing this bit in the input changes that bit in the output.

(ii) Linear Cryptanalysis

Linear cryptanalysis uses linear approximations to model nonlinear steps in the encryption process. Applying the approximation to a vast amount of known plaintext will eventually recover a key bit that is correct with a certain probability. Cipher-specific refinement of this method can recover multiple key bits. In 1993, Matsei used linear cryptanalysis to attack a DES algorithm. The calculation of the key bit is easy when all correlations during the encryption process are linear. Assuming some correlations are nonlinear, one can still recover a key bit with high probability by finding good linear approximations for the nonlinear calculation steps. This is exactly the key idea behind linear cryptanalysis: build equations that linearly approximate a key bit by combining some of the plaintext and ciphertext bits. If these equations have a bias, that is they hold true for a certain probability $P = 0.5$, then this bias can be exploited to mount a known plaintext attack.

(iii) Exploitation of Weak Keys

Weak keys are keys for which the strength of the encryption function is weaker than other keys. Several attack techniques may work against weak keys. In general if the size of the key space is K bits, then the complexity of an exhaustive search is 2^K . The complexity of the weak keys is size $2f$. If the complexity of an attack against this set of weak keys is 2^w , where $2^w < 2f$, then exploiting weak keys is more efficient than an exhaustive search. Then the complexity can be expressed as 2^{K-f+w} . The attacker can compare this complexity to the complexity of the conventional attack to determine whether to exploit weak keys.

(iv) Algebraic Attack

Given a particular cipher algebraic cryptanalysis consists of two steps. First, one must convert the cipher and possibly some supplemental information into a system of polynomial equations, usually over $GF(2)$, but sometimes over other things. Second, one must solve the system of equations and obtain from the solution the secret key of the cipher.

An algebraic attack is a method of cryptanalysis against a cipher. It involves:

- a. Expressing the cipher operations as a system of equations.
- b. Substituting in known data for some of the variables.
- c. Solving for the key.

SUMMARY

- Block ciphers are the most prominent and important elements in many cryptographic systems. Individually they provide confidentiality. As a fundamental building block their versatility allows construction of pseudorandom number generators, stream ciphers, MACs, and hash functions. Modern block ciphers can be designed as a substitution cipher or a transposition cipher.
- Based on the size of the key used, transposition and substitution ciphers are categorized as full-size, partial-size, and keyless ciphers.
- Modern block ciphers are made of a combination of transposition units for diffusion units (D-boxes), substitution units (S-boxes), XORs, and also other units.
- A product cipher is a combination of two or more elementary ciphers with a goal of producing a cipher which is more secure compared to that of any cipher with only individual components.
- The product cipher that uses both invertible and non-invertible components is known as a Feistel cipher. The cipher that uses only invertible components is called a non-Feistel cipher.

REVIEW QUESTIONS

1. Differentiate between block and stream ciphers.
2. Explain operations of various transposition ciphers in detail.
3. Explain in detail about Feistel encryption with a diagram.
4. Discuss the different categories of product ciphers.
5. Show and explain Feistel encryption and decryption algorithms.
6. Briefly explain the design principles of block ciphers.

7. Discuss in detail the block cipher mode of operation.
 8. Compare invertible and non-invertible ciphers.
 9. What are the various block cipher design principles? Explain how different cryptographic algorithms use the Feistel cipher structure.
 10. Discuss the design principles of the block cipher technique.
 11. What is a Feistel cipher? Name the ciphers that follow the Feistel structure.
 12. What are the confusion and diffusion properties of modern ciphers?

MULTIPLE CHOICE QUESTIONS

CHAPTER 5

DATA ENCRYPTION STANDARD (DES)

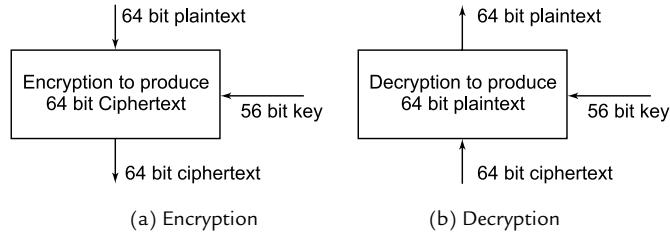
5.1 INTRODUCTION

In 1972 and 1974, the National Bureau of Standards (now the National Institute of Standards and Technology or NIST) issued the first public request for an encrypted standard. Many researchers and organizations proposed different encryption techniques. As a result of this in the United States in 1977, the data encryption standard (DES) was adopted as a standard for secret key encryption schemes. The DES is a block cipher that uses shared secret encryption. The algorithm is based on LUCIFER, designed by Horst Feistel, and was developed by IBM and the government of the United States in 1972 with an objective that everyone could use a standard security algorithm to communicate with each other.

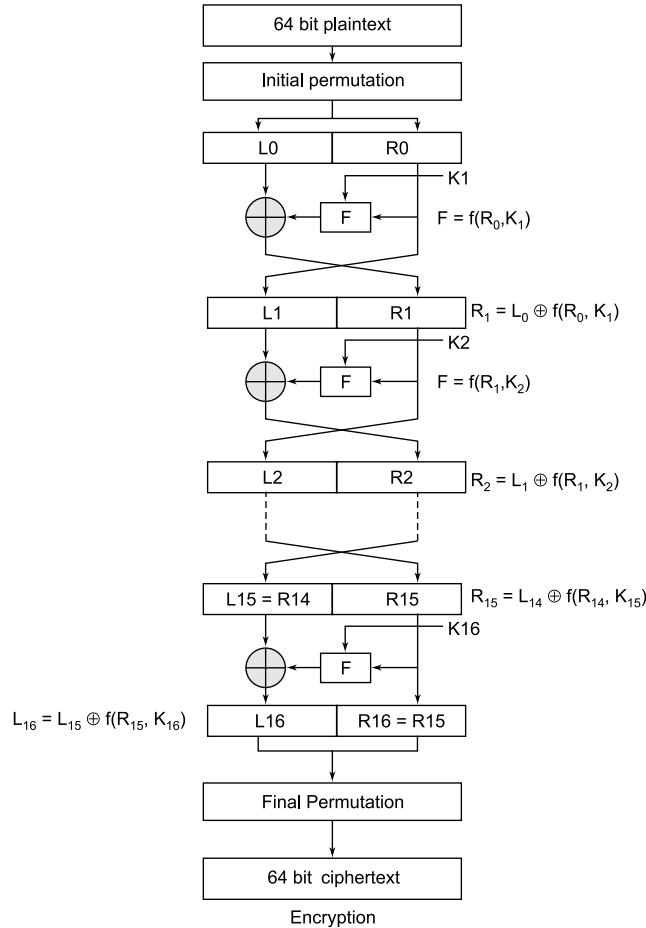
It is mainly based on a symmetric key algorithm that uses a 56 bit key. DES encrypts 64 bit blocks of plaintext at a time. The use of a 56 bit key is not accepted by many banks and commercial organizations because the key is too short. Other versions such as Triple DES (TDES or 3 DES) were developed to address this issue. This uses a longer key and is more secure but has never become popular. In response to a growing desire to replace DES, NIST announced the Advanced Encryption Standard (AES) program in 1997.

5.2 DES STRUCTURE

Figures 5.1 (a) and (b) show the schematic representation of the DES mechanism. The encryption process of the DES algorithm is shown in Figure 5.2. The encryption of a block of messages takes place in sixteen rounds. The initial permutation is a keyless permutation (P-Box).

**FIGURE 5.1** DES encryption and decryption mechanisms.

It takes 64 bit input and permutes it according to a predefined rule. From the input key provided, sixteen 48 bit sub-keys are generated. These keys are used, one for each round. Each round contains 8 S-boxes. The content of these S-Boxes has been determined by the U.S. National Security Agency (NSA).

**FIGURE 5.2** DES encryption process.

The 64 bit plaintext message block is divided into two halves (32 bits each). The right half is expanded from 32 bits to 48 bits using another fixed table as shown in Table 5.1. This 48 bit output is XORed with the 48 bit sub-key for that round. The output of this is transformed into a 32 bit block using the S-boxes. The 32 bit outputs are subsequently permuted again using another fixed table. This is by now thoroughly shuffled. The right half is combined with the left half using the XOR operation. In the next round this combination is used as the new left half.

5.2.1 DES Function

After the initial permutation the 64 bit plaintext is divided into two halves as a block of 32 bits each. These sub-blocks are passed through an encryption ladder called a *Feistel structure*. As in a Feistel cipher, each round of the DES Feistel structure contains two components, a *mixer* and a *swapper*. These components are invertible in operation, and there is a non-invertible component inside the function F. Where,

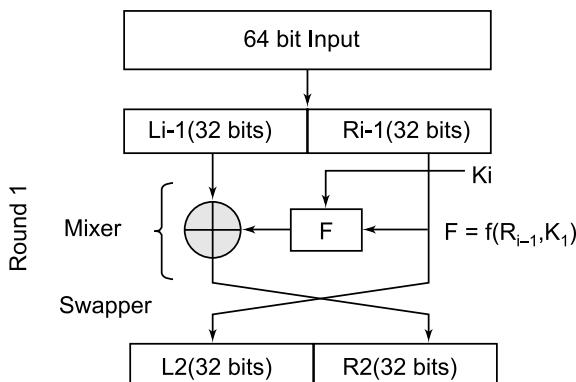


FIGURE 5.3 Feistel cipher rounds.

The output of the function F is XORed with the left half of the plaintext message as shown in Figure 5.3. The following four steps describe the operations performed in the Feistel box:

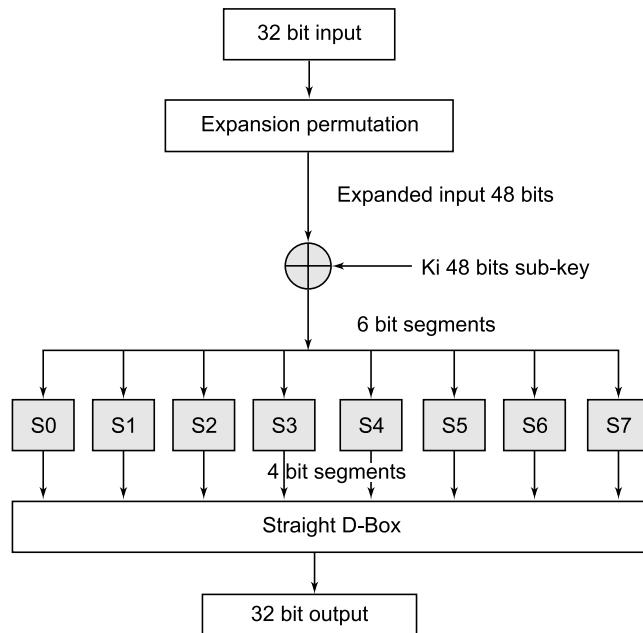
1. Expand the input half block from 32 bits to 48 bits by reshuffling and copying some bits. If the bits are numbered 1–32, the expansion process is as in Table 5.1.

TABLE 5.1 32 Bit to 48 Bit Expansion.

32 Bit Input	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
Expanded 48 Bit Output	[32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17, 16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25, 24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1]

The bit sequence obtained starts with bit 32, duplicates the pairs (4, 5), (8, 9), (12, 13), (16, 17), (20, 21), (24, 25), (28, 29), and ends with bit (32, 1).

2. XOR the expanded half block with 48 bit sub-key K_i .
3. Break the resulting 48 bits into eight groups of 6 bit segments.
4. Convert this 6 bit to 4 bit output bypassing substitution (S-boxes).

**FIGURE 5.4** DES function.

The results of the S-boxes are concatenated together. They produce a 32 bit output which is then put through a permutation, giving the final output of the Feistel box.

The full process of DES can be thought of as a rather complicated series of permutations and substitutions. One of the main functions of the encryption key is that it effectively selects when the copies of bits are going to be passed through into the output of the S-boxes.

The purpose of the Feistel functions is to split the blocks as in Figure 5.4. It performs the encryption, contraction, and permutation of the half block as it moves through Feistel boxes representing confusion and diffusion. Shuffling things around, it diffuses the bits of the plaintext message. The final result of this entire process is a very complex string of information which mixes the plaintext, the encryption key, and various other information in a thoroughly confused and diffuse way.

5.3 ROUND KEY GENERATION: TO GENERATE 16 SUB-KEYS ($K_1, K_2 \dots K_{16}$)

The round key generation algorithm shown in Figure 5.5 gives the steps involved in the generation of 48 bit sub-keys from the initial 64 bit keys.

The key generation process is explained in the following example:

Let K be the hexadecimal key ($16 \times 4 = 64$ bits)

$$K = 1\ 3\ 3\ 4\ 5\ 7\ 7\ 9\ 9\ B\ B\ C\ D\ F\ F\ 1$$

The binary representation of this 64 bit key is:

$$K = 00010011\ 00110100\ 01010111\ 01111001\ 10011011\ 10111100\\ 11011111\ 11110001$$

This 64 bit key is permuted according to Table 5.2.

TABLE 5.2 64 Bit Key Generation.

PC-1	57	49	41	33	25	17	9
	1	58	50	42	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36
	63	55	47	39	31	23	15
	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

In the Table PC-1, the first entry is 57; this means that the 57th bit of the original key K becomes the first bit of the permuted key K_p . The 49th bit of the original key becomes the second bit of the permuted key, and so on. The 4th bit of the original key is the last bit of the permuted key. This gives the 56 bit permutation from the original 64 bit key.

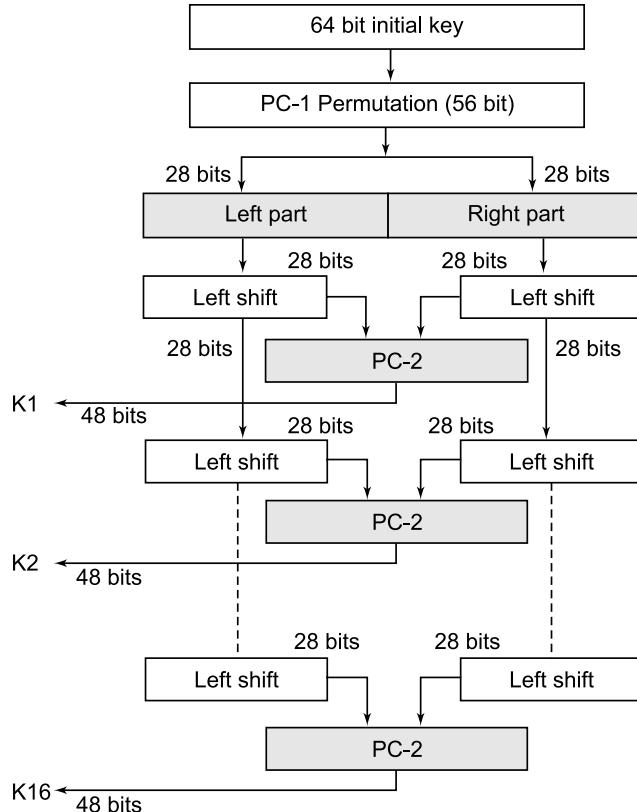


FIGURE 5.5 Round key generation.

Original 64 bit key:

K = 00010011 00110100 01010111 01111001 10011011 10111100
 11011111 11110001

The 56-bit permutation from the original 64 bit key is:

$$K_p = \begin{matrix} 1111000 & 0110011 & 0010101 & 0101111 & 0101010 & 1011001 & 1001111 \\ 0001111 \end{matrix}$$

Split this key into left and right halves as M_0 and N_0 , where each half has 28 bits.

M_0	N_0
111100001100110010101010101111	01010101011001100111100011111

With M_0 and N_0 defined the other 15 round key blocks M_i and N_i can be created.

Where $1 \leq i \leq 16$

Each pair of blocks M_i and N_i is formed from the previous pair M_{i-1} and N_{i-1} respectively for $i = 1, 2, 3, \dots, 16$.

Using the following schedule of the left shift of the previous block (M_{i-1} and N_{i-1}) the other 15 blocks of the sub-keys are created. In the left shift operation, move each bit one place to the left, except for the first bit. This is cycled to the end of the block. The number of times left shift operations are performed in different rounds to generate the sub-keys is given in Table 5.3.

TABLE 5.3 Number of Left Shift Operations in Different Rounds of Feistel to Generate Sub-Keys.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

From original pair M_0 and N_0 we obtain:

$$\begin{aligned} M_0 &= 111100001100110010101010101111 \\ N_0 &= 01010101011001100111100011111 \end{aligned}$$

$M1 = 111000011001100101010101111$	$M9 = 010101010111111100001100110$
$N1 = 1010101011001100111100011110$	$N9 = 0011110001111010101010110011$
$M2 = 1100001100110010101010111111$	$M10 = 010101011111110000110011001$
$N2 = 0101010110011001111000111101$	$N10 = 111100011110101010101011001100$
$M3 = 0000110011001010101011111111$	$M11 = 010101111111000011001100101$
$N3 = 0101011001100111100011110101$	$N11 = 110001111010101010101100110011$
$M4 = 0011001100101010101111111100$	$M12 = 010111111100001100110010101$
$N4 = 0101100110011110001111010101$	$N12 = 000111101010101010110011001111$
$M5 = 1100110010101010111111110000$	$M13 = 011111110000110011001010101$
$N5 = 0110011001111000111101010101$	$N13 = 0111101010101011001100111100$
$M6 = 0011001010101011111111000011$	$M14 = 11111111000011001100101010101$
$N6 = 1001100111100011110101010101$	$N14 = 1110101010101100110011110001$
$M7 = 1100101010101111111100001100$	$M15 = 1111100001100110010101010111$
$N7 = 0110011110001111010101010110$	$N15 = 10101010101100110011110001111$
$M8 = 0010101010111111110000110011$	$M16 = 1111000011001100101010101111$
$N8 = 1001111000111101010101011001$	$N16 = 0101010101100110011110001111$

By applying the following permutation table to each of the concatenated pairs $M_i N_i$, each pair gives a 56 bit block. Out of this, as in Table 5.4, PC-2 uses only a 48 bit key.

TABLE 5.4 PC-2.

PC-2	14	17	11	24	1	5
	3	28	15	6	21	10
	23	19	12	4	26	8
	16	7	27	20	13	2
	41	52	31	37	47	55
	30	40	51	45	33	48
	44	49	39	56	34	53
	46	42	50	36	29	32

In the 48 bit key obtained, the first bit of K_i is the 14th bit of $M_i N_i$, the second bit is the 17th bit, and so on. The last bit is the 32nd bit of $M_i N_i$.

K_1	000110 110000 001011 101111 111111 000111 000001 110010
K_2	011110 011010 111011 011001 110110 111100 100111 100101
K_3	010101 011111 110010 001010 010000 101100 111110 011001
K_4	011100 101010 110111 010110 110110 110011 010100 011101
K_5	011111 001110 110000 000111 111010 110101 001110 101000
K_6	011000 111010 010100 111110 010100 000111 101100 101111
K_7	111011 001000 010010 110111 111101 100001 100010 111100
K_8	111101 111000 101000 111010 110000 010011 101111 111011
K_9	111000 001101 101111 101011 111011 011110 011110 000001
K_{10}	101100 011111 001101 000111 101110 100100 011001 001111
K_{11}	001000 010101 111111 010011 110111 101101 001110 000110
K_{12}	011101 010111 000111 110101 100101 000110 011111 101001
K_{13}	100101 111100 010111 010001 111110 101011 101001 000001
K_{14}	010111 110100 001110 110111 111100 101110 011100 111010
K_{15}	101111 111001 000110 001101 001111 010011 111100 001010
K_{16}	110010 110011 110110 001011 000011 100001 011111 110101

For the first key we have:

$$M_1 N_1 = 1110000 \ 1100110 \ 0101010 \ 1011111 \ 1010101 \ 0110011 \ 0011110 \\ 0011110$$

After applying the permutation as in Table PC-2 it becomes:

$$K_1 = 000110 \ 110000 \ 0010111 \ 1011111 \ 111111 \ 000111 \ 000001 \ 110010$$

The 16 keys ($K_1, K_2 \dots K_{16}$) generated by this method are:

The previous process is repeated to generate all 16 sub-keys.

5.4 ENCODING EACH 64 BIT BLOCK OF PLAINTEXT

There is an initial permutation (IP). It is a 64 bit plaintext. These 64 bits are rearranged according to the IP in Table 5.5. The entry in the table shows the new arrangement of the bits from their initial order.

For example, the 58th bit of the plaintext block becomes the first bit of the IP. The 50th bit of the plaintext block becomes the second bit of the IP and so on. The 7th bit of the plaintext block is the last bit of the IP.

TABLE 5.5 Initial Permutation (IP).

IP	58	50	42	34	26	18	10	2
	60	52	44	36	28	20	12	4
	62	54	46	38	30	22	14	6
	64	56	48	40	32	24	16	8
	57	49	41	33	25	17	9	1
	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5
	63	55	47	39	31	23	15	7

For example, if the plaintext block with 64 bits is:

$$= 0000 \ 0011 \ 0010 \ 0011 \ 0100 \ 0101 \ 0110 \ 0111 \ 1000 \ 1001 \ 1010 \ 1011 \\ 1100 \ 1101 \ 1010 \ 1111$$

Applying the initial permutation (IP) to this block of plaintext as in Table 5.5, we get the initial permutation IP as:

$$= 1100 \ 1100 \ 0000 \ 0000 \ 1100 \ 1100 \ 1111 \ 1111 \ 0000 \ 1110 \ 1010 \\ 1111 \ 0000 \ 1010 \ 1011$$

That is, the 58th bit of the plaintext block is “0,” which becomes the first bit of the IP. The 50th bit of the plaintext block is “1,” which becomes the second bit of the IP. The 7th bit of the plaintext block is “1,” which becomes the last bit of the IP. Now divide the permuted block IP into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

L_0	R_0
1100 1100 0000 0000 1100 1100 1111 1111	1111 0000 1110 1010 1111 0000 1010 1011

The same procedure is repeated through 16 iterations, for $1 \leq i \leq 16$, using a function F , where

$$F = f(R_{i-1}, K_i)$$

This operates on two blocks:

A data block of 32 bits and a key K_i of 48 bits (where $i = 1, 2, \dots, 16$), to produce a block of 32 bits.

In the sixteen different iterations, the bits for the left and right blocks are calculated as:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \end{aligned}$$

The final block for $i = 16$ obtained has the left and right segments L_{16} and R_{16} . That is, in each round, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation F .

For example, in the first round, $i = 1$, we have:

$$\begin{aligned} K_1 &= 000110 110000 001011 101111 111111 000111 000001 110010 \\ L_1 &= R_0 = 1111 0000 1110 1010 1111 0000 1010 1011 \\ R_1 &= L_0 \oplus f(R_0, K_1) \end{aligned}$$

How does function F work?

Let $F = f(R_{i-1}, K_i)$

First expand each block R_{i-1} from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in R_{i-1} .

Use the selection table (Table 5.6) having a 32 bit input block and a 48 bit output block. The resulting 48 bit output is written as 8 blocks of 6 bits each as in Table 5.6.

The 32 bits expanded to 48 bits is represented as: $E(R_{i-1})$

TABLE 5.6 32 Bits Expanded to 48 Bits.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

For example: The $E(R_0)$ is calculated from R_0 as follows:

$$\begin{aligned} R_0 &= 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1011 \\ E(R_0) &= 111110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010111 \end{aligned}$$

That is, each block of 4 original bits has been expanded to an output block of 6 bits.

Next in the F calculation, XOR the output $E(R_{i-1})$ with the key K_i as:

$$K_i \oplus E(R_{i-1}).$$

For example: If $i = 1$

Then for K_1 and $E(R_0)$, we have

$$\begin{aligned} K_1 &= 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001 \\ &\quad 110010 \\ E(R_0) &= 111110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010111 \\ K_1 \oplus E(R_0) &= 111000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100 \\ &\quad 100101. \end{aligned}$$

This operation gives eight groups of 6 bits. Use these groups of six bits as addresses in tables called S-boxes. Each group of six bits will give us an address in a different S-box, and located at that address will be a 4 bit numbers. Now the original 6 bit numbers are replaced by these 4 bit numbers. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits for generating the required 32 bits.

Write the previous result, which is 48 bits, in the form:

$$K_1 \oplus E(R_{i-1}) = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8,$$

Where each B_i is a group of 6 bits.

We now calculate $S_1(B_1)$ $S_2(B_2)$ $S_3(B_3)$ $S_4(B_4)$ $S_5(B_5)$ $S_6(B_6)$ $S_7(B_7)$ $S_8(B_8)$
Where $S_i B_i$ represents the output of the i^{th} S-box.

TABLE 5.7 S-box to Generate S_1 .

	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Row No.	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6

To repeat, each of the functions S_1 , S_2 , S_8 takes a 6-bit block as input and yields a 4-bit block as output. Table 5.7 represents the S-box to generate S_1 . If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows:

From the blocks of 6 bits the first and last bits of B are represented in the rows 0–3 (or in binary 00 to 11). The middle 4 bits of B represent in base 2 a number on the decimal range 0 to 15 (binary 0000 to 1111) in the columns.

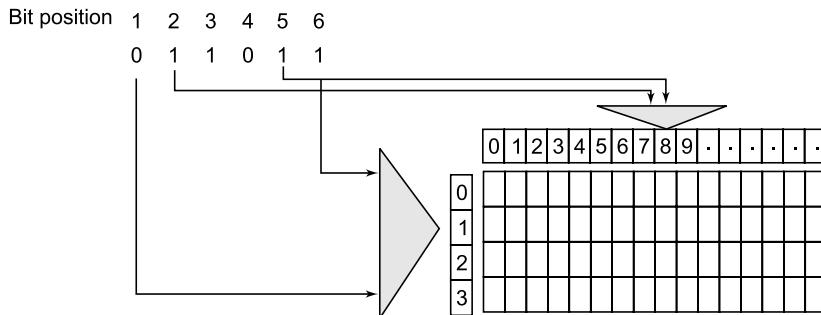
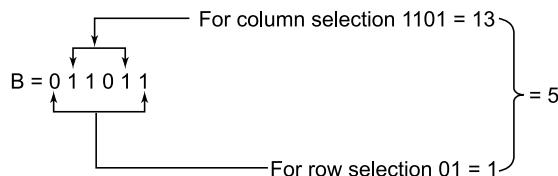


FIGURE 5.6 S-box procedures.

The i^{th} row and j^{th} column elements in Table 5.7 represent the number in the decimal ranging from 0 to 15 and are uniquely represented by a 4 bit block. This block is the output $S_1(B)$ of S_1 for the input B . For example, a block $B = 011011$.



The first bit is 0 and the last bit is 1, giving 01 for the row selection. This selects row 1.

The middle four bits are “1101.” This is the binary equivalent of decimal 13. That is, in row 1 and column 13, 5 appears. This determines the output as 5 (binary 0101).

Hence, $S_1(011011) = 0101$. The following table defines the functions S_1 , S_2 , ..., S_8 .

TABLE 5.8 Generation of S_1 , S_2 , ..., S_8 .

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

(continued)

S₇	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S₈	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

For the first round, we get the output of the eight S-boxes as:

$$K_1 \oplus E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\\ 100111.$$

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)\\ = 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0111$$

The final stage in the calculation of F is to do a permutation P of the S-box output to obtain the final value of F :

$$F = P(S_1(B_1)S_2(B_2)S_3(B_3)\dots S_8(B_8))$$

The permutation P , which is a 32 bit output, is given in Table 5.9.

TABLE 5.9 Permutation P to Generate 32 Bit Output.

P	16	7	20	21
	29	12	28	17
	1	15	23	26
	5	18	31	10
	2	8	24	14
	32	27	3	9
	19	13	30	6
	22	11	4	25

Example: The output of the eight S-boxes is:

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)\\ = 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0111$$

This gives:

$$F = 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$$

$$R_1 = L_0 \oplus F$$

$$R_1 = L_0 \oplus f(R_0, K_1) \quad \text{where } F = f(R_0, K_1)$$

$$= (1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111) \text{ (perform XOR operation)} \oplus \\ (0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011)$$

$$= 1110\ 1111\ 0100\ 1010\ 0110\ 0101\ 0100\ 0100$$

In the next round, we will have $L_2 = R_1$, which is the block we just calculated.

R_2 is calculated as $R_2 = L_1 \oplus f(R_1, K_2)$.

The same procedure is repeated for the other rounds. At the end of the sixteenth round we have the blocks L_{16} and R_{16} . We then reverse the order of the two blocks into the 64-bit block $R_{16}L_{16}$ and apply a final permutation IP^{-1} as defined by the following Table 5.10.

TABLE 5.10 IP^{-1} .

IP^{-1}	40	8	48	16	56	24	64	32
	39	7	47	15	55	23	63	31
	38	6	46	14	54	22	62	30
	37	5	45	13	53	21	61	29
	36	4	44	12	52	20	60	28
	35	3	43	11	51	19	59	27
	34	2	42	10	50	18	58	26
	33	1	41	9	49	17	57	25

In the output table (Table 5.10), bit 40 is the first bit, bit 8 is the second bit, and so on. Finally, bit 25 of the output block is the last bit of the output.

Example:

If we process all 16 rounds using the method defined previously, we get, on the 16th round,

$$L_{16} = 0100\ 0011\ 0100\ 0010\ 0011\ 0010\ 0011\ 0100 \\ R_{16} = 0000\ 1010\ 0100\ 1100\ 1101\ 1001\ 1001$$

We reverse the order of these two blocks and apply the final permutation to:

$$R_{16}L_{16} = 00001010\ 01001100\ 11011001\ 10010101\ 01000011\ 01000010 \\ 00110010\ 00110100$$

$$IP^{-1} = 10000101\ 11101000\ 00010011\ 01010100\ 00001111\ 00001010 \\ 10110100\ 00000101$$

The hexadecimal representation of this is: 85E813540F0AB405.
 This is the encrypted form of the plaintext: 0123456789ABCDEF.

Plaintext: “0123456789ABCDEF”
 Ciphertext: “85E813540F0AB405”

The decryption process is simply the inverse of the encryption process. It follows the same steps as previously, but reverses the order in which the sub-keys are applied. In the decryption process the DES algorithm takes the ciphertext as the input but uses the keys K_i in reverse order. That is, key K_{16} is used for the first iteration, K_{15} for the second, until K_1 , which is used on the 16th and last iteration.

5.4.1 Advantages and Disadvantages of the DES Algorithm

Advantages of the DES algorithm:

- The DES algorithm is ideally suitable for implementation in hardware (bit-shift, lookups).
- A dedicated hardware could run DES at a speed of 200 Mbps.
- The DES algorithm is well suited for the encryption of voice and video applications.

Limitations of the DES Algorithm:

There have been concerns about the level of security provided by DES in two areas, key size and the nature of the algorithm.

- On initial consideration, with a 56 bit key length (forms approximately 7.2×10^{16}), brute force attacks appear impractical. However, with massively parallel computing of about 5000 nodes with each node capable of 50 million keys/sec, the time taken for a brute force search is approximately 100 hours.
- Using the DES algorithm the greater concern is cryptanalysis. It is possible by exploiting the characteristics of DES. The focus is on the 8 S-boxes used in each iteration. These boxes were constructed in such a way that cryptanalysis is possible by an attacker who knows the weakness in the S-boxes.

5.5 DES CRYPTANALYSIS

Major cryptanalytic attacks against DES are:

- 1976: For a very small class of weak keys, DES can be broken with complexity 1.
- 1977: Exhaustive search will become possible within 20 years, breaking DES with complexity 2^{56} .
- 1980: A time/memory trade-off can break DES faster at the expense of more memory.
- 1982: For a very small class of semi-weak keys, DES can be broken with complexity 1.
- 1985: A meet-in-the-middle attack can break 6-round DES with complexity 2^{52} .
- 1987: The “Davies’ attack” can break DES with complexity $2^{56.2}$, slightly worse than brute force.
- 1990: Differential cryptanalysis can break DES with 2^{47} chosen plaintexts (full 16-round).
- 1993: Linear cryptanalysis can break DES with 2^{43} known plaintexts.
- 1994: Differential-linear cryptanalysis can break 8-round DES with 768 chosen plaintexts plus a 2^{46} brute-force search.
- 1994: The Davies’ attack can be improved, and can break DES with 2^{52} known plaintexts.

5.5.1 Brute Force Attacks

- Diffie and Hellman defined the “brute force” attack on DES. This is done by developing a special purpose parallel processing computing system, which is capable of mapping millions of keys per second. (The “brute force” attack in DES means that you try as many of the $2^{56} = 72$ quadrillion possible keys you have to before decrypting the ciphertext into a sensible plaintext message.)
- In 1998, a team of cryptanalysts under the direction of John Gilmore built a computing system that could go through the entire 56-bit DES key space in an average of 4.5 days. On July 17, 1998, they announced they had cracked a 56-bit key in 56 hours. The computer, called “Deep Crack,” uses 27 boards, each containing 64 chips, and is capable of testing 90 billion keys per second.

5.5.2 Attacks Faster than Brute Force Attacks

- **Differential cryptanalysis:** Differential cryptanalysis is a chosen plaintext attack that analyzes how the difference in two plaintext messages affects the differences between the corresponding ciphertext. Differential cryptanalysis involves the analysis of the effect of the plaintext pair defense on the resulting ciphertext differences. The most common difference utilized is the fixed XORed value of the plaintext pairs. By exploiting these differences, the partial sub-key used in the cipher algorithm can be guessed. This guess is done statistically by using counting procedures for each key in which the key with the highest count is assumed to be the most partial sub-key. The scheme can successfully cryptanalyze DES with an effort of the order of 2^{47} chosen plaintext.
- **Linear Cryptanalysis:** Block ciphers commonly use nonlinear operations in their schedule. In DES, the only nonlinear stages are the s-Boxes. All other operations are linear and can be easily analyzed. Linear cryptanalysis is known as a known plaintext attack, which is one of the most commonly used attacks against block ciphers.
The S-boxes have more features in common with linear transformation than one would expect if they were chosen completely at random. Thus, one can be convinced that the DES S-boxes are not optimized against linear cryptanalysis.
- **Improved Davies' Attack:** Davies described a potential attack on DES that is based on the nonuniformity of the distribution of the outputs of pairs of adjacent S-boxes. It produces an unbalanced output. Thus, the output of pairs of successive S-boxes is nonuniformly distributed, even when the inputs are uniformly distributed. The general idea of a Davies' attack is to take advantages of this property to break the full 16 round DES faster than brute force.

5.6 TRIPLE DATA ENCRYPTION STANDARD (TDES)

DES is a standard symmetric block cipher. It uses a 56-bit key to encrypt/decrypt a 64-bit block of data. The algorithm is best suited to implementation in hardware, probably to discourage implementations in software, which tend to be slow by comparison. However, modern computers are so fast that satisfactory software implementations are readily available. Since the time DES was adopted, it has been widely guessed that some kind of back door

was designed into the cryptic S-boxes, effectively allowing the cracking of DES. In 1998, the Electronic Frontier Foundation built a DES Cracker for less than \$250,000 that can decode DES messages in less than a week.

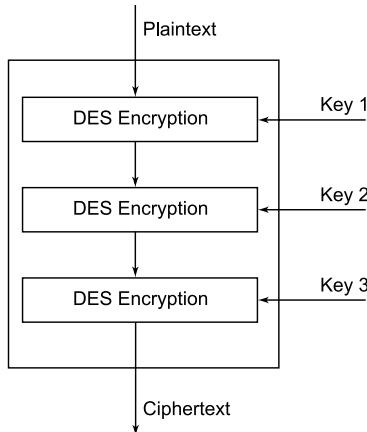


FIGURE 5.7 Triple data encryption standard.

Ever since DES was first announced, controversy has raged about whether a 56 bit key is long enough to guarantee security. To overcome the problem, the use of double or triple length keys are introduced. The use of multiple length keys leads us to the Triple-DES algorithm, in which DES is applied three times.

TDES is developed with a minor variation of the DES. It takes three 64 bit keys for an overall key length of 192 bits as shown in Figure 5.7. Its encryption procedure is the same as that of the DES algorithm, but it is repeated three times, hence the name Triple-DES. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key.

The three keys follow the three executions of the DES algorithm. The function uses an Encryption-Decryption-Encryption (EDE) sequence. Figure 5.8 represents the encryption and decryption process of TDES using three keys, K_1 , K_2 , and K_3 .

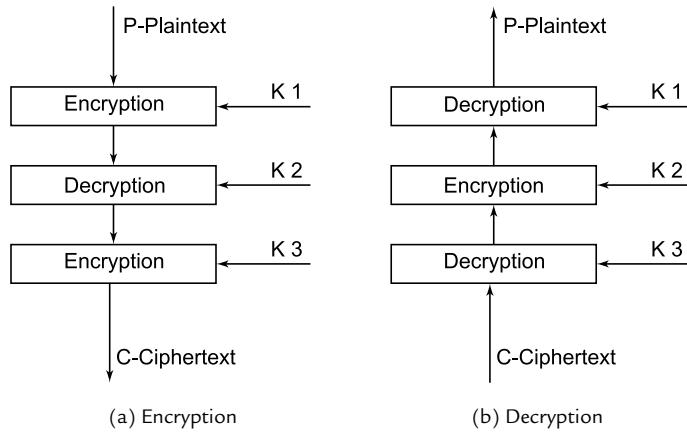
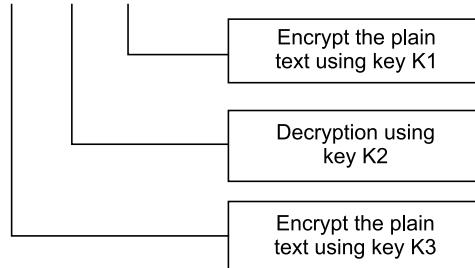


FIGURE 5.8 Encryption and decryption process using TDES.

The ciphertext obtained in the encryption process is represented as

$$C = EK_3 [DK_2 [EK_1[P]]]$$



Where C - ciphertext, P - Plaintext

$EK_i[x]$ - Encryption of x using key K_i

$DK_i[y]$ - Decryption of y using key K_i

The decryption is the reverse operation of the encryption as shown in Figure 5.8 (b) to result the plaintext.

$$P = Dk_1[Ek_2[Dk_3[C]]]$$

In the second stage of the encryption process there are no cryptographic significances. Its only advantages are that it allows users of TDES to decrypt data encrypted by users of the older single DES.

$$C = Ek_1[Dk_1[Ek_1[P]]] = Ek_1[p]$$

The key is always presented as a 64 bit block, every 8th bit of which is ignored. This results in a triple-length key of three 56 bit keys K_1 , K_2 , and K_3 . With three distinct keys, TDES has an effective key length of 168 bits. Having this 168 bit key length, brute force attacks are effectively impossible.

TDES can be a billion times more secure if used properly, but it is three times slower than the regular DES. The advantages of TDES have been proven. Its reliability and longer key length eliminate many of the short-cut attacks that can be used to reduce the amount of time it takes to break the DES.

5.7 INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA)

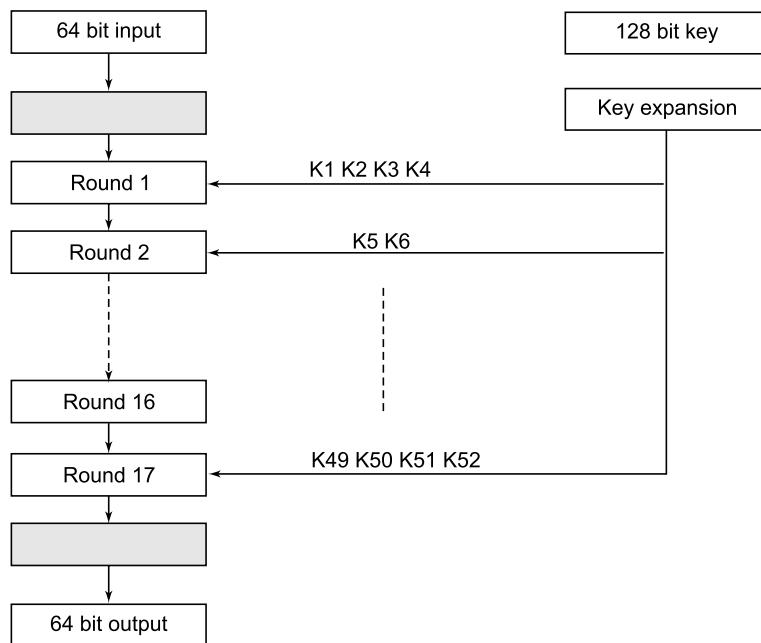


FIGURE 5.9 Representation of IDEA.

IDEA is a universally applicable block encryption algorithm. It protects the transmitted and stored data against unauthorized access by third parties. With a key size of 128 bits, IDEA is far more secure than the widely

used DES algorithm. The algorithm is used worldwide in various banking and commercial applications. The block cipher IDEA operates with 64 bit plaintext and ciphertext blocks and is controlled by a 128 bit key. The fundamental consideration in the design of this algorithm is the use of operations from three different algebraic groups, such as addition, multiplication, and bit-by-bit XOR.

5.7.1 Key Generation

In IDEA, the 52 sub-keys are generated from the 128 bit original key. It uses fifty-two 16 bit key sub-blocks, that is, six sub-blocks for each of the first eight rounds (i.e., $6 \times 8 = 48$) and four more for the ninth round of output transformation. The 128 bit key used for encryption is divided into eight 16 bit sub-keys ($8 \times 16 = 128$). These eight 16 bit sub-keys are $K_1, K_2, K_3, \dots, K_8$ (each 16 bits). The first 8 sub-keys for the algorithm are assigned directly. Out of this six sub-keys are used in the first round, and two are used in the second round. Then the key is circularly left shifted for 25 bits. It is again divided into 6 sub-keys. This procedure is repeated until all 52 sub-keys have been generated. The process of key generation provides an effective technique for changing the key bits used for sub-keys in the eight rounds.

Algorithm for encryption key generation:

Input: 128 bit key K_1, \dots, K_{128}

Output: Six 16 bit sub-keys $K_i^{(r)}, \quad 1 \leq i \leq 6$
 $1 \leq r \leq 8$

Four 16 bit sub-keys $K_i^{(r)}, r = 9, 1 \leq i \leq 4$

Step 1: Arrange the sub-keys

$$\begin{array}{ccccccc}
 K_1^{(1)} & \dots & \dots & K_6^{(1)} \\
 K_1^{(2)} & \dots & \dots & K_6^{(2)} \\
 & \cdot & & \cdot \\
 & \cdot & & \cdot \\
 & \cdot & & \cdot \\
 K_1^{(8)} & \dots & \dots & K_6^{(8)} \\
 K_9^{(1)} & \dots & \dots & K_9^{(4)}
 \end{array}$$

Step 2: Partition K into 16 bit sub-blocks. Assign these blocks to the first eight sub-keys.

Step 3: Repeat the following process:

- (i) Cyclic left shift of key K by 25 bits
- (ii) Partition the result into 8 sub-blocks
- (iii) Assign these blocks to the next 8 sub-keys, until all the keys are assigned.

For example: If the original 128 bit key is labeled $K = (1, 2, 3, \dots, 128)$, then all of the sub-key blocks of the eight rounds have the arrangement as shown in Table 5.11. In the encryption process only six 16 bit sub-keys are needed in each round, whereas the final transformation uses four 16 bit sub-keys. The 9 sub-keys are extracted from the 128 bit key with the left shift of 25 bits. As a result the first sub-key of each round is not in order as shown in Table 5.12.

TABLE 5.11 Generation of IDEA 16 Bit Sub-Key.

		Round 1
1	$K_1^{(1)}$	= $K(1, 2, \dots, 16)$
2	$K_2^{(1)}$	= $K(17, 18, \dots, 32)$
3	$K_3^{(1)}$	= $K(33, 34, \dots, 48)$
4	$K_4^{(1)}$	= $K(49, 50, \dots, 64)$
5	$K_5^{(1)}$	= $K(65, 66, \dots, 80)$
6	$K_6^{(1)}$	= $K(81, 82, \dots, 96)$
		Round 2
7	$K_1^{(2)}$	= $K(97, 98, \dots, 112)$
8	$K_2^{(2)}$	= $K(113, 114, \dots, 128)$
9	$K_3^{(2)}$	= $K(26, 27, \dots, 41)$
10	$K_4^{(2)}$	= $K(42, 43, \dots, 47)$
11	$K_5^{(2)}$	= $K(58, 59, \dots, 73)$
12	$K_6^{(2)}$	= $K(74, 75, \dots, 89)$
		Round 3
13	$K_1^{(3)}$	= $K(90, 91, \dots, 105)$
14	$K_2^{(3)}$	= $K(106, 107, \dots, 121)$
15	$K_3^{(3)}$	= $K(122, 123, \dots, 128, 1, 2, \dots, 9)$
16	$K_4^{(3)}$	= $K(10, 11, \dots, 25)$
17	$K_5^{(3)}$	= $K(51, 52, \dots, 66)$
18	$K_6^{(3)}$	= $K(67, 68, 75, \dots, 82)$

(continued)

Round 4		
19	$K_1^{(4)}$	=K(83, 84,98)
20	$K_2^{(4)}$	=K(99, 100,114)
21	$K_3^{(4)}$	=K(115, 116,128, 1, 2)
22	$K_4^{(4)}$	=K(3, 4,18)
23	$K_5^{(4)}$	=K(19, 20,34)
24	$K_6^{(4)}$	=K(35, 36,50)
Round 5		
25	$K_1^{(5)}$	=K(76, 77,91)
26	$K_2^{(5)}$	=K(92, 93,107)
27	$K_3^{(5)}$	=K(108, 109,123)
28	$K_4^{(5)}$	=K(124, 125,128, 1, 2....11)
29	$K_5^{(5)}$	=K(12, 13,27)
30	$K_6^{(5)}$	=K(28, 29,43)
Round 6		
31	$K_1^{(6)}$	=K(44, 45,59)
32	$K_2^{(6)}$	=K(60, 61,75)
33	$K_3^{(6)}$	=K(101, 102,115)
34	$K_4^{(6)}$	=K(117, 118,128, 1, 2, 3, 4)
35	$K_5^{(6)}$	=K(5, 6,20)
36	$K_6^{(6)}$	=K(21, 22,36)
Round 7		
37	$K_1^{(7)}$	=K(37, 38,52)
38	$K_2^{(7)}$	=K(53, 54,68)
39	$K_3^{(7)}$	=K(69, 70,84)
40	$K_4^{(7)}$	=K(85, 86,100)
41	$K_5^{(7)}$	=K(126, 127, 128.....1, 2,13)
42	$K_6^{(7)}$	=K(14, 15,29)
Round 8		
43	$K_1^{(8)}$	=K(30, 31,45)
44	$K_2^{(8)}$	=K(46, 47,61)
45	$K_3^{(8)}$	=K(62, 63,77)

46	$K_4^{(8)}$	= $K(78, 79, \dots, 93)$
47	$K_5^{(8)}$	= $K(94, 95, \dots, 109)$
48	$K_6^{(8)}$	= $K(110, 111, \dots, 125)$
Round 9		
49	$K_1^{(9)}$	= $K(23, 24, \dots, 38)$
50	$K_2^{(9)}$	= $K(39, 40, \dots, 54)$
51	$K_3^{(9)}$	= $K(55, 56, \dots, 70)$
52	$K_4^{(9)}$	= $K(71, 72, \dots, 86)$

The fifty-two 16 bit sub-key blocks are computed from the given key K as follows:

For the first round, the first 8 sub-keys are taken directly from K . After that the key K is circularly left shifted for 25 bits and again divided into eight 16 bit sub-keys.

The fifty-two 16 bit key sub-blocks, which are generated from the 128-bit key, are explained in the following steps:

1. First, the 128-bit key is partitioned into *eight* 16-bit sub-blocks, which are then directly used as the first eight key sub-blocks.
2. The 128-bit key is then cyclically left shifted by 25 positions, after which the resulting 128-bit block is again partitioned into *eight* 16-bit sub-blocks to be directly used as the next eight key sub-blocks.
3. The cyclic shift procedure described previously is repeated until all of the required *fifty-two* 16-bit key sub-blocks have been generated.

TABLE 5.12 The Key Sub-Blocks Used for Encryption and Decryption in the Individual Rounds.

Round 1	$K_1^{(1)}$	$K_2^{(1)}$	$K_3^{(1)}$	$K_4^{(1)}$	$K_5^{(1)}$	$K_6^{(1)}$
Round 2	$K_1^{(2)}$	$K_2^{(2)}$	$K_3^{(2)}$	$K_4^{(2)}$	$K_5^{(2)}$	$K_6^{(2)}$
Round 3	$K_1^{(3)}$	$K_2^{(3)}$	$K_3^{(3)}$	$K_4^{(3)}$	$K_5^{(3)}$	$K_6^{(3)}$
Round 4	$K_1^{(4)}$	$K_2^{(4)}$	$K_3^{(4)}$	$K_4^{(4)}$	$K_5^{(4)}$	$K_6^{(4)}$
Round 5	$K_1^{(5)}$	$K_2^{(5)}$	$K_3^{(5)}$	$K_4^{(5)}$	$K_5^{(5)}$	$K_6^{(5)}$
Round 6	$K_1^{(6)}$	$K_2^{(6)}$	$K_3^{(6)}$	$K_4^{(6)}$	$K_5^{(6)}$	$K_6^{(6)}$
Round 7	$K_1^{(7)}$	$K_2^{(7)}$	$K_3^{(7)}$	$K_4^{(7)}$	$K_5^{(7)}$	$K_6^{(7)}$
Round 8	$K_1^{(8)}$	$K_2^{(8)}$	$K_3^{(8)}$	$K_4^{(8)}$	$K_5^{(8)}$	$K_6^{(8)}$
Output Transform	$K_1^{(9)}$	$K_2^{(9)}$	$K_3^{(9)}$	$K_4^{(9)}$		

a. Encryption of the key sub-blocks

Round 1	$K_1^{(9)-1} - K_2^{(9)} - K_3^{(9)} K_4^{(9)-1} K_5^{(8)} K_6^{(8)}$
Round 2	$K_1^{(8)-1} - K_3^{(8)} - K_2^{(8)} K_4^{(8)-1} K_5^{(7)} K_6^{(7)}$
Round 3	$K_1^{(7)-1} - K_3^{(7)} - K_2^{(7)} K_4^{(7)-1} K_5^{(6)} K_6^{(6)}$
Round 4	$K_1^{(6)-1} - K_3^{(6)} - K_2^{(6)} K_4^{(6)-1} K_5^{(5)} K_6^{(5)}$
Round 5	$K_1^{(5)-1} - K_3^{(5)} - K_2^{(5)} K_4^{(5)-1} K_5^{(4)} K_6^{(4)}$
Round 6	$K_1^{(4)-1} - K_3^{(4)} - K_2^{(4)} K_4^{(4)-1} K_5^{(3)} K_6^{(3)}$
Round 7	$K_1^{(3)-1} - K_3^{(3)} - K_2^{(3)} K_4^{(3)-1} K_5^{(2)} K_6^{(2)}$
Round 8	$K_1^{(2)-1} - K_3^{(2)} - K_2^{(2)} K_4^{(2)-1} K_5^{(1)} K_6^{(1)}$
Output Transform	$K_1^{(1)-1} - K_2^{(1)} - K_3^{(1)} K_4^{(1)-1}$

b. Decryption of the key sub-blocks

5.7.2 IDEA Encryption

The entire scheme of IDEA encryption is illustrated in Figure 5.10. This encryption process has two input parameters to the encryption function, that is, the plaintext block and the encryption key. The plaintext is a 64 bit block and the key size is 128 bits long. The design methodology behind the IDEA algorithm is based on mixing three different algebraic operations.

These three operations are incompatible, in the sense that no pair of the three operations satisfies distributive law or associative law. Confusion is provided by the incompatibility of these operations and the diffusion by means of a structure known as the *multiplication or addition* (MA) structure. The MA structure is the basic building block of the algorithm.

This entire process consists of eight identical encryption rounds followed by an output transformation. In the first encryption round, the first four 16-bit key sub-blocks are combined with two of the 16-bit plaintext blocks using addition modulo 2^{16} , and with the other two plaintext blocks using multiplication modulo $2^{16} + 1$. The results are then processed further as shown in Figure 5.10, whereby two more 16-bit key sub-blocks enter the calculation and the third algebraic group operator, the bit-by-bit XOR, is used. At the end of the first encryption round four 16-bit values are produced, which are used as input to the second encryption round in a partially changed order. The previously described process for round one is repeated in each of the subsequent seven encryption rounds using different 16-bit key sub-blocks for each combination. During the subsequent output transformation, the four 16-bit values are produced at the end of the 8th encryption round. These are combined with the last four of the 52 key sub-blocks using

addition modulo 2^{16} and multiplication modulo $2^{16} + 1$ to form the resulting four 16-bit ciphertext blocks.

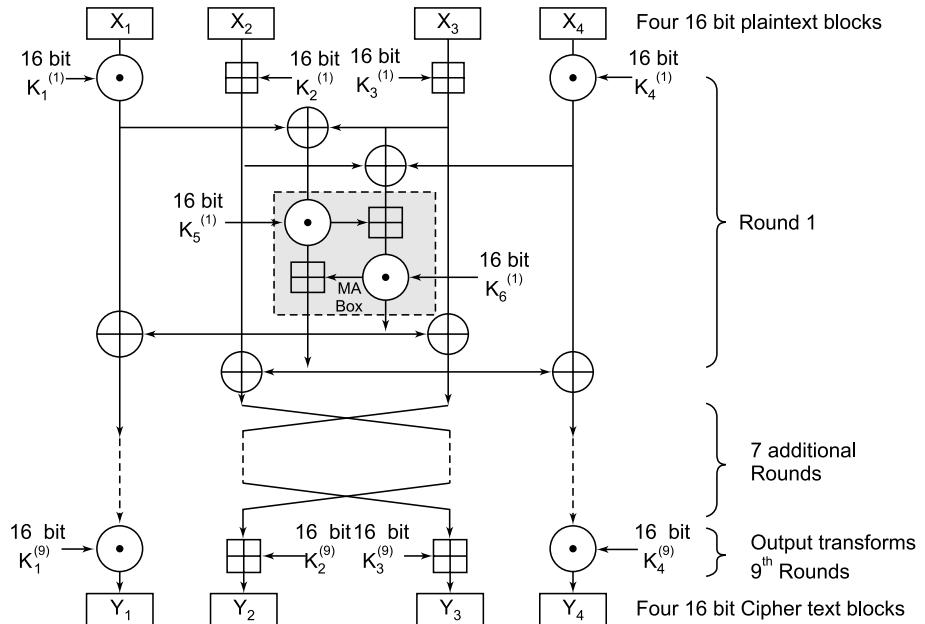


FIGURE 5.10 IDEA encryption process.

1. \boxplus Addition of 16 bit integer modulo 2^{16} .
2. ε Multiplication of 16 bit integer modulo $2^{16} + 1$.
3. \oplus Bit-by-bit XOR operation of 16 bit sub-blocks.

5.7.3 IDEA Algorithm

Input: Plaintext: 64 bit $P = P_1, \dots, P_{64}$ Key: 128 bits $K = K_1, \dots, K_{128}$
Output: Ciphertext: 64 bit $Y = Y_1, \dots, Y_{64}$

Step 1: Compute 16 bit sub-key:

$K_1^{(1)}, \dots, K_6^{(1)}$ for $1 \leq r \leq 8$ and $K_1^{(9)}, \dots, K_4^{(9)}$ for the final transformation, using the key scheduling algorithm.

Step 2: Plaintext blocks:

$$\begin{aligned} X_1 &\leftarrow P_1, \dots, P_{16} \\ X_2 &\leftarrow P_{17}, \dots, P_{32} \\ X_3 &\leftarrow P_{33}, \dots, P_{48} \\ X_4 &\leftarrow P_{49}, \dots, P_{64} \end{aligned}$$

Step 3: For the rounds from 1 to 8 do:

- $$\begin{array}{ll}
 (i) & X_1 \leftarrow X_1 * K_1^{(r)}, & X_4 \leftarrow X_4 * K_4^{(r)} \\
 (ii) & X_2 \leftarrow X_2 + K_2^{(r)}, & X_3 \leftarrow X_3 + K_3^{(r)} \\
 (iii) & t_0 \leftarrow K_5^{(r)} * (X_1 \cdot X_3) \\
 & t_1 \leftarrow K_6^{(r)} * (t_0 + (X_2 \cdot X_4)) \\
 (iv) & t_2 \leftarrow t_0 + t_1 \\
 (v) & X_1 \leftarrow X_1 \cdot t_1, & X_4 \leftarrow X_4 \cdot t_2 \\
 (vi) & a \leftarrow X_2 \cdot t_2 \\
 (vii) & X_2 \leftarrow X_3 \cdot t_3 \\
 (viii) & X_3 \leftarrow a
 \end{array}$$

Step 4: Final transformation (Ciphertext blocks)

- $$\begin{array}{ll}
 (i) & Y_1 \leftarrow X_1 * K_1^{(a)}, & Y_4 \leftarrow X_4 * K_4^{(a)} \\
 (ii) & Y_2 \leftarrow X_3 * K_3^{(a)}, & Y_3 \leftarrow X_2 * K_2^{(a)}
 \end{array}$$

5.7.4 Decryption

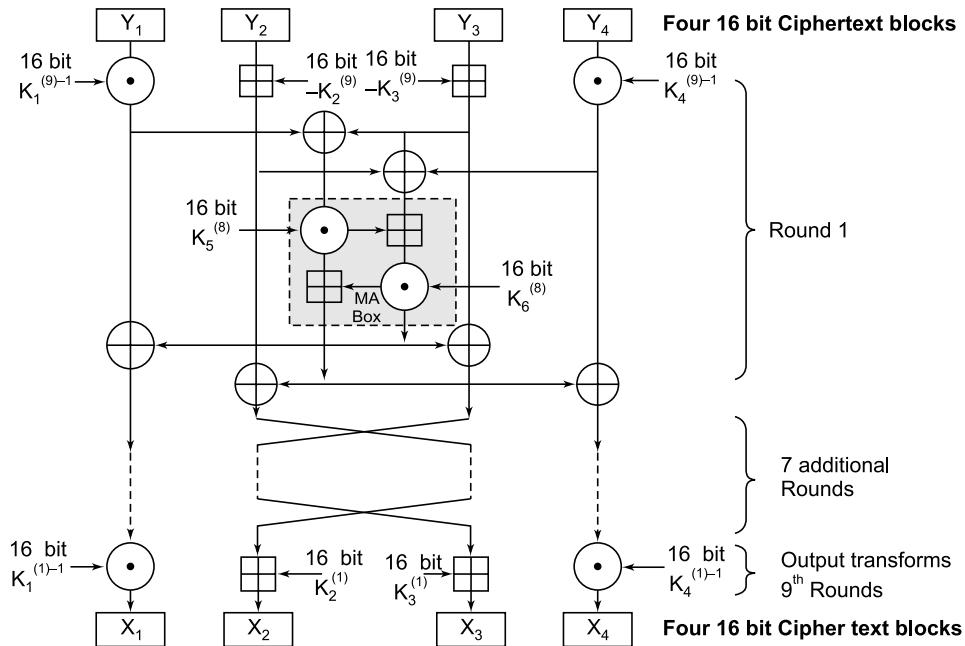


FIGURE 5.11 IDEA decryption process.

The decryption operation is similar to that of the encryption process, except that the key sub-blocks are reversed and a different selection of sub-keys is used. Decryption sub-keys are either the additive or multiplicative inverse of the encryption sub-keys. The decryption of sub-blocks is derived from the encryption key sub-blocks as shown in Table 5.13. Referring to the values in the table, it is clear that the first four decryption sub-keys at round i are derived from the first four sub-keys at encryption round $(10 - i)$, where the output transformation stage is counted as round 9.

For example, the first four decryption sub-keys are equal to the multiplicative inverse modulo $(2^{16} + 1)$ of the corresponding first and fourth encryption. For rounds 2 to 8 the second and third decryption sub-keys are the inverse modulo 2^{16} of the corresponding third and second encryption sub-keys. For round one and nine the second and third decryption sub-keys are equal to the additive inverse modulo 2^{16} of the corresponding second and third encryption sub-keys. Note also that for the first eight rounds, the last two sub-keys of decryption round i are equal to the last two sub-keys of encryption round $(9 - i)$.

TABLE 5.13 Decryption Sub-Keys Derived from Encryption Sub-Keys.

Round i	First Four Decryption Sub-Keys at i	Round $(10 - i)$	First Four Encryption Sub-Keys at $(10 - i)$
Round 1	$Z_1^{(9)-1} - Z_2^{(9)} - Z_3^{(9)}Z_4^{(9)-1}$	Round 9	$Z_1^{(9)}Z_2^{(9)}Z_3^{(9)}Z_4^{(9)}$
Round 2	$Z_1^{(8)-1} - Z_3^{(8)} - Z_2^{(8)}Z_4^{(8)-1}$	Round 8	$Z_1^{(8)}Z_2^{(8)}Z_3^{(8)}Z_4^{(8)}$
.	.	.	.
.	.	.	.
.	.	.	.
Round 8	$Z_1^{(2)-1} - Z_3^{(2)} - Z_2^{(2)}Z_4^{(2)-1}$	Round 2	$Z_1^{(2)}Z_2^{(2)}Z_3^{(2)}Z_4^{(2)}$
Round 9	$Z_1^{(1)-1} - Z_2^{(1)} - Z_3^{(1)}Z_4^{(1)-1}$	Round 1	$Z_1^{(1)}Z_2^{(1)}Z_3^{(1)}Z_4^{(1)}$

More precisely, each of the fifty-two 16-bit key sub-blocks used for decryption is the inverse of the key sub-block used during encryption with respect to the applied algebraic group operation. Additionally, the key sub-blocks must be used in reverse order during decryption in order to reverse the encryption process, as shown in Table 5.13.

Example: Encryption Algorithm: The simplified IDEA encrypts a 16-bit block of plaintext to a 16-bit block of ciphertext. It uses a 32-bit key. The simplified algorithm consists of four identical rounds and a “half round”

final transformation. The simplified algorithm mixes three algebraic operations on nibbles (4-bit blocks):

- Bitwise XOR
- Addition modulo $2^4 (= 16)$
- Multiplication modulo $2^4 + 1 (= 17)$.

There are 16 possible nibbles: 0000... 1111, which represent 0, ..., 15, for addition *modulo* 16.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

The 16 nibbles are thought of as 0001, ..., 1111, 0000, which represent 1, ..., 15, 16, for multiplication *modulo* 17. Notice that 0000, which is 16, is congruent to $-1 \text{ modulo } 17$. 0000 is its own inverse under multiplication *modulo* 17.

The 32-bit key, say 11011100011011110011111101011001, is split into eight nibbles.

1101	1100	0110	1111	0011	1111	0101	1001
------	------	------	------	------	------	------	------

The first six nibbles are used as the sub-keys for round 1. The remaining two nibbles are the first two sub-keys for round 2. Then the bits are shifted cyclically 6 places to the left, and the new 32-bit string is split into eight nibbles that become the next eight sub-keys. The first four of these nibbles are used to complete the sub-keys needed for round 2, and the remaining four sub-keys are used in round 3. The shifting and splitting process is repeated until all 28 sub-keys are generated.

The 32-bit key is 1101 1100 0110 1111 0011 1111 0101 1001.

TABLE 5.14 Encryption Key Schedule.

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
Round 1	1101	1100	0110	1111	0011	1111
Round 2	0101	1001*	0001	1011	1100	1111
Round 3	1101	0110	0111	0111*	1111	0011
Round 4	1111	0101	1001	1101	1100	0110*
Round 5	1111	1101	0110	0111		

*Denotes a shift of bits.

Six sub-keys are used in each of the 4 rounds. The final 4 sub-keys are used in the fifth “half round” final transformation. As an example, we will encrypt the plaintext message 1001110010101100 using the key 11011100011011110011111.

The ciphertext message is 1011101101001011.

Decryption Algorithm: The simplified IDEA decrypts using the same steps as encryption, but new keys must be generated for decryption.

TABLE 5.15 Inverse of Nibble for Addition Modulo 16.

Number in Binary	Integer	Inverse in Binary	Inverse in Integer
0000	0	0000	0
0001	1	1111	15
0010	2	1110	14
0011	3	1101	13
0100	4	1100	12
0101	5	1011	11
0110	6	1010	10
0111	7	1001	9
1000	8	1000	8
1001	9	0111	7
1010	10	0110	6
1011	11	0101	5
1100	12	0100	4
1101	13	0011	3
1110	14	0010	2
1111	15	0001	1

K_i^j denotes the j^{th} decryption key of decryption round i . Z_i^j denotes the j^{th} encryption key of encryption round i . For the first decryption round: $K_1^1 = (Z_1^5)^{-1}$, where $(Z_1^5)^{-1}$ denotes the multiplicative inverse of the first encryption key of encryption round 5 – the “half round” final transformation – modulo 17;

$K_2^1 = -Z_2^5$, where $-Z_2^5$ denotes the additive inverse of the second encryption key of encryption round 5 modulo 16; $K_3^1 = -Z_3^5$; $K_4^1 = (Z_4^5)^{-1}$;

$K_5^1 = Z_5^4$; and $K_6^1 = Z_6^4$. The decryption keys are similarly generated in the remaining complete decryption rounds. The decryption keys for the final transformation “half round” are:

$$K_1^5 = (Z_1^1)^{-1}, K_2^5 = -Z_2^1, K_3^5 = -Z_3^1, \text{ and } K_4^5 = (Z_4^1)^{-1}.$$

For our example the decryption keys are:

Ciphertext: 1011101101001011

Key: 110111000110111100111111

Plaintext: 1001110010101100

TABLE 5.16 Inverse Nibble for Multiplication Modulo 17.

Number in Binary	Integer	Inverse in Binary	Inverse in Integer
0001	1	0001	1
0010	2	1001	9
0011	3	0110	6
0100	4	1101	13
0101	5	0111	7
0110	6	0011	3
0111	7	0101	5
1000	8	1111	15
1001	9	0010	2
1010	10	1100	12
1011	11	1110	14
1100	12	1010	10
1101	13	0100	4
1110	14	1011	11
1111	15	1000	8
0000	16 = -1	0000	16 = -1

TABLE 5.17 Decryption Key Schedule.

	K₁	K₂	K₃	K₄	K₅	K₆
Round 1	1000	0011	1010	0101	1100	0110
Round 2	1000	1011	0111	0100	1111	0011
Round 3	0100	1010	1001	0101	1100	1111
Round 4	0111	0111	1111	1110	0011	1111
Round 5	0100	0100	1010	1000		

5.7.5 IDEA Applications

The IDEA algorithm can easily be embedded in any encryption software. The typical areas of application are:

- Audio and video data for cable TV, Pay TV, video conferencing, distance learning, business TV, and VoIP.
- Sensitive financial and commercial data transactions.
- Transmission links via modem, routers or ATM links, and GSM technology.
- Smart cards.
- IDEA has been an effective measure in the protection of data communication authentication for Pretty Good Privacy (PGP) and other applications.

5.8 BLOWFISH ALGORITHM

Blowfish is a symmetric block cipher that can be effectively used for encryption and to protect data. It was designed by Bruce Schneier in 1993. Blowfish is included in a large number of cipher suites and encryption products. A significant amount of cryptanalysis has proven that the security level of this algorithm is very high. It takes a variable length key block cipher of a key size ranging from 32 bits to 448 bits, making it ideal for securing data. The algorithm uses a Feistel network iterating a simple encryption function 16 times. The plaintext block size is 64 bits and the key can be any length (up to 448 bits). It is also one of the fastest block ciphers when implemented on 32-bit microprocessors with large data caches. The key features of the Blowfish algorithm are given as follows:

1. **Fast:** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.
2. **Compact:** It can run with less than 5 kilobytes of memory.
3. **Simple:** It uses addition, XOR, and lookup tables with 32-bit operands.
4. **Secure:** The key length is variable; it can be in the range of 32–448 bits: the default length is 128 bits.
5. It is suitable for applications where the key does not change often, like communications links or automatic file encryption.

5.8.1 Algorithm Details

The two important components of the Blowfish algorithm are:

- (i) A key expansion part
- (ii) A data encryption part

In Blowfish, the key length is variable and can be as long as 448 bits; the key expansion part converts this 448 bit key into several sub-key arrays totaling 4168 bytes. The data encryption occurs via a 16 round Feistel network. Each round consists of a key dependent permutation along with key- and data-dependent substitution. The operations in these rounds include addition and XOR operations on 32 bit words. The only additional operations are four indexed array data lookups per round.

The Feistel structure of Blowfish, is as shown in Figure 5.12. The algorithm keeps two sub-key arrays; the 18-entry P -array and four 256 array S-boxes. The S-boxes accept 8 bit input and produce 32 bit output. One entry of the P -array is used every round, and after the final round, each half of the data block is XORED with one of the two remaining unused P -entries.

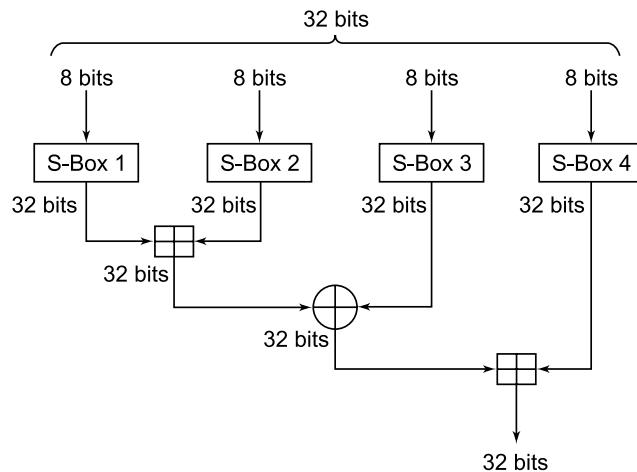


FIGURE 5.12 Feistel structures of Blowfish.

5.8.1.1 Sub-Keys

Blowfish uses a large number of sub-keys. These keys must be pre-computed before any data encryption or decryption. The K -array consists of eighteen 32 bit sub-keys represented as:

K_1, K_2, \dots, K_{18} . There are four 32 bit S-boxes with 256 entries each.

S1,0	S1,1	S1,2.....S1,255
S2,0	S2,1	S2,2.....S2,255
S3,0	S3,1	S3,2.....S3,255
S4,0	S4,1	S4,2.....S4,255

5.8.1.2 Generating the Sub-Key

The procedure for sub-key generation using the Blowfish algorithm is explained as follows:

1. Initialize first the K -array and then the four S-boxes, in order, with a fixed string. This string consists of hexadecimal digits of $K_i \quad i = 1, 2, \dots, 18$.

$$\begin{aligned}K_1 &= 0 \times 243f\ 6a88 \\K_2 &= 0 \times 85a308d3 \\K_3 &= 0 \times 13198a2e \\K_4 &= 0 \times 03707344\end{aligned}$$

2. XOR K_1 with the first 32 bits of the key.

XOR K_2 with the second 32 bits of the key and so on for all bits of the key (possible for up to K_{14}). Repeatedly cycle through the key bits until the entire K -array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64 bit key, then AA, AAA, etc. are equivalent keys.)

3. Encrypt all the zero strings with the Blowfish algorithm, using the sub-keys described in steps (1) and (2).
4. Replace K_1 and K_2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified sub-keys.
6. Replace K_3 and K_4 with the output of step (5).
7. Continue the process, replacing all entries of the K -array and then all four S-boxes in order with the output of the continuously changing Blowfish algorithm.
8. To generate the sub-keys, it completes 521 iterations.

5.8.1.3 Encryption Process

The input X is a 64 bit plaintext element. Divide X into two 32 bit blocks X_L and X_R . Then,

$$\begin{array}{ll} \text{for} & i = 1 \text{ to } 16 \\ & X_L = X_L \oplus K_i \\ & X_R = F(X_L) \oplus X_R \end{array}$$

Swap X_L and X_R Where $i = 1, 2, \dots, 16$ rounds

After the sixteenth round swap X_L and X_R again to undo the last swap operation.

Then $X_R = X_R \oplus K_{17}$ and $X_L = X_L \oplus K_{18}$.

Finally, combine X_L and X_R to get the ciphertext as in Figure 5.13.

For example, consider a plaintext of 64 bit length, the 18 sub-keys are K_1, K_2, \dots, K_{18} , and the round function is $F(\cdot)$. Using the Blowfish algorithm the encryption process results in the 64 bit ciphertext. Using the algorithm the entire encryption process is discussed as follows:

Plaintext message P is divided into two 32 bit parts as:

$$(L_0, R_0) = P$$

TABLE 5.18 Blowfish Encryption Process.

$L_1 = L_0 \text{ XOR } K_1$		$L_{11} = L_9 \text{ XOR } F(R_{10}) \text{ XOR } K_{11}$
$R_2 = R_0 \text{ XOR } F(L_1) \text{ XOR } K_2$		$R_{12} = R_{10} \text{ XOR } F(L_{11}) \text{ XOR } K_{12}$
$L_3 = L_1 \text{ XOR } F(R_2) \text{ XOR } K_3$		$L_{13} = L_{11} \text{ XOR } F(R_{12}) \text{ XOR } K_{13}$
$R_4 = R_2 \text{ XOR } F(L_3) \text{ XOR } K_4$		$R_{14} = R_{12} \text{ XOR } F(L_{13}) \text{ XOR } K_{14}$
$L_5 = L_3 \text{ XOR } F(R_4) \text{ XOR } K_5$		$L_{15} = L_{13} \text{ XOR } F(R_{14}) \text{ XOR } K_{15}$
$R_6 = R_4 \text{ XOR } F(L_5) \text{ XOR } K_6$		$R_{16} = R_{14} \text{ XOR } F(L_{15}) \text{ XOR } K_{16}$
$L_7 = L_5 \text{ XOR } F(R_6) \text{ XOR } K_7$		$L_{17} = L_{15} \text{ XOR } F(R_{16}) \text{ XOR } K_{17}$
$R_8 = R_6 \text{ XOR } F(L_7) \text{ XOR } K_8$		$R_{18} = R_{16} \text{ XOR } K_{18}$
$L_9 = L_7 \text{ XOR } F(R_8) \text{ XOR } K_9$		$C = (R_{18}, L_{17})$
$R_{10} = R_8 \text{ XOR } F(L_9) \text{ XOR } K_{10}$		

The ciphertext obtained is $C = (R_{18}, L_{17})$.

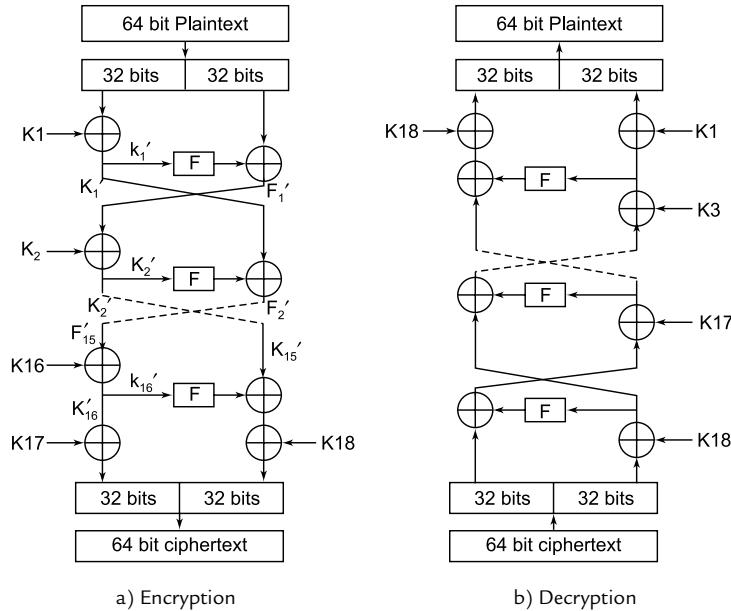


FIGURE 5.13 Blowfish algorithm.

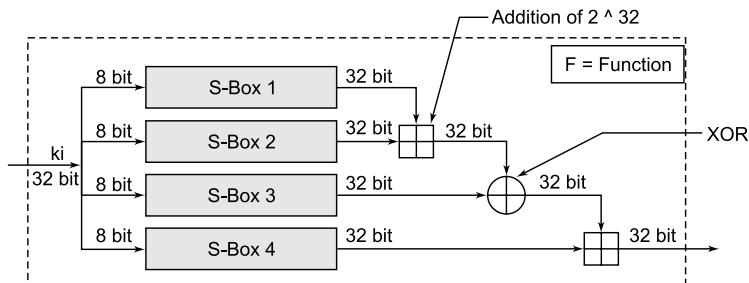


FIGURE 5.14 Function “F” in the Blowfish algorithm.

5.8.1.4 Decryption

The Blowfish decryption algorithm is identical to the encryption algorithm step by step in the same order, except that K_1, K_2, \dots, K_{18} are used in the reverse order. The Decryption process is performed by considering the 64 bit ciphertext C , obtained in the encryption process. Use the 18 sub-keys K_1, K_2, \dots, K_{18} and the function $F()$. The output will be the 64 bit plaintext message P .

Divide the 64 bit ciphertext C into two 32 bit parts as: $(LL_0, RR_0) = C$ [LL_0 and RR_0 are the representation of the 32 left and 32 right bit portions of the ciphertext message].

TABLE 5.19 Blowfish Decryption Process.

LL_1	$= LL_0 \text{XOR } K_{18}$
RR_2	$= RR_0 \text{XOR } F(LL_1) \text{ XOR } K_{17}$
LL_3	$= LL_1 \text{XOR } F(RR_2) \text{ XOR } K_{16}$
RR_4	$= RR_2 \text{XOR } F(LL_3) \text{ XOR } K_{15}$
LL_5	$= LL_3 \text{XOR } F(RR_4) \text{ XOR } K_{14}$
RR_6	$= RR_4 \text{XOR } F(LL_5) \text{ XOR } K_{13}$
LL_7	$= LL_5 \text{XOR } F(RR_6) \text{ XOR } K_{12}$
RR_8	$= RR_6 \text{XOR } F(LL_7) \text{ XOR } K_{11}$
LL_9	$= LL_7 \text{XOR } F(RR_8) \text{ XOR } K_{10}$
RR_{10}	$= RR_8 \text{XOR } F(LL_9) \text{ XOR } K_9$
LL_{11}	$= LL_9 \text{XOR } F(RR_{10}) \text{ XOR } K_8$
RR_{12}	$= RR_{10} \text{XOR } F(LL_{11}) \text{ XOR } K_7$
LL_{13}	$= LL_{11} \text{XOR } F(RR_{12}) \text{ XOR } K_6$
RR_{14}	$= RR_{12} \text{XOR } F(LL_{13}) \text{ XOR } K_5$
LL_{15}	$= LL_{13} \text{XOR } F(RR_{14}) \text{ XOR } K_4$
RR_{16}	$= RR_{14} \text{XOR } F(LL_{15}) \text{ XOR } K_3$
LL_{17}	$= LL_{15} \text{XOR } F(RR_{16}) \text{ XOR } K_2$
RR_{18}	$= RR_{16} \text{XOR } K_1$
$P = (RR_{18}, LL_{17})$	

To validate the decryption algorithm:

$$P = 64 \text{ bit initial plaintext message}$$

$$P = (L_0, R_0)$$

After encryption the resulting ciphertext is:

$$C = 64 \text{ bit}$$

$$C = (R_{18}, L_{17})$$

When we apply the decryption process using the sub-key K_{18} ,

$$\begin{aligned}
 LL_1 &= LL_0 \text{ XOR } K_{18} \\
 &= R_{18} \text{ XOR } K_{18} \text{ where } LL_0 = R_{18} \\
 &= R_{16} \text{ XOR } K_{18} \text{ XOR } K_{18}; \text{ Applying } K_{18} \text{ in encryption} \\
 &= R_{16} \\
 RR_2 &= RR_0 \text{ XOR } F(LL_1) \text{ XOR } K_{17} \\
 &= L_{17} \text{ XOR } F(R_{16}) \text{ XOR } K_{17} \\
 &= L_{15} \text{ XOR } F(R_{16}) \text{ XOR } K_{17} \text{ XOR } F(R_{16}) \text{ XOR } K_{17} \\
 &= L_{15}
 \end{aligned}$$

Repeat the same steps for all other rounds.

Finally:

$$\begin{aligned}
 LL_{17} &= LL_{15} \text{ XOR } F(RR_{16}) \text{ XOR } K_2 \\
 &= R_2 \text{ XOR } F(L_1) \text{ XOR } K_2 \\
 &= R_0 \text{ XOR } F(L_1) \text{ XOR } K_2 \\
 &= R_0 \\
 RR_{18} &= RR_{16} \text{ XOR } K_1 \\
 &= L_1 \text{ XOR } K_1 \\
 &= L_0 \text{ XOR } K_1 \text{ XOR } K_1 \\
 &= L_0 \\
 P &= (RR_{18}, LL_{17}) \\
 &= L_0, R_0 \\
 &= P
 \end{aligned}$$

SUMMARY

- The DES algorithm is a block cipher. The design of the DES algorithm is based on the Feistel structure.
- The DES algorithm takes a 64 bit key of which only 56 bits are used. From these 56 bits sixteen 48 bit sub-keys are generated.
- In the DES algorithm the message is split into 64 bit blocks, and a complex series of steps encrypt the message.
- To break DES with a 56 bit key length was initially considered as impractical. However, with a massive parallel processing of about 5000 nodes, the time taken for a brute-force search took approximately 100 hours.
- The next version of DES is double DES. Double DES suffers by meeting the man in the middle attack.

- The next version of DES is triple DES, also known as TDES. TDES uses three keys and it also has the advantage of proven reliability and a longer key length, which eliminates many attacks.
- The International Data Encryption Algorithm (IDEA) is a block cipher. It operates with 64 bit plaintext. The fundamental features in the design of this algorithm are the use of operations from three different algebraic groups.
- The algorithm structure has been chosen such that, with the exception of those that are different, key sub-blocks are used. The encryption process is identical to the decryption process.
- The Blowfish algorithm is a new secret key block cipher. It uses a Feistel network with 16 encryption functions. The size of the plaintext is 64 bits, and the key can be any length up to 448 bits.
- The most efficient way to break Blowfish is through exhaustive search of the key space.

REVIEW QUESTIONS

1. Explain the working of DES in detail.
2. Explain the design criteria for DES.
3. Draw the general structure of DES and explain the encryption and decryption process.
4. Explain the process of key generation in DES with an example.
5. Discuss the advantages and disadvantages of the DES algorithm.
6. How do different attacks crack the DES standard? Explain.
7. What is TDES? Explain the working of TDES.
8. Explain different cryptanalysis.
9. Describe the detailed structure of IDEA.
10. Explain the operation of IDEA.
11. Explain the working model of a single round DES encryption algorithm with a neat sketch. Also compare DES and 3DES.
12. With suitable sketches, explain the working of the DES algorithm.

13. What is the role of S-boxes in DES?
14. How is key expansion done in Blowfish?
15. Describe IDEA encryption and decryption. Write the applications which use IDEA.
16. Explain the Data Encryption Standard (DES) in detail.
17. What are the weaknesses of DES?
18. Explain the round transformation of IDEA. Also explain the key scheduling of IDEA.
19. How is the expansion permutation function done in DES?

MULTIPLE CHOICE QUESTIONS

1. The encryption algorithm having a 48 bit round key is called:
(a) DES (b) AES (c) IDEA (d) Blowfish
2. The number of keys used in each round of IDEA is:
(a) 5 (b) 4 (c) 6 (d) 8
3. Triple DES was first proposed by:
(a) Tuchman (b) Rivest
(c) both (a) and (b) (d) None of the above
4. What is the original key length in DES?
(a) 64 bit (b) 128 bit (c) 72 bit (d) 56 bit
5. _____ is a block cipher.
(a) DES (b) IDEA (c) AES (d) RSA
6. The Data Encryption Standard is also called:
(a) Data Encryption Algorithm (b) Double DES
(c) AES (d) RSA

7. Which one of the following modes of operation in DES is used for operating short data?
(a) Cipher Feedback Mode (CFB) (b) Cipher Block Chaining (CBC)
(c) Electronic Code Book (ECB) (d) Output Feedback Modes (OFB)
8. _____ is the first step in DES.
(a) Key transformation (b) Expansion permutation
(c) S-box substitution (d) P-box substitution
9. _____ substitution is a process that accepts 48 bits from the XOR operation.
(a) S-box (b) P-box
(c) Expansion permutation (d) Key transformation
10. Which of the following is the best-known example of a symmetric key cipher system?
(a) RSA (b) DES (c) MD5 (d) All the above

ADVANCED ENCRYPTION STANDARD (AES)

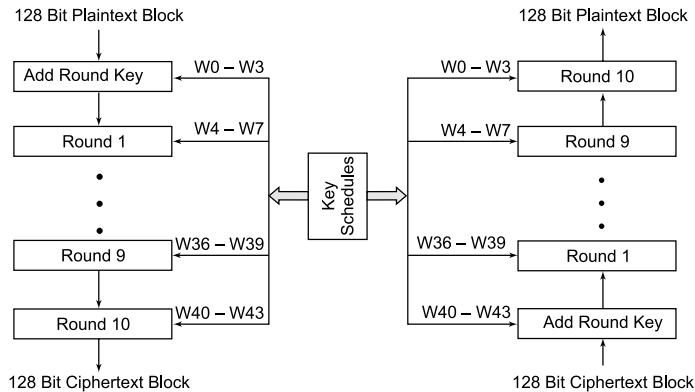
6.1 INTRODUCTION

The Advanced Encryption Standard (AES) also known as Rijndael is the modern encryption standard approved by the National Institute of Standards and Technology (NIST). It has been submitted to the International Standard Organization (ISO) and the Internet Engineering Task Force (IETF) as well as the Institute of Electrical and Electronics Engineers (IEEE) for adopting as a standard. The AES algorithm is an example of a symmetric block cipher. It uses the same key for both the encryption and the decryption process. The AES algorithm was originally created by two Belgians named Joan Daemen and Vincent Rijmen.

AES is a non-Feistel cipher. Since its adaption as a standard, AES has been one of the world's most popular encryption algorithms that use symmetric keys for encryption and decryption. The data block size for encryption and decryption is 128 bits. It allows three different key lengths: 128, 192, and 256 bits. Normally this key size depends on the number of rounds used. AES uses 10, 12, or 14 rounds. For encryption with 10 rounds of processing it uses a 128 bit key, 12 rounds with a 192 bit key, and for 14 rounds a 256 bit key. Except for the last round, all rounds are identical. The overall structure of the AES algorithm for a 128 bit encryption key is shown in Figure 6.1.

The AES algorithm is designed to have the following important characteristics:

- Resistance against all known attacks.
- Speed and code compactness on a wide range of platforms.
- Design simplicity.

**FIGURE 6.1** Overall structure of the AES algorithm.

Each round of processing includes the following four steps:

- One single byte based substitution step
- A row-wise permutation step
- A column-wise mixing step
- The addition of the round key

The order in which these four steps are executed is different for the encryption and decryption operations. If N_r is the number of rounds, the total number of round keys generated by the key-expansion algorithm is always one more than the number of rounds used. In other words, the number of round keys will be $= N_r + 1$

6.2 DATA REPRESENTATION

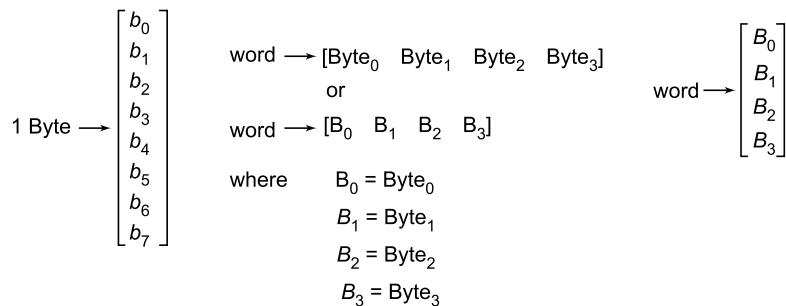
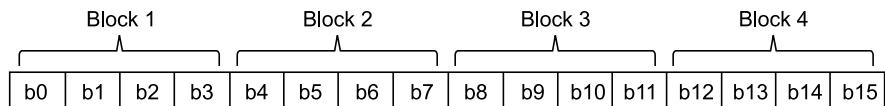
AES uses the data representation terms *bit*, *byte*, *word*, *block*, and *state*.

Bit: A bit is a binary digit with a value 0 or 1.

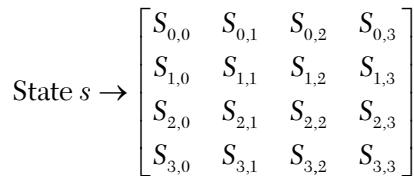
Byte: A group of 8 bits form a byte. A byte may be arranged as a row matrix (1×8) of eight bits or a column matrix (8×1) of eight bits.

Word: A word is a group of 4 bytes that can be treated as a single entity, a row matrix of 4 bytes, or a column matrix of 4 bytes.

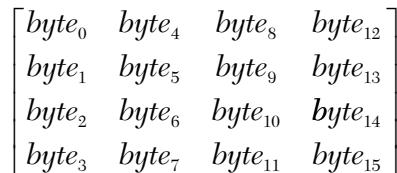
Block: AES is a block cipher. This means that the number of bytes that it encrypts is fixed. AES uses a block length of 16 bytes at a time. If the bytes being encrypted are larger than the specified block, then AES is executed concurrently. If the plaintext is smaller than 16 bytes, then it must be padded.

**FIGURE 6.2** Byte and word representation.**FIGURE 6.3** Block representation.

State: It defines the current condition (state) of the block. That is the block of bytes that is currently being worked on. AES uses several rounds in which each round is made of several stages. A data block is transferred from one stage to another. The state in different stages is normally called “s.”

**FIGURE 6.4** State representation.

States, like blocks, are made of 16 bytes. A state is a block of 128 bits ($16 \times 8 = 128$) consisting of a 4×4 matrix of bytes arranged as follows:



Each element of the state is represented as $S_{R,C}$:

Where R is the number of rows [0.....3]
 C is the number of columns [0.....3]

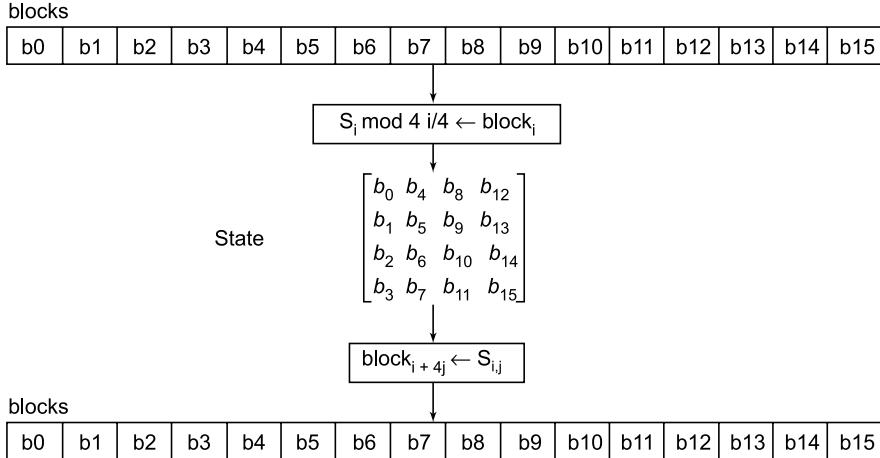


FIGURE 6.5 Transformations from block to state and state to block.

The first 4 bytes of the 128 bit input block occupy the first column in the 4×4 matrix of bytes. The next 4 bytes occupy the second column, and so on; that is, at the beginning of the cipher, bytes in a data block are inserted into a state, column by column, and in each column from top to bottom. At the end of the cipher, bytes in the state are extracted in the same way.

Plaintext Message	P L A I N T E X T T O S T A T E
Value	15 11 00 08 13 19 04 23 19 19 14 18 19 00 19 04
Text in Hexadecimal representation	0F 0B 0A 08 0D 13 04 17 13 13 0E 12 13 0A 13 04

$$\begin{bmatrix} 0F & 0D & 13 & 13 \\ 0B & 13 & 13 & 0A \\ 0A & 04 & 0E & 13 \\ 08 & 17 & 12 & 04 \end{bmatrix}$$

FIGURE 6.6 State representation of a text stream.

This 4×4 matrix is also referred to as the state array. Each round of processing works in the input state array and produces an output state array.

The output state array produced by the last round is rearranged into a 128 bit output block. If the block size is 192, then the state array consists of 4 rows and 6 columns. At the end of the process, the bytes in the state are extracted in the same way as shown in Figure 6.5. The state representation of a text stream is illustrated in Figure 6.6.

6.3 STRUCTURE OF AES

AES is an iterated block cipher. All that means is the same operations are performed many times on a fixed number of bytes. An iteration of the previous steps is called a *round*. The number of rounds of the algorithm depends on the key size, as shown in Table 6.1.

TABLE 6.1 AES Key Size and Number of Rounds.

Key Size (Bytes)	Block Size (Bytes)	Rounds
16	16	10
24	16	12
32	16	14

The AES for a data block and key size of 128 bits using ten rounds is as shown in Figure 6.7. The generic structure of one internal round both in encryption and decryption is shown in Figures 6.8 and 6.9. As a first step before any round-based processing for encryption, the input state array is XORed with the first 4 words of the key schedule. The same process is repeated for decryption also. During decryption the ciphertext state array is XORed with the last 4 words of the key scheduler.

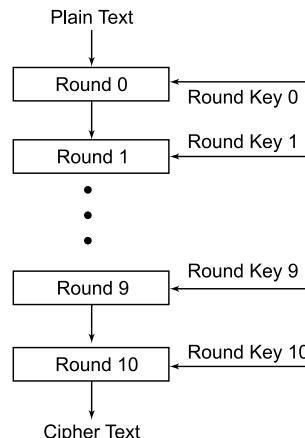


FIGURE 6.7 AES data path for data block and key size of 128 bits.

6.3.1 AES Encryption

Each encryption round of AES consists of the following four steps:

1. Substitute bytes
2. Shift rows
3. Mix column
4. Add round key

In the last step, the output obtained in the previous three steps is XORed with four words of the key scheduler. The algorithm begins with an *AddRoundKey* stage followed by nine rounds of four stages: substitute bytes, shift rows, mix column, and add round key. The tenth round has three transformations. This applies for both encryption and decryption, with the exception that each stage of a round of the decryption algorithm is the inverse of its counterpart in the encryption algorithm. The tenth round simply leaves out the mix columns stage. The first nine rounds of the decryption algorithm consist of the following operations, as shown in Figure 6.9:

1. Inverse shift rows
2. Inverse substitute bytes
3. Inverse add round key
4. Inverse mix columns

Again, the tenth round simply leaves out the inverse mix columns stage. Each of these stages will now be considered in more detail.

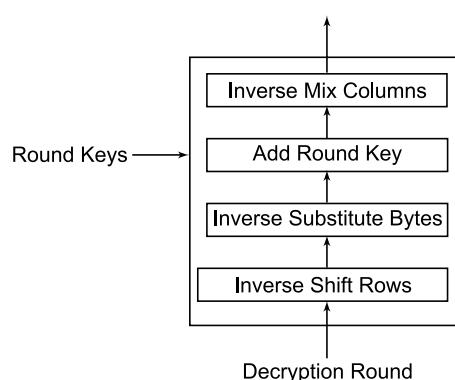
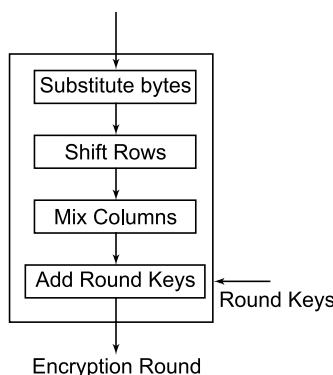


FIGURE 6.8. AES encryption stages.

FIGURE 6.9 AES decryption stages.

Add Round Key Transformation

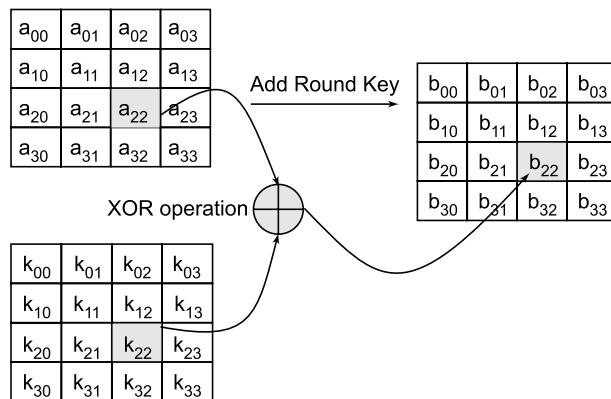


FIGURE 6.10 Add round key.

In this stage, each of the 16 bytes (128 bits) of state are bitwise XORed with the 16 bytes of a portion of the expanded key for the current round. Once the first 16 bytes are XORed against the first 16 bytes of the expanded key, then the expanded key bytes 1 to 16 are never reused again. The next time the *Add Round Key* function is called, bytes 17 to 32 are XORed against the state, and so on for each round of execution.

state	1	2	3	4	16
XOR	⊕	⊕	⊕	⊕										⊕
Expanded key	1	2	3	4	16

First time *AddRoundKey* gets executed

state	1	2	3	4	16
XOR	⊕	⊕	⊕	⊕										⊕
Expanded key	17	18	19	20	32

Second time *AddRoundKey* gets executed

Substitution Bytes

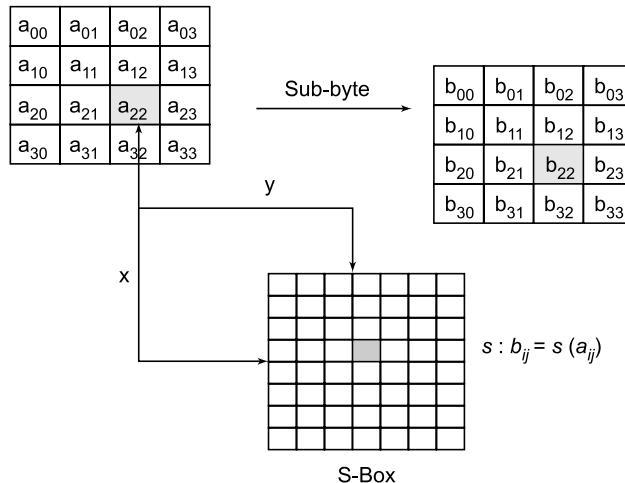


FIGURE 6.11 Substitution byte.

This stage is also known as *sub-byte*. It uses 16×16 matrixes of a byte lookup table called an *S-box*. This matrix consists of all the possible combinations of an 8 bit sequence ($2^8 = 16 \times 16 = 256$). However, the *S-box* is not just a random permutation of these values, and there is a well-defined method for creating the *S-box* tables. Again the matrix that gets operated upon throughout the encryption is known as the state. Let us consider how this matrix is affected in each round. For this particular round each byte is mapped into a new byte in the following way:

In the *S-box* the leftmost nibble is used to represent a particular row and the rightmost nibble specifies a column.

For example: byte {95} represents row 9 and column 5. This turns out to contain the value {2A}. This is then used to update the state matrix, as shown in Figure 6.11. During encryption each value of the state is replaced with the corresponding *S-box* value.

For example, Hex 19 would get replaced with Hex D4 from the *S-box* lookup table. During decryption each value in the state is replaced with the corresponding inverse of the *S-box*. For example, Hex D4 would get replaced with Hex 19.

The inverse substitute byte transformation (also known as *InvSubByte*) makes use of an inverse *S-box*. In this case what is desired is to select the value {2A} and get the value {95} as in Table 6.2. The *S-box* is designed to be

resistant to known cryptanalytic attacks. Specifically, the Rijndael developers sought a design that has a low correlation between input bits and output bits, and the property that the output cannot be described as a simple mathematical function of the input.

TABLE 6.2 AES Forward S-box Lookup Table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

For example, Hex 53 is replaced with Hex ED.

In addition, the *S-box* has no fixed points ($s - \text{Box}(a) = a$) and no opposite fixed points ($s - \text{Box}(a) = a$) where a is the bitwise complement of a .

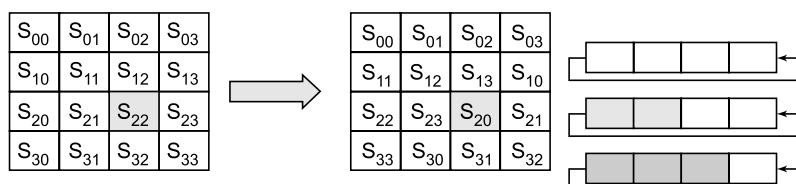
The *S-box* must be invertible if decryption is to be possible ($\text{Is} - \text{box}$ [$s - \text{Box}(a) = a$]).

However, it should not be its self-inverse.

$$\text{i.e., } s - \text{Box}(a) \neq \text{Is} - \text{Box}(a)$$

TABLE 6.3 AES Inverse S-box Lookup Table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Shift Row Transformation:**FIGURE 6.12** Shift row transformation.

It operates on the rows of the state. This operation cyclically shifts the bytes in each row by a certain offset. In AES the first row is left unchanged, whereas in the second row each byte is shifted one to the left as in Figure 6.12. Similarly, the third and fourth row are also shifted by offsets of two and three respectively. For a block size of 128 bits and 192 bits, the shifting pattern is

the same. In this way, each column of the output state of the shift rows step is composed of bytes from each column of the input state. If the block size is 256 bits, the first row is unchanged and the shifting for the second, third, and fourth row is one byte, 2 bytes, and 3 bytes respectively.

For example, the shift row transformation for the state is given as follows

$$\begin{bmatrix} 0F & 0D & 13 & 13 \\ 0B & 13 & 13 & 0A \\ 0A & 04 & 0E & 13 \\ 08 & 17 & 12 & 04 \end{bmatrix}_{\text{before}} \Rightarrow \begin{bmatrix} 0F & 0D & 13 & 13 \\ 13 & 13 & 0A & 0B \\ 0E & 13 & 0A & 04 \\ 04 & 08 & 17 & 12 \end{bmatrix}_{\text{after}}$$

The inverse shift row transformation performs these circular shifts in the opposite direction for each of the last three rows. It rotates each i^{th} row by i elements to the right as follows:

$$\begin{bmatrix} 0F & 0D & 13 & 13 \\ 0B & 13 & 13 & 0A \\ 0A & 04 & 0E & 13 \\ 08 & 17 & 12 & 04 \end{bmatrix}_{\text{before}} \Rightarrow \begin{bmatrix} 0F & 0D & 13 & 13 \\ 0A & 0B & 13 & 13 \\ 0E & 13 & 0A & 04 \\ 17 & 12 & 04 & 08 \end{bmatrix}_{\text{after}}$$

Mix Column Transformation

In mix column each column of the state is replaced by another column obtained by multiplying that column with a matrix in a particular field GF (2⁸) (Galois field). There are two parts to this step:

- The first will explain which parts of the state are multiplied against which parts of the matrix.
- The second will explain how this multiplication is implemented over on the Galois field.

(i) Matrix Multiplication

The state is arranged into a 4×4 matrix with 4 rows and 4 columns. Multiplication is performed one column at a time (4 bytes). Each value in the column is eventually multiplied against every value of the matrix (a total of 16 of multiplications). The results of these multiplications are XORed together to produce only 4 result bytes for the next state. Therefore, there are 4 bytes input, 16 multiplications, 12 XORs, and 4 bytes output. The

multiplication is performed one matrix row at a time against each value of a state column.

Multiplication Matrix

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

16 byte state

b ₀₀	b ₀₁	b ₀₂	b ₀₃
b ₁₀	b ₁₁	b ₁₂	b ₁₃
b ₂₀	b ₂₁	b ₂₂	b ₂₃
b ₃₀	b ₃₁	b ₃₂	b ₃₃

The first result byte is calculated by multiplying 4 values of the state column against 4 values of the first row of the matrix. The result of each multiplication is then XORed to produce 1 byte.

$$b_0 = (b_{00} \times 2) \oplus (b_{10} \times 3) \oplus (b_{20} \times 1) \oplus (b_{30} \times 1)$$

The second result byte is calculated by multiplying the same 4 values of the state column against 4 values of the second row of the matrix. The result of each multiplication is then XORed to produce 1 byte.

$$b_1 = (b_{00} \times 1) \oplus (b_{10} \times 2) \oplus (b_{20} \times 3) \oplus (b_{30} \times 1)$$

The third result byte is calculated by multiplying the same 4 values of the state column against 4 values of the third row of the matrix. The result of each multiplication is then XORed to produce 1 byte.

$$b_2 = (b_{00} \times 1) \oplus (b_{10} \times 1) \oplus (b_{20} \times 2) \oplus (b_{30} \times 3)$$

The fourth result byte is calculated by multiplying the same 4 values of the state column against 4 values of the fourth row of the matrix. The result of each multiplication is then XORed to produce 1 byte.

$$b_3 = (b_{00} \times 3) \oplus (b_{10} \times 1) \oplus (b_{20} \times 1) \oplus (b_{30} \times 2)$$

This procedure is repeated again with the next column of the state, until there are no more state columns.

Putting it all together:

The first column will include state bytes 1–4 and will be multiplied against the matrix in the following manner:

$$b_0 = (b_{00} \times 2) \oplus (b_{10} \times 3) \oplus (b_{20} \times 1) \oplus (b_{30} \times 1)$$

$$b_1 = (b_{00} \times 1) \oplus (b_{10} \times 2) \oplus (b_{20} \times 3) \oplus (b_{30} \times 1)$$

$$\begin{aligned} b_2 &= (b_{00} \times 1) \oplus (b_{10} \times 1) \oplus (b_{20} \times 2) \oplus (b_{30} \times 3) \\ b_3 &= (b_{00} \times 3) \oplus (b_{10} \times 1) \oplus (b_{20} \times 1) \oplus (b_{30} \times 2) \\ (b_0 &= \text{specifies the first byte of the state}) \end{aligned}$$

The second column will be multiplied against the second row of the matrix in the following manner:

$$\begin{aligned} b_4 &= (b_{01} \times 2) \oplus (b_{11} \times 3) \oplus (b_{21} \times 1) \oplus (b_{31} \times 1) \\ b_5 &= (b_{01} \times 1) \oplus (b_{11} \times 2) \oplus (b_{21} \times 3) \oplus (b_{31} \times 1) \\ b_6 &= (b_{01} \times 1) \oplus (b_{11} \times 1) \oplus (b_{21} \times 2) \oplus (b_{31} \times 3) \\ b_7 &= (b_{01} \times 3) \oplus (b_{11} \times 1) \oplus (b_{21} \times 1) \oplus (b_{31} \times 2) \end{aligned}$$

And so on until all columns of the state are exhausted.

Each column is individually operated. Each byte of a column is mapped into a new value that is a function of all 4 bytes in the column. The transformation can be determined by the following matrix multiplication on the state.

(ii) Galois Field Multiplication

The multiplication mentioned previously is performed over a Galois field. Implementation of the multiplication can be done with the use of Tables 6.4 and 6.5 (*E*-Table and *L*-Table) containing hexadecimal values.

The result of the multiplication is obtained by adding the results of the lookup Table *L* and lookup Table *E*. All numbers being multiplied using the mix column function converted to a hex value will form a maximum of a two-digit hex number. We use the first digit in the number on the vertical index and the second number on the horizontal index. If the value being multiplied is composed of only one digit, we use 0 on the vertical index.

For example, if the two hex values being multiplied are $AF \times 8 = AF \times 08$

The value of AF in lookup L is $B7$

Value of 08 in lookup L is $4B$

Once the *L* Table lookup is complete, we can then simply add the numbers together. The only trick is that if the addition result is greater than FF , we subtract FF from the addition result.

For example, $B7 + 4B = 102H$.

Where $102 > FF$

$\therefore 102 - FF = 03$

TABLE 6.4. E Table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	ARC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	FA	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

The last step is to look up the addition result on the *E* table. Again we take the first digit to look up the vertical index and the second digit to look up the horizontal index.

For example, $E(03) = 0F$

Therefore, the result of multiplying $AF \times 08$ over a Galois field is $0F$.

(Where the AF value in the *L* Table is $B7$ and the 08 value is $4B$)

Also, any number multiplied by one is equal to itself and does not need to go through the previous procedure.

For example, $FF \times 1 = FF$

6.4 AES KEY EXPANSION

In an AES algorithm each round has its own round key that is derived from the original 128 bit encryption key. The key expansion algorithm is designed to ensure that if we change one bit of the encryption key, it should affect

the round keys for several rounds. Prior to the encryption and decryption process, the key must be expanded. The expanded key is used in the *AddRoundKey* function. One of the four steps of each round for both encryption and decryption involves XORing the round key with the state array. In order for this to work the expanded key must be large enough so that it can provide key material for every time the *AddRoundKey* function executed.

TABLE 6.5 L Table.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	36	D0	CE	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

The *AddRoundkey* function gets called for each round as well as one extra time at the beginning of the algorithm. In the same way the 128 bit input block is arranged in the form of a state array, the algorithm first arranges the 16 bytes of the encryption key in the form of 4×4 arrays of bytes.

The first four bytes of the encryption key constitute the word w_0 , the next four bytes the word w_1 , and so on.

The algorithm subsequently expands the words $[w_0 \ w_1 \ w_2 \ w_3]$ into a 44 word key schedule that can be represented as: $w_0 \ w_1 \ w_2 \dots \ w_{43}$.

$$\begin{bmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{bmatrix} \downarrow [w_0 \quad w_1 \quad w_2 \quad w_3]$$

The words w_0 , w_1 , w_2 and w_3 are bitwise XORed with the input block before the round-based processing begins for the remaining four words one at a time in each of the ten rounds. The same process with reverse order is also true for decryption as shown in Figure 6.1. The last four words of the key schedule are bitwise XORed with the 128 bit ciphertext block before any round-based processing begins. Subsequently, each of the four words in the remaining 40 words of the key schedule is used in each of the ten rounds of processing.

6.4.1 AES Key Expansion Algorithm

It expands four words w_0 , w_1 , w_2 , w_3 into 44 words as: w_0 , w_1 , w_2 , w_{43} . The key expansion takes place on a four-word to four-word basis as shown in Figure 6.13. Here each grouping of four words decides what the next grouping of four words will be. The algorithm generates four words of the round key for a given round from the corresponding four words of the round key of the previous round. For example, we have the four words of the round key for the i^{th} round.

$$w_i \quad w_{i+1} \quad w_{i+2} \quad w_{i+3}$$

For these to serve as the round key for the i^{th} round, i must be a multiple of 4. This will obviously serve as the round key for the $(i/4)^{th}$ round. Also, w_4 , w_5 , w_6 , w_7 is the round key for round 1, and the sequence of words w_8 , w_9 , w_{10} , w_{11} are the round key for round 2, and so on.

The words w_{i+4} , w_{i+5} , w_{i+6} , w_{i+7} are determined from the words w_i , w_{i+1} , w_{i+2} , w_{i+3} . In Figure 6.13 we can write the key expansion operation as:

$$w_{i+5} = w_{i+4} \oplus w_{i+1} \quad w_{i+6} = w_{i+5} \oplus w_{i+2} \quad w_{i+7} = w_{i+6} \oplus w_{i+3}$$

In the resulting new four-word grouping, each word is an XOR of the previous word and the corresponding word in the previous four-word grouping. The beginning word of each four-word grouping in the key expansion is

obtained by $w_{i+4} = w_i \oplus g(w_{i+3})$. That is, the first word of the new four-word grouping is obtained by XORing the first word of the last grouping with what is returned by applying a function $g()$ to the last word of the previous four-word grouping. The key expansion routine executes a maximum of 4 consecutive functions.

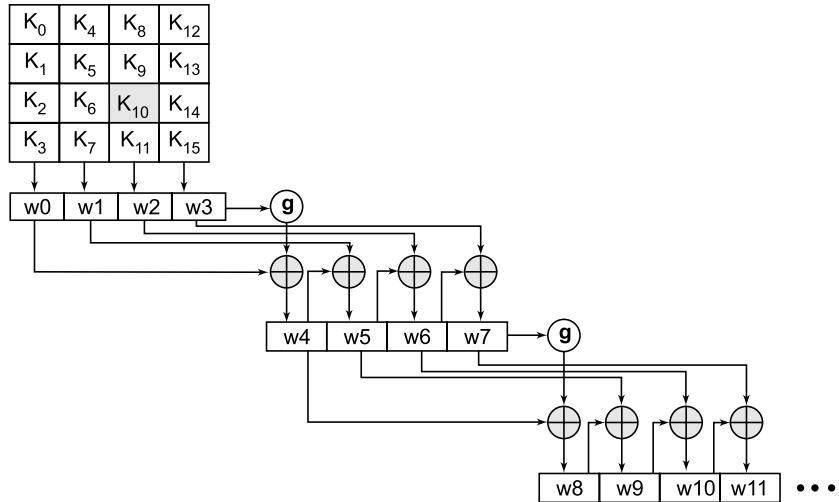


FIGURE 6.13 Key expansion.

These functions are:

1. RotWord
2. SubWord
3. Recon
4. EK
5. K

The iteration of all these steps is called a *round*. The number of rounds of the key expansion algorithm depends on the key size. Table 6.6 gives the details about the key size and expansion algorithm rounds.

TABLE 6.6 Key Size and Expansion Algorithm Rounds.

Key Size (Bytes)	Block Size (Bytes)	Expansion Algorithm Rounds	Expanded Bytes/Rounds	Round Key Copy	Round Key Expansion	Expanded Key (Bytes)
16	16	44	4	4	40	176
24	16	52	4	6	46	208
32	16	60	4	8	52	240

The first byte of the expanded key is always equal to the key size. If the key is 16 bytes long, the first byte of the expanded key will be the same as that of the original key. If it is 32 bytes, then the first 32 bytes of the expanded key will be the same as the original key. In each round it adds 4 bytes to the expanded key, with the exception of the first round. Each round also takes the previous round's 4 bytes as input, operates, and returns 4 bytes. The algorithm calls all four of the functions on every:

- 4 rounds for a 16 byte key
- 6 rounds for a 24 byte key
- 8 rounds for 32 byte key

For the rest of the rounds only a K function result is XORed with the result of the *EK* function. Also note that there is an exception of this rule, where if the key is 32 bytes long, an additional call to the *SubWord* function is called every 8 rounds starting on the 13th round.

6.4.2 AES Key Expansion Function

The function $g()$ performs the following three steps:

1. **RotWord:** It performs a one byte left circular rotation on the argument's four-byte word.
2. **SubWord:** It performs a byte of substitution for each byte of the word returned by the previous step. This uses the same 16×16 lookup table as used in the SubByte step of the encryption round.
3. **Rcon:** It XORs the byte obtained from the previous step with what is known as a round constant. The RoundConstant is a word whose three rightmost bytes are always zero. Therefore, XORing with the RoundConstant amounts to XORing with just its leftmost byte. This function returns a 4-byte value based on Table 6.5.

$$Rcon ((Round/key\ Size/4))-1$$

RoundConstant (Rcon)

The round constant for the i^{th} round is denoted $Rcon [i]$.

$$Rcon[i] = (Rc[i], \quad 0 \times 00, \quad 0 \times 00, \quad 0 \times 00, \quad 0 \times 00)$$

The left-hand side of the previous equation stands for the round constant to be used in the i^{th} round. The right-hand side of the equation represents the rightmost three bytes of the round constant, which zero. The only non-zero byte in the round constant $Rc[i]$ follows the following recursion:

$$Rc[1] = 0 \times 00$$

$$Rc[j] = 0 \times 02 \times Rc [j - 1]$$

For Example: a 16 byte key $Rcon$ is first called in the 4th round ($4/(16/4)) - 1 = 0$

In this case $Rcon$ will return 01 00 00 00

For a 24 byte key $Rcon$ is first called in the 6th round: $(6/(24/4)) - 1 = 0$

In this case $Rcon$ will also return 01 00 00 00

EK (Offset): The EK function returns a 4 byte expanded key after the specified offset. If the offset is “0,” then EK will return bytes 0, 1, 2, 3 of the expanded key.

K (Offset): The K function returns 4 bytes of the key after the specified offset. If the offset is “0,” then K will return bytes 0, 1, 2, 3 of the expanded key.

The key expansion algorithm for key size 16, 24, and 32 is explained as in Table 6.6 with the following three fields.

1. **Round:** It is a counter representing the current step in the key expansion algorithm. It is like a loop counter.
2. **Expanded Key Bytes:** Expanded key bytes affected by the result of the function(s).
3. **Function:** The function(s) that will return the 4 bytes to the affected expanded key bytes.

TABLE 6.7 RoundConstant.

Round Constant			
Rcon (0)	= 01 00 00 00	Rcon (8)	= 1B 00 00 00
Rcon (1)	= 02 00 00 00	Rcon (9)	= 36 00 00 00
Rcon (2)	= 04 00 00 00	Rcon (10)	= 6C 00 00 00
Rcon (3)	= 08 00 00 00	Rcon (11)	= D8 00 00 00
Rcon (4)	= 10 00 00 00	Rcon (12)	= AB 00 00 00
Rcon (5)	= 20 00 00 00	Rcon (13)	= 4D 00 00 00
Rcon (6)	= 40 00 00 00	Rcon (14)	= 9A 00 00 00
Rcon (7)	= 80 00 00 00		

TABLE 6.8 Process of 16 Byte Key Expansion.

Round	Expanded Key Bytes				Function
0	0	1	2	3	K(0)
1	4	5	6	7	K(4)
2	8	9	10	11	K(8)
3	12	13	14	15	K(12)
4	16	17	18	19	SubWord(RotWord(EK(4-1)*4))) XOR Rcon((4/4)-1) XOR EK ((4-4)*4)
5	20	21	22	23	EK((5-1)*4)XOR EK((5-4)*4)
6	24	25	26	27	EK((6-1)*4)XOR EK((6-4)*4)
7	28	29	30	31	EK((7-1)*4)XOR EK((7-4)*4)
8	32	33	34	35	Sub Word(Rot Word(EK((8-4)*4))) XOR Rcon((8/4)-1) XOR EK((8-4)*4)
9	36	37	38	39	EK((8-1)*4)XOR EK((9-4)*4)
10	40	41	42	43	EK((10-1)*4)XOR EK((10-4)*4)
11	44	45	46	47	EK((11-1)*4)XOR EK((11-4)*4)
12	48	49	50	51	Sub Word(Rot Word(EK((12-4)*4))) XOR Rcon((12/4)-1) XOR EK((12-4)*4)
13	52	53	54	55	EK((13-1)*4)XOR EK((13-4)*4)
14	56	57	58	59	EK((14-1)*4)XOR EK((14-4)*4)
15	60	61	62	63	EK((15-1)*4)XOR EK((15-4)*4)
16	64	65	66	67	Sub Word(Rot Word(EK((16-4)*4))) XOR Rcon((16/4)-1) XOR EK((16-4)*4)

Round	Expanded Key Bytes				Function
17	68 69 70 71				$EK((17-1)*4) \text{XOR } EK((17-4)*4)$
18	72 73 74 75				$EK((18-1)*4) \text{XOR } EK((18-4)*4)$
19	76 77 78 79				$EK((19-1)*4) \text{XOR } EK((19-4)*4)$
20	80 81 82 83				Sub Word(Rot Word($EK((20-4)*4)$)) XOR Rcon($(20/4)-1$) XOR $EK((20-4)*4)$
21	84 85 86 87				$EK((21-1)*4) \text{XOR } EK((21-4)*4)$
22	88 89 90 91				$EK((22-1)*4) \text{XOR } EK((22-4)*4)$
23	92 93 94 95				$EK((23-1)*4) \text{XOR } EK((23-4)*4)$
24	96 97 98 99				Sub Word(Rot Word($EK((24-4)*4)$)) XOR Rcon($(24/4)-1$) XOR $EK((24-4)*4)$
25	100 101 102 103				$EK((25-1)*4) \text{XOR } EK((25-4)*4)$
26	104 105 106 107				$EK((26-1)*4) \text{XOR } EK((26-4)*4)$
27	108 109 110 111				$EK((27-1)*4) \text{XOR } EK((27-4)*4)$
28	112 113 114 115				Sub Word(Rot Word($EK((28-4)*4)$)) XOR Rcon($(28/4)-1$) XOR $EK((28-4)*4)$
29	116 117 118 119				$EK((29-1)*4) \text{XOR } EK((29-4)*4)$
30	120 121 122 123				$EK((30-1)*4) \text{XOR } EK((30-4)*4)$
31	124 125 126 127				$EK((31-1)*4) \text{XOR } EK((31-4)*4)$
32	128 129 130 131				Sub Word(Rot Word($EK((32-4)*4)$)) XOR Rcon($(32/4)-1$) XOR $EK((32-4)*4)$
33	132 133 134 135				$EK((33-1)*4) \text{XOR } EK((33-4)*4)$
34	136 137 138 139				$EK((34-1)*4) \text{XOR } EK((34-4)*4)$
35	140 141 142 143				$EK((35-1)*4) \text{XOR } EK((35-4)*4)$
36	144 145 146 147				Sub Word(Rot Word($EK((36-4)*4)$)) XOR Rcon($(36/4)-1$) XOR $EK((36-4)*4)$
37	148 149 150 151				$EK((37-1)*4) \text{XOR } EK((37-4)*4)$
38	152 153 154 155				$EK((38-1)*4) \text{XOR } EK((38-4)*4)$
39	156 157 158 159				$EK((39-1)*4) \text{XOR } EK((39-4)*4)$
40	160 161 162 163				Sub Word(Rot Word($EK((40-4)*4)$)) XOR Rcon($(40/4)-1$) XOR $EK((40-4)*4)$
41	164 165 166 167				$EK((41-1)*4) \text{XOR } EK((41-4)*4)$
42	168 169 170 171				$EK((42-1)*4) \text{XOR } EK((42-4)*4)$
43	172 173 174 175				$EK((43-1)*4) \text{XOR } EK((43-4)*4)$

24 Byte Key Expansion

Each round (except rounds 0, 1, 2, 3, 4, and 5) will take the result of the previous round and produce a 4 byte result for the current round. Notice the first 6 rounds simply copy the total 24 bytes of the key as shown in Table 6.9.

TABLE 6.9 Process of 24 Byte Key Expansion.

Round	Expanded Key Bytes				Function
0	0	1	2	3	$K(0)$
1	4	5	6	7	$K(4)$
2	8	9	10	11	$K(8)$
3	12	13	14	15	$K(12)$
4	16	17	18	19	Sub Word (Rot Word($EK(4-1)*4$)) XOR Rcon($(4/4)-1$) XOR $EK((4-4)*4)$
5	20	21	22	23	$EK((5-1)*4)$ XOR $EK((5-4)*4)$
6	24	25	26	27	$EK((6-1)*4)$ XOR $EK((6-4)*4)$
7	28	29	30	31	$EK((7-1)*4)$ XOR $EK((7-4)*4)$
8	32	33	34	35	Sub Word(Rot Word($EK((8-4)*4$))) XOR Rcon($(8/4)-1$) XOR $EK((8-4)*4)$
9	36	37	38	39	$EK((8-1)*4)$ XOR $EK((9-4)*4)$
10	40	41	42	43	$EK((10-1)*4)$ XOR $EK((10-4)*4)$
11	44	45	46	47	$EK((11-1)*4)$ XOR $EK((11-4)*4)$
12	48	49	50	51	Sub Word(Rot Word($EK((12-4)*4$))) XOR Rcon($(12/4)-1$) XOR $EK((12-4)*4)$
13	52	53	54	55	$EK((13-1)*4)$ XOR $EK((13-4)*4)$
14	56	57	58	59	$EK((14-1)*4)$ XOR $EK((14-4)*4)$
15	60	61	62	63	$EK((15-1)*4)$ XOR $EK((15-4)*4)$
16	64	65	66	67	Sub Word(Rot Word($EK((16-4)*4$))) XOR Rcon($(16/4)-1$) XOR $EK((16-4)*4)$
17	68	69	70	71	$EK((17-1)*4)$ XOR $EK((17-4)*4)$
18	72	73	74	75	$EK((18-1)*4)$ XOR $EK((18-4)*4)$
19	76	77	78	79	$EK((19-1)*4)$ XOR $EK((19-4)*4)$
20	80	81	82	83	Sub Word(Rot Word($EK((20-4)*4$))) XOR Rcon($(20/4)-1$) XOR $EK((20-4)*4)$
21	84	85	86	87	$EK((21-1)*4)$ XOR $EK((21-4)*4)$
22	88	89	90	91	$EK((22-1)*4)$ XOR $EK((22-4)*4)$
23	92	93	94	95	$EK((23-1)*4)$ XOR $EK((23-4)*4)$

Round	Expanded Key Bytes				Function
24	96 97 98 99				Sub Word(Rot Word(EK((24-4)*4))) XOR Rcon((24/4)-1) XOR EK((24-4)*4)
25	100 101 102 103				EK((25-1)*4)XOR EK((25-4)*4)
26	104 105 106 107				EK((26-1)*4)XOR EK((26-4)*4)
27	108 109 110 111				EK((27-1)*4)XOR EK((27-4)*4)
28	112 113 114 115				Sub Word(Rot Word(EK((28-4)*4))) XOR Rcon((28/4)-1) XOR EK((28-4)*4)
29	116 117 118 119				EK((29-1)*4)XOR EK((29-4)*4)
30	120 121 122 123				EK((30-1)*4)XOR EK((30-4)*4)
31	124 125 126 127				EK((31-1)*4)XOR EK((31-4)*4)
32	128 129 130 131				Sub Word(Rot Word(EK((32-4)*4))) XOR Rcon((32/4)-1) XOR EK((32-4)*4)
33	132 133 134 135				EK((33-1)*4)XOR EK((33-4)*4)
34	136 137 138 139				EK((34-1)*4)XOR EK((34-4)*4)
35	140 141 142 143				EK((35-1)*4)XOR EK((35-4)*4)
36	144 145 146 147				Sub Word(Rot Word(EK((36-4)*4))) XOR Rcon((36/4)-1) XOR EK((36-4)*4)
37	148 149 150 151				EK((37-1)*4)XOR EK((37-4)*4)
38	152 153 154 155				EK((38-1)*4)XOR EK((38-4)*4)
39	156 157 158 159				EK((39-1)*4)XOR EK((39-4)*4)
40	160 161 162 163				Sub Word(Rot Word(EK((40-4)*4))) XOR Rcon((40/4)-1) XOR EK((40-4)*4)
41	164 165 166 167				EK((41-1)*4)XOR EK((41-4)*4)
42	168 169 170 171				EK((42-1)*4)XOR EK((42-4)*4)
43	172 173 174 175				EK((43-1)*4)XOR EK((43-4)*4)
44	176 177 178 179				EK((44-1)*4)XOR EK((44-6)*4)
45	180 181 182 183				EK((45-1)*4)XOR EK((45-6)*4)
46	184 185 186 187				EK((46-1)*4)XOR EK((46-6)*4)
47	188 189 190 191				EK((47-1)*4)XOR EK((47-6)*4)
48	192 193 194 195				Sub Word(Rot Word(EK((48-1)*4))) XOR Rcon((48/6)-1) XOR EK((48-6)*4)
49	196 197 198 199				EK((49-1)*4)XOR EK((49-6)*4)
50	200 201 202 203				EK((50-1)*4)XOR EK((50-6)*4)
51	204 205 206 207				EK((51-1)*4)XOR EK((51-6)*4)

32 Byte Key Expansion

Each round (except rounds 0, 1, 2, 3, 4, 5, 6, and 7) will take the result of the previous round and produce a 4 byte result for the current round. Notice the first 8 rounds simply copy the total 32 bytes of the key as in Table 6.10.

TABLE 6.10 Process of 32 Byte Key Expansion.

Round	Expanded Key Bytes					Function
0	0	1	2	3	4	K(0)
1	5	6	7	8	9	K(4)
2	10	11	12	13	14	K(8)
3	15	16	17	18	19	K(12)
4	19	Sub Word (Rot Word(EK(4-1)*4)) XOR Rcon((4/4)-1) XOR EK ((4-4)*4)	16	17	18	
5	23	EK((5-1)*4)XOR EK((5-4)*4)	20	21	22	
6	27	EK((6-1)*4)XOR EK((6-4)*4)	24	25	26	
7	31	EK((7-1)*4)XOR EK((7-4)*4)	28	29	30	
8	35	Sub Word(Rot Word(EK((8-4)*4))) XOR Rcon((8/4)-1) XOR EK((8-4)*4)	32	33	34	
9	39	EK((8-1)*4)XOR EK((9-4)*4)	36	37	38	
10	43	EK((10-1)*4)XOR EK((10-4)*4)	40	41	42	
11	47	EK((11-1)*4)XOR EK((11-4)*4)	44	45	46	
12	51	Sub Word(Rot Word(EK((12-4)*4))) XOR Rcon((12/4)-1) XOR EK((12-4)*4)	48	49	50	
13	55	EK((13-1)*4)XOR EK((13-4)*4)	52	53	54	
14	59	EK((14-1)*4)XOR EK((14-4)*4)	56	57	58	
15	63	EK((15-1)*4)XOR EK((15-4)*4)	60	61	62	
16	67	Sub Word(Rot Word(EK((16-4)*4))) XOR Rcon((16/4)-1) XOR EK((16-4)*4)	64	65	66	
17	71	EK((17-1)*4)XOR EK((17-4)*4)	68	69	70	
18	75	EK((18-1)*4)XOR EK((18-4)*4)	72	73	74	
19	79	EK((19-1)*4)XOR EK((19-4)*4)	76	77	78	
20	83	Sub Word(Rot Word(EK((20-4)*4))) XOR Rcon((20/4)-1) XOR EK((20-4)*4)	80	81	82	
21	87	EK((21-1)*4)XOR EK((21-4)*4)	84	85	86	
22	91	EK((22-1)*4)XOR EK((22-4)*4)	88	89	90	

Round	Expanded Key Bytes				Function
23	92	93	94	95	$EK((23-1)*4) \text{XOR } EK((23-4)*4)$
24	96	97	98	99	Sub Word(Rot Word($EK((24-4)*4)$)) XOR Rcon($(24/4)-1$) XOR $EK((24-4)*4)$
25	100	101	102	103	$EK((25-1)*4) \text{XOR } EK((25-4)*4)$
26	104	105	106	107	$EK((26-1)*4) \text{XOR } EK((26-4)*4)$
27	108	109	110	111	$EK((27-1)*4) \text{XOR } EK((27-4)*4)$
28	112	113	114	115	Sub Word(Rot Word($EK((28-4)*4)$)) XOR Rcon($(28/4)-1$) XOR $EK((28-4)*4)$
29	116	117	118	119	$EK((29-1)*4) \text{XOR } EK((29-4)*4)$
30	120	121	122	123	$EK((30-1)*4) \text{XOR } EK((30-4)*4)$
31	124	125	126	127	$EK((31-1)*4) \text{XOR } EK((31-4)*4)$
32	128	129	130	131	Sub Word(Rot Word($EK((32-4)*4)$)) XOR Rcon($(32/4)-1$) XOR $EK((32-4)*4)$
33	132	133	134	135	$EK((33-1)*4) \text{XOR } EK((33-4)*4)$
34	136	137	138	139	$EK((34-1)*4) \text{XOR } EK((34-4)*4)$
35	140	141	142	143	$EK((35-1)*4) \text{XOR } EK((35-4)*4)$
36	144	145	146	147	Sub Word(Rot Word($EK((36-4)*4)$)) XOR Rcon($(36/4)-1$) XOR $EK((36-4)*4)$
37	148	149	150	151	$EK((37-1)*4) \text{XOR } EK((37-4)*4)$
38	152	153	154	155	$EK((38-1)*4) \text{XOR } EK((38-4)*4)$
39	156	157	158	159	$EK((39-1)*4) \text{XOR } EK((39-4)*4)$
40	160	161	162	163	Sub Word(Rot Word($EK((40-4)*4)$)) XOR Rcon($(40/4)-1$) XOR $EK((40-4)*4)$
41	164	165	166	167	$EK((41-1)*4) \text{XOR } EK((41-4)*4)$
42	168	169	170	171	$EK((42-1)*4) \text{XOR } EK((42-4)*4)$
43	172	173	174	175	$EK((43-1)*4) \text{XOR } EK((43-4)*4)$
44	176	177	178	179	Sub Word($EK((44-1)*4)$) XOR $EK((44-8)*4)$
45	180	181	182	183	$EK((45-1)*4) \text{XOR } EK((45-8)*4)$
46	184	185	186	187	$EK((46-1)*4) \text{XOR } EK((46-8)*4)$
47	188	189	190	191	$EK((47-1)*4) \text{XOR } EK((47-8)*4)$
48	192	193	194	195	Sub Word(Rot Word($EK((48-1)*4)$)) XOR Rcon($(48/8)-1$) XOR $EK((48-8)*4)$
49	196	197	198	199	$EK((49-1)*4) \text{XOR } EK((49-8)*4)$

(continued)

Round	Expanded Key Bytes				Function
50	200	201	202	203	EK((50-1)*4)XOR EK((50-8)*4)
51	204	205	206	207	EK((51-1)*4)XOR EK((51-8)*4)
52	208	209	210	211	Sub Word(EK((52-1)*4))XOR EK((52-8)*4)
53	212	213	214	215	EK((53-1)*4)XOR EK((53-8)*4)
54	216	217	218	219	EK((54-1)*4)XOR EK((54-8)*4)
55	220	221	222	223	EK((55-1)*4)XOR EK((55-8)*4)
56	224	225	226	227	Sub Word(Rot Word(EK((56-1)*4))) XOR Rcon((56/8)-1) XOR EK((56-8)*4)
57	228	229	230	231	EK((57-1)*4)XOR EK((57-8)*4)
58	232	233	234	235	EK((58-1)*4)XOR EK((58-8)*4)
59	236	237	238	239	EK((59-1)*4)XOR EK((59-8)*4)

6.5 AES DECRYPTION

The decryption process of AES is basically the reverse of the encryption algorithm. The basic components in the AES decryption process are:

- Key Expansion
- Inverse AddRoundKey
- Inverse Mix Column

6.5.1 Key Expansion

This module provides the details for generating the ten rounds of the round key. The fourth column of the $(i - 1)$ key is rotated such that each element is moved up by one row as in Figure 6.14. The content of these columns after rotation is replaced with the corresponding values from the S-box (Table 6.1).

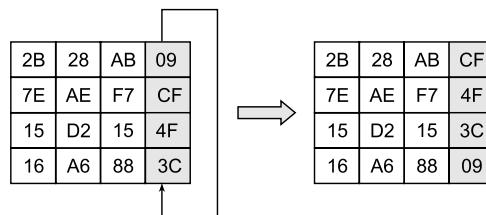


FIGURE 6.14 Key expansion for the decryption process.

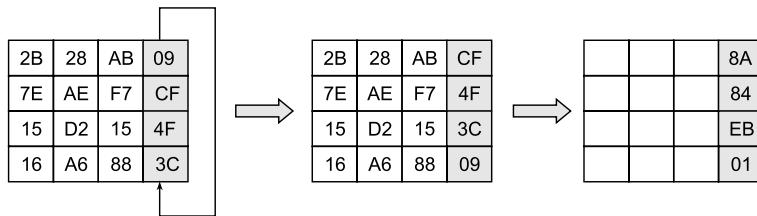
For example: From Table 6.1 (S-box lookup) substitute the elements of the matrix with the corresponding elements of the S-box lookup table.

$$CF = 8A$$

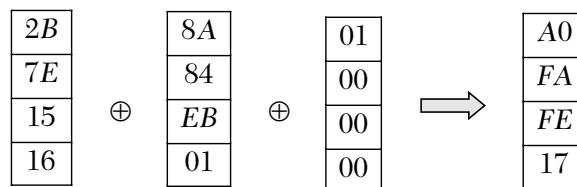
$$4F = 84$$

$$3C = EB$$

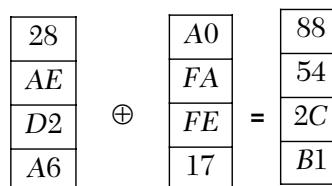
$$09 = 01$$



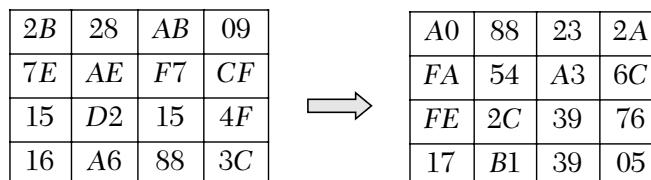
To generate the first column of the i^{th} key this result is XORed with the first column of the $(i-1)^{th}$ key as well as a constant (row constant or $Rcon$) which depends on i .



The second column is generated by XORing the first column of the i^{th} key with the second column of the $(i-1)^{th}$ key.



This continues iteratively for the other two columns in order to generate the entire i^{th} key.



Additionally this entire process continues iteratively to generate ten keys. All these keys are stored statically once they have been computed. The i^{th} key generated is required for the $(10-i)^{th}$ round of decryption.

6.5.2 Inverse AddRoundKey

It is the reverse process of *AddRoundKey*. In Inverse *AddRoundKey* an XOR operation is performed between the ciphertext and intermediate expanded key corresponding to the particular iteration. In the following example, on the left-hand side we have the ciphertext and key value; the final value after it has been generated by this step is shown on the right.

1A	A4	95	3E
A9	B9	4C	3A
13	CD	78	27
86	F1	33	A8

 \oplus

D0	C9	E1	B6
14	EE	3F	63
F9	25	0C	0C
A8	89	C8	A6



CA	6D	74	88
BD	57	73	59
EA	E8	74	2B
2E	78	FB	0E

Inverse Shift Row: This step rotates each i^{th} row by i elements to the right by 1, 2, and 3 cells.

CA	6D	74	88
BD	57	73	59
EA	E8	74	2B
2E	78	FB	0E



CA	6D	74	88
59	BD	57	73
74	2B	EA	E8
78	FB	0E	2E

CA	6D	74	88
59	BD	57	73
74	2B	EA	E8
78	FB	0E	2E



10	B3	CA	97
15	CD	DA	8F
CA	0B	BB	C8
C1	63	07	C3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	AD	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

FIGURE 6.15 Inverse sub-byte.

Inverse sub-bytes: This step replaces each entry in the matrix from the corresponding entry in the inverse S-box.

6.5.3 Inverse Mix Column

The inverse mix column operation performed by the Rijndael cipher, along with the shift-row steps, is the main source of all the ten rounds of diffusion in Rijndael. Each column is treated as a polynomial over GF(2⁸) and is then multiplied modulo X⁴ + 1 with a fixed inverse polynomial as: C⁻¹(X) = 11X³ + 13X² + 9X + 14

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 09 \\ 09 & 14 & 11 & 13 \\ 13 & 09 & 14 & 11 \\ 11 & 13 & 09 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

6.6 AES ENCRYPTION AND DECRYPTION EXAMPLE

The different stages of the AES decryption process are represented in the block diagram in Figure 6.16.

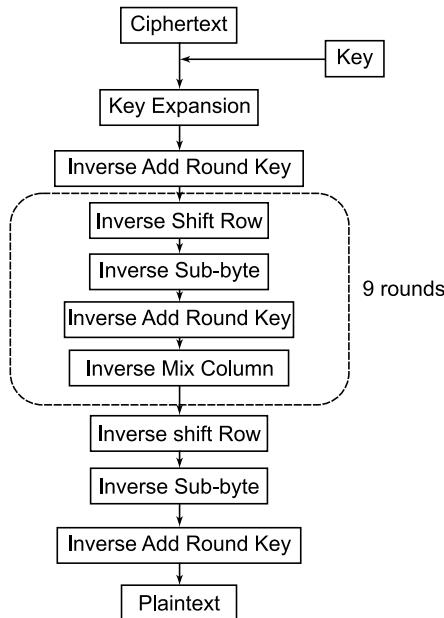


FIGURE 6.16 AES decryption process.

Example: Let's consider the 16 bit plaintext input for the encryption as:

P: 1101 0111 0010 1000

The 16-bit Key: K : 0100 1010 1111 0101

Key Generation: The first step is key generation or key expansion. This step generates the sub-keys. The input key, K, is split into 2 words w_0 and w_1 :

Where, $w_0 = 0100\ 1010$ $w_1 = 1111\ 0101$

The first sub-key, Key_0 , is in fact just the input key: $Key_0 = w_0 w_1 = K$
 The other sub-keys are generated as follows:

$$w_2 = w_0 \text{ XOR } 10000000 \text{ XOR } SubNib(RotNib(w_1))$$

(Note: $RotNib()$ is “rotate the nibbles,” which is equivalent to swapping the nibbles)

$$= 0100 1010 \text{ XOR } 10000000 \text{ XOR } SubNib(0101 1111)$$

(Note: $SubNib()$ is “apply S-box substitution on nibbles using encryption S-box”)

$$= 1100 1010 \text{ XOR } SubNib(0101 1111)$$

$$= 1100 1010 \text{ XOR } 0001 0111$$

$$= 1101 1101$$

$$w_3 = w_2 \text{ XOR } w_1 = 1101 1101 \text{ XOR } 1111 0101$$

$$= 0010 1000$$

$$w_4 = w_2 \text{ XOR } 0011 0000 \text{ XOR } SubNib(RotNib(w_3))$$

$$= 1101 1101 \text{ XOR } 0011 0000 \text{ XOR } SubNib(1000 0010)$$

$$= 1110 1101 \text{ XOR } 0110 1010$$

$$= 1000 0111$$

$$w_5 = w_4 \text{ XOR } w_3 = 1000 0111 \text{ XOR } 0010 1000$$

$$= 1010 1111$$

Now the sub-keys are:

$$Key_0 = w_0 w_1 = 0100 1010 1111 0101$$

$$Key_1 = w_2 w_3 = 1101 1101 0010 1000$$

$$Key_2 = w_4 w_5 = 1000 0111 1010 1111$$

Encryption: The encryption process includes:

1. The Initial operation—Add Round Key
2. The Main Round
3. The Final Round

The output of each operation is used as the input to the next operation, always operating on 16 bits. The 16 bits can be viewed as a state matrix of nibbles.

Add Round 0 Key: Plaintext XOR $Key_1 = 1101 0111 0010 1000 \text{ XOR }$
 $0100 1010 1111 0101$
 $= 1001 1101 1101 1101$

Round 1: Nibble substitution (S-boxes). Each nibble in the input is used in the encryption S-box to generate an output nibble.

Input = 1001 1101 1101 1101
 Output = 0010 1110 1110 1110

Shift row. Swap 2nd nibble and 4th nibble (note, in this example, it's not so easy to see since the 2nd and 4th nibbles are the same!)

= 0010 1110 1110 1110

Mix columns: Apply the matrix multiplication with the constant matrix, M_e , using GF(2⁴). For GF(2⁴), the addition operation is simply an XOR, and for the multiplication operation you can use a lookup table.

$$M_e = \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix} \quad S = \begin{pmatrix} 0010 \\ 1110 \end{pmatrix} \quad \begin{pmatrix} 1110 \\ 1110 \end{pmatrix} = \begin{pmatrix} S'_{00} \\ S'_{10} \end{pmatrix} \quad \begin{pmatrix} S'_{01} \\ S'_{11} \end{pmatrix}$$

$S' = M_e \times S$			
S'_{00}	$= 0010 \text{ XOR } (4 \times 1110)$ $= 0010 \text{ XOR } (4 \times E)$ $= 0010 \text{ XOR D}$ $= 0010 \text{ XOR } 1101$ $= 1111$	S'_{10}	$= (4 \times 0010) \text{ XOR } 1110$ $= 1000 \text{ XOR } 1110$ $= 0110$
S'_{01}	$= 1110 \text{ XOR } (4 \times 1110)$ $= 1110 \text{ XOR } (4 \times E)$ $= 1110 \text{ XOR } 1101$ $= 0011$	S'_{11}	$= (4 \times 1110) \text{ XOR } 1110$ $= 1101 \text{ XOR } 1110$ $= 0011$
Output = $S'_{00} S'_{10} S'_{01} S'_{11}$ = 1111 0110 0011 0011			

Add Round 1 Key: = 1111 0110 0011 0011 XOR 1101 1101 0010 1000
= 0010 1011 0001 1011

Final Round: Nibble substitution (S-boxes) = 1010 0011 0100 0011
ShiftRow (2nd and 4th) = 1010 0011 0100 0011

Add Round 2 Key: = 1010 0011 0100 0011 XOR 1000 0111 1010 1111
= 0010 0100 1110 1100

After the completion of the encryption process, the ciphertext obtained is:
Ciphertext = 0010 0100 1110 1100

Decryption: For decryption the same keys generated during encryption are used. That is, the decryption process would generate the round sub-keys using the input key K , using the encryption S-box.

$$\begin{aligned}\text{Add Round 2 Key} &= 0010\ 0100\ 1110\ 1100\ \text{XOR} 1000\ 0111\ 1010\ 1111 \\ &= 1010\ 0011\ 0100\ 0011\end{aligned}$$

$$\text{Inverse Shift Row} = 1010\ 0011\ 0100\ 0011$$

$$\begin{aligned}\text{Inverse Nibble Sub (use the inverse or decryption S-box)} \\ &= 0010\ 1011\ 0001\ 1011\end{aligned}$$

$$\begin{aligned}\text{Add Round 1 Key} &= 0010\ 1011\ 0001\ 1011\ \text{XOR} 1101\ 1101\ 0010\ 1000 \\ &= 1111\ 0110\ 0011\ 0011\end{aligned}$$

Inverse Mix Columns

$$\begin{array}{lll}S &= & S_{00} \quad S_{01} \\ && S_{10} \quad S_{11}\end{array} \quad \begin{array}{lll}&= & 1111 \quad 0011 \\ && 0110 \quad 0011\end{array}$$

$S' = \begin{matrix} S_{00}', & S_{01}', \\ S_{10}', & S_{11}' \end{matrix}$ $= 9 \times S_{00} \text{ XOR } 2 \times S_{10} \quad 9 \times S_{01} \text{ XOR } 2 \times S_{11}$ $2 \times S_{00} \text{ XOR } 9 \times S_{10} \quad 2 \times S_{01} \text{ XOR } 9 \times S_{11}$	
$S_{00}' = (9 \times 1111) \text{ XOR } (2 \times 0110)$ $= 9 \times F \text{ XOR } 2 \times 6$ $= E \text{ XOR } C$ $= 1110 \text{ XOR } 1100$ $= 0010$	$S_{10}' = 2 \times 1111 \text{ XOR } 9 \times 0110$ $= 2 \times F \text{ XOR } 9 \times 6$ $= D \text{ XOR } 3$ $= 1101 \text{ XOR } 0011$ $= 1110$
$S_{01}' = 9 \times 0011 \text{ XOR } 2 \times 0011$ $= 9 \times 3 \text{ XOR } 2 \times 3$ $= 8 \text{ XOR } 6$ $= 1000 \text{ XOR } 0110$ $= 1110$	$S_{11}' = 2 \times 0011 \text{ XOR } 9 \times 0011$ $= 1110$
Output = 0010 1110 1110 1110	

$$\text{Inverse Shift Row} = 0010\ 1110\ 1110\ 1110$$

$$\text{Inverse Nibble Sub} = 1001\ 1101\ 1101\ 1101$$

$$\begin{aligned}\text{Add Round 0 Key:} &= 1001\ 1101\ 1101\ 1101\ \text{XOR} 0100\ 1010\ 1111\ 0101 \\ &= 1101\ 0111\ 0010\ 1000\end{aligned}$$

$$\text{Plaintext} = 1101\ 0111\ 0010\ 1000$$

$$\text{Original} = 1101\ 0111\ 0010\ 1000$$

6.7 ADVANTAGES OF AES ALGORITHM

1. Design of AES algorithm is simple.
2. AES supports keys of variable length.
3. AES supports a variable number of rounds.
4. Code for AES is small, so small storage and processing devices support the implementation.
5. Parallel processing can be applied.

6.8 COMPARISON BETWEEN AES AND DES ALGORITHMS

TABLE 6.11 Comparison Between DES and AES Algorithms.

S. No.	Details	DES	AES
1.	Developed in the year	1977	2000
2.	Key Length used	56 bits	128, 192 or 256 bits
3.	Cipher type	Symmetric Block Cipher	Symmetric Block Cipher
4.	Block Size	64 bits	128, 192 or 256 bits
5.	Security	Proven inadequate	Considered Secure
6.	Cryptanalysis resistant	Vulnerable to differential and linear cryptanalysis; weak substitution table	Strong against differential, truncated, differential linear interpolation and square attack
7.	Possible Key combinations	2^{56} keys	2^{128} , 2^{192} , or 2^{256} keys
8.	Possible ASCII printable characters (there are 95 printable characters of ASCII).	95^7	95^{16} , 95^{24} , or 95^{32}
9.	Time required to check all possible keys at 50 billion keys per second (the key may be found after checking half of the key space, the time shown is 100% of the key space).	For a 56 bit key 400 days	For 128 bit key 5×10^{21} years

SUMMARY

- AES is a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits.
- AES has been submitted to the ISO, IETF, and IEEE for adopting it as a standard.
- The key size depends on the number of rounds used. It uses 10, 12, or 14 rounds.
- AES is designed to have important characteristics like resistance against all known attacks, speed, code compactness, and design simplicity.
- Each round of processing in AES includes: substitution steps, row-wise permutation, column-wise mixing steps, and addition of the round key.
- AES key expansion is performed before the encryption and decryption operation. Each time the *AddRoundKey* is XORed with the state.
- The decryption process of AES is basically the reverse of the encryption algorithm. Its basic components are key expansion, inverse *AddRoundKey*, and inverse mix column.

REVIEW QUESTIONS

1. What are the important features of the advanced encryption standard (AES)? How is AES different from DES?
2. List out the parameters of AES.
3. Discuss the details of the encryption process using the AES algorithm.
4. Explain in detail key generation in the AES algorithm and its expansion format.
5. Explain the key expansion algorithm.
6. Explain the decryption process using the AES algorithm.
7. Which four tasks are performed in each round of AES cipher? Explain.
8. Explain the key expansion process in AES.
9. Write about the following in an AES cipher:
 - (a) Substitute Bytes Transformation
 - (b) ShiftRows Transformation
 - (c) MixColumns Transformation
 - (d) AddRoundKey Transformation

- 10.** Give the structure of AES. Explain how encryption/decryption is done in AES.

MULTIPLE CHOICE QUESTIONS

- 1.** The AES technique uses the _____ algorithm.
(a) Blowfish (b) Twofish (c) Rijndael (d) Kryptotel
- 2.** AES is an example of the _____ encryption technique
(a) Symmetric (b) Asymmetric
(c) Public key (d). Elliptical curve
- 3.** If the key size is 192 in AES, what would the number of rounds be?
(a) 10 (b) 12 (c) 15 (d) 14
- 4.** The key length used for encryption with 10 rounds of processing is:
(a) 256 bits (b) 128 bits (c) 64 bits (d) 32 bits
- 5.** In the NIST competition for AES there were _____ finalist ciphers.
(a) 2 (b) 3 (c) 4 (d) 5
- 6.** The name of the cipher that was chosen to be AES is:
(a) Twofish (b) Lucifer (c) MARS (d) Rijndael
- 7.** The Advanced Encryption Standard (AES) has three different configurations with respect to number of rounds and:
(a) Data Size (b) Round Size (c) Key Size
(d) Encryption Size (e) None
- 8.** The 4×4 byte matrix in the AES algorithm is called:
(a) Words (b) States
(c) Transactions (d) Permutations
- 9.** In AES the 4×4 matrix is transformed into a key of size:
(a) 32 words (b) 44 words
(c) 54 words (d) 64 words

ASYMMETRIC KEY ENCRYPTION

7.1 INTRODUCTION

There are number of encryption technologies using symmetric key algorithms, such as DES, Triple DES (TDES), and AES. These algorithms provide efficient and powerful cryptographic solutions. Also they are suitable for encrypting bulk data. However, there are some shortcomings in these algorithms, namely key exchange and trust. In this chapter, we consider these two important limitations of symmetric key encryption and discuss how the asymmetric encryption method solves them. We begin with a detailed discussion of the asymmetric algorithm, followed by a more detailed analysis of the RSA, Rabin, ElGamal, and elliptical curve algorithms.

In a symmetric key encryption technique, sharing the key between the parties is a challenge. During key exchange both the parties ensure that the key must remain secret. The direct key exchange is not always feasible due to risk and vulnerability. The other important aspect of symmetric key encryption is data integrity and the identity of the source of the data. A symmetric key can be used to check the identity of the individual who originated a particular set of data, but this authentication scheme encounters some thorny problems involving trust. So the idea of the asymmetric key mechanism is to solve the key exchange and trust problem related to symmetric cryptography by replacing the single secret key with a pair of mathematically related keys. Out of these keys one is made publically available and the other one must be kept secret by the individual who generated the pair.

7.2 PUBLIC KEY CRYPTOGRAPHY

Public key cryptography is a well-established technique and standard for protecting communications from eavesdropping, tampering, and impersonation attacks. The private key systems suffer from key distribution problems. In order to have secured communication the key must be securely sent to the other party. To secure the key and communicate it to the intended receiver in a public Internet, the public key cryptosystem is used. In this case the messages are encrypted with a *public key* and decrypted with a *private key*. No keys need to be distributed for a secure communication to occur. Anyone with a copy of your public key can then encrypt information that only you can read, but they cannot decrypt it. Only the person who has the private key can decrypt the message.

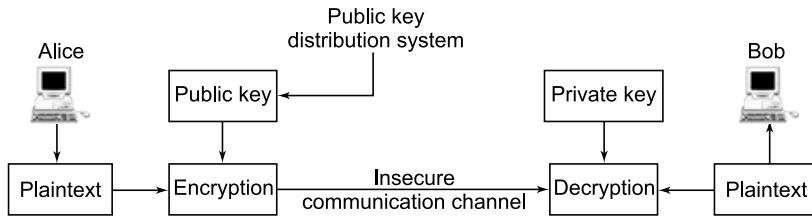


FIGURE 7.1 Asymmetric key encryption techniques.

The primary benefit of public key cryptography is that it allows people who have no preexisting security arrangement to exchange messages securely. The need for sender and receiver to share a secret key via some unsecured channel is eliminated.

7.2.1 Public Key Encryption

In this cryptosystem each user generates a pair of keys to be used for encryption and decryption. Out of this one of the two keys is placed in a public distribution system to be accessed by the others as shown in Figure 7.1. This key is known as the public key. The other key is kept private. Each user maintains a collection of public keys obtained from others. As in Figure 7.2 (a) if Alice wants to send a secret message to Bob, Alice encrypts the plaintext message using Bob's public key; when Bob receives the message, he decrypts it using his private key, and no other recipient can decrypt the message because only Bob knows his private key. In Figure 7.2 (b), Alice encrypts the plaintext message using her private key, whereas Bob uses the public

key to decrypt the ciphertext to obtain the plaintext. This type of asymmetric key cryptosystem is used in e-banking to send encrypted messages to a large number of customers.

Some examples of public key cryptosystems are:

- (i) RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman)
- (ii) Rabin (Integer factorization problem, square root modulo composite “ n ”)
- (iii) ElGamal (named for its inventor Toher Elgamal)
- (iv) Diffie-Helman (Whitefield Diffie and Martin Hellman)
- (v) Elliptic Curve Cryptography (ECC)
- (vi) DSA –Digital Signature Algorithm (invented by David Kravitz)

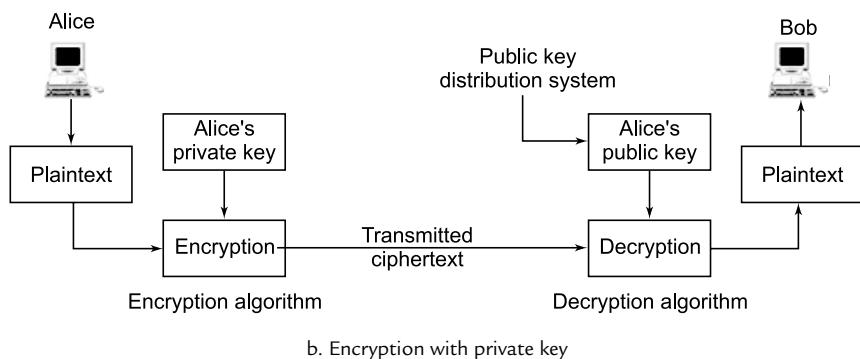
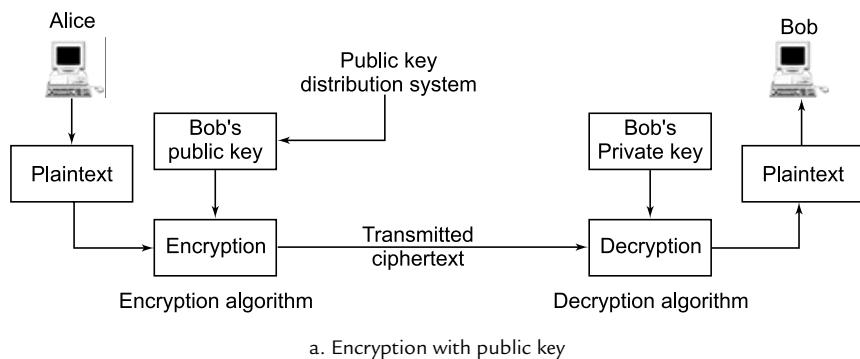


FIGURE 7.2 (a) and (b). Public key encryption.

7.3 RSA ALGORITHM

The algorithm is named after Ron Rivest, Adi Shamir, and Len Adleman, who invented it in 1977. It is one of the most widely used public key cryptosystems in the world. It can be used for encryption and digital signatures. Its security is mainly based on the difficulty in factoring large integers.

A sender Alice can send an encrypted message to the receiver Bob, without any prior exchange of secret keys. Alice just uses Bob's public key to encrypt the message and Bob decrypts it using the private key, which only he knows. The RSA cryptosystem is based on the assumption that factoring large integers is computationally hard. RSA keys are often the length of a power of two, like 512, 1024, and 2048 bits.

7.3.1 RSA Encryption Algorithm

Let us define integer parameters P as a plaintext, C as an encrypted text (ciphertext), E as the encryption key, D as the decryption key, and M as the modulo number. The encryption process is represented by following equation:

$$C = P^E \bmod M \quad \dots(1)$$

Where “mod” represents the modular operation. The following equation represents the decryption process:

$$P = C^D \bmod M \quad \dots(2)$$

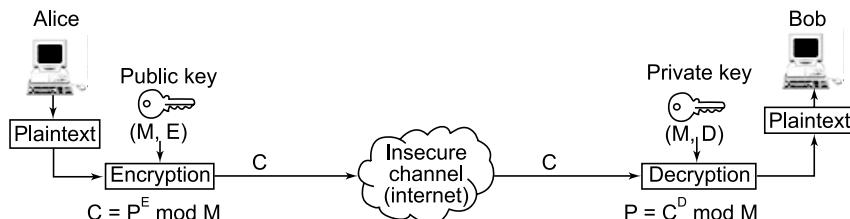


FIGURE 7.3 RSA algorithm.

Select the public exponent E between 3 and $(L - 1)$ that is relatively prime to $(p - 1)$ and $(q - 1)$; also compute the private exponent D from E , p , and q . The relationship between the exponents E and D ensures that encryption and decryption are inverse, so that the decryption operation recovers the original plaintext message P . Without the private keys (M, D) , it's difficult

to recover P from C . Consequently, M and E can be made public without compromising security, which is the basic requirement for public-key cryptography. Referring to equations (1) and (2), the E , D , and M are defined.

We randomly choose quite large numbers of two prime factors p and q ($p \neq q$).

Then modulo M is defined, where $M = p \times q$

Using p and q , L is computed as:

$$L = LCM(p - 1, q - 1) \quad \dots(3)$$

Where LCM stands for Least Common Multiplier. The encryption key is chosen as a certain number which is less than and mutually prime with L .

Therefore, E can be expressed as:

$$GCD(L, E) = 1$$

Where GCD stands for Greatest Common Divisor.

The decryption key D should satisfy the following equation for an arbitrary integer:

$$E \cdot D = L + 1$$

By using these parameters E , D , and M , the encryption and decryption operation can be performed according to equations (1) and (2).

Example 7.1: Bob chooses two primes $p = 11$ and $q = 3$

M is calculated as $M = p \times q$

$$M = 11 \times 3 = 33$$

Using p and q the value of L is obtained as:

$$L = LCM(p - 1, q - 1)$$

$$\text{Where } t = (p - 1)(q - 1)$$

$$= (11 - 1)(3 - 1)$$

$$= 20$$

Choose a prime E in such a way that E is between 3 and $L - 1$, that is, relatively prime to $(p - 1)$ and $(q - 1)$. E needs to be relatively prime to t .

Let $E = 3$

$$\begin{aligned} GCD(E, p - 1) &= 1 \\ &= GCD(3, 2) = 1 \\ \therefore &\quad GCD(E, t) \\ &= GCD(E, (p - 1)(q - 1)) \\ &= GCD(3, 20) = 1 \end{aligned}$$

$$\begin{aligned}\text{The private key } D \text{ is computed as: } & D \times E \equiv 1 \pmod{t} \\ &= E^{-1} \pmod{t} \\ &= 3^{-1} \pmod{20}\end{aligned}$$

i.e., find a value for D such that t divides $(D \times E - 1)$

i.e., 20 divides $(D \times E - 1)$

A simple testing ($D = 1, 2, \dots$) gives $D = 7$

$$\begin{aligned}(D \times E - 1) &= 3 \times 7 - 1 \\ &= 20\end{aligned}$$

Which is divisible by t . The public and private keys are:

$$\textbf{Public Key} = (M, E) = (33, 3)$$

$$\textbf{Private Key} = (M, D) = (33, 7)$$

Encryption: Alice encrypts the message using the RSA encryption equation $C = P^E \pmod{M}$

If the plaintext message ($P = 7$)

$$\begin{aligned}C &= 7^3 \pmod{33} \\ &= 343 \pmod{33} \\ &= 13\end{aligned}$$

Hence, the ciphertext is $C = 13$

This ciphertext is transmitted in the unsecured channel.

Decryption: Bob receives the ciphertext and uses his private key (M, D)

Bob decrypts the ciphertext to obtain the plaintext message using the RSA equation

$$\begin{aligned}P &= C^D \pmod{M} \\ P &= 13^7 \pmod{33} \\ &= 7\end{aligned}$$

Example 7.2: Bob chooses two primes $p = 11$ and $q = 13$

M is calculated as $M = p \times q$

$$M = 11 \times 13 = 143$$

Using p and q the value of L is obtained as:

$$L = \text{LCM} (p - 1, q - 1)$$

Where $t = (p - 1, q - 1)$

$$\begin{aligned}&= (11 - 1)(13 - 1) \\ &= 120\end{aligned}$$

Choose a prime E that is between 7 and $L-1$, and that is relatively prime to $(p-1)$ and $(q-1)$. E needs to be relatively prime to t (it turns out that 7 is between 1 and 120, and the GCD of 120 is 1).

Let $E = 3$

$$\begin{aligned} GCD(E, p - 1) &= 1 \\ &= GCD(7, 102) = 1 \\ \therefore & GCD(E, t) \\ &= GCD(E, (p - 1)(q - 1)) \\ &= GCD(7, 120) = 1 \end{aligned}$$

The private key D is computed as: $D \times E \equiv 1 \pmod{t}$

$$\begin{aligned} &= E^{-1} \pmod{t} \\ &7^{-1} \pmod{120} = 103 \end{aligned}$$

Which is divisible by t . The public and private keys are:

$$\begin{aligned} \textbf{Public Key} &= (E, M) = (7, 143) \\ \textbf{Private Key} &= (D, M) = (103, 143) \end{aligned}$$

Encryption: Alice encrypts the message using the RSA encryption equation $C = P^E \pmod{M}$.

If the plaintext message $P = 9$

$$\begin{aligned} C &= 9^7 \pmod{143} \\ &= 48 \end{aligned}$$

Hence, the ciphertext is $C = 48$.

This ciphertext is transmitted in the unsecured channel.

Decryption: Bob receives the ciphertext and uses his private key (M, D) .

Bob decrypts the ciphertext to obtain the plaintext message using the RSA equation.

$$\begin{aligned} P &= C^D \pmod{M} \\ P &= 48^{103} \pmod{143} \\ &= 9 \end{aligned}$$

Example 7.3: Encrypt the plaintext message “ATTACK AT SEVEN” using the RSA method and decrypt the ciphertext to obtain the plaintext message.

Solution: Divide the plaintext message into a group of three letters (including the blank space).

ATT ACK _AT _SE VEN

These block of three characters can be represented as the sum of powers of 27 (blank space is considered as “0”). Therefore, the positional values of the letters in the given plaintext message are:

$$\begin{array}{lllll} A = 1; & C = 3; & E = 5; & K = 11; & N = 14; \\ S = 19; & T = 20; & & & V = 22 \end{array}$$

Using this system of integer representation, the maximum value of a block (ZZZ) is $27^3 - 1 = 19682$. So we require a modulo M value greater than this value.

Select the public exponent $E = 3$, a pair of random primes $p = 173$ and $q = 149$, and check:

$$\begin{aligned} GCD(E, p - 1) &= GCD(3, 172) = 1 \\ GCD(E, q - 1) &= GCD(3, 148) = 1 \end{aligned}$$

$$\text{Calculate } M = p \times q = 173 \times 149 = \mathbf{25777}$$

$$\begin{aligned} L &= (p - 1)(q - 1) = (173 - 1)(149 - 1) \\ &= 172 \times 148 \\ &= \mathbf{25456} \end{aligned}$$

$$\begin{aligned} \text{The private exponent } D \text{ is computed as } D &= E^{-1} \bmod L \\ &= 3^{-1} \bmod 25456 \\ &= 16971 \\ \therefore & \quad ED = 3 \times 16971 \\ &= 50913 \\ &= 2 \times 25456 + 1 \end{aligned}$$

$$\text{i.e., } ED \equiv 1 \bmod 25456 = 1 \bmod L$$

From the previous calculations:

The public key is $(M, E) = (25777, 3)$ and the private key is $(M, D) = (25777, 16971)$. The values of p, q, d , and L are kept secret.

For example, to encrypt the first integer that represents “ATT” we have:

$$\begin{aligned} C &= P^E \bmod M \\ &= 1289^3 \bmod 25777 \\ &= 18524 \end{aligned}$$

The plaintext message ATTACK AT SEVEN is represented by a sequence of a five-integer block as: P_1, P_2, P_3, P_4 , and P_5

$$\begin{aligned} P_1 &= \text{ATT} & P_2 &= \text{ACK} & P_3 &= \text{_AT} & P_4 &= \text{_SE} & P_5 &= \text{VEN} \\ \text{ATT} \rightarrow P_1 &= 1 \times 27^2 + 20 \times 27^1 + 20 \times 27^0 = 1289 \\ \text{ACK} \rightarrow P_2 &= 1 \times 27^2 + 3 \times 27^1 + 11 \times 27^0 = 821 \\ \text{_AT} \rightarrow P_3 &= 0 \times 27^2 + 1 \times 27^1 + 20 \times 27^0 = 47 \\ \text{_SE} \rightarrow P_4 &= 0 \times 27^2 + 19 \times 27^1 + 5 \times 27^0 = 518 \\ \text{VEN} \rightarrow P_5 &= 22 \times 27^2 + 5 \times 27^1 + 14 \times 27^0 = 16187 \\ \therefore P_i &= (1289, 821, 47, 518, 16187) \end{aligned}$$

The corresponding ciphertext integers are calculated as: $C_i = P_i^E \bmod M$

$$\begin{aligned} C_1 &= 1289^3 \bmod 25777 = 18524 \\ C_2 &= 821^3 \bmod 25777 = 7025 \end{aligned}$$

$$\begin{aligned}C_3 &= 47^3 \bmod 25777 = 715 \\C_4 &= 518^3 \bmod 25777 = 2248 \\C_5 &= 16187^3 \bmod 25777 = 24465\end{aligned}$$

This sequence of integers as ciphertext is sent as C_i to the receiver who has the private key (M, D) to decrypt the ciphertext.

Where $C_i = (18524, 7025, 715, 2248)$

The receiver uses the private key $(M, D) = (25777, 16971)$

The plaintext integers are obtained by decryption process as: $P = C^D \bmod M$

$$\begin{aligned}P_1 &= 18524^{16971} \bmod 25777 = 1289 \\P_2 &= 7025^{16971} \bmod 25777 = 821 \\P_3 &= 715^{16971} \bmod 25777 = 47 \\P_4 &= 2248^{16971} \bmod 25777 = 518 \\P_5 &= 24465^{16971} \bmod 25777 = 16187\end{aligned}$$

These integers are then converted back into the block of three characters of plaintext message as:

$$\begin{aligned}P_1 &= 1289 \div 27^2 \\&= 1289 \div 729 && 1 = A \\&= 1 \text{rem} 560 \\&560 \div 27^1 = 560 \div 27 && 20 = T \\&= 20 \text{rem} 20 \\&20 \div 27^0 = 20 \div 1 && 20 = T \\&20 \text{ rem} 0\end{aligned}$$

$$\therefore P_1 = 1289 = ATT$$

$$\begin{aligned}P_2 &= 821 \div 27^2 \\&= 821 \div 729 && 1 = A \\&= 1 \text{rem} 92 \\&92 \div 27^1 = 92 \div 27 && 3 = C \\&= 3 \text{rem} 11 \\&11 \div 27^0 = 11 \div 1 && 11 = K \\&11 \text{ rem} 0\end{aligned}$$

$$\therefore P_2 = 821 = ACK$$

$$\begin{aligned}P_3 &= 47 \div 27^2 \\&= 47 \div 729 && 0 = Space \\&= 0 \text{rem} 47 \\&47 \div 27^1 = 47 \div 27 && 1 = A \\&= 1 \text{ rem} 20 \\&20 \div 27^0 = 20 \div 1 && 20 = T \\&20 \text{ rem} 0\end{aligned}$$

$$\begin{aligned}
 \therefore P_3 &= 47 = \text{_AT} \\
 P_4 &= 518 \div 27^2 \\
 &= 518 \div 729 & 0 &= \text{Space} \\
 &= 0 \text{rem}518 \\
 &\quad 518 \div 27^1 = 518 \div 27 & 19 &= S \\
 &= 19 \text{rem}5 \\
 &\quad 5 \div 27^0 = 5 \div 1 5 = E \\
 &\quad 5 \text{ rem}0 \\
 \therefore P_4 &= 518 = \text{_SE} \\
 P_5 &= 16187 \div 27^2 \\
 &= 16187 \div 729 & 22 &= V \\
 &= 22 \text{rem}149 \\
 &\quad 149 \div 27^1 = 149 \div 27 & 5 &= E \\
 &= 5 \text{rem}14 \\
 &\quad 14 \div 27^0 = 14 \div 1 & 14 &= N \\
 &\quad 14 \text{ rem}0 \\
 \therefore P_5 &= 16187 = \text{VEN}
 \end{aligned}$$

7.3.2 Attacks on RSA Cryptosystems

The RSA algorithm has been analyzed for its vulnerability, and while a number of fascinating attacks were identified, none of them are devastating. They mostly illustrate the dangers of improper use of RSA. It was found that security implementing RSA is a non-trivial task. The RSA function is an example of a one-way trapdoor function. It is not known to be easily invertible without the trapdoor D . The general attacks on RSA involve trying to compute D , or some other ways to get to the plaintext message P . There are three straight methods to enumerate all elements in the multiplicative group of M until plaintext exponential running time $O(n^e)$. Therefore, the requirement in any efficient encryption process is an algorithm with a substantial lower running time. The attacks on RSA are classified into two categories:

1. Structural attacks
2. Implementation attacks

The structural attacks focus on attacking the underlying structure of the RSA function.

Structural Attacks: The structural attacks are categorized as:

- a. Factorizing attack
- b. Elementary attack

- c. Small private key attack
- d. Small public key attack
- e. Hastad's broadcast attack

Factorization Attack: This attack is the attempt to factor the modules M , because knowing the factorization of M , one may easily obtain $F(M)$ from which D can be determined by $D = E^{-1} \bmod \phi(M)$. However, at present the fastest factoring algorithm runs in exponential time.

Elementary Attack: Elementary attack clears about the deliberate misuse of RSA. To understand such misuse would be closing common modules M to serve multiple users. For example, Alice is sending a plaintext message P to Bob, which has been encrypted by the RSA function $C = P^E \bmod M$.

Let us assume the same M is used by all users. If adversary Eve (E) want to decrypt C , she cannot, because she does not know M , the private key (M, D). However, in fact, Eve is able to use her own keys E and D to factor M , and in turn recover Bob's private key (M, D). So the resulting system is no longer secure.

Small Private Key Attack: The performance of an RSA cryptosystem depends on the size of the key. The decryption performance depends on the size of the private key used. To enhance the decryption performance, Alice might tend to use a small value of (M, D) rather than a large random number, but a smaller private key value leads to a total collapse of the RSA cryptosystem. This break of RSA is based on Wiener's Theorem, which in general provides a lower constraint for D . Wiener has proved that Eve may efficiently find the value of D when $D < (1/3 \times N^{1/4})$ and where $N = p \times q$.

Small Public Key: Similar to small private key selection to enhance the performance of the RSA decryption process, it is customary to use a small public key, but unlike the case with a small private key, attacks on small public keys turn out to be much less effective. The most powerful attacks on small public keys are based on Coppersmith's theorem.

Hastad's Broadcast Attacks: In Hastad's broadcast attack, Hastad showed that adding linear padding to a plaintext message P prior to encryption is insecure. Suppose Bob wishes to send an encrypted message (ciphertext) to a number of parties P_1, P_2, \dots, P_k .

Each party has its own RSA key (M_i, E_i) .

Eve can learn $C_i = f_i(M)^{E_i} \bmod N_i$ for $i = 1, 2, \dots, k$ if even though other parties are involved, Eve can recover the plaintext M_i from all the ciphertext.

7.4 RABIN CRYPTOSYSTEM

The Rabin cryptosystem is a public key cryptosystem that was discovered by Michael Rabin in 1979. Its security mechanism is similar to that of RSA the cryptosystem, though both depend on the difficulty of integer factorization for their security. It is an alternative of the RSA cryptosystem for which factorization of modulo “ N ” has almost the same computational complexity as obtaining the decryption transformation from the encryption transformation. The Rabin public key encryption was the first example of a provably secure public key encryption scheme. The problem faced by a passive adversary of recovering plaintext from some given ciphertext is computationally equivalent to factoring. The Rabin system uses $N = p \times q$, where p and q are prime numbers, just as in the RSA cryptosystem.

7.4.1 Rabin’s Cryptosystem Algorithm

Rabin’s cryptosystem is based on two prime numbers p and q , and forms the product $N = p \times q$, where the public key is N , and the private key is the numbers p and q . These integers are congruent to 3 modulo 4. To encrypt a plaintext message P , the equation used to get the ciphertext is $C = P^2 \pmod{N}$. The encryption and decryption steps are as given as follows:

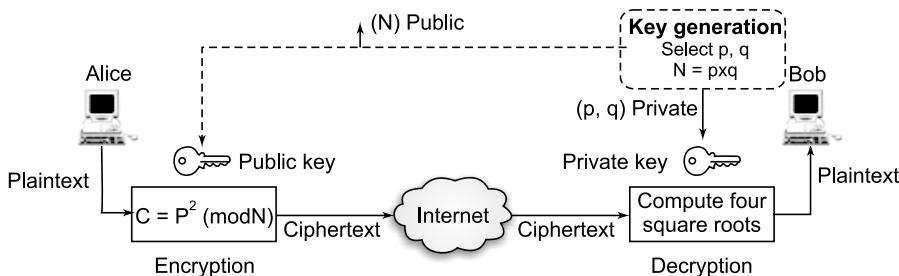


FIGURE 7.4 Encryption and decryption by the Rabin cryptosystem.

Encryption: Alice wants to send a plaintext message P to Bob. Alice encrypts the message using the following steps:

1. Alice obtains Bob’s authentic public key N , where $N = p \times q$.
2. The plaintext message is represented as an integer P in the range $\{0, 1, 2, \dots, (n - 1)\}$.
3. Alice computes and obtains the ciphertext as $C = P^2 \pmod{N}$.
4. Alice now sends the ciphertext C to Bob.

Decryption: To recover the plaintext P from C , the receiver Bob calculates the four square roots of C modulo N and chooses the correct message from the four possibilities. The four possibilities of the ciphertext are calculated as follows:

1. Determine a and b , satisfying the equation $a \times p + b \times q = 1$ (using Euclidian algorithm a and b generation stage).
2. Compute:

$$\begin{aligned} r &= C^{(p+1)/4} \bmod p \\ s &= C^{(q+1)/4} \bmod q \\ x &= (a \times p \times s + b \times q \times r) \bmod N \\ y &= (a \times p \times s - b \times q \times r) \bmod N \end{aligned}$$

3. Now the four square roots of $C \bmod N$ are:

$$\begin{aligned} m_1 &= x \\ m_2 &= -x \bmod N \\ m_3 &= y \\ m_4 &= -y \bmod N \end{aligned}$$

The problem with the Rabin cryptosystem is the decryption into four possible messages. The solution is to pad it in such a way that only one of the four possible messages fits the padding. This is done by replicating the bits of the last portion of the message and adding them to the end of the message, so only the correct decryption will have the trailing bit duplicated.

Example 7.4: Consider two prime numbers $p = 7$ and $q = 11$. $n = p \times q$

$$n = 7 \times 11 = 77$$

Using Euclidian algorithm $(-3 \times 7 + 2 \times 11) = 1$

$$\therefore \quad a = -3, \quad b = 2$$

If the plaintext message is $(5)_{10} = (101)_2$

Padding to the plaintext message $P = 101101_2 = 45_{10}$

The ciphertext $C = 45^2 \bmod 77 = 23$

Decryption computes the four square roots as: $r = 23^2 \bmod 7 = 4$

$$s = 23^2 \bmod 11 = 1$$

$$x = (-3) \times 7 \times 1 + 2 \times 11 \times 4 \bmod 77 = 67$$

$$y = (-3) \times 7 \times 1 - 2 \times 11 \times 4 \bmod 77 = 45$$

Thus two of the square roots are 67 and 45, and the other two are $77 - 67 = 10$ and $77 - 45 = 32$

Now $10_{10} = 001010_2$

$$32_{10} = 100000_2$$

$$45_{10} = 101101_2$$

$$67_{10} = 1000011_2$$

So only 101101_2 has the required redundancy and the payload is $101_2 = 5_{10}$

Example 7.5: Choose two large distinct prime numbers p and q

Choose two primes $p = 277$, $q = 331$

$$N = p \times q = 277 \times 331 = 91687$$

Alice's public key is $N = 91687$

While Alice's private key is $p = 277$, $q = 331$

Encryption: Consider a 10 bit plaintext message as: $P = 1001111001_2 = 633_{10}$. The last six bits of original message are required to be replicated prior to encryption.

Alice replicates the last 6 bits of P to obtain the 16 bit message $100111100111001_2 = 40569_{10}$.

Its decimal equivalent is $P = 40569$

Alice then computes $C = P^2 \bmod N$

$$40569^2 \bmod 91687 = 62111$$

This ciphertext is transmitted to Bob in an unsecured channel.

Decryption: Bob decrypts the ciphertext C to obtain the plaintext message P by using the decryption algorithm and his knowledge of the factor of N to compute the four square roots of $C \bmod N$:

$$r = C^{(p+1)/4} \bmod p$$

$$s = C^{(q+1)/4} \bmod q$$

$$x = (a \times p + b \times q) \bmod N \quad (a \text{ and } b \text{ satisfying the equation } a \times p + b \times q = 1)$$

$$y = (a \times p \times s - b \times q \times r) \bmod N$$

$$m_1 = x$$

$$m_2 = -x \bmod N$$

$$m_3 = y$$

$$m_4 = -y \bmod N$$

$$m_1 = 6965414 = 10001000000010110$$

$$m_2 = 22033 = 101011000010001$$

$$m_3 = 40569 = 1001111001111001$$

$$m_4 = 51118 = 1100011110101110$$

Since only m_3 has the required redundancy, Bob decrypts C to m_3 to recover the original message $P = 1001111001$.

7.5 DIFFIE-HELLMAN KEY EXCHANGE

In symmetric key encryption both the parties want to share the secret key for use. This communication in a public channel is highly unsecured. Every piece of information that they exchange is observed by their adversary Eve. How can both Alice and Bob share the key without making it available to the adversary Eve? The Diffie-Hellman key exchange mechanism provides the possible solution for this issue.

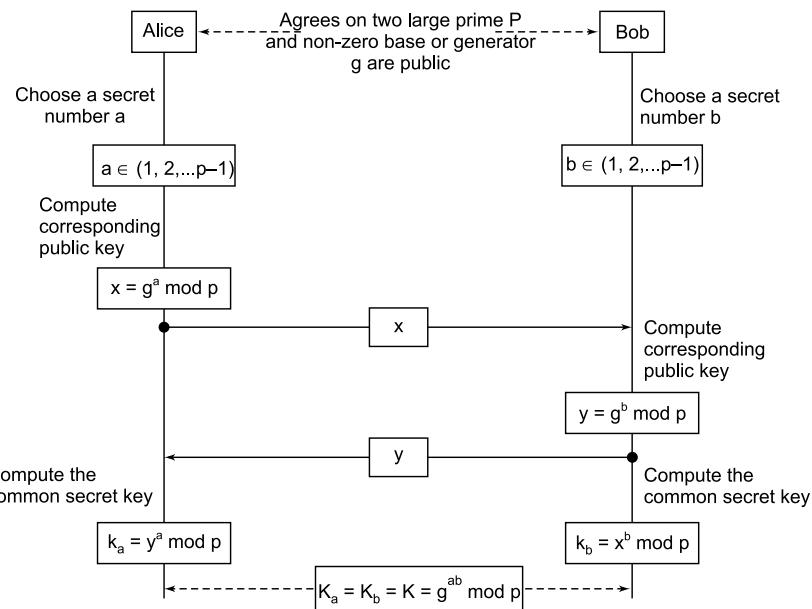


FIGURE 7.5 Diffie-Hellman key exchange.

The Diffie-Hellman key exchange algorithm is proposed in a key exchange protocol and not used for encryption. It solves the problem in sharing a secret key for use in a symmetric key cipher, but their only means of communication is insecure. The goal of the Diffie-Hellman key exchange process is for Alice and Bob to be able to agree upon a shared secret to be used by them. Both Alice and Bob independently generate keys for a symmetric encryption algorithm that will be used to encrypt the data stream to be exchanged between them. The main object of this mechanism is that neither the shared secret nor the encryption key ever travel over the network.

The steps followed for the key exchange process using the Diffie-Hellman method are:

1. Alice and Bob agree on a large prime p and a nonzero base or generator g . They make the value of p and g public knowledge.
2. Alice chooses a secret number a , that she does not reveal to anyone, and Bob chooses a secret number b that he keeps secret. Alice computes her public number x as $x = g^a \bmod p$ and Bob computes his public number y as $y = g^b \bmod p$.
3. Both Alice and Bob exchange these computed values; that is, Alice sends x to Bob and Bob sends y to Alice. By the end of this step, Alice knows p, g, a, x , and y ; and Bob knows p, g, b, x , and y .
4. Eve gets to see the values of x and y , since they are communicated over the unsecured channel.
5. Finally, using the available numbers, Alice and Bob again use their secret integers to compute the secret key as:

Alice computes the key $K_a = y^a \bmod p$

Where $y = g^b \bmod p$

$$\begin{aligned} \therefore K_a &= (g^b \bmod p)^a \bmod p \\ &= g^{ba} \bmod p \\ \therefore K_a &= g^{ba} \bmod p \end{aligned}$$

Also Bob computes the key $K_b = x^b \bmod p$

Where $x = g^a \bmod p$

$$\begin{aligned} \therefore K_b &= (g^a \bmod p)^b \bmod p \\ &= g^{ab} \bmod p \\ \therefore K_b &= g^{ab} \bmod p \end{aligned}$$

The keys both Alice and Bob computed, K_a and K_b respectively, are actually the same.

$$\therefore K_a = K_b = K = g^{ab} \bmod p$$

The Diffie-Hellman key exchange algorithm is represented in Figure 7.5.

Example 7.6: Alice and Bob agree to use the prime $p = 23$ and the primitive root $g = 5$. Alice chooses the secret key $a = 6$ and computes:

$$\begin{aligned} x &= g^a \bmod p \\ x &= 5^6 \bmod 23 \\ x &= 15625 \bmod 23 \\ &= 8 \end{aligned}$$

Bob chooses the secret key $b = 15$ and computes:

$$\begin{aligned}y &= g^b \bmod p \\y &= 5^{15} \bmod 23 \\&= 19\end{aligned}$$

Alice sends Bob the number 8 and Bob sends Alice the number 19. These values are communicated over an unsecured communication channel.

$\therefore x = 8$ and $y = 19$ are considered as public

The number $a = 6$ and $b = 15$ are not transmitted and remain secret.

Using this, Alice and Bob are both able to compute the number:

$$\begin{array}{ll}K_a = y^a \bmod p & K_b = x^a \bmod p \\& = 19^6 \bmod 23 \\& = 2 & = 8^{15} \bmod 23 \\& & = 2\end{array}$$

Then 2 is the shared secret.

In case Eve sees the entire information exchange between Alice and Bob, she can reconstitute Alice's and Bob's shared secret if she can solve either of the congruences:

$$\begin{array}{ll}g^a = y \bmod p & \text{or } g^a = y \bmod p \\5^a = 8 \bmod 23 & \text{or } 5^b = 19 \bmod 23\end{array}$$

In this example the numbers used are too small. It takes very little time for Eve's computer to check all possible powers of $g^a \bmod p$ as $5^6 \bmod 23$. It is suggested that Alice and Bob choose a prime p having approximately 1000 bits (i.e., $p \approx 2^{1000}$). Then it is very difficult for Eve to crack to get the unknown values to compute the key.

7.5.1 Diffie-Hellman Problem (DHP)

In the Diffie-Hellman key exchange process, Eve is about to eavesdrop the values of x and y , so she knows the value of g^a and g^b . She also knows the values of g and p . So if she can solve the discrete logarithm problem (DLP), then she can find the shared secret key value g^{ab} .

The security of Alice and Bob's shared key rests on the difficulty of the following, potentially easier problems. Consider p to be a prime number and g a base (integer). The Diffie-Hellman problem is the problem of computing the value of $g^{ab} \bmod p$ from the known values of $g^a \bmod p$ and $g^b \bmod p$.

It is clear that DHP is no harder than the DLP. If Eve can solve the DLP, then she can compute Alice and Bob's secret exponents a and b from the intercepted values $x = g^a$ and $y = g^b$, and then it is easy for her to compute their shared key g^{ab} . Conversely, if Eve has an algorithm that effectively solves DHP, can she use it to also efficiently solve the DLP?

7.6 ELGAMAL PUBLIC KEY CRYPTOSYSTEM

The Diffie-Hellman key exchange algorithm provides a mechanism for publicly sharing a random secret key in an unsecured communication channel. It does not meet the complete requirements of being a public key cryptosystem. ElGamal is a public key cryptosystem described by Taher ElGamal in 1985. The ElGamal algorithm provides an alternative to the RSA for public key encryption. In RSA, the security level depends on the difficulty of factorizing large integers. In the ElGamal algorithm, it is based on the discrete log problem and is closely related to the Diffie-Hellman key exchange algorithm. This is because $x \log(g) \bmod n = \log(g^x) \bmod n$, and division is easy to compute in a group, so x is easy to compute given $\log(g^x) \bmod n$.

7.6.1 ElGamal Key Generation

In an ElGamal public key cryptosystem, Alice needs a large prime number p for which the discrete logarithm problem FFP (Fork-Finder algorithm) is difficult. She also needs an element g mod p of large (prime) order where p and g are preselected from a trusted party and $g < p$, which is securely shared among Alice and Bob. Alice now selects a secret number “ a ” which she considers as her private key, and then computes the quantity:

$$A = g^a \pmod{p}$$

Where A is considered as a public key, Alice publishes her public key A and she keeps her private key “ a ” secret. Now suppose Bob wants to encrypt a plaintext message “ P ” and he uses Alice’s public key A , where Bob’s message P is an integer between 2 and p .

ElGamal Encryption: To encrypt P , Bob first randomly chooses another number $K \bmod p$. Bob uses K to encrypt only one message and then decrypts it. The number K is known as an ephemeral key, and it is used only for the purpose of encrypting a single message. Using K Bob computes two quantities C_1 and C_2 ,

$$\begin{aligned} C_1 &= g^k \pmod{p} \\ C_2 &= P \cdot A^k \pmod{p} \end{aligned}$$

This pair of numbers (C_1, C_2) is considered as Bob’s ciphertext.

ElGamal Decryption: To decrypt the ciphertext (C_1, C_2) , Alice uses her private key “ a ” and she now computes:

$$x = C_1^a \pmod{p} \text{ and hence also } x^{-1} = (C_1^a)^{-1} \pmod{p}$$

She then multiplies C_2 by x^{-1} . This results the plaintext message P given as follows:

$$\begin{aligned} \text{If } C_1 &= g^K & x^{-1} \cdot C_2 &= (C_1^a)^{-1} \cdot C_2 \pmod{p} & \text{Where } x = C_1^a \pmod{p} \\ \therefore & \quad C_2 = P A^K \pmod{p} \\ & \quad x^{-1} \cdot C_2 & &= ((g^{ak})^{-1} \cdot (P A^K) \pmod{p}) \\ & & &= ((g^{ak})^{-1} \cdot (P(g^a)^K) \pmod{p}) & \text{Since } A = g^a \pmod{p} \\ & & &= P(g^{ak})^{-1} \cdot (g^{ak}) \pmod{p} \\ & & &= P \end{aligned}$$

To decrypt the message P , Eve intercepts the public key parameters p and g and Alice's public key $A = g^a \pmod{p}$.

If Eve can solve the discrete logarithm problem, she can find a and decrypt the message; otherwise, it is difficult for Eve to decrypt the ciphertext.

Example 7.7: Alice uses the prime $p = 2579$ and the primitive root $g = 2$. She chooses $a = 765$ to be her private key and computes her public key.

$$\begin{aligned} A &= g^a \pmod{p} \\ A &= 2^{765} \pmod{2579} \\ &= 949 \end{aligned}$$

Bob decides to send Alice the plaintext message $P = 1299$. He chooses an ephemeral key at random, say he chooses $K = 853$, and then he computes the ciphertext pair (C_1, C_2) as:

$$\begin{aligned} C_1 &= g^k \pmod{p} \\ C_1 &= 2^{853} \pmod{2579} \\ &= 435 \\ \text{And } C_2 &= P \cdot A^k \pmod{p} \\ C_2 &= 1299 \cdot 949^{853} \pmod{2579} \\ &= 2396 \end{aligned}$$

When Alice receives the ciphertext as (C_1, C_2) or $(435, 2396)$, she uses her private key to decrypt it to obtain the plaintext message P .

$$\begin{aligned} \text{She computes } x &= C_1^a \pmod{p} \\ &= 435^{765} \pmod{2579} \end{aligned}$$

$$\begin{aligned} \text{She then multiplies } C_2 &\text{ by } x^{-1} \\ x^{-1} \cdot C_2 &= P \cdot (g^{ak})^{-1} \cdot (g^{ak}) \pmod{p} \\ i.e., \quad P &= \frac{C_2}{x} = \frac{2396}{435^{769} \pmod{2579}} \end{aligned}$$

$$\begin{aligned} P &= 2396 \times (435^{769})^{-1} \pmod{2579} \\ P &= 1299 \end{aligned}$$

Example 7.8: Alice uses the prime $p = 2357$ and the primitive root $g = 2$. She chooses $a = 175$ to be her private key. Show the encryption and decryption process.

Encryption: Alice computes her public key as:

$$\begin{aligned} P &= 2035, & K &= 1520 \\ C_1 &= g^k \pmod{p} \\ C_1 &= 2^{1520} \pmod{2357} \\ &= 1430 \\ C_2 &= P \cdot A^k \pmod{p} \\ &= 2035 \times (1185)^{1520} \pmod{2357} \\ &= 697 \end{aligned}$$

Bob sends the ciphertext as $(C_1, C_2) = (1430, 697)$

Decryption: $x = C_1^a \pmod{p}$

$$\begin{aligned} x^{-1} \cdot C_2 &= P(g^{ak})^{-1} \cdot g^{ak} \pmod{p} \\ P &= \frac{C_2}{x} = \frac{697}{C_1^a \pmod{p}} = \frac{697}{1430^{175} \pmod{2357}} \\ &= 697 \times (1430^{175})^{-1} \pmod{2357} \\ &= 697 \times 872 = 2035 \end{aligned}$$

Example 7.9: Alice uses the prime $p = 17$ and the primitive root $g = 6$, and she chooses $a = 5$ to be her private key and computes her public key as:

$$A = g^a \pmod{p} = 6^5 \pmod{17} = 7$$

Bob decides to send Alice the plaintext message $P = 13$. He chooses a random ephemeral key $K = 10$, then computes the ciphertext pair (C_1, C_2) :

$$\begin{aligned} C_1 &= g^k \pmod{p} & C_2 &= P \times A^k \pmod{p} \\ &= 6^{10} \pmod{17} & &= 13 \times 7^{10} \pmod{17} \\ &= 5 & &= 9 \end{aligned}$$

Bob sends $C_1 = 15$ and $C_2 = 9$ to Alice.

Alice receives $C_1 = 15$ and $C_2 = 9$ from Bob.

Her public key is $(p, g, A) = (17, 6, 7)$ and her private key is $a = 15$.

Alice now decrypts the ciphertext using the private key.

$$\begin{aligned} \text{Decryption factor } (C_1^{-a}) \times C_2 \pmod{p} &= 15^{-5} \times 9 \pmod{17} \\ &= 15^{11} \times 9 \pmod{17} \\ &= 9 \end{aligned}$$

$$\text{Decryption } (C_2 \times 9) \pmod{p} = 9 \times 9 \pmod{17} = 13$$

Alice has now decrypted the ciphertext and received the plaintext message $P = 13$.

7.7 ELLIPTICAL CURVE CRYPTOSYSTEM (ECC)

An elliptical curve cryptosystem is a public key cryptography method. Unlike other popular algorithms such as RSA and ElGamal, ECC is based on a discrete logarithm that is much more difficult to challenge at an equivalent key length. ECC is an asymmetric cryptography algorithm which involves some high level calculation using mathematical curves to encrypt and decrypt data. It is similar to RSA in application. The modular multiplication and modular exponentiation in RSA is equivalent to ECC operations of addition and multiplication of the points on an ellipsis, by an integer respectively.

The ECC was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington). The ECC is relatively new and uses an elliptical curve as a base for the cryptographic system. The elliptical cryptosystem is considered as the first encryption system applicable for handheld devices such as mobile phones and sensor networks for secured information communication by these devices. Their physical resources such as processor capacity and memory are not nearly as abundant as on their desktop and laptop counterparts. A normal desktop or laptop system has no problems working with 2048 bit keys and higher, but these small embedded devices do, since we do not want to spend a lot of their resources and bandwidth securing traffic.

The security of RSA mainly depends on the difficulty of factorizing a large integer which can be done in sub-exponential times. For the elliptical curve discreet logarithm problem (ECDLP), however, only exponential algorithms are known, which means we can use shorter keys for security levels, whereas RSA and ElGamal would need much bigger keys. For example, the security levels offered by both a 160 bit ECC key and a 1024 bit RSA key are similar. Also to reach the same level of security, one needs a 15360 bit RSA key, while one only needs a 512 bit ECC key. Another advantage of using ECC is it reduces the processing overhead compared to RSA with the same level of security.

7.7.1 Elliptic Curve

An elliptic curve will simply be the set of points described by the equation

$$y^2 = x^3 + ax + b \quad \dots(1)$$

Equation (1) is known as the Weierstrass normal form for elliptic curves. The different shapes of elliptic curves for $b = 1$ and “ a ” varying from 2 to -3 is given in Figure 7.6.

Based on the value of “ a ” and “ b ,” an elliptic curve may assume different shapes on the plane. As it can be easily seen and verified, elliptic curves are symmetric about the x -axis. For our representation we consider a point on infinity (also known as an ideal point) to be part of our curve. This point is denoted as a point at infinity with the symbol 0 (zero).

7.7.2 The Group Law of Elliptic Curves

A group is a set for which the binary operation defined is known as addition. In order for the set “ G ” to be a group, addition must be defined so that it represents the following four properties:

1. **Closure:** If a and b are members of G , then $a+b$ is a member of G .
2. **Associativity:** $(a + b) + c = a + (b + c)$.
3. **Existence of Identity:** There exists an identity element “ 0 ” such that $a + 0 = 0 + a = a$.
4. **Existence of Inverse:** Every element has an inverse; that is, for every “ a ” there exists b such that $a + b = 0$.
5. **Commutativity:** $a + b = b + a$.

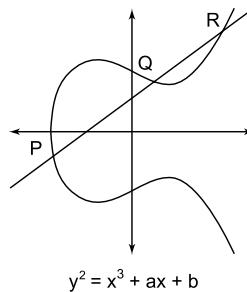


FIGURE 7.6 Elliptic curves.

Abelian Group: If it is an Abelian group, along with the four properties such as closure, associativity, identity, and inverse, the commutative property must also be satisfied. A group in which the group operation is not commutative is called a *non-abelian group* or *non-commutative group*. The group over an elliptic curve has the following important features:

- The elements of the group are the points of an elliptic curve.
- The identity element is the point at infinity.
- The inverse of the point P is the one symmetric about the x -axis as in Figure 7.7.
- Addition is given as:

For a three aligned, nonzero points P , Q , and R , this sum is: $P + Q + R = 0$.

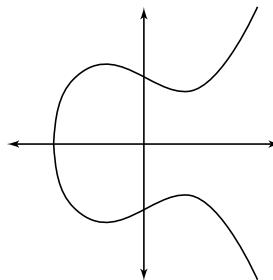


FIGURE 7.7 Symmetric nature of an elliptic curve.

With these three alignments of points P , Q , and R , it proves that the addition operator is both associative and commutative. It is in an Abelian group.

An elliptic curve is represented as $E_p(a,b)$

Where P is a prime number and a, b are restricted to $\text{mod } P$.

The equation of an elliptic curve is described by the calculation of the circumference of an ellipse (i.e., a cubic equation with highest degree of 3).

The maximum number of points lie between:

$$P + 1 - 2\sqrt{P} \leq N \leq P + 1 + 2\sqrt{P}$$

7.7.3 Geometric Addition

In an Abelian group, we can write:

$$P + Q + R = 0 \quad \text{as } P + Q = -R \quad \dots(2)$$

A geometric method can be used to find the sum between two points P and Q . Draw a line passing through the points P and Q ; this line will intersect a third point on the curve at R as shown in Figure 7.8.

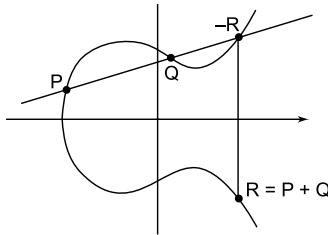


FIGURE 7.8 Addition operation using an elliptic curve.

The point symmetric to point R is $-R$, which is the result of $P+Q$ as in equation (2).

$$\begin{array}{lll} \text{If } P = 0 \text{ or } Q = 0 & P + 0 = P & Q + 0 = Q \\ \text{For any } P \text{ and for any } Q, & & \end{array}$$

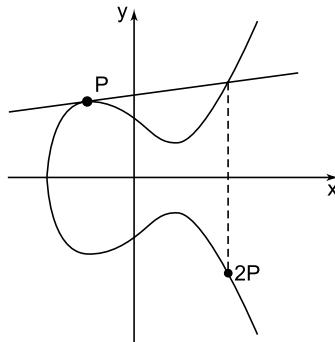


FIGURE 7.9 ECC when $P=Q$.

If $P = -Q$:

The line passing through these two points is vertical and does not intersect any third point.

$$P + Q = P + (-P) = 0 \text{ from the definition of the inverse.}$$

If $P = Q$:

There are infinite lines passing through the point. As the two points become closer together, the line passing through them becomes tangent to the curve as in Figure 7.9.

$$P + P = -R$$

where R is the point of intersection between the curve and the line tangent to the curve in P .

If $P \neq Q$ then there is no third point R in the graph. In this case, the line passing through P and Q is also tangent to the curve, as in Figure 7.10.

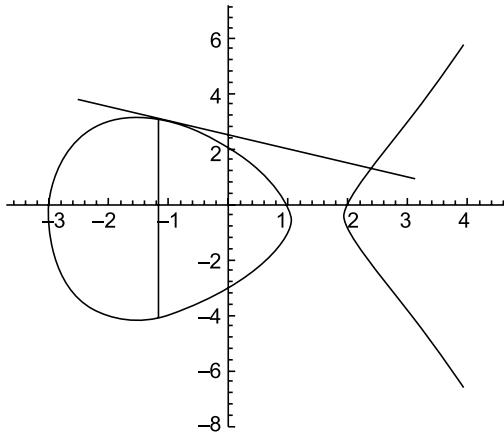


FIGURE 7.10 ECC when $P \neq Q$.

Since the line intersects just two points, then it means that it is tangent to the curve. Let P be the tangency point. In the previous case $P + P = -Q$.

Or $P + Q = -P$

If on the other hand Q is the tangency point, the correct equation would have been $P + Q = -Q$.

7.7.4 Algebraic Addition

Algebraic addition can be performed by transforming the previous rule into a set of equations.

If a straight line intersects an elliptic curve at three points, the coordinates of the first two points are:

$$P = (x_p, y_p) \text{ and } Q = (x_Q, y_Q)$$

Then the coordinates of the third point are:

$$x_R = \lambda^2 - x_p - x_Q \quad y_R = y_p + \lambda(x_R - x_p)$$

With the relationship between P , Q , and R

$$P + Q = -R$$

We can therefore write the expression for the x and y coordinates with the addition of two points P and Q .

$$x_{P+Q} = \lambda^2 - x_p + x_Q \quad y_{P+Q} = \lambda(x_p + x_R) - y_p$$

Since the y-coordinate of the reflection $-R$ is the negative of the y-coordinate of the point R on the intersecting straight line.

Where λ is the slope of the line:

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}$$

In the same way, for the curve $E : y^2 + xy = x^3 + ax^2 + b$

$R = P + Q = (x_R, y_R)$ can be determined by the formula:

$$x_R = \lambda^2 + \lambda + x_p + x_Q + a \quad y_R = \lambda(x_p + x_R) + x_R + y_p$$

$$\text{Where } \lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

For a curve $E: y^2 = x^3 + ax + b$

Let $P = (x_p, y_p)$ and $Q = (x_Q, y_Q) \in E$ with $P \neq Q$

Then $R = P + Q = (x_R, y_R)$

$$\begin{aligned} \text{where: } x_R &= \lambda^2 - x_p - x_Q \\ y_R &= y_p + \lambda(x_R - x_p) \quad \text{or} \quad y_R = y_Q + \lambda(x_R - x_Q) \end{aligned}$$

$$\text{Hence: } (x_p, y_p) + (x_Q, y_Q) = (x_R, -y_R) \quad (\because P + Q = -R)$$

Example 7.10: Consider the point addition in the curve

$$\begin{aligned} y^2 &= x^3 - 7x \\ x_R &= \lambda^2 - x_p - x_Q \\ &= 1.1982^2 + 2.35 + 0.1 \\ &= 0.389 \end{aligned}$$

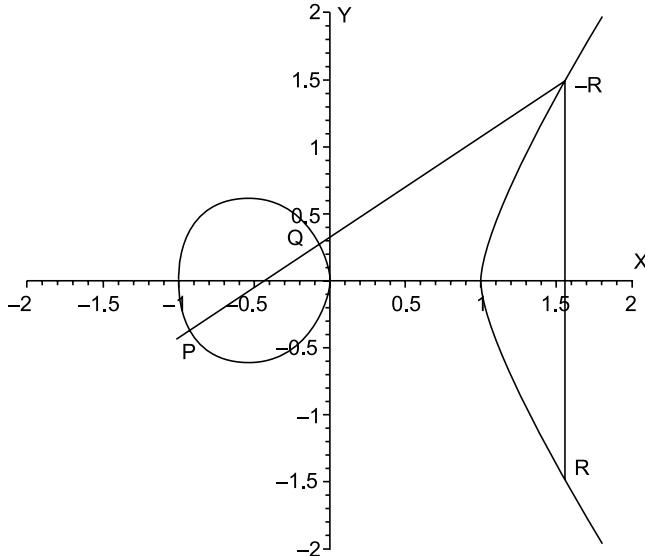
$$\begin{aligned} y_R &= \lambda(x_p + x_R) - y_p \\ &= 1.1982(-2.35 - 3.89) + 1.86 \\ &= -5.62 \end{aligned}$$

$$\text{Where } \lambda = \frac{y_Q - y_P}{x_Q - x_P}$$

$$\lambda = \frac{0.836 - 1.86}{-0.1 + 2.35} = 1.1982$$

The points are:

$$\begin{array}{ll} P = (-2.35, -1.86) & Q = (-0.1, 0.836) \\ -R = (3.89, 5.6) & R = (3.89, -5.62) \\ P + Q = R = (3.89, -5.62) \end{array}$$



Example 7.11: Find the point in elliptic curve $E_{11}(1, 1)$ where $P = 11$, $a = 1$, and $b = 1$. The elliptic curve equation is $y^2 = x^3 + ax + b$ (use modP if needed).

Substitute P , a , and b values in the previous equation:

$$y^2 = x^3 + x + 1 \quad x = 0; y = +1, -1$$

The points are $(0, 1)$ and $(0, -1)$.

Since $(0, -1)$ is negative, take modulo P.

One of the points on the curve is $(0, 1)$, $(0, 10)$.

Point Doubling

To the point P on an elliptic curve, draw the tangent line. The line intersects the elliptic curve at the point $-R$. The reflection of the point $-R$ with respect to the x -axis gives the point R , which is the result of doubling point P as in Figure 7.11.

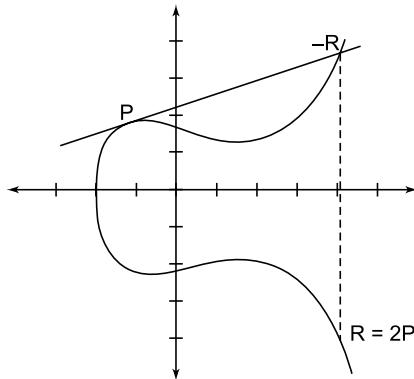


FIGURE 7.11 Point doubling.

For the curve $E : y^2 = x^3 + ax + b$

Let $P = (x_p, y_p) \in E$ with $P \neq -P$

Then $R = 2P = (x_R, y_R)$ is determined by the following equations:

$$x_R = \lambda^2 - 2x_p$$

$$\lambda = \frac{3x_p^2 + a}{2y_p}$$

Where $\lambda = \frac{3x_p^2 + a}{2y_p}$

In the same way for the curve: $E : y^2 + xy = x^3 - ax^2 + b$

$R = 2P = (x_R, y_R)$ can be determined by the following equations:

$$x_R = \lambda^2 + \lambda + a$$

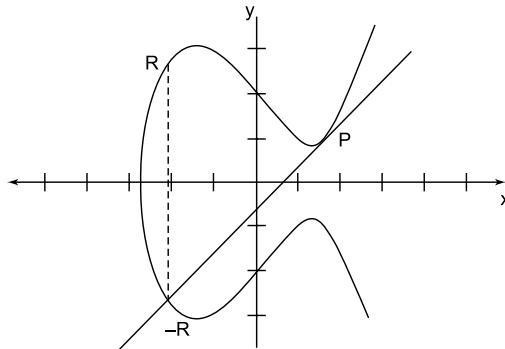
$$y_R = x_p^2 + \lambda x_R + x_R$$

Where $\lambda = x_p + y_p/x_p$

Example 7.12: Consider the point doubling curve: $y^2 = x^3 - 3x + 5$

$$\begin{aligned} x_R &= \lambda^2 - 2x_p \\ &= 1.698^2 - 2 \times 2 \\ &= -1.11 \end{aligned} \quad \begin{aligned} y_R &= \lambda(x_p - x_R) - y_p \\ &= 1.698(2 + 1.11) - 2.65 \\ &= 2.64 \end{aligned}$$

$$\lambda = \frac{3x_p^2 + a}{2y_p} = \frac{3 \times 2^2 \times (-3)}{2 \times 2.65} = 1.698$$



The points are;

$$\begin{aligned} P &= (2, 2.65) & -R &= (-1.11, -2.64) & R &= (-1.11, 2.64) \\ 2P &= R = (-1.11, 2.64) \end{aligned}$$

7.7.5 Multiplication

Here only scalar multiplication can be allowed. It is the process of a repeated addition operation. In scalar multiplication a natural number “n” is multiplied as $nP = p + p + \dots + p$ (n – times).

Where n is a natural number, computing nP requires n addition. If n has K binary digits, then the algorithm would be $\mathcal{O}(2^K)$. This increases the time complexity. The alternative for this is the double and add algorithm. Its operation can be explained with an example as follows:

Take $n = 151$

Its binary representation is $151_{10} = 1001\ 0111_2$.

This binary representation can be turned into a sum of powers of two.

$$\begin{aligned} 151 &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + \\ &\quad 1 \times 2^1 + 1 \times 2^0 \\ &= 2^7 + 2^4 + 2^2 + 2^1 + 2^0 \\ \therefore \quad nP &= 151 \times P \\ &= 2^7 P + 2^4 P + 2^2 P + 2^1 P + 2^0 P \end{aligned}$$

Based on the double and add algorithm:

- Take P
- Double it, so that we get $2P$
- Add $2P$ to P (in order to get the result $2^1 P + 2^0 P$)
- Double $2P$, so that we get $2^2 P$
- Add it to the result (so that we get $2^2 P + 2^1 P + 2^0 P$)

- Don't perform any addition involving $2^3 P$
- Double $2^3 P$ to get $2^4 P$
- Add it to our result (so that we get $2^4 P + 2^2 P + 2^1 P + 2^0 P$)

The result $151P$ is obtained by performing just seven doubling and four addition operations. Every doubling and adding operation for both $\Theta(1)$ operations. Then this algorithm is $\Theta(\log n)$. This is better than the initial $\Theta(n)$ algorithm.

7.7.6 Operation on an Elliptic Curve

ECC is similar to RSA in application. The modular multiplication and modular exponentiation in RSA is equivalent to ECC operations of addition and multiplication of points on an elliptic curve by an integer respectively. In RSA the security is based on the assumption that it is difficult to factor a large integer composed of two large prime factors. So RSA needs a large key size to be secure and unbreakable. But for ECC it is possible to use smaller primes, or a smaller finite field, with an elliptic curve to achieve the same degree of security.

7.7.7 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Fundamentally every cryptosystem is a hard mathematical problem that is computationally infeasible to solve. The discrete logarithm problem is the basis for the security of many cryptosystems including ECC. More specifically the ECC relies upon the difference of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

In the multiplicative group, the discrete logarithm problem is given elements r and q of the group and a prime p ; find a number k such that $r = qk \text{ mod } p$.

If the elliptic curve group is described using multiplicative notation, then the elliptic curve discrete logarithm problem is:

Given point P and Q in the graph, find the number so that $Pk = Q$

where k is called the discrete logarithm of Q to the base P . When the elliptic curve group is described using additive notation, then the elliptic curve discrete logarithm problem is:

Given P and Q in the group, find a number k such that $Pk = Q$.

Example 7.13: Consider an elliptic curve defined by $y^2 = x^3 + 9x + 17$.

Find the discrete logarithm k of point $Q = (4, 5)$ to the base $P = (16, 5)$.

One way to find k is to compute multiples of P until Q is found. The first few multiples of P are:

$$\begin{aligned} P &= (16, 5) & 2P &= (20, 20) & 3P &= (14, 14) & 4P &= (19, 20) \\ 5P &= (13, 10) & 6P &= (7, 3) & 7P &= (8, 7) & 8P &= (12, 17) \\ 9P &= (4, 5) \end{aligned}$$

Since $9P = (4, 5) = Q$, the discrete logarithm of Q to the base P is $k = 9$. In the real application k would be selected to be large enough such that it would be infeasible to determine k in simple steps.

7.7.8 ECC Advantages and Disadvantages

Advantages:

1. Greater flexibility in choosing a cryptographic system
2. No known sub-exponential time algorithm for ECDLP
 - Smaller key sizes (with the same security). Current recommendation for the minimum key size for ECC should be 132 bits vs. 952 bits for RSA.
3. The computational speed of ECC is greater than its other counterparts. It consumes less storage space. As a result ECC can be used in smart cards, cellular phones, pagers, etc.

Disadvantages:

1. Hyperelliptic cryptosystems offer even smaller key sizes.
2. ECC is mathematically more complex than RSA or SDL.

7.8 ELGAMAL ENCRYPTION USING ELLIPTIC CURVE CRYPTOGRAPHY

The ElGamal cryptosystem described in the previous section is directly based on ECDLP. In the elliptic curve version of the encryption P and Q are two points on the elliptic curve. The multiplication operation is replaced by addition and the exponentiation operation by multiplication. The entire process is described as follows:

Let us assume that Alice wants to send a message M to Bob using an unsecured communication channel that can be accessed by Eve. The message is divided into two pairs of message blocks M_1 , and M_2 . Bob chooses an elliptic curve (C) and secrete integer “ a ” where “ a ” is his private key. Select a fixed point $P(x_1, y_1)$, which lies on the curve C . Bob with his private key computes his public key $R = aP$.

Alice then transmits the pair of messages M_1 and M_2 and chooses a random integer “ k .” She uses fixed point P to compute $Q = kP$, and Bob’s public key to compute $kR = kaP = (x', y')$.

Then Alice sends Bob point Q and a pair of field elements $(m_1, m_2) = (M_1x', M_2y')$. To read the message Bob uses his private key to compute $aQ = akP = (x', y')$, and he can decrypt the message by two division $M_1 = m_1(x')^{-1}$ and $M_2 = m_2(y')^{-1}$. The ElGamal ECC scheme is shown in Figure 7.12.

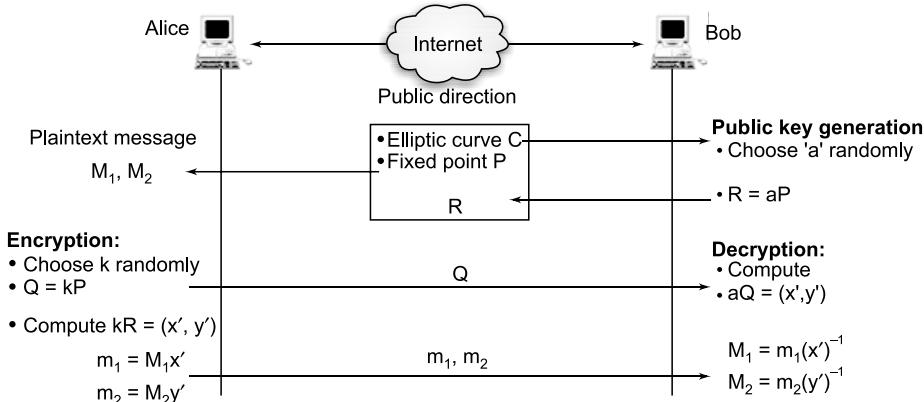


FIGURE 7.12 ElGamal ECC scheme.

7.9 ELLIPTIC CRYPTOGRAPHY AND DIFFIE-HELLMAN KEY EXCHANGE

There are several different standardized key exchange protocols. A very basic one is the Elliptic Curve Diffie-Hellman protocol described as follows. It works in the following way. To agree on a shared key Alice and Bob individually gerate key pairs (Pa, Qa) and (Pb, Qb) . They then exchange the public keys Qa and Qb , such that each can compute the point $P = PaQb = PbQa$ using their respective private keys.

The shared secret key is derived from P by a key derivation function generally being applied to its x -coordinates.

The steps involved in Diffie-Hellman by exchange for an elliptic curve is represented in Figure 7.13 and explained as follows:

1. Alice and Bob agree on an elliptic curve E over a finite field Fq , so the discrete logarithm problem is hard in $E(Fq)$. They also agree on a point $P \in E(Fq)$ such that the sub group generated by P has a large order (usually prime).
2. Alice chooses a secret integer “ a ” to compute $P_a = aP$ and sends P_a to Bob.
3. Bob chooses secret integer “ b ” compute and sends Pb to Alice.
4. Alice computes $aP_b = abP$
Bob computes $bP_a = abP$
5. Alice and Bob agree on a method to exchange a key from abP .

The only information the eavesdropper Eve has is the curve E , the first filed Fq , and the points P , aP , and bP .

She will therefore need to solve:

The Diffie-Hellman problem for Elliptic curves—given P , aP , and bP in $E(Fq)$. If Eve can solve discrete logs in $E(Fq)$ then she could use P and aP to find “ a .” However, if E and Fq are chosen carefully, then this is considered computationally infeasible.

Example 7.14: Consider an elliptic curve $y^2 = x^3 + 11x + b \bmod(167)$

Let us consider a point $P(x, y) = (2, 7) \in E(F_q)$ on the elliptic curve given in the previous equation on substituting the value of (x, y) $x = 2, y = 7$.

In the equation for the elliptic curve $y^2 = x^3 + 11x + b \bmod(167)$ we get $b = 19$.

The elliptic curve $y^2 = x^3 + 11x + b \bmod(167)$ and the point $(x, y) = (2, 7)$ are agreed publicly by Alice and Bob.

Alice chooses a secret integer $a = 15$ and calculates $P_a = aP = 15(2, 7) = (102, 88)$.

Alice sends P_a to Bob.

Bob chooses a secret integer $b = 22$ and calculates $P_b = bP = 22(2, 7) = (9, 43)$.

Bob sends P_b to Alice.

Alice computes $aP_b = 15(9, 43) = (131, 140)$

Similarly, Bob computes $bP_a = 22(102, 88) = (131, 140)$

Here Alice and Bob have securely generated the point (131, 140). They will have previously agreed some way to extract a key from this point.

Any eavesdropper would know the elliptic curve equation (1) and the points (2, 7), (102, 88) and (9, 43) obtains (131, 140). Though Eve would have to solve the Diffie-Hellman problem for the elliptic curve.

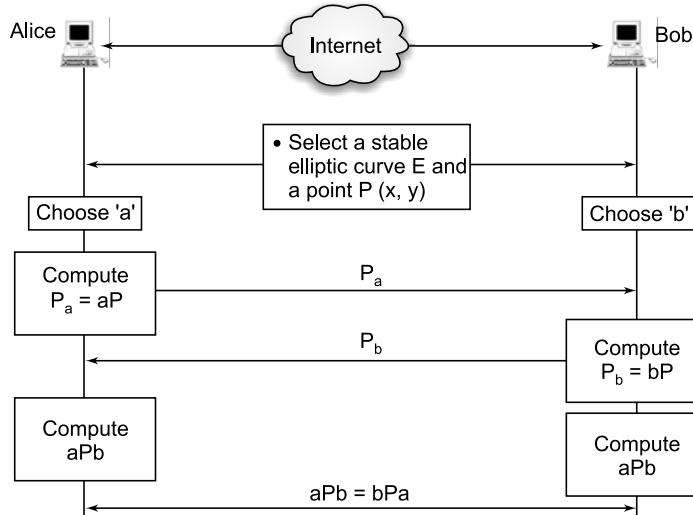


FIGURE 7.13 ECC-DH scheme.

TABLE 7.1 Comparison Between Symmetric and Asymmetric Key Cryptography.

Symmetric	Asymmetric
1. Uses identical keys to both encrypt and decrypt the data.	1. An asymmetric uses two related keys (public and private) to encrypt and decrypt the data.
2. Symmetric key algorithms are much faster computationally than asymmetric key algorithms.	2. Asymmetric key algorithms are slower due to the computational complexity in the encryption process.
3. Symmetric encryption is typically more efficient than the asymmetric encryption process and is often used for bulk data encryption.	3. Asymmetric encryption algorithms typically involve exponential operations; they are not lightweight in terms of performance. For this reason asymmetric algorithms are often used to secure key exchange rather than used for bulk data encryption.

SUMMARY

- There are some shortcomings in a symmetric key cryptosystem.
- Some of the popular asymmetric algorithms are RSA, Robin, and ElGamal algorithms.
- In a public key cryptosystem the messages are encrypted with public keys and decrypted with a private key.
- Some of the examples of public key algorithms are: RSA, Robin, ElGamal, Diffie-Hellman, Elliptic Curve Cryptosystem, and Digital Signature Algorithm.
- RSA's security is mainly based on two prime numbers p and q and their product $N = pq$, where N is the public key and the private key is p and q .
- The Diffie-Hellman key exchange mechanism provides the possible solution for the key exchange between the parties.
- The ElGamal algorithm is based on the discrete log problem and is closely related to the Diffie-Hellman key exchange algorithm.
- Elliptic curve cryptography is an asymmetric cryptography algorithm which involves some high level calculation using mathematical curves to encrypt and decrypt data.
- The security levels offered by both a 160 bit ECC key and a 1024 bit RSA key are similar.

REVIEW QUESTIONS

1. What drawbacks to symmetric and asymmetric encryption are resolved by using a hybrid method like Diffie-Hellman?
2. The values of the public key and private key are $(N, E) = (33, 3)$ and $(N, D) = (33, 7)$. Use the RSA algorithm to encrypt the word “Technology” and also show how the word can be decrypted from its encrypted form.
3. Illustrate the ElGamal encryption and decryption algorithm.
4. Write about key generation, encryption, and decryption in ElGamal cryptosystems.
5. Let $p = 353$ and $g = 3$, $a = 97$, and $b = 233$. Use the Diffie-Hellman key exchange algorithm to find K_a , B_b and secret key K .
6. Describe public and private keys in the ECC system and explain about security of ECC.

7. Briefly explain the Diffie-Hellman key exchange algorithm.
8. What are discrete logarithms? Explain how are they used in public key cryptography.
9. What are the attacks that are possible on RSA?
10. Explain the ElGamal cryptosystem with examples.
11. Discuss the security of the ElGamal cryptosystem.
12. Differentiate between symmetric and asymmetric encryption schemes.

MULTIPLE CHOICE QUESTIONS

1. An asymmetric key is also called:
(a) Secret key (b) Public key
(c) Private Key (d) None of the above
2. RSA stands for:
(a) Rivest Shamir and Adleman (b) Rock Shane and Amozen
(c) Rivest Shane and Amozen (d) Rock Shamir and Adleman
3. The RSA algorithm uses a variable sized key that is usually between _____ and _____ bits.
(a) 256, 1028 (b) 256, 2048 (c) 512, 1024 (d) 512, 2048
4. If an efficient algorithm for factoring large numbers is discovered, which of the following schemes will be known to be not secure?
(a) Diffie-Hellman (b) RSA
(c) AES (d) None of the above
5. The symmetric (shared) key in the Diffie-Hellman protocol is:
(a) $K = g^{xy}$ and p (b) $K = g^{xy} \bmod q$
(c) $K = (R^2)x$ (d) All of the above

6. The performance of the RSA cryptosystem depends on:
- (a) Size of the key
 - (b) Ciphertext
 - (c) Sender and Receiver System
 - (d) Algorithm
7. In RSA p and q are the two prime numbers, _____ is the public key and _____ is the private key.
- (a) $p \times q; M$
 - (b) $M; p \times q$
 - (c) $p; q$
 - (d) $M; p$
8. ECC is a _____ type of cryptographic algorithm.
- (a) Private key
 - (b) Symmetric key
 - (c) Asymmetric key
 - (d) Session key
9. If the straight line is tangential to the elliptic curve at P , $R =$
- (a) P^2
 - (b) $-P$
 - (c) P
 - (d) $2P$
10. In an ECC it is possible to use _____ primes, or a _____ finite field with an elliptic curve to achieve same degree of security.
- (a) smaller; smaller
 - (b) larger; smaller
 - (c) smaller; larger
 - (d) larger; larger

PART

2

SECURITY SYSTEMS

MESSAGE INTEGRITY AND MESSAGE AUTHENTICATION

8.1 INTRODUCTION

Message integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle. Data must not be changed in transit, and steps must be taken to ensure that data cannot be altered by unauthorized people. These measures include file permission and user access control. Message authentication determines whether the contents of a message come from their supposed source without being modified by a malicious interloper. The two services provided by message authentication are to ensure message integrity and to verify who sent the message. This chapter deals with various techniques used for message integrity and message authentication.

Message confidentiality alone cannot provide all the security requirements of a transmitted message. In a computer system it is not always possible for humans to scan the information to determine if data has been erased, added, or modified. Even if scanning were possible, the individual may not have an idea about what the correct data is. For example, a financial transaction of ₹ 10,000.00 may be changed to ₹ 100,000.00. The message integrity is the validity of a transmitted message. It deals with the methods of ensuring that the contents of a message have not been tampered with. By adopting data integrity, the internal and external data consistency can be achieved by preventing unauthorized changes in data. This ensures the data attributes like timeliness and completeness are consistent with requirements. One of the commonly used techniques for message integrity is called the *hash*

function. A hash function takes a message of any length and computes a product value of fixed length. The product is referred to as a *hash value*. The length of the original message does not alter the length of the hash value. Hash functions are used to ensure the integrity of a message. It provides a digital fingerprint of message content, which ensures that the message has not altered by an intruder.

8.2 MESSAGE INTEGRITY

8.2.1 Documents and Fingerprints

Document fingerprinting makes it easier to protect the information by identifying standard formats that are used throughout the organization. Document fingerprinting is performed by algorithms that map data such as documents and files to shorter text strings, also known as *fingerprints*. These fingerprints are unique identifiers for their corresponding data and file. It is used to identify sensitive information so that it can be tracked and protected accordingly. Generally document fingerprinting is a traditional data loss prevention (DLP) technique.

How Does Document Fingerprinting Work?

Different documents and data types often follow unique word patterns. Traditional DLP solutions identify those unique word patterns within the documents and then create document fingerprints based on them. To obtain the document fingerprinting, the DLP agent uses an algorithm. It converts the word pattern of the document into a document fingerprint, which is a small Unicode XML file containing a unique hash value representing the original text, and the fingerprint is saved as a data classification in the active directory. The document now becomes a sensitive information type that you can associate with a DLP policy.

Benefits of Document Fingerprinting

The primary benefit of file fingerprinting is the ability to automate and scale the process of identifying and tagging sensitive information on a network. After a DLP solution creates document fingerprints and associates files with their appropriate DLP policies, the program detects network traffic such as e-mail, TCP, FTP, or web uploads that contain documents matching fingerprinted data in order to apply protection based on those DLP policies. Protections deployed based on document fingerprints may include blocking transmission, preventing file access, or encrypting sensitive data.

8.2.2 Message Digest (MD)

A message digest is a cryptographic hash function containing a string of digits created by the one-way hash formula. The purpose of its design is to protect the integrity of data or information. It detect the changes and alteration to any part of the data or information. Message digest functions are also called *hash functions*. They are used to produce the digital summary of information called a *message digest*. Message digests are commonly of 128 bits or 160 bits in length and provide a digital identifier for each digital file or document.

The message digest functions are mathematical functions that process information to produce a different message digest for each unique document. Identical documents have the same message digest, but if even one of the bits for the document changes, the message digest changes. Figure 8.1 shows the 160 bit message digest produced for different documents. The size of the message digest is much shorter than the data from which the digests are generated, and the digests have a finite length. Good message digests use the one-way function to ensure that it is mathematically and computationally infeasible to detect the message digest process and discover the original data.

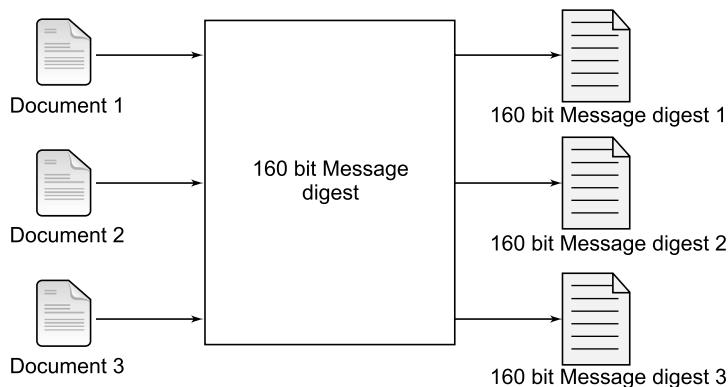


FIGURE 8.1 Message digest process using a 160 bit message digest.

A collision can exist when there are duplicate message digests for different data sets. Finding collisions for good message digest functions is also mathematically and computationally infeasible but possible, given enough time and computational effort. Message digests are commonly used in conjunction with public key technology to create digital signatures that are used for authentication, integrity, and non-repudiation.

8.2.3 Message Integrity Checking

For message integrity the message is passed through an algorithm called a cryptographic hash function. This creates a message digest. The receiver runs the cryptographic hash function again and generates a new message digest, which is then compared with the message digest received. If both are the same, it is certain that the original message was not modified during transit. Figure 8.2 shows integrity verification.

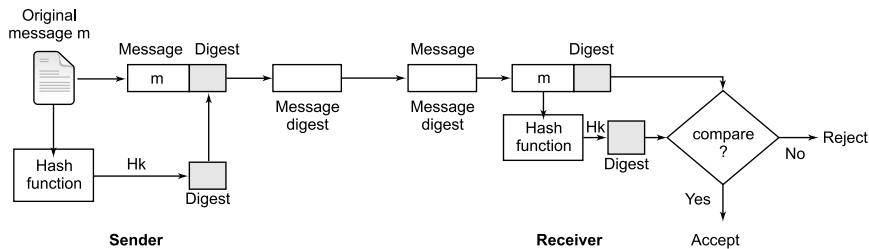


FIGURE 8.2. Message integrity checking.

8.3 HASH FUNCTION

A hash function is a mathematical function that compresses an arbitrary length message into a fixed small size message digest. The input to a hash function is typically called a “*message*” or the “*plaintext*,” and the output is often referred to as the “*message digest*” or hash value. The basic idea is that the message digest should serve as a compact representative image of an input string and can be used as if it is uniquely identifiable with that string. That is, the output of the hash function should serve as a digital fingerprint for the input and should be the same each time the message is hashed.

8.3.1 Features of Hash Functions

The important features of hash functions are:

Fixed Length Output

- Hash functions convert data of arbitrary length to a fixed length as in Figure 8.3. This process is often referred to as hashing the data.
- The hash or message digest is much smaller than the message input, hence hash functions are sometime called *compression functions*.
- A hash function with an n bit output is referred to as an n -bit hash function. Popular hash functions generate values between 160 bits and 512 bits.

Efficiency of Operation

- Generally for any hash function H with input x , computation of $H(x)$ is a fast operation.
- Computationally hash functions are much faster than symmetric key encryption.

8.3.2 Properties of Cryptographic Hash Functions

The properties of hash functions are:

1. Pre-image Resistance
2. Secondary Pre-image resistance
3. Collision Resistance

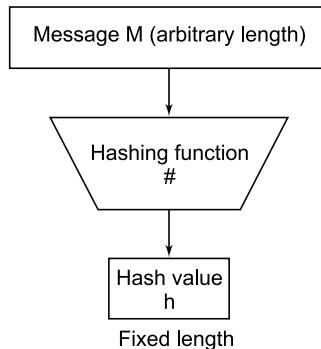


FIGURE 8.3. Hash function.

Pre-Image Resistance

A hash function is pre-image resistant if, given a hash value h , it is difficult to find any message m such that $h = \text{hash}(K, m)$ where K is the hash key. This property indicates that unlike a decryption there should not be any “*de-hash*” function. A good pre-image resistant function should be “*hard*” to invert. An example of a hash function that is not pre-image resistant is:

$$\begin{aligned} h &= \text{hash}(K, m) \\ &= m \bmod 2^K \end{aligned}$$

since it is very easy to invert the function after guaranteeing that for any value of h a message of size “ m ” can be found. Basically every message is of the form $h + x2^K$ where x is an integer.

Second Pre-Image Resistance

The second pre-image property of a hash function is defined as follows:

A hash function F is second pre-image resistant if for a message m_1 it is difficult to find a different message m_2 which hashes to the same image. Such messages are called *partners* (m_1 and m_2).

$$\text{i.e., } \text{hash}(K, m_1) = \text{hash}(K, m_2) \text{ where } K \text{ is the key.}$$

The one-way hash function H operates on the arbitrary length input message m , returning $h = H(m)$.

The important properties are:

- For a given m , it is easy to generate $h = H(m)$.
- For a given h , it is difficult to compute m such that $h = H(m)$ is pre-image resistant.
- Given m_1 it is hard to find m (different from M) such that $H(m) = H(M^1)$ is second pre-image resistant.
- It is hard to find MM^1 such that $H(m) = H(M^1)$ is collision resistant.

Collision Resistant

A very efficient cryptographic hash function is collision resistant. A hash function is collision resistant if it is hard to find two different messages that hash to the same image. A hash function is collision resistant if given two messages m_1 and m_2 it is hard to find a hash h , such that

$$h = m = \text{hash}(K, m_1) = \text{hash}(K, m_2)$$

For example: If there is no collision resistant hash function, then $\text{hash}(K, m) = 8$. Since all hashes result in 8, it makes it 100% likely that two messages will have the same hashes. Since hash functions have an infinite domain space but a finite range space (12 bits or 256 bits for (SHA-256)), a good collision resistant hash function should have each hash value be about as evenly distributed as possible.

For a given hash function with a range space of 2^{128} and a message m , any number between 0 and $2^{128} - 1$ should have the same chance (1 out of 2^{128}) of being $\text{hash}(K, m)$.

8.4 RANDOM ORACLE MODEL

Most of the cryptographic models theoretically work with provable security, only at the cost of efficiency. Some of the schemes are probably secured but are not efficient. Encryption techniques like ElGamal encryption are

efficient and secured at the same time. There is a need to construct more encryption schemes which are provably secure but also efficient. A random oracle model is one such technique used to achieve this.

A random oracle model (ROM) is a well-known idealized model of computation for proving the secret of cryptosystems. The random oracle model was introduced by Bellare and Rogaway in order to make it possible to give rigorous proofs of security for certain basic cryptographic protocols.

A RO is a theoretical black box that responds to every query with a random response chosen uniformly from its output domain, except that for any specific query, it responds the same way every time it receives that query; that is, a random oracle is a mathematical function mapping every possible query to a random response from its output domain.

8.4.1 Random Oracle Model

In the ROM, one assumes that some hash function is replaced by a publicly accessible random function (the random oracle). This means that the adversary cannot compute the result of the hash function by himself. He must query the random oracle. It is hard to see that such an oracle cannot exist in reality, but that model provides a formal methodology to design and validate the security of cryptographic schemes following the typical two-step approach:

1. The design of the scheme and providing proof of security in the RO model. The construction might be based on “standard” cryptographic assumptions.
2. To use the scheme in the real world, each party uses a real-world hash function (e.g., SHA1) and we adjust the scheme appropriately. That is, whenever the scheme asks to evaluate the RO on a value x , then the function is computed locally.

The random oracle H can be seen as a black box. It is accessible by both authentic and malicious parties. Whenever it is queried with a binary string x , then its outputs bring string $y = H(x)$; that is, whenever any party queries x , it outputs the same response y . The queries to the box are hidden from the other parties, which means that if the party P queries x to H then the party P^1 learns neither the query nor its answer. This models the fact that a hash function should be a publicly known function that is computable by everyone, since the internal function is hidden for both parties.

8.4.2 Pigeonhole Principle

If m pigeons are in n holes and $m > n$, then at least two pigeons are in the same hole. In fact at least m/n pigeons must be in the same hole.

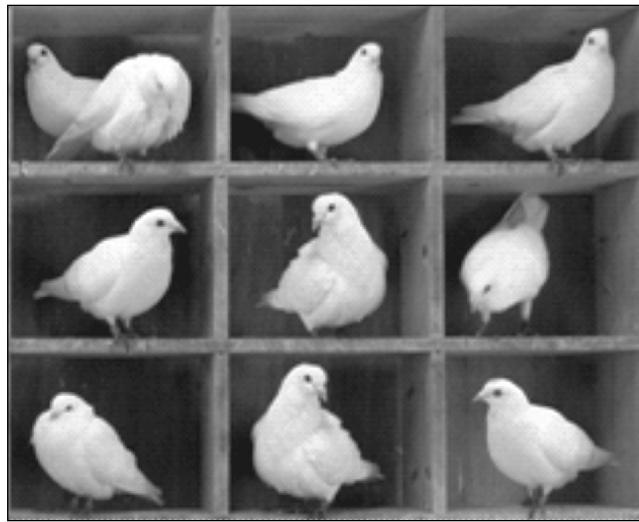


FIGURE 8.4. Pigeonhole principle.

In problem-solving, the “*pigeons*” are often numbers or objects, and the “*pigeonholes*” are properties that numbers/objects might possess.

For example, if we have 5 pigeons sitting in 2 pigeonholes, then one of the pigeonholes must have at least $5/2 = 2.5$ pigeons, but since (hopefully) the boxes can't have half pigeons, then one of them must contain 3 pigeons.

To prove the generalized pigeonhole principle:

Assume there were no pigeonholes with at least m/n pigeons. Then every hole has $< m/n$ pigeons, so the total number of pigeons is $< (m/n) \times (\# \text{holes}) = (m/n) \times n = m$.

8.4.3 Birthday Attack

A birthday attack is a type of cryptographic attack that exploits the mathematics behind the birthday problem in probabilistic theory. This attack can be used to abuse communication between two or more parties.

Birthday Problem

To understand the birthday problem consider the scenario in which a teacher with a class of 30 students asks for everybody's birthday, to determine whether any two students have the same birthday. Intuitively, this chance may seem small. If the teacher picked a specific day (say September 16), then the chance that at least one student was born on that specific day is $1 - \left[\frac{364}{365} \right]^{30}$. This is about 7.9%.

However, the probability that at least one student has same birthday as any other student is around 70%. This is by using the formula:

$$1 - \frac{365!}{[(365-n)! \cdot 365^n]} \text{ for } n = 30$$

The probability of two or more people having the same birthday is:

No. of People	Possibilities	Different Possibilities
2	365^2	365×364
3	365^3	$365 \times 364 \times 363$
.	.	.
.	.	.
.	.	.
K	365^K	$365 \times 364 \times 363 \times \dots \times (365 - (k + 1))$

$$P(\text{no common birthday}) = \frac{365 \times 364 \times 363 \times \dots \times (365 - (K + 1))}{365^K}$$

With 22 people in a room there is a better than 50% chance that two people have a common birthday.

With 40 people in a room there is almost a 90% chance that two people have a common birthday.

If there are K people, there are $\frac{K(K-1)}{2}$ pairs.

The probability that one pair has a common birthday is $\frac{K(K-1)}{2 \times 365}$.

If $K >> \sqrt{365}$ then this probability is more than half.

In general if there are $-n$ possibilities, then on average \sqrt{n} trials are required to find a collision.

Mathematics Behind the Birthday Paradox

Given a function f , the goal of the attack is to find two different inputs x_1, x_2 such that $f(x_1) = f(x_2)$; such a pair x_1 and x_2 is called a collision.

The method used to find a collision is simply to evaluate the function f for different input values that may be chosen randomly until the same result is found more than once. Because of the birthday problem this method can be rather efficient. Specifically, if a function $f(x)$ yields any of H different outputs with equal probability and H is sufficiently large, then we expect to

obtain a pair of different arguments x_1 and x_2 with $f(x_1) = f(x_2)$ after evaluating the function for about $1.25\sqrt{H}$ different arguments on average. Consider the following experiments:

For a set of H values we choose n values uniformly at random, thereby allowing repetition. Let $p(n; H)$ be the probability that during this experiment at least one value is chosen more than once.

This probability can be approximated as:

$$\begin{aligned} p(n; H) &\approx 1 - e^{-n(n-1)/(2H)} \\ &\approx 1 - e^{-n^2/(2H)} \end{aligned}$$

Let $n(p; H)$ be the smallest number of values we have to choose, such that the probability for finding collision is at least p . By inverting the previous expression, we find the following approximation:

$n(p; H) \approx \sqrt{2H \ln \frac{1}{1-p}}$ and assigning a 0.5 probability of collision, we arrive at $n(O-5; H) \approx 1.774 \sqrt{H}$.

Let $Q(H)$ be the expected number of values we have to choose before finding the first collision. The number can be approximated by:

$$Q(H) \approx \sqrt{\frac{\Pi}{2}}$$

As an example, if a 64 bit hash is used, there are approximately 1.8×10^{19} different outputs. If these are equally probable, then it would take only approximately 5 billion attempts (5.1×10^9) to generate a collision using brute force. This value is called the birthday bound, and for n -bit code it could be computed as $2^{n/2}$.

8.5 MESSAGE AUTHENTICATION

The message authentication technique determines whether the content of the message has come from the intended source without being modified by a malicious intruder. Authenticating for message integrity ensures that no one has tampered with the message or changed its content. There are different message authentication techniques discussed as follows.

8.5.1 Different Ways to Use Hashing for Message Authentication

The different ways in which message hashing can be incorporated in a communication network are discussed in the following sections.

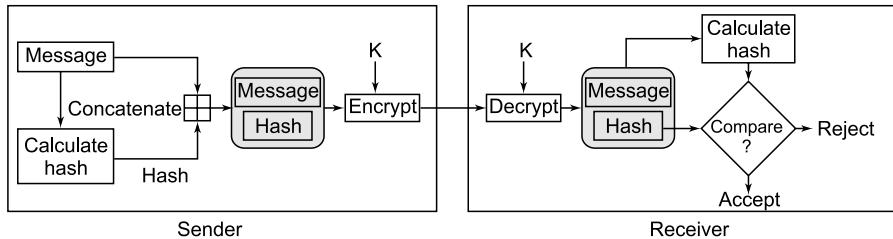


FIGURE 8.5. Message and its hash encrypted together.

In a symmetric key encryption technique as shown in Figure 8.5, the message and its hash code are concatenated together to form a composite message, which is then encrypted and placed on the communication network. The receiver decrypts the message and separates out its hash code, which is then compared with the hash code calculated from the received message. The hash code provides authentication and the encryption provides confidentiality.

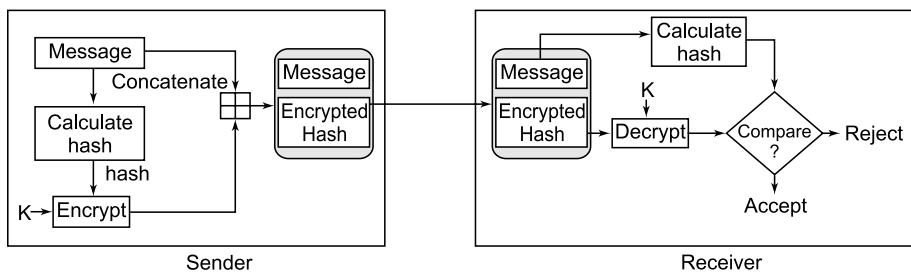


FIGURE 8.6. Hash code is encrypted.

In the encryption technique shown in Figure 8.6, only the hash code is encrypted. This scheme is efficient to use when confidentiality is not the issue but message authentication is critical. Only the receiver with the secret key can decrypt the hash code and get the real hash code of the message, so the receiver can verify whether or not the message is authentic. A hash code produced in this way is also known as a *Message Authentication Code* (MAC) and the hash function is known as a *keyed hash function*.

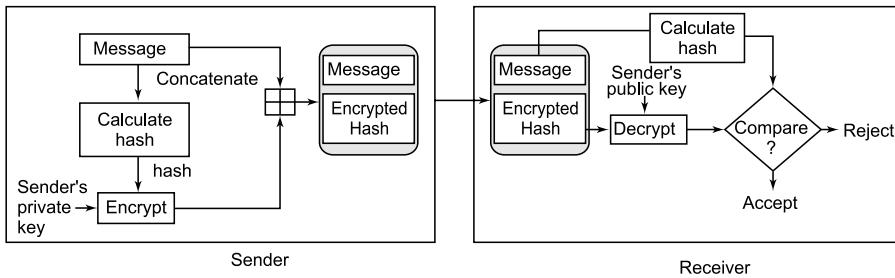
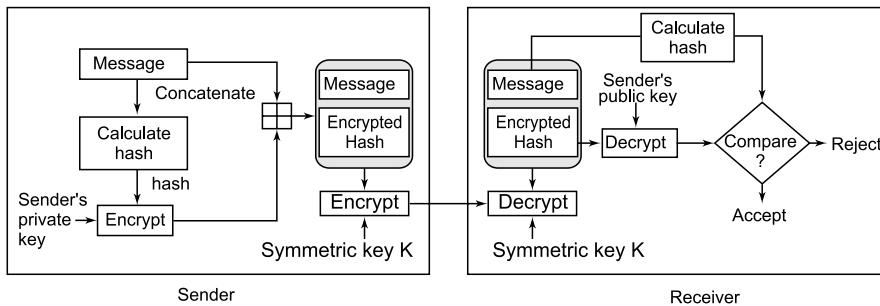
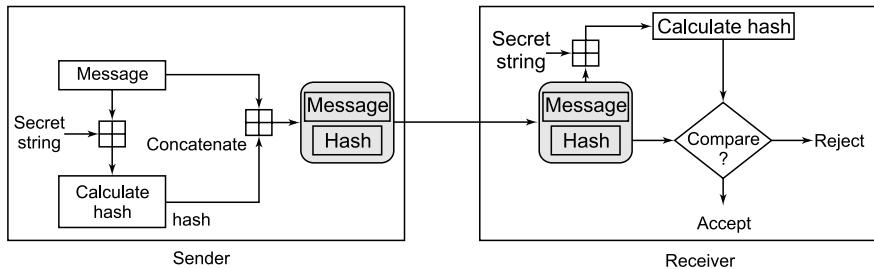
**FIGURE 8.7.** Using public key encryption.

Figure 8.7 is a public key encryption technique. The hash code of the message is encrypted with the sender's private key. The receiver can recover the hash code with the sender's public key and authenticate the message as indeed coming from the alleged sender. Confidentiality, again, is not the issue here. The sender encrypting with her private key the hash code of her message constitutes the basic idea of a digital signature.

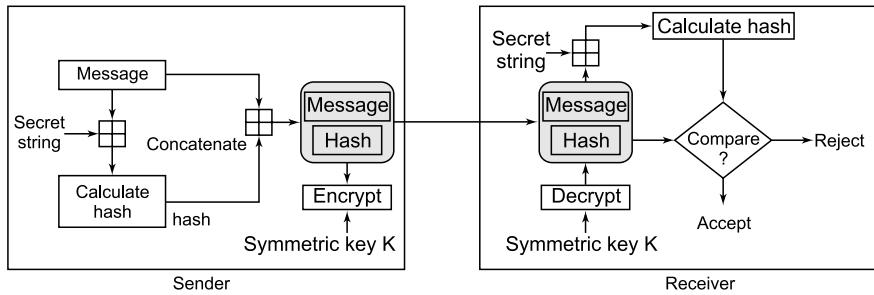
If both confidentiality and authentication are required, Figure 8.7 can be modified as in Figure 8.8. This also adds symmetric key based confidentiality.

**FIGURE 8.8.** Authentication with confidentiality.

A different approach of hashing for authentication is as shown in Figure 8.9. In this technique nothing is encrypted. However, the sender appends a secret string to the message before computing its hash code. This string is also known to the receiver. On the receiving side, before checking the hash code of the received message for its authentication, the receiver appends the same secret string to the message. Using this technique, it would not be possible for anyone to alter such a message, even though they have access to both the original message and the overall hash code.

**FIGURE 8.9** Using a secret string.

An extension of the technique shown in Figure 8.9 is shown in Figure 8.10. In this we have added symmetric key based confidentiality to the transmission between the sender and receiver.

**FIGURE 8.10.** Authentication with symmetric key confidentiality.

8.5.2 Modification Detection Code (MDC)

Modification Detection Code (MDC) is a 128 bit value that is generated by using a one-way cryptographic calculation (one-way hash function). The sender of the message transmits the MDC with integrity to the intended receiver of the message. The receiver of the message can use a similar algorithm to generate another MDC. If the two MDCs are identical, the receiver assumes that the message is genuine. If they differ the receiver assumes that someone or some event altered the message.

8.5.3 Message Authentication Code (MAC)

A MAC is a block of a few bytes that is used to authenticate a message. For verification of the integrity of the message and the data origin authentication, the receiver can check this block and be sure that the message hasn't

been modified during transit. A very simple way to mark message authentication of a message is to compute its checksum. This can be done by using a CRC algorithm. The main limitation of this method is the lack of protection against intentional modifications in the message content. The intruder can change the message itself, then calculate a new checksum and eventually replace the original checksum by the new value.

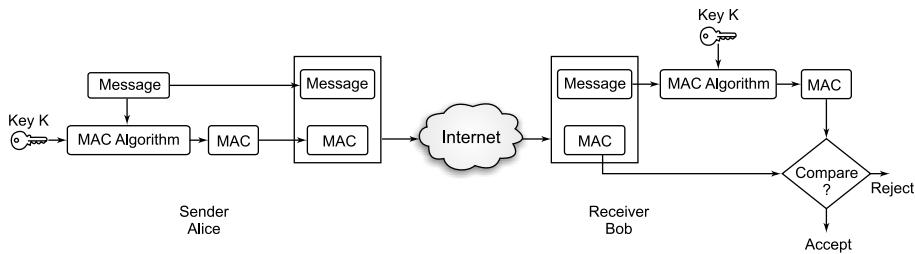


FIGURE 8.11. MAC using an encrypted MAC.

To obtain the required MAC, Alice uses some publicly known MAC algorithm and inputs the message and the secret key K . This produces a MAC value which is similar to a hash; that is, the MAC function also compresses an arbitrarily long input message into a fixed length output. The major difference between a hash and a MAC is that a MAC uses a secret key during the compression. Alice then forwards the message along with the MAC to the receiver Bob as shown in Figure 8.11. If the message is to be secured, it needs encryption. We are concerned with providing message authentication, not confidentiality.

At the receiving side on receipt of the message and the MAC, Bob separates the message from the MAC. The MAC value is recomputed using the received message, the shared secret key K , and the MAC algorithm. Bob now compares the freshly computed MAC with the MAC received from Alice. If they match, then he accepts the message and assures himself that the message has been sent by the intended sender. The different types of MAC algorithms that provide security against intentional changes of authentication code are:

1. CBC MAC
2. Nested MAC (NMAC)
3. Parallel MAC
4. HMAC

Cipher Block Chaining (CBC) MAC or CMAC

CBC MAC is based on a pseudorandom function. It works similar to encryption performed in a CBC mode, with a difference that intermediate values are not returned. After the last block encryption, an additional encryption is performed on the current result using the second secret key K_2 . This additional encryption is to protect the calculated code. With this additional step it is often referred to as ECBC MAC (Encrypted CBC MAC). CBC MAC can protect a message of any length from one to many blocks. To ensure security while using CBC MAC, one should change the secret key from time to time. It can be proven that after sending the number of messages that is equal roughly to the square of the number of all possible values of data blocks, the key is no longer safe. The plaintext message is divided into " n " message blocks, each with a length of m bits, as in Figure 8.12. The last data block should be filled up to the full length using previously agreed upon bits. If the number of bits in the last block is less than the full-length bits (m bits), one should append an additional bit equal to 1 and then fill the rest of the block up with bits equal to 0. If there is not enough free space in the last block, one should add one more extra block and fill it with the additional padding bits.

The first message block m_1 is encrypted using the symmetric key K_1 to create an m bit block of encrypted message. This block is XORed with the next block of plaintext message m_2 , and the output obtained is encrypted using the symmetric key K_1 in function F to obtain the new m bit block. The CBC process is repeated to encrypt all n data blocks of the plaintext message. In addition to the encryption using symmetric key K_1 , the CMAC also uses another key K_2 as shown in Figure 8.12. Adding the additional key in the last encryption step protects against appending new blocks of modified messages by a potential intruder. It is not necessary to use an additional encryption, unlike in other MAC algorithms. CMAC is considered to be secure. It provides a safe way for message authentication. It is certified, for example, by the U.S. institute NIST.

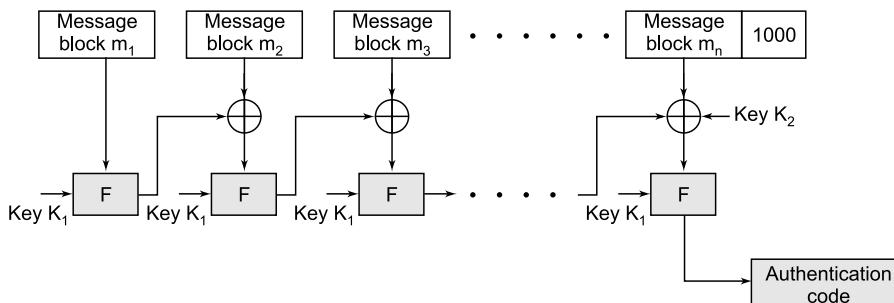


FIGURE 8.12. CBC MAC.

Nested MAC (NMAC)

The function of NMAC is similar to the CBC MAC. It uses a slightly different pseudorandom function F . The function F returns numbers that are correct values of secret keys. In NMAC also after the encryption of the last block, an additional encryption of the result is performed using the second secret encryption key K_2 , as in Figure 8.13. This is because the previous result of encryption of the last data block consists of the same amount of bits as the secret key K_1 . An additional sequence of bits (a fix pad) should be appended to assure that the result has the same size as the data block. The last additional encryption is performed to protect the calculated code as in the case of the CBC MAC. Without the second encryption using key K_2 , an intruder would be able to append any number of blocks to the intercepted message with the corrected calculated authentication code and attach it to the modified message. As input to the first new added function F , the attacker would use the original authentication code of the original message.

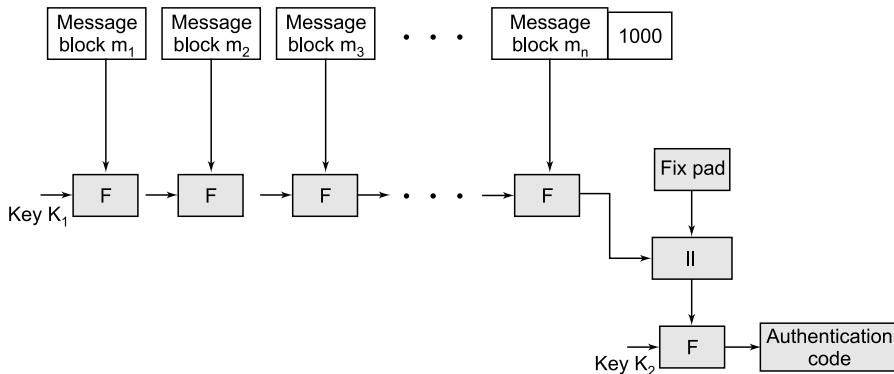


FIGURE 8.13. Nested MAC.

Parallel MAC (PMAC)

The PMAC algorithm performs a sequential data process of data blocks. PMAC uses two secret encryption keys K_1 and K_2 , and two encryption functions P and F . The first secret key is used in P functions. All these P functions also receive the subsequent numbers of the additional counter. The output bits of the P functions are added XOR to message blocks. The result of this XOR operation is encrypted by a pseudorandom function F that uses the second secret key K_2 . Output bits from all F functions and output bits from the last data block (which is not encrypted by the F function) are added

XOR together as in Figure 8.14. The result obtained is encrypted using the F function algorithm, with the second secret encryption key K_2 . Using PMAC a higher level of security can be proven if the secret key is changed from time to time.

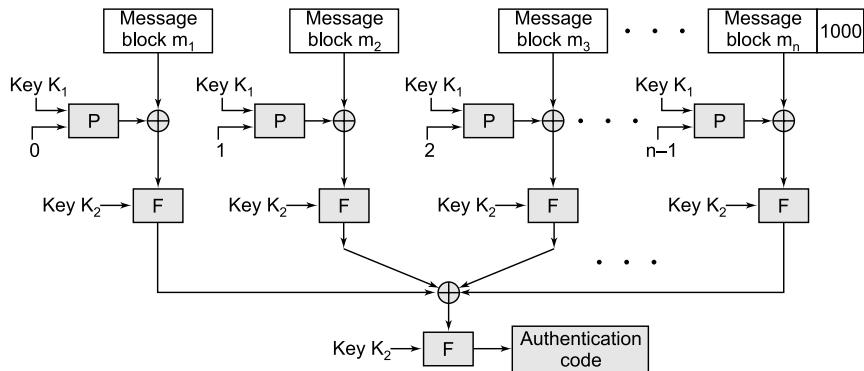


FIGURE 8.14. Parallel MAC.

Hashed MAC (HMAC)

The Hashed MAC is quite similar to the Nested MAC, but its implementation is much more complex than the NMAC. It is also a popular system of checking message integrity. The HMAC produces a unique MAC value using a one-way hash function.

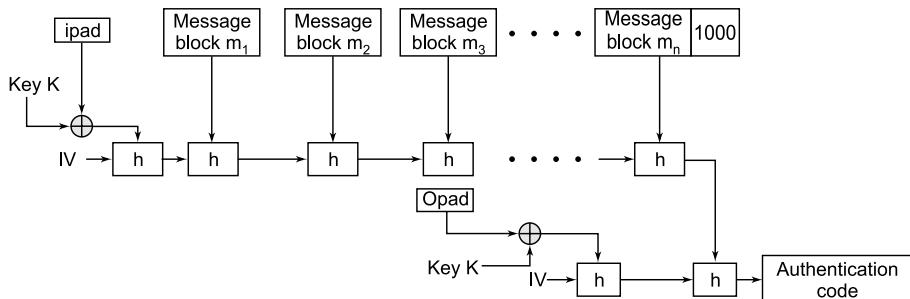


FIGURE 8.15. Hashed MAC (HMAC).

The message is divided into n blocks of m bits long as in Figure 8.15. The input parameters *ipad* (input pad) and *opad* (output pad) are used to modify the secret key. They may have various values assigned. It is recommended to choose the values that would make both inputs to the hash

functions look as dissimilar as possible (that is, that modify the secret key in two different ways).

Using a secure hash function (that means the function that doesn't produce the same outputs for different input data) guarantees the security of the HMAC algorithm. The HMAC algorithm is used in many popular Internet protocols such as SSL, IPsec, and SSH.

SUMMARY

- Message integrity involves maintaining the consistency, accuracy, and trustworthiness of data over the entire life cycle.
- Integrity refers to the ability to protect the information, data, or transmission from unauthorized or accidental alterations.
- A document fingerprint makes it easier to protect information by identifying standard formats that are used throughout an organization.
- Message digest is a cryptographic function containing a string of digits created by a one-way hash formula. Message digest functions are also called hash functions.
- A hash function is a mathematical function that compresses an arbitrary length message into a fixed small size message digest.
- The properties of a hash function are: pre-image resistance, secondary pre-image resistance and, collision resistance.
- The random oracle model assumes some hash function is replaced by a publicly accessible random function.
- A birthday attack is a type of cryptographic attack that exploits the mathematics behind the birthday problem in probabilistic theory.
- Authenticating for message integrity ensures that no one has tampered with the message or changed its content.
- A MAC is a block of a few bytes that is used to authenticate a message.

REVIEW QUESTIONS

1. What are the criteria of the cryptographic hash function?
2. Explain the Chinese remainder theorem with an example.

3. Explain the Chinese remainder theorem. Using CRT find “ x ” from the equations
 $x \equiv 7 \pmod{13}$ and $x \equiv 11 \pmod{12}$.
4. What is the birthday attack on digital signatures?
5. Describe the Chinese remainder theorem and explain its application.
6. Give the structure of CMAC. What is the difference between CMAC and HMAC?
7. Describe the process involved in digital signatures. Explain about different digital signatures.
8. Write about the HMAC algorithm. What needs to be done to speed up the HMAC algorithm?

MULTIPLE CHOICE QUESTIONS

1. Message authentication is a service beyond:
(a) Message Confidentiality (b) Message Integrity
(c) Message Splashing (d) Message Sending
2. This technology is used to measure and analyze human body characteristics for authentication purposes.
(a) Footprinting (b) Biometrics
(c) Anthropomorphism (d) Optical character recognition
3. This enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority.
(a) Security Identifier (SID)
(b) Public Infrastructure (PKI)
(c) Internet Assigned Numbers Authority (IANA)
(d) Trusted Computing Platform Alliance (TCPA)

12. Hash collision means:

- (a) Two keys for one message
- (b) One key for two messages
- (c) Two different keys for different messages
- (d) Always the same key

13. Encryption strength is based on:

- | | |
|---------------------------|----------------------|
| (a) Strength of algorithm | (b) Secrecy of key |
| (c) Length of key | (d) All of the above |

HASH FUNCTIONS

9.1 INTRODUCTION

A hash function usually means a mathematical function that compresses the message, meaning the output is shorter than the input. An ideal hash function makes it very difficult to get the original message from the message digest. Using a hash algorithm it is very easy to compute a hash for a given message and make it infeasible to modify the message without changing the resultant hash, and infeasible to find two messages with the same hash. Hash functions are used in many parts of cryptography. There are many different types of hash functions such as SHA1, MD4, MD5, RACE, and Whirlpool, with differing security properties. All these hash functions are discussed in detail in this chapter.

9.2 CONSTRUCTIONS OF HASH FUNCTIONS

Hash function constructions are classified as:

- (i) Iterated construction
- (ii) Merkle-Damgard strengthening

Iterated Construction

Hash functions constructed by iterative mode are more efficient than other types of hash functions. These hash functions are used in information processing applications, such as the computation of a digital signature, in which a function accepts an arbitrary length message as input and is iterated until a

fixed size message is processed completely. The iterative mode of operation is also adopted in the popular hash functions MD5 and SHA-1. It is in the iterative mode of operation of their respective compression functions. One of the best examples of iterative construction is Merkle-Damgard iterative construction, which is commonly used to design cryptographic hash function, as explained in the next section.

Merkle-Damgard Strengthening

Merkle-Damgard construction has influenced the design of popular hash functions such as MD4, MD5, and SHA-0 and hash functions SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 from the secure hash standard. It is a particular method of construction of hash function where if the original compression function h is collision resistant, then the hash function H is also collision resistant. The construction of the Merkle-Damgard hash function is depicted in Figure 9.1 and is described as follows.

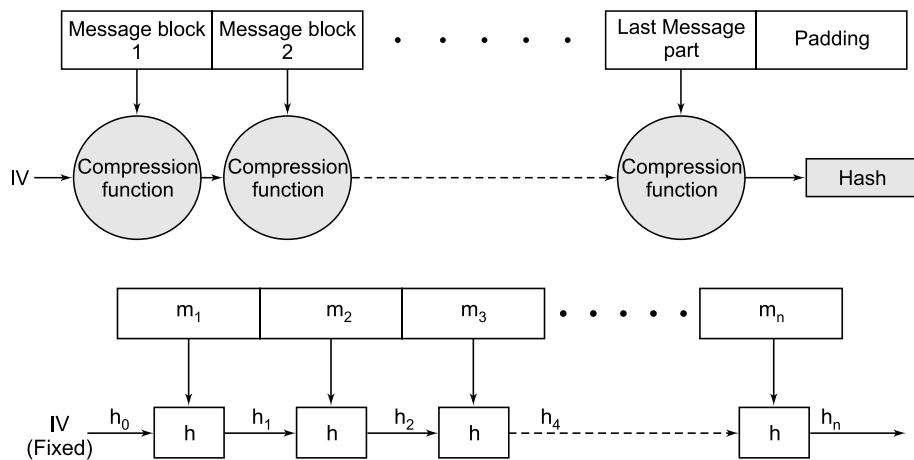


FIGURE 9.1 Merkle-Damgard hash function construction.

A message m is split into n -blocks, m_1, m_2, \dots, m_n . Each block is of fixed length, with appropriate padding whenever needed. In case the last message block is not of b -bits, it is padded to be of the same length. The first block and the IV ($h_0 = \text{IV}$) is passed as an input to a compression function h to yield the first chaining vector h_1 .

Then $h_1 = H(h_0, m_1)$, h_1 and m_2 are passed as input to h again to yield the second chaining vector $h_2 = H(h_1, m_2)$. This chaining vector (CV) goes until

m_n and $h_{(n-1)}$ are passed to h , which outputs $h_n = H(h_{(n-1)}, m_n)$. The entire construction is denoted by H .

Therefore the output of the hash function is:

$$\begin{aligned} h_0 &= \text{IV a fixed initial vector (IV)} \\ h_1 &= H(h_0, m_1) \\ h_2 &= H(h_1, m_2) \\ &\vdots && \vdots \\ h_i &= H(h_{(i-1)}, m_i) \\ h_n &= H(h_{(n-1)}, m_n) \\ H_{(m)} &= h_n \end{aligned}$$

The key property of Merkle-Damgård is that it has an avalanche effect. In other words even if a single bit in the original message m is modified, then $H_{(m)}$ becomes completely different and uncorrelated. Thus, if in a message only a single bit is changed during transit, its hash value will be completely different than that of the original one. This will reduce the likelihood of finding collisions.

Different Hash Functions

Table 9.1 indicates the different hash functions and their variants.

TABLE 9.1 Hash Functions and their Variants.

Hash Functions	Different Hash functions
Secure Hash Algorithm (SHA)	<p>The family of SHAs comprises four SHA algorithms: SHA-0, SHA-1, SHA-2, and SHA-3.</p> <ul style="list-style-type: none"> - SHA-0: Is a 16 bit hash function. - SHA-1: The most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols like secure socket layer (SSL) security. - SHA-2: Its family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512, depending upon the number of bits in their hash value. - SHA-3 In October 2012 NIST chose the Keccak Algorithm as the new SHA-3. It offers many benefits, such as efficient performance and good resistance for attacks.

(continued)

Hash Functions	Different Hash functions
Message Digest (MD)	<p>The MD family comprises hash functions MD2, MD4, MD5, and MD6.</p> <ul style="list-style-type: none"> - MD2: the last value of the MD2 chaining variable is a 128 bit (16 byte) hash value. - MD4: it compresses an input with a maximum length of 2^{64} to a 128 bit hash value. - MD5: Digest has been widely used due to its assurance about integrity of a transferred file of 256 bits. It produces a 128 bit message digest. - MD6: It accepts an input message of length $2^{64}-1$ bits and produces a message digest of any desired size (224, 256, 384, and 512 bits).
RACE Integrity Primitive Evaluation Message Digest (RIPEMD)	<p>RIPEMD is a set of hash functions that was designed by an open research community and is generally known as a family of European hash functions.</p> <p>The RIPEMD family includes RIPEMD, RIPEMD-128, and RIPEMD-160.</p> <ul style="list-style-type: none"> - RIPEMD-128: is based upon the design principles used in MD4. It produces a 128 bit message digest. - RIPEMD-160: is an improved version and the most widely used version. It is a hash algorithm with a 160 bit message digest.
Whirlpool	<p>Whirlpool is a 512 bit hash function. The different versions of Whirlpool functions are:</p> <ul style="list-style-type: none"> - WHIRLPOOL-0 - WHIRLPOOL – T

9.3 MESSAGE DIGEST

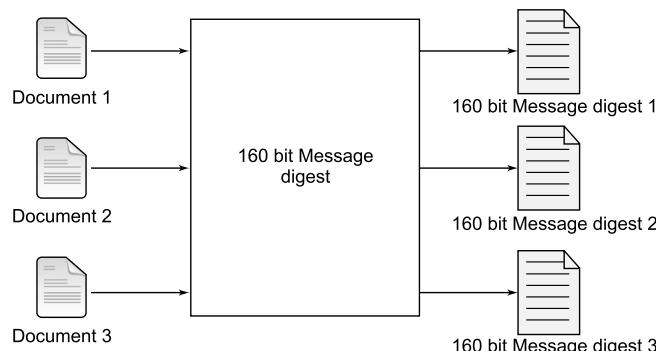


FIGURE 9.2 Message digest process using a 160 bit message digest.

A message digest is a cryptographic hash function containing a string of digests created by a one-way hash formula. A message digest is designed to protect the integrity of data or information. It detect the changes and alterations to any part of data or information. Message digest functions are also called *hash functions*. They are used to produce the digital summary of information called a *message digest*. Message digests are generally 128 bits to 160 bits in length and provide a digital identifier for each digital file or document. Message digest functions are mathematical functions that process information to produce a different message digest for each unique document. Identical documents have the same message digest, but if even one of the bits for the document changes, the message digest changes. The message digests are much shorter than the data from which the digests are generated, and the digits have a finite length. A good message digest uses a one-way function to ensure that it is mathematically and computationally infeasible to uncover the message digest process and discover the original data.

A collision can exist when there are duplicate message digests for different data sets. Finding collisions for good message digest functions is also mathematically and computationally infeasible but possible given enough time and computational efforts. Message digests are commonly used in conjunction with public key technology to create digital signatures that are used for authentication, integrity, and non-repudiation for electronic files and documents.

Message Digest Overview

MD2 was developed in 1989. It generates a 128 bit message digest. The MD2 hash function description is found in Figures 9.3 (a) and (b).

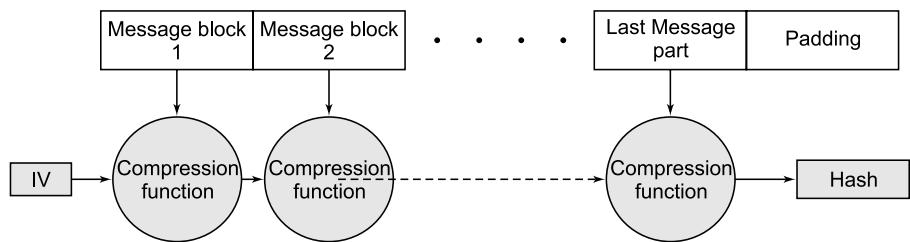


FIGURE 9.3 (a) General representation of MD2.

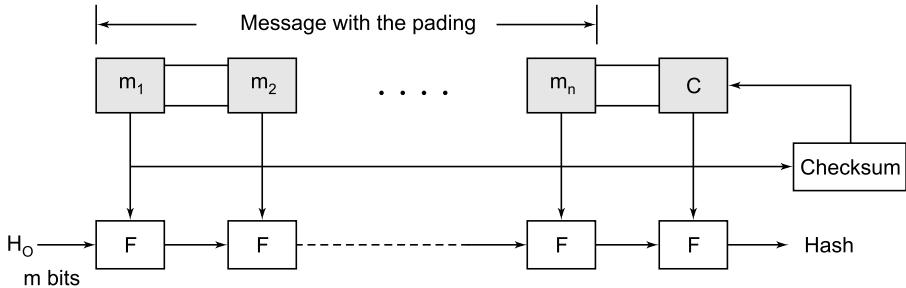


FIGURE 9.3 (b) MD2 hash function.

The input message blocks are referred to as m_1, m_2, \dots, m_n . Outputs obtained from each compression function are of length 128 bits. The first step of MD2 is to append a padding to the initial message, then to compute a checksum block (C). This increases the length of the message by one block. Finally, the compression function F is applied iteratively to produce the hash value. The value of i^{th} intermediate hash is $H_{i+1} = F(H_i, M_i)$. The IV of the hash function is H_0 and is set by default to 0. A_0^0

The Compression Function

A precise representation of the compression function F is as shown in Figure 9.4. Each box in this figure contains one byte. F is decomposed into three metrics denoted by A , B , and C with 16 columns and 19 rows each as shown in Figure 9.4. The first row of the metrics is initialized respectively with H_i , M_i , and $H_i \oplus M_i$. Then the row of each metric is computed recursively from top to bottom. The last rows of B and C are not used. The \oplus symbol denotes addition modulo 256.

The computation in the compression function is based on a function f from 16 bits to 8 bits. In case of metrics "A," this can be described by the equation:

$$A_i^t = \phi(A_i^{t-1}, A_{i-1}^t) = A_i^{t-1} \oplus S(A_{i-1}^t)$$

where S is a fixed S-box of size 8 bits. The equations for metrics B and C are exactly the same. The function ϕ is represented in Figure 9.5.

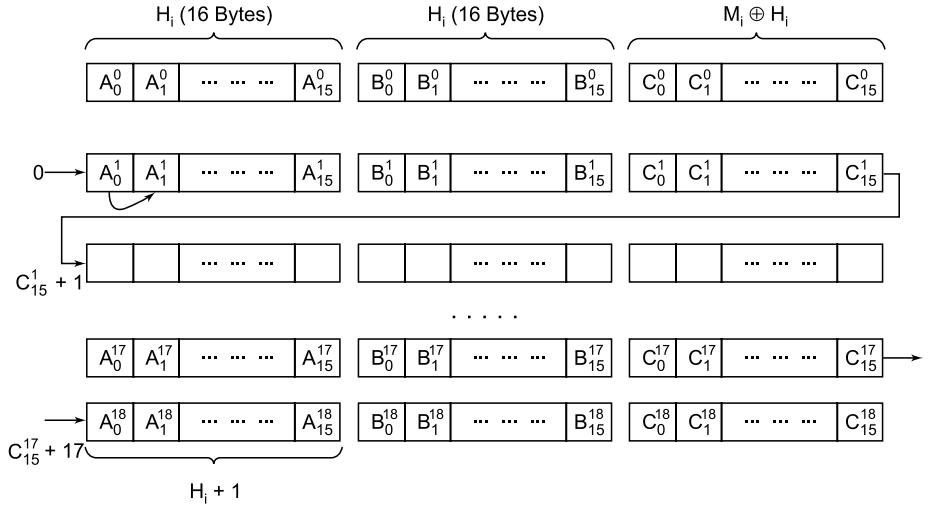
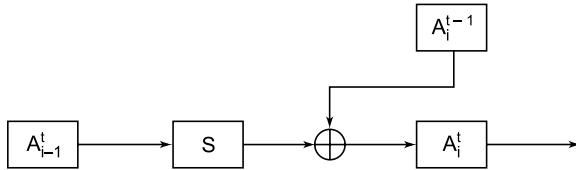


FIGURE 9.4 The compression function of MD2.

FIGURE 9.5 The ϕ function.

The Checksum Function

The checksum C is computed from the message block by iterating a nonlinear checksum function. It is considered as G . It uses only basic operations like XOR and the S-box S . The checksum function is described by the equation as:

$$\begin{aligned} IC_0 &= 0 \\ IC_{i+1} &= G(IC_i, M_i) \end{aligned}$$

The function G is used to compute the intermediate checksum IC_i from IC_{i+1} and M_i . The final value IC_{n+1} is the appended checksum C .

MD 5 Algorithm

MD 5 is a message digest algorithm developed by Ron Rivest at MIT. This has been the most widely used secure hash algorithm, particularly for Internet standard message authentication. The algorithm takes as input a message of arbitrary length and produces as output a 128 bit message digest of the input. The following five steps are performed to compute the message digest for an arbitrary length message as input in MD 5.

Step 1: Padding: This step is for appending padding bits. The original message is padded so that its length is congruent to $448 \bmod 512$. The padding rules are:

- The original message is always padded with one bit “1” first.
- Then zero or more bits “0” are padded to increase the length of the message up to 64 bits less than a multiple of 512.

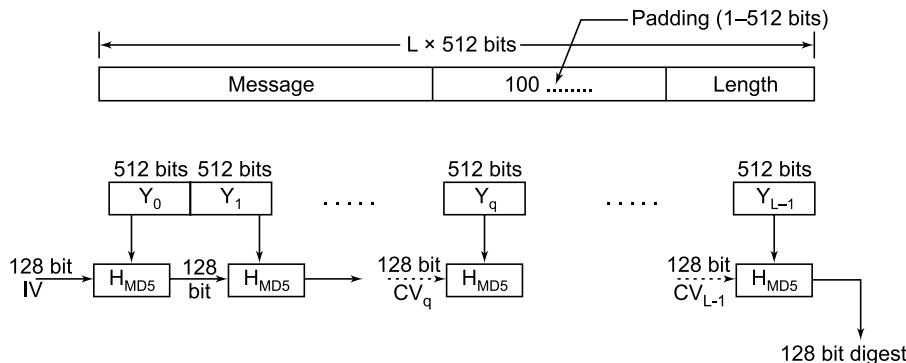


FIGURE 9.6 Generation of message digest.

Step 2: Appending length: 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules for appending lengths are:

1. The length of the original message is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used.
2. Break the 64 bit length into 2 words (32 bits each).
3. The low-order word is appended first and followed by the high-order word.

The expanded message at this level will exactly be a multiple of 512 bits. This expanded message is represented as a sequence of $L \times 512$ bit blocks $Y_0, Y_1, \dots, Y_q, \dots, Y_{L-1}$ as shown in Figure 9.6.

Step 3: Initialize MD Buffer: MD 5 requires a 128 bit buffer with a specific initial value. The variables IV and CV are represented by a four word buffer ($A\ B\ C\ D$) to generate the message digest. Here, A , B , C , and D are 32 bit registers and they are initialized as IV to the following values in hexadecimal form.

Word A	01	23	45	67
Word B	89	AB	CD	EF
Word C	FE	DC	BA	98
Word D	76	54	32	10

- Word buffer A is initialized to: 0×67452301
- Word buffer B is initialized to: $0 \times EFCDAB89$
- Word buffer C is initialized to: $0 \times 98BADCCE$
- Word buffer D is initialized to: 0×10325476

Step 4: Processing message in 16 word blocks: This is the main section of the MD 5 algorithm. It includes four rounds of processing. These rounds are represented by H_{MD5} as in Figure 9.6. For each input block four rounds of operations are performed; that is, there are sixteen operations in each round. The four rounds have a similar structure, but each uses different auxiliary functions F , G , H , and I .

$$\begin{aligned} F(x, y, z) &= (x \text{ AND } y) \text{ OR } (\text{NOT } x \text{ AND } z) \\ G(x, y, z) &= (x \text{ AND } z) \text{ OR } (y \text{ AND } (\text{NOT } z)) \\ H(x, y, z) &= (x \text{ XOR } y \text{ XOR } z) \\ I(x, y, z) &= y \text{ XOR } (x \text{ OR } (\text{NOT } y)) \end{aligned}$$

Each round consists of 16 steps, and each step uses a 64 element table, $t_{(1)}, t_{(2)}, \dots, t_{(64)}$, constructed from the sine functions. Let $t_{(i)}$ denote the i^{th} element of the table,

where $i = 1, 2, 3 \dots 64$.

Since there are four rounds, we use 16 out of the 64 values of i in each round.

$$t_{(i)} = \text{int}(\text{abs}(\sin e(i)) \times 2^{32})$$

Each round also takes as input the current 512 bit block (Y_q) and the 128 bit chaining variable (CV_q). An array of 32 bit words holds the current 512 bit block (Y_q). For the first round the words are used in their original order. Let $m_{(1)}, m_{(2)}, \dots, m_{(15)}$ be blocks of padded and appended messages. In each round we have 16 input sub-blocks. In general it is represented as $M_{(i)}$, where $i = 1, 2, \dots, 15$.

The round 1 operation is $R_1(a, b, c, d, x, s, i)$, it is defined as:

$$a = b + ((a + F(b, c, d) + x + t_{(i)}) \ll s)$$

For $K = 1$ to N do the following:

$$AA = A; \quad BB = B; \quad CC = C; \quad DD = D$$

$(x[0], x[1], \dots, x[15]) = M [K]$ Divide $M [K]$ into 16 words

$R1(A B C D x[0]7, 1)$	$R1(A B C D x[4]7, 5)$	$R4(A B C D x[4]6, 61)$
$R1(D A B C x[1]12, 2)$	$R1(D A B C x[5]12, 6)$	$R4(D A B C x[11]10, 62)$
$R1(C D A B x[2]17, 3)$	$R1(C D A B x[6]17, 7)$	$R4(C D A B x[2]15, 63)$
$R1(B C D A x[3]22, 4)$	$R1(B C D A x[7]22, 8)$	$R4(B C D A x[9]21, 64)$

$$A = A + AA; \quad B = B + BB; \quad C = C + CC; \quad D = D + DD$$

Round 1			
[abcd k s i] denote the operation $a = b + ((a + F(b, c, d) + X[k] + T[i]) \ll s)$			
[ABCD 0 7 1]	[DABC 1 12 2]	[CDAB 2 17 3]	[BCDA 3 22 4]
[ABCD 4 7 5]	[DABC 5 12 6]	[CDAB 6 17 7]	[BCDA 7 22 8]
[ABCD 8 7 9]	[DABC 9 12 10]	[CDAB 10 17 11]	[BCDA 11 22 12]
[ABCD 12 7 13]	[DABC 13 12 14]	[CDAB 14 17 15]	[BCDA 15 22 16]
Round 2			
[abcd k s i] denote the operation $a = b + ((a + G(b, c, d) + X[k] + T[i]) \ll s)$			
[ABCD 1 5 17]	[DABC 6 9 18]	[CDAB 11 14 19]	[BCDA 0 20 20]
[ABCD 5 5 21]	[DABC 10 9 22]	[CDAB 15 14 23]	[BCDA 4 20 24]
[ABCD 9 5 25]	[DABC 14 9 26]	[CDAB 3 14 27]	[BCDA 8 20 28]
[ABCD 13 5 29]	[DABC 2 9 30]	[CDAB 7 14 31]	[BCDA 12 20 32]
Round 3			
[abcd k s t] denote the operation $a = b + ((a + H(b, c, d) + X[k] + T[i]) \ll s)$			
[ABCD 5 4 33]	[DABC 8 11 34]	[CDAB 11 16 35]	[BCDA 14 23 36]
[ABCD 1 4 37]	[DABC 4 11 38]	[CDAB 7 16 39]	[BCDA 10 23 40]
[ABCD 13 4 41]	[DABC 0 11 42]	[CDAB 3 16 43]	[BCDA 6 23 44]
[ABCD 9 4 45]	[DABC 12 11 46]	[CDAB 15 16 47]	[BCDA 2 23 48]

Round 4			
[abcd k s t] denote the operation $a = b + ((a + I(b,c,d) + X[k] + T[i]) \ll\ll s)$			
[ABCD 0 6 49]	[DABC 3 10 54]	[CDAB 14 15 51]	[BCDA 5 21 52]
[ABCD 12 6 53]	[DABC 3 10 54]	[CDAB 10 15 55]	[BCDA 1 21 56]
[ABCD 8 6 57]	[DABC 15 10 58]	[CDAB 6 15 59]	[BCDA 13 21 60]
[ABCD 4 6 61]	[DABC 11 10 62]	[CDAB 2 15 63]	[BCDA 9 21 64]

Perform the following additions, which increment each of the four registers by the value it had before this block was started.

$$A = A + AA$$

$$B = B + BB$$

$$C = C + CC$$

$$D = D + DD$$

Step 5: output

Output $A, B, C, D \rightarrow$ is the message digest

The message digest produced as output is A, B, C , and D , beginning with the low-order byte of A , and ending with the high-order byte of D . In each case the output of the intermediate stage and the final iteration is copied into the register $a b c d$. There are 16 such iterations in each round.

9.4 SECURE HASH ALGORITHM (SHA)

The secure hash algorithm (SHA) was developed by NIST in association with the NSA and was first published in 1993 as the Secure Hash Standard. The first revision to this algorithm was published in 1995.

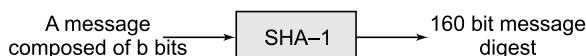


FIGURE 9.7 SHA hash function.

The revised version of SHA was called SHA-1. The SHA hash function is similar to the MD4 hash function, as shown in Figure 9.7. The major difference between MD5 and SHA-1 is it adds some complexity to the algorithm, and also there is a change in the block size. SHA was originally intended as part of the digital signature standard (DSS), which is used for signing data

and requires a hash function to sign. In addition to the SHA-1 hash, the NIST also published a set of more complex hash functions for which the output ranges from 224 bits to 512 bits. These hash algorithms are referred to as SHA-224, SHA-256, SHA-384, and SHA-512.

Description of the SHA-1 Algorithm

SHA-1 takes the input message of maximum length $2^{64}-1$ bits and returns a 160 bit message digest.

1. Appending padding bits: The input is processed in parts of 512 bits each and is padded so that its length (in bits) is congruent of 448 modulo 512. The padding rules are:

- (i) The original message is always padded with one bit “1” first. The reason for using 1 first is that otherwise, collisions occur between padding and messages.
- (ii) Then zero or more bits “0” are padded as in Figure 9.8.

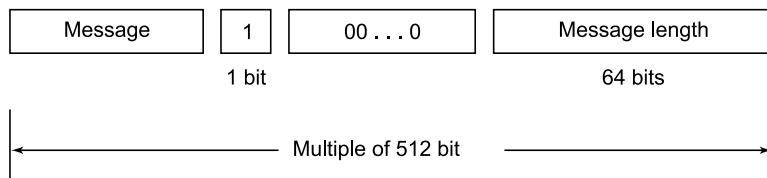


FIGURE 9.8 Appending padding bits.

2. Appending Length: The 64 bits appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending lengths are:

- (i) The length of the original message in bytes is converted to its binary format of 64 bits. In case of overflow only the low-order 64 bits are used.
- (ii) Divide the 64 bit length into 2 words of 32 bits each.
- (iii) The lower order word is appended first and followed by the high-order word.

For example: If the message M is 30 bits length → 011001 00001010 11101011 11001100 (30 bits)

The padding is done by appending to the input:

- A single bit, 1
- Number of 0s → 417
- 64 bit integer representing 30

$\text{Pad}(M) \rightarrow 011001\ 00001010\ 11101011\ 11001100\ 1\ 000\ \dots\dots\dots\ 00011110$

If M is 500 bits

Padding is done by appending to the input:

- A single bit 1
- Number of 0s → 459
- A 64 bit integer representing 500

Length of $\text{Pad}(M)$ → 1024 bits

3. Preparing processing functions: SHA-1 requires 80 processing functions defined as:

- (i) $f(t; B, C, D) = ((B \text{ AND } C) \text{ OR } (\text{NOT } B) \text{ AND } D))$ ($0 \leq t \leq 19$)
- (ii) $f(t; B, C, D) = (B \text{ XOR } C \text{ XOR } D)$ ($20 \leq t \leq 39$)
- (iii) $f(t; B, C, D) = ((B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ or } (C \text{ AND } D))$ ($40 \leq t \leq 59$)
- (iv) $f(t; B, C, D) = ((B \text{ XOR } C \text{ XOR } D))$ ($60 \leq t \leq 79$)

Where $t = 0, 1, 2, \dots, 79$

4. Preparing processing constants: SHA-1 requires four processing constant words, defined as:

$$\begin{aligned} k_0 &= 0 \times 5A827999 & (0 \leq t \leq 19) \\ k_1 &= 0 \times 6ED9EBA1 & (20 \leq t \leq 39) \\ k_2 &= 0 \times 8F1BBCDC & (40 \leq t \leq 59) \\ k_3 &= 0 \times CA62C1D6 & (60 \leq t \leq 79) \end{aligned}$$

5. Buffer initialization: The intermediate results of each block are stored in five 32 bit registers denoted with A through E . These five registers are initialized with the following hexadecimal values.

A	$= 0 \times 67452301$
B	$= 0 \times EFCDAB89$
C	$= 0 \times 98BADCFE$
D	$= 0 \times 10325476$
E	$= 0 \times C3D2E1F0$

Using this we want to produce a message digest length of 160 bits. We need five variables as $5 \times 32 = 160$ bits.

6. Processing message in 512 bit blocks: This is the main processing part of the SHA-1 algorithm, which loops through the padded and appended message in blocks of 512 bits each. The operations performed on each input block are discussed as follows. The input and predefined functions are:

$M[1, 2, 3, \dots, N]$ Blocks of the padded and appended message.
 $f(0; B, C, D), f(1; B, C, D) \dots f(79; B, C, D)$ defined as previously.

A, B, C, D, E word buffers with initial values copy these value A through E and store them in a temporary register A through E . First take the 512 bit message block and divide it into 16 sub-blocks each consisting of 32 bits. SHA has four rounds. In each round there are 20 steps (0 – 19, 20 – 39, . . . 60 – 79). Each round takes the current 512 bit block, the register contents of $a b c d$, and a constant $k(t)$, where $t = 0$ to 79, as inputs. Finally, we define four functions: f_{EXP} , f_{if} , f_{MAJ} , and f_{XOR} .

It first takes a 512 bit message as an argument; each of the other functions take three 32 bits.

The f_{EXP} function expands the initial 512 bit input message M (consisting of sixteen 32 bit words).

$$M_i \text{ with } 0 \leq i \leq 15 \text{ to } 80$$

32 bit words w_i with $0 \leq i \leq 79$

$$w_i = \begin{cases} M_i & \text{if } 0 \leq i \leq 15 \\ w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \ll 1 & \text{if } 16 \leq i \leq 79 \end{cases}$$

The other functions are defined as:

$$\begin{aligned} f_{if}(b, c, d) &= b \wedge c \oplus \neg b \wedge d \\ f_{MAJ}(b, c, d) &= b \wedge c \oplus b \wedge d + c \wedge d \\ f_{XOR}(b, c, d) &= b \oplus c \oplus d \end{aligned}$$

The SHA-1 process includes four rounds. Figure 9.9 represents a single SHA-1 round. Each round contains 20 iterations. This makes the total 80 iterations. The operation in an iteration is described as:

$$a b c d e = s^5(A_i) + E_i + K_i + f + w_i, A_i, s^{30}(B_i)$$

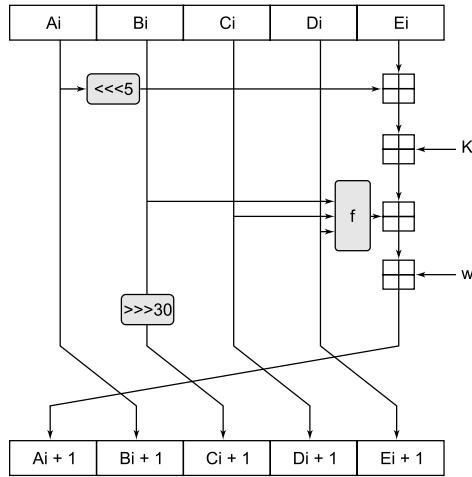


FIGURE 9.9 Single SHA-1 round.

Where

a, b, c, d, e = The variables

s^t = Circular-left-shift of the 32 bit sub-blocks by t-bytes.

w_i = A 32 bit derived from the current 32 bit sub-block.

f = One of the five additive constants.

w_i is calculated as:

For the first 16 words of w (i.e., $i=0$ to 15) the contents of the input message sub-block $M(i)$ are copied to w_i . The remaining 64 values of w are derived using the equation:

$$w_i = s^1(w[i - 16] \text{ XOR } w[1 - 14] \text{ XOR } w[i - 8] \text{ XOR } w[i - 3])$$

s^1 - indicates a circular left-shift operation.

TABLE 9.2 Comparison Between MD5 And SHA-1.

Parameters	MD5	SHA-1
• Message Digest length	128 bits	160 bits
• Number of rounds	64	80
• Initialization variables	4	5
• Collision complexity	2^{64}	2^{80}
• Speed	It is faster (64 iterations and 128 bit buffer)	It is slower than MD5 (80 iterations and 160 bit buffer)

SHA-256

SHA-256 is a 256 bit hash and is meant to provide 128 bits of security against collision attacks. The operation of SHA-256 is similar to that of MD4, MD5, and SHA-1. The message to be hashed is first padded with its length in such a way that the result is a multiple of 512 bits long and is then passed into 512 bit message blocks $M_{(1)}, M_{(2)}, \dots, M_{(N)}$; out of this only one message block is processed at a time.

Description of SHA-256

The entire process of the SHA-256 operation is explained in the following steps:

Step 1: Appending padding bits: The original message to be hashed is first padded with its length in such a way that the result is a multiple of 512 bits long. Suppose the length of the message M in bits is l ; append the bit “1” to the end of the message and then k zero bits, where k is the smallest nonnegative solution to the equation:

$$l + 1 + k = 448 \bmod 512$$

To this append the 64 bit block which is equal to the number l , written in binary.

For example: The 8 bit ASCII message “abc” has a length of $8 \times 3 = 24$.

This number is to be padded with a 1, then $448 - (24 + 1) = 423$ zero bits, and then its length to become the 512 bit padded message.

01100001 01100010 01100011 100.....0 00.....011000

The length of the padded message should now be a multiple of 512 bits.

Step 2: Preparing message blocks: The original message with padded bits is parsed into 512 bit message blocks $M_{(1)}, M_{(2)}, \dots, M_{(N)}$.

Parse the message into N 512 bits blocks $M_{(1)}, M_{(2)}, \dots, M_{(N)}$. The first 32 bits of message block i are denoted $M_{0(i)}$, the next 32 bits are $M_{1(i)}$, and so on up to $M_{15(i)}$. We use the big-endian conversion throughout, so within each 32 bit word, the leftmost bit is stored in the most significant bit.

Step 3: Preparing processing functions: The message blocks are processed one at a time, beginning with the fixed initial hash value $H(0)$ sequentially computed:

$$H(i) = H(i=1) + C_{M(i)} [H_{(i-1)}]$$

Where C is the SHA-256 compression function. The compression function operates on a 512 bit message block and a 256 bit intermediate hash value. It is essentially a 256 bit block cipher algorithm which encrypts the intermediate hash value using the message block as a key.

There are two main components:

1. The SHA-256 compression function.
2. The SHA-256 message schedule.

Consider the initial hash value $H(0)$ is the following sequence of 32 bit words, which are obtained by taking the factorial parts of the square roots of the first eight primes.

$$\begin{array}{ll} H_1^{(0)} = 6a09e667 & H_5^{(0)} = 510e527f \\ H_2^{(0)} = bb67ae85 & H_6^{(0)} = 9b05688c \\ H_3^{(0)} = 3c6ef372 & H_7^{(0)} = 1f83d9ab \\ H_4^{(0)} = a54ff53a & H_8^{(0)} = 5be0cd19 \end{array}$$

Step 4: Preprocessing: Computation of the hash of a message begins by preparing the message:

Hash Computation Procedure

For $M_{(i)}$, where $i = 1, 2, \dots, N$ number of blocks in the padded message.

Initialize the register: a, b, c, d, e, f, g, h with the $(i - 1)^{\text{st}}$ the intermediate hash value. The initial hash value is, when $i = 1$.

$$\begin{aligned} a &\leftarrow H_1^{(i-1)} \\ b &\leftarrow H_2^{(i-1)} \\ c &\leftarrow H_3^{(i-1)} \\ &\vdots \\ h &\leftarrow H_s^{(i-1)} \end{aligned}$$

Apply the SHA – 256 compression function to update registers a, b, c, \dots, h .

The six logical functions used in SHA-256 are listed as follows. Each of these functions operate on 32 bit words and produce a 32 bit word as output.

$$\begin{aligned} Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ \Sigma_0(x) &= s^2(x) \oplus s^{13}(x) \oplus s^{22}(x) \\ \Sigma_1(x) &= s^6(x) \oplus s^{11}(x) \oplus s^{25}(x) \\ \sigma_0(x) &= s^7(x) \oplus s^{18}(x) \oplus R^3(x) \\ \sigma_1(x) &= s^{17}(x) \oplus s^{19}(x) \oplus R^{10}(x) \end{aligned}$$

For $j = 0, 1, \dots, 63$

Compute $ch(e, f, g)$, $Maj(a, b, c)$

$$\Sigma_0(a), \Sigma_1(e) \text{ and } w_j$$

$$T_1 \leftarrow h + \Sigma_1(e) + Ch(e, f, g) + k_j + w_j$$

$$T_2 \leftarrow \Sigma_0(a) + Maj(a, b, c)$$

$$h \leftarrow g \quad g \leftarrow f \quad f \leftarrow e$$

$$e \leftarrow d + T_1 \quad d \leftarrow c \quad c \leftarrow b$$

$$b \leftarrow a \quad a \leftarrow T_1 + T_2$$

The i^{th} intermediate hash value $H(i)$ is computed as:

$$H_1^{(i)} \leftarrow a + H_1^{(i-1)}$$

$$H_2^{(i)} \leftarrow b + H_2^{(i-1)}$$

⋮ ⋮

$$H_8^{(i)} \leftarrow h + H_8^{(i-1)}$$

The hash of the message M is:

$$H^{(N)} = (H_1^{(N)}, H_2^{(N)}, \dots, H_8^{(N)})$$

The SHA-256 compression function is as given in Figure 9.10.

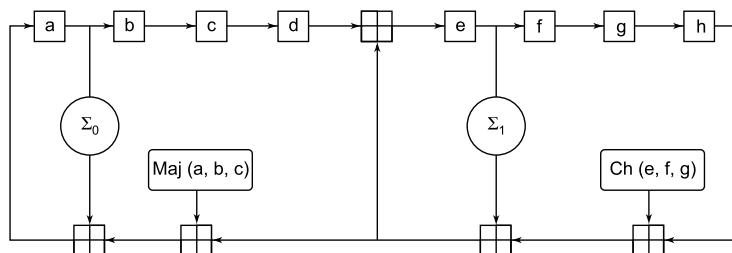


FIGURE 9.10 j^{th} internal steps of the SHA-256 compression process.

Where \square denotes mod2³² addition. The message scheduler register loaded with $w_0, w_1, w_2, \dots, w_{15}$ is represented in Figure 9.11.

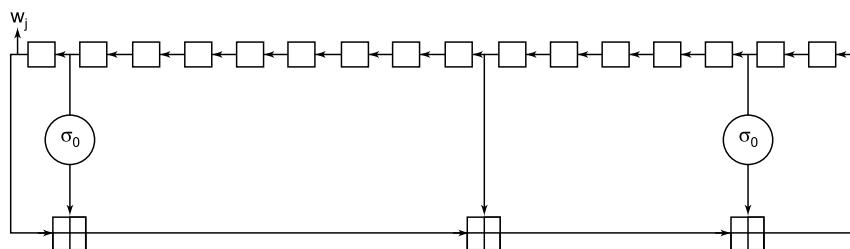


FIGURE 9.11 SHA-256 message scheduler.

SHA-512

SHA-512 is a variant of SHA-256, which is an evolution of SHA-1. SHA-512 is similar to SHA-256 except that it uses 1024 bit blocks and accepts as input a 2^{128} bit maximum length string. SHA-512 also has other algorithmic modifications in comparison with SHA-256. The different steps for generating the “hash” value of the message digest are discussed as follows:

Step 1: Padded with its length in such a way that the result is a multiple of 1024 bits long.

Step 2: Parsed into 1024 bit message blocks:

$$M^{(1)}, M^{(2)}, \dots M^{(N)}$$

The message blocks are processed one at a time, beginning with a fixed initial hash value $H(0)$ sequentially computed:

$$H^{(i)} = (H_1^{(i-1)} + C_M^{(i)}, [H^{(i-1)}])$$

The initial hash value $H(0)$ is the following sequence of 64 bit words (which are obtained by taking the fractional parts of the square roots of the first eight primes).

$$\begin{array}{ll} H_1^{(0)} = 6a09e667f3bcc908 & H_5^{(0)} = 510e527fade682d1 \\ H_2^{(0)} = bb67ae8584caa73b & H_6^{(0)} = 9b05688c2b3e6c1f \\ H_3^{(0)} = 3c6ef372fe94f82b & H_7^{(0)} = 1f83d9abfb41bsd6b \\ H_4^{(0)} = a54ff53a5f1d36f1 & H_8^{(0)} = 5be0cd19137e2179 \end{array}$$

The SHA-512 compression function operates on a 1024 bit message block and a 512 bit intermediate hash value. It is essentially a 512 bit block cipher algorithm which encrypts the intermediate hash value using the message block as keys. There are two main components:

1. The SHA-512 compression function.
2. The SHA-512 message schedule.

Step 3: Preprocessing

Computation of the hash of a message begins by preparing the message.

Hash Computation Procedure

For $M(i)$, where $i = 1, 2, \dots, N$ number of blocks in the padded message, initialize the registers a, b, c, d, e, f, g, h with the $(i - 1)^{\text{st}}$ intermediate hash value. The initial hash value is when $i = 1$.

$$\begin{aligned} a &\leftarrow H_1^{(i-1)} \\ b &\leftarrow H_2^{(i-1)} \end{aligned}$$

$$\vdots$$

$$h \leftarrow H_8^{(i-1)}$$

Apply the SHA-512 compression function to update the registers $a, b, c \dots h$. The six logical functions used in SHA-512 are listed as follows. Each of these functions operates on 64 bit words and produces a 64 bit output.

$$\begin{aligned} Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ \Sigma_0(x) &= s^{28}(x) \oplus s^{34}(x) \oplus s^{39}(x) \\ \Sigma_1(x) &= s^{14}(x) \oplus s^{18}(x) \oplus s^{41}(x) \\ \sigma_0(x) &= s^1(x) \oplus s^8(x) \oplus R^7(x) \\ \sigma_1(x) &= s^{19}(x) \oplus s^{61}(x) \oplus R^6(x) \end{aligned}$$

For $j = 0, 1, \dots, 79$

Compute	$Ch(e, f, g)$	$Maj(a, b, c)$	$\Sigma_0(a)$	$\Sigma_1(e)$	w_j
	$T_1 \leftarrow h + \Sigma_1(e) + Ch(e, f, g) + k_j + w_j$		$T_2 \leftarrow \Sigma_0(a) + Maj(a, b, c)$		
	$h \leftarrow g$	$g \leftarrow f$	$f \leftarrow e$		
	$e \leftarrow d + T_1$	$d \leftarrow c$	$c \leftarrow b$		
	$b \leftarrow a$	$a \leftarrow T_1 + T_2$			

The i^{th} intermediate value $H(i)$ is computed as

$$\begin{aligned} H_1^{(i)} &\leftarrow a + H_1^{(i-1)} \\ H_2^{(i)} &\leftarrow b + H_2^{(i-1)} \end{aligned}$$

$$\vdots$$

$$H_8^{(i)} \leftarrow h + H_8^{(i-1)}$$

The SHA-512 compression function is as given in Figure 9.12.

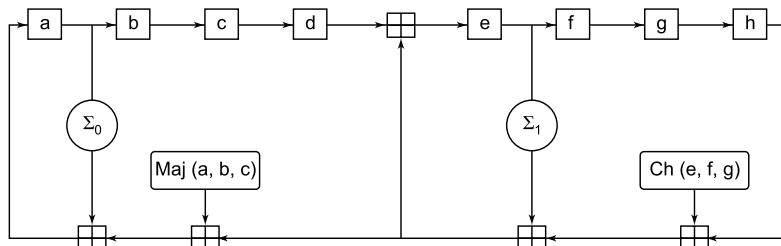


FIGURE 9.12 j^{th} internal step of the SHA-512 compression function C.

Where \boxplus denotes mod 2^{64} addition. The message schedule register loaded with $w_0, w_1, w_2, \dots, w_{15}$ is represented as in Figure 9.13.

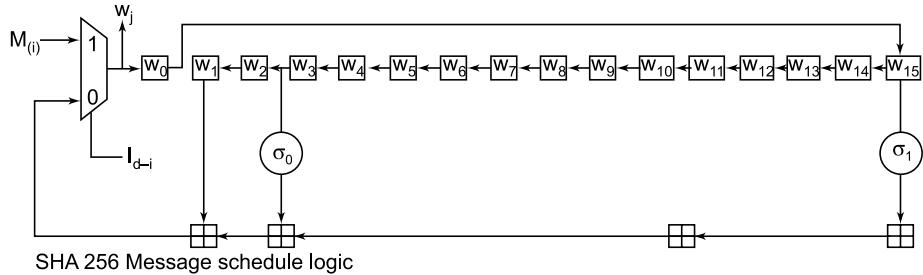


FIGURE 9.13 SHA-512 message schedule.

The message schedule can be implemented as a compact 16 word circular buffer $w_0, w_1, w_2, \dots, w_{15}$ and combination functions (σ_0, σ_1) that are cycled for the 64 clock cycle. The core logic as a word shift register with combinational logic that computes the next state for the word shifter registers. The hash core logic path with the core registers (a, b, \dots, h) and the combinational functions ($\text{Maj}, \text{ch}, \Sigma_0, \Sigma_1$). The whole operation is processed in a single clock cycle for each of the 64 cycles of the algorithm. In the combinational functions ($\text{Maj}, \text{ch}, \Sigma_0, \Sigma_1, \sigma_0, \sigma_1$), the sigma function shifters are implemented as fixed bit remap (zero - logic) and the Maj and Ch are canonical implementations of FIPS.

Comparisons of Different Hash Functions

TABLE 9.3 Standard Hash Functions at a Glance.

Name	Block Size (Bits)	Word Size (Bits)	Output Size (Bits)	Rounds
MD4	512	32	128	48
MD5	512	32	128	64
SHA-0	512	32	160	80
SHA-1	512	32	160	80
SHA-224	512	32	224	64
SHA-256	512	32	256	64
SHA-384	1024	64	384	80
SHA-512	1024	64	512	80

Collision Attacks on Hash Functions

Informally, a hash function is said to be collision resistant if it is hard to find any two inputs that map to the same output or digest for a given specification of the hash function. A hash function is said to be near-collision resistant if it is hard to find any two inputs such that their digests differ in only a few bits for a given specification of the hash function. Based on the IV used in finding collisions, collision attacks in hash functions are classified as follows:

1. **Collision attack:** collisions using a fixed IV for two distinct message inputs (*Type 1* collisions). A collision attack is considered to be a stronger attack compared to the other two, as the other two attacks deviate from the strict definition of a collision, which is to find two distinct inputs that map to the same output for a given specification of the hash function.
2. **Semi-free-start collision attack:** collisions using the same random (or arbitrary) IV for two distinct message inputs (*Type 2* collisions). Semi-free-start collision attacks are not so dangerous if the IV used in the attack is an outcome of a pseudorandom process, as the probability of hitting the right IV is negligible.
3. **Pseudo-collision attack:** free-start collision attack using two different IVs for two distinct message inputs (*Type 3* collision). Pseudo-collision attacks deviate significantly from the strict definition of a collision. In addition, many hash functions might not have considered this attack in their design criteria.

Multi-Block Collision Attacks on Hash Functions

The collision attacks on the hash functions MD5, SHA-0, and SHA-1 are multi-block collision attacks. Collisions have been found in these hash functions without ever getting a *Type 1* collision on the first application of the compression function (CF). The attacks use near-collisions obtained after processing the first distinct message blocks (x_1, x'_1) as a tool to get *Type 3* collisions for the second distinct message blocks (x_2, x'_2) , as shown in Figure 9.14. For example, 2-block collisions were found in MD5 and SHA-1 and 4-block collisions were found in SHA-0 (this was later improved in the collision format $H(M, M') = H(M, M'')$). One can append extra blocks after the collided blocks to extend the collisions, and this is possible due to the length extension property of the Merkle-Damgård structure. These collisions are basically a chain of trivial *Type 2* collisions.

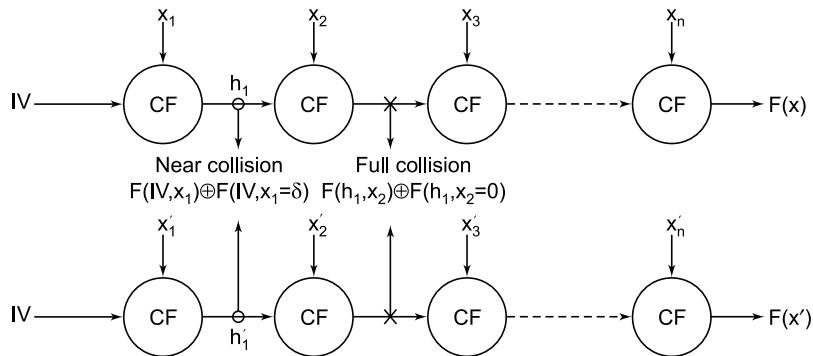


FIGURE 9.14 Two block collisions in hash algorithms.

SUMMARY

- A hash function usually means a mathematical function that compresses the message meaning so the output is shorter than the input.
- Hash functions contracted by iterative mode are more efficient than other types of functions.
- Merkle-Damgård construction has influenced the design of popular hash functions such as MD 4, MD 5, and SHA-0 and hash functions SHA-1, SHA-256, and SHA-512.
- A message digest is a cryptographic hash function containing a string of digits created by a one-way hash formula.
- MD 5 is a message digest algorithm and the most widely used secure hash algorithm, particularly for Internet standard message authentication.
- SHA was originally intended as the part of the digital signature standard (DSS).

REVIEW QUESTIONS

1. What is the one-way property in a hash function?
2. Explain about MD5 in detail.

3. Illustrate about the SHA algorithm and explain.
4. Define hash function.
5. What are the functions of hash functions in cryptography?
6. Explain SHA.
7. What are the requirements for hash functions?
8. What properties must be satisfied by a hash function?
9. What is meant by a message digest? Give an example.
10. Clearly discuss the Secure Hash Algorithm (SHA).
11. Compare the distinct features of the SHA-1 and MD5 algorithm.
12. Why do MD4, MD5, and SHA-1 require padding of messages that are already in multiples of 512 bits? What is the minimal and maximal amount of padding in each of these cases?
13. Discuss the properties required for a hash function to produce a secure message digest. Consider a brute force attack on a digitally signed message of length n bits and determine the time complexities of the computational properties of the hash function.
14. Compare the distinct features of the SHA-1 and MD5 algorithms.
15. In cryptography MD5 is a widely used cryptographic hash function with a 128 bit hash value. Explain the MD5 algorithm and its workings in detail.
16. What are the two basic attacks on a hash function? Explain them briefly.
17. Give an interpretation of the term collision resistance.
18. Illustrate the Secure Hash algorithm in brief.
19. What is the difference between a hash function and a message authentication code?
20. Give the structure of the SHA-512 compression function. Explain the structure of each round. Is the man in the middle attack possible on SHA-512?
21. Explain the compression of the Secure Hash Algorithm.

- 22.** What are the requirements of hash functions?
- 23.** Describe the steps in finding the message digest using the SHA-512 algorithm. What is the order of finding two messages having the same message digest?

MULTIPLE CHOICE QUESTIONS

- 1.** Which of the following are used to generate a message digest by the network security protocols?
(i) RSA (ii) SHA-1 (iii) DES (iv) MD5
(a) (i) and (iii) (b) (ii) and (iii) (c) (ii) and (iv) (d) (iii) and (iv)
- 2.** A digital signature needs a:
(a) Public key system (b) Private key system
(c) Public and private key system (d) None of the above
- 3.** The secure hash function or algorithm was developed by:
(a) NIST (b) ANSI (c) IEEE (d) None of the above
- 4.** A hash function is:
(a) Used to produce a Fingerprint of a file
(b) Useful for message authentication
(c) Both (a) and (b)
(d) None of the above
- 5.** Hash collision means:
(a) Two keys for one message
(b) One key for two message
(c) Two different keys for different messages
(d) Always the same key

- 6.** SHA-1 is similar to:
- (a) RSA (b) DES (c) MD5 (d) Rijndael
- 7.** What are MD4 and MD5?
- (a) Symmetric Encryption Algorithms
(b) Asymmetric Encryption Algorithms
(c) Hashing Algorithms
(d) Digital Certificates
- 8.** Design of modern hash function uses
- (a) Merkle-Damgård construction (b) Feistel Cipher
(c) XOR (d) D-Boxes
- 9.** The number of rounds in MD5 is
- (a) 6 (b) 4 (c) 8 (d) 16
- 10.** The block size of SHA-512 is
- (a) 256 (b) 1024 (c) 512 (d) 128

CHAPTER 10

DIGITAL SIGNATURE

10.1 INTRODUCTION

Authentication is any process through which one proves and verifies certain information. Sometimes one may want to verify the origin of a document, the identity of the sender, the time and date a document was sent and/or signed, the identity of a computer or user, and so on. A *digital signature* is a cryptographic means through which many of these may be verified. The digital signature of a document is a piece of information based on both the document and the signer's private key. It is typically created through the use of a hash function and a private signing function (encrypting with the signer's private key), but there are other methods also.

A conventional signature has the following salient characteristics: relative ease of establishing that the signature is authentic, the difficulties of forging a signature, the non-transferability of the signature, the difficulty of altering the signature, and the non-repudiation of the signature to ensure that the signer cannot deny signing.

A digital signature should have all the afore mentioned features of conventional signatures and also the ability to verify author, date, and time of signature and authenticate message contents. Their applications are also extended to secure e-mail and credit card transactions over the Internet.

Digital signatures and handwritten signatures both rely on the fact that it is very hard to find two people with the same signature. People use public key cryptography to compute digital signatures by associating something unique with each person. When public key cryptography is used to encrypt a message, the sender encrypts the message with the public key of the intended recipient. When public key cryptography is used to calculate a

digital signature, the sender encrypts the “*digital fingerprint*” of the document with his or her own private key. Anyone with access to the public key of the signer may verify the signature.

Suppose Alice wants to send a signed document or message to Bob. The first step is generally to apply a hash function to the message, creating what is called a message digest. The message digest is usually considerably shorter than the original message. In fact, the job of the hash function is to take a message of arbitrary length and shrink it down to a fixed length. To create a digital signature, one usually signs (encrypts) the message digest as opposed to the message itself. This saves a considerable amount of time, though it does create a slight insecurity. Alice sends Bob the encrypted message digest and the message, which she may or may not encrypt. In order for Bob to authenticate the signature he must apply the same hash function as Alice to the message she sent him, decrypt the encrypted message digest using Alice’s public key, and compare the two. If the two are the same he has successfully authenticated the signature. If the two do not match there are a few possible explanations. Either someone is trying to impersonate Alice, the message itself has been altered since Alice signed it, or an error occurred during transmission.

There is a potential problem with this type of digital signature. Alice not only signed the message she intended to but also signed all other messages that happen to hash to the same message digest. When two messages hash to the same message digest it is called a *collision*; the collision-free properties of hash functions are a necessary security requirement for most digital signature schemes. A hash function is secure if it is very time consuming, if at all possible, to figure out the original message given its digest. However, there is an attack called the *birthday attack* that relies on the fact that it is easier to find two messages that hash to the same value than to find a message that hashes to a particular value. In this chapter, we first discuss the concepts of digital signature and then the different digital signature schemes.

10.2 DIGITAL SIGNATURE

A digital signature is basically a way to ensure that an electronic document like e-mail, spreadsheets, text files, and so on are authentic. Authentic means that you know who created the document and also guaranteed that it has not been altered in any way since that person created it. Digital signatures rely on certain types of encryption to ensure authentication. Encryption is the

process of taking all the data that one computer is sending to another and encoding it into a form that only the other computer will be able to decrypt. Authentication is the process of verifying that information is coming from a trusted source. These two processes work hand in hand for digital signatures.

10.2.1 Characteristics of Digital Signatures

A conventional digital signature has the following salient characteristics:

1. Ease of establishing that the signature is authentic.
2. The difficulty of forging a signature.
3. The non-transferability of the signature.
4. The difficulty of altering the signature.
5. The non-repudiation of the signature to ensure that the signer cannot deny signing.

A digital signature should have all the aforementioned features of a conventional signature, plus a few more, as digital signatures are being used in practical but sensitive applications such as secure e-mail and credit card transactions over the Internet. Since a digital signature is just a sequence of zeros and ones, it is desirable for it to have the following properties:

- The signature must be a bit pattern that depends on the message being signed; i.e., for the same sender, the digital signature is different for different documents.
- The signature must use some information that is unique to the sender to prevent both forgery and denial.
- It must be relatively easy produce.
- It must be relatively easy to recognize and verify the authenticity of the digital signature.
- It must be computationally infeasible to forge a digital signature either by constructing a new message for an existing digital signature or developing a fraudulent digital signature for the given message.

10.2.2 Creating and Verifying Digital Signatures

Digital signatures are computed based on the message that needs to be signed. In practice instead of using the whole message, a hash function is applied to the message to obtain the message digest. Among the commonly used hash functions in practice are MD5 and SHA. These algorithms are

fairly sophisticated and ensure that it is highly improbable for two different messages to be mapped to the same hash value. There are two broad techniques used in digital signature computation—symmetric key cryptography and public key cryptosystems. In symmetric key systems a single key is shared between the messages sent and received; a public key cryptosystem uses a pair of keys, a private key to encrypt the message and a public key for decryption. For the purpose of confidentiality of the message to be sent, it would be encrypted with the owner's public key, which is now only to be decrypted by the receiver with the corresponding private key. For authentication a message would be encrypted with the private key of the originator or sender. This message could be decrypted by anyone using the public key of the sender. The process of creation of a digital signature is as shown in Figure 10.1.

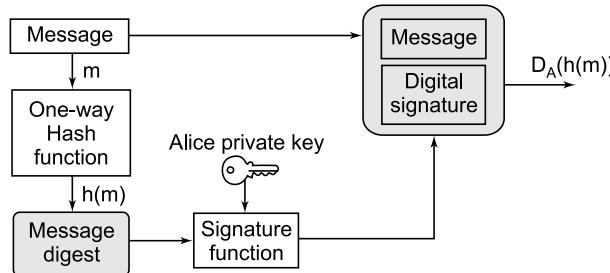


FIGURE 10.1 Creating a digital signature attached to the message.

A hash function is applied to the message that results a fixed size message digest. The signature function uses the message digest and sender's (Alice) private key to generate the digital signature. A very simple form of digital signature is computed as follows:

- Of the message a unique message digest is obtained with the help of a one-way hash function $h(m)$.
- The message digest is then *signed* (encrypted) with the private key of sender (Alice) D_A . Thus, the digital signature now consists of $D_A[h(m)]$.
- Alice (sender) then sends the message m along with her signature $D_A[h(m)]$ to Bob. However, the signature ensures authenticity of the sender.

At the receiver, the inverse signature function is applied to the digital signature to recover the original message digest. The received message is subjected to the same hash function to which the original message was

subjected. The resulting message digest is compared with the one recovered from the signature. If they match, then it ensures that the message has indeed been sent by the sender and that it has not been altered. The process of verifying a digital signature is shown in Figure 10.2. Bob will do the following after receiving the message:

- Of the message he will generate the one-way hash function $h(m)$.
- He will then decrypt the unique digital signature with the help of Alice's public key, E_A , which is available in the public file $E_A(D_A[h(m)])$.
- Bob will then compare the message digest obtained with the message digest computed by Alice. If the two match, then the message has been signed by Alice and the message is unaltered.

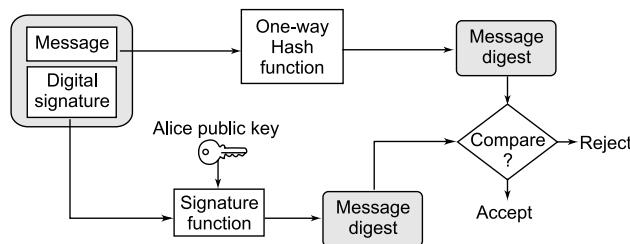


FIGURE 10.2 Verifying digital signature.

10.3 DIGITAL ENVELOPE

A digital envelope is a secure electronic data container that is used to protect a message through encryption and data authentication. A digital envelope allows users to encrypt data with the speed of secret key encryption and the convenience and security of public key encryption. To increase the speed of encryption the best solution is to combine public and secret keys in order to get both the security advantage of the public key system and the speed advantages of the secret key system. The outline of creating a digital envelope is as shown in Figure 10.3.

The digital envelope is created as follows:

1. A symmetric key is generated.
2. The symmetric key is then used to encrypt the file or message.
3. The symmetric key is then encrypted with the help of the receiver's public key system.

4. The encrypted message and the encrypted symmetric key are put together to form the digital envelope.

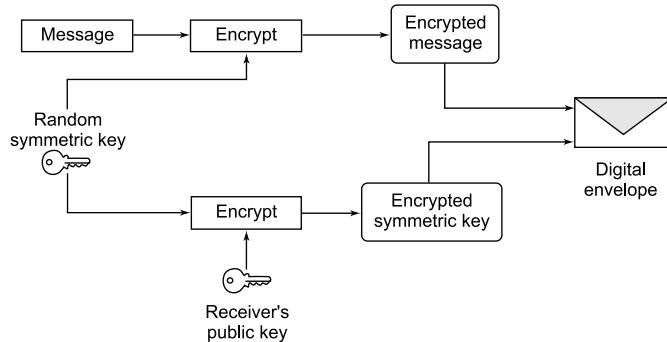


FIGURE 10.3 Creating a digital envelope.

The process of opening the digital envelope and recovering the content is as shown in the Figure 10.4.

1. The symmetric key is recovered by a decryption process using the recipient's private key.
2. The encrypted message obtained from the digital signature is decrypted by using the random symmetric key.

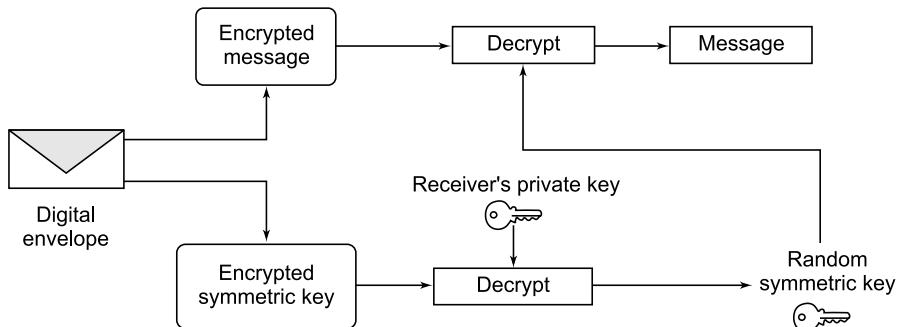


FIGURE 10.4 Opening a digital envelope.

10.3.1 Digital Envelope Carrying Signed Message

The process of creating a digital envelope containing a signed message is as shown in Figure 10.5.

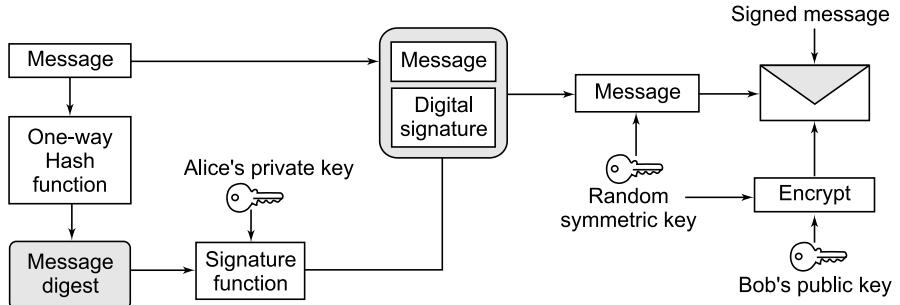


FIGURE 10.5 Creating a digital envelope carrying a signed message.

The following steps are used for the creation of a digital envelope carrying the signal message:

1. A digital signature is created by the signature function using the message digest of the message and the sender's (Alice) private key.
2. The original message and the digital signature are then encrypted by the sender (Alice) using a randomly generated symmetric key algorithm.
3. The symmetric key itself is encrypted using the recipient's (Bob's) public key.
4. The combination of encrypted message and signature, together with the encrypted symmetric key, form the digital envelope containing the signed message.

The process of opening the digital envelope, recovering the message, and verifying the signature is as shown in Figure 10.6.

1. First, the symmetric key is recovered using the recipient's private key.
2. This symmetric key is then used to decrypt and recover the message and digital signature.
3. The message is converted into a message digest using a hash function.
4. The message digest obtained in the previous steps is compared to verify.

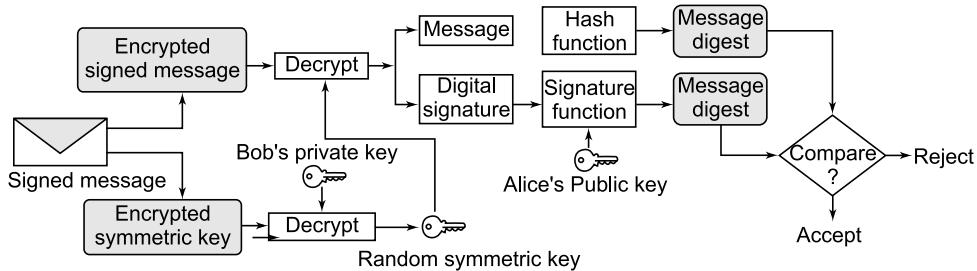


FIGURE 10.6 Opening a digital envelope and verifying a digital signature.

10.4 DIGITAL SIGNATURE STANDARD (DSS)

DSS is part of the U.S. government Capstone program to develop cryptography and security standards that must be used by government agencies and private companies doing business with the government as specified by the Federal Information Processing Standard (FIPS). NIST and the National Security Agency (NSA) are responsible for Capstone. Capstone's major components include an encryption algorithm called Skipjack (and a Skipjack encryption chip called Clipper), a hash function like SHA-1 and DSA. Capstone is also working in a key exchange protocol.

The following algorithms are suitable for digital signature generation under the DSS standard:

1. Digital Signature Algorithm (DSA)
2. The RSA algorithm
3. ElGamal Signature Scheme
4. The Elliptic Curve Digital Signature Algorithm (ECDSA)

Also in DSS a hash function is used for the signature generation process. These hash functions are known as the Secure Hash Standard (SHS), which are the specifications for the Secure Hash Algorithm (SHA).

10.4.1 Digital Signature Algorithm (DSA)

DSA is a standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in 1991 for use in the United States as the Digital Signature Standard, specified in FIPS-186 in 1993. A

DSA digital signature is computed using a set of domain parameters, a private key (x), a pre-message secret number (K), data to be signed, and a hash function. A digital signature is verified using the same domain parameters, a public key (y) that is mathematically associated with the private key (x) used to generate the digital signature, data to be verified, and the same hash function that was used during signature generation.

The DSA algorithm can be understood in two different phases:

Phase I: public and private key generation.

Phase II: signature generation and signature verification.

Public and private key generation: It can be described in the following steps:

1. Choose a prime number q which is called the *prime divisor*.
2. Choose another prime number p , such that $p - 1 \bmod q = 0$ is called the *prime modulus*.
3. Choose an integer g , such that $1 < g < p$
 $g^p \bmod p = 1$ and $g = h^{((p-1)/q)} \bmod p$, q is also called g 's multiplicative order modulo p .
4. Choose an integer such that $0 < x < q$.
5. Compute y as $g^x \bmod p$. The parameters p , q , and g are made public.
6. Package the public key as $\{p, q, g, y\}$.
7. Package the private key as $\{p, q, g, x\}$.

Signature generation and signature verification: To generate a message signature, the sender can follow these steps:

1. Generate the message digest h using a hash algorithm like SHA-1.
2. Generate a random number k , such that $0 < k < g$.
3. Compute r as $(g^k \bmod p) \bmod q$. If $r = 0$ select a different k .
4. Compute i such that $k^i \bmod q = 1$; i is called the *modular multiplicative inverse* of k modulo q .
5. Compute $s = i*(h + r^s) \bmod q$ if $s = 0$, and select a different k .
6. Package the digital signature as $\{r, s\}$.

To verify a message signature, the receiver of the message and the digital signature can follow these steps:

1. Generate the message digest using the same hash algorithm.
2. Compute w such that $s*w \bmod q = 1$; w is called the *modular multiplicative inverse* of s modulo q .
3. Compute $u_1 = r*w \bmod q$

$$u_2 = r*w \bmod q$$

$$v = (((g^{u_1})(y^{u_2})) \bmod p) \bmod q.$$

4. If $v = r$, the digital signature is valid.

10.4.2 RSA Digital Signature

The RSA digital signature scheme applies the sender's private key to a message to generate a signature. The signature can then be verified by applying the corresponding public key to the message and putting the signature through the verification process, providing either a valid or invalid result. Any signature generated by the first operation will always verify correctly with the second operation if the corresponding public key is used. If the signature was generated differently or if the message was altered after being signed, then the digest generated at the receiver differs. The RSA digital signature is as shown in Figure 10.7.

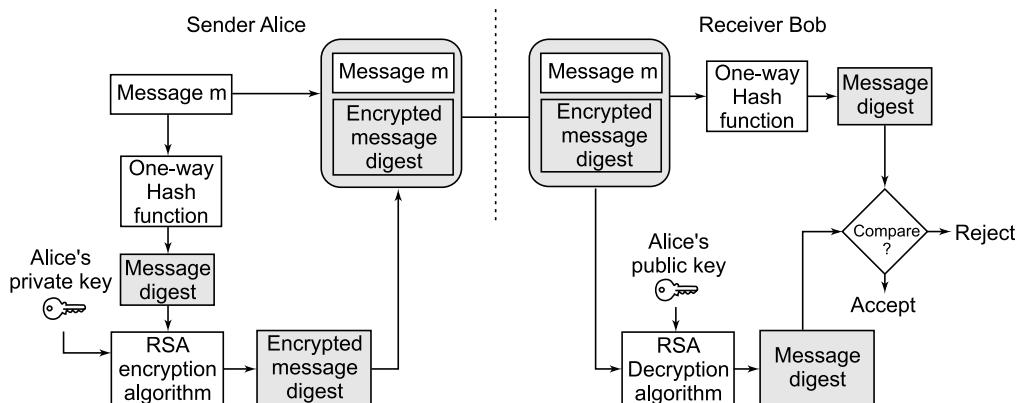


FIGURE 10.7 Digital signature on RSA.

The signing process in DSS using DSA is as shown in Figure 10.8. It also makes use of a hash function. The hash code is provided as input to a signature function together with a random number generated for this particular signature. The signature function also uses the sender's private key and a set of parameters known to a group of communicating parties, referred to as a global public key. The output signature consists of two components.

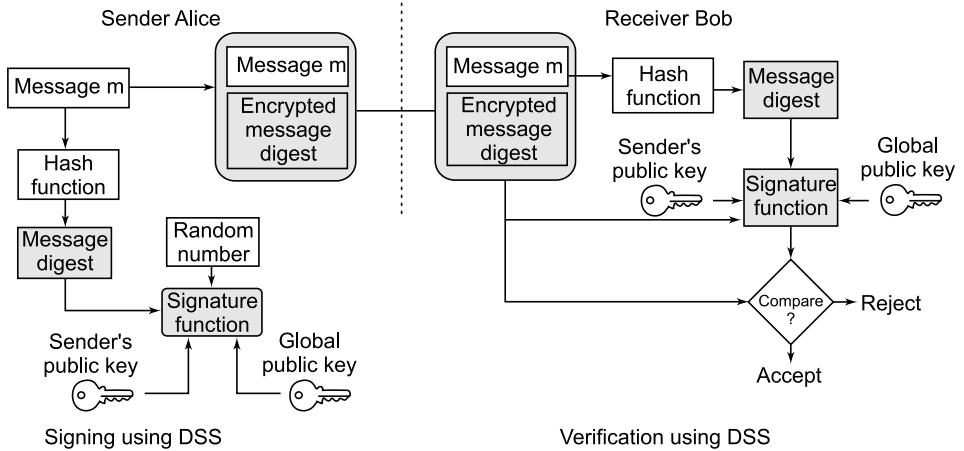


FIGURE 10.8 Signing and verification process using DSS.

The signature verification process is as shown in Figure 10.8. At the receiving end the hash code of the incoming message is generated and the message digest is input to a verification function together with two components of the signature. The verification function uses the global public key as well as the sender's public key and recreates the original message digest. A match between the recreated and the original signature indicates the authenticity of the message.

Bob has a document that Alice agrees to sign. They do the following:

1. Alice generates two large prime numbers p, q and computes $n = pq$

She chooses e_A such that $1 < e_A < \phi(n) = 1$ and calculates d_A

such that $e_A d_A \equiv 1 \pmod{\phi(n)}$.

Alice publishes (e_A, n) and keeps private d_A, p, q .

2. Alice's signature is $y \equiv m^{d_A} \pmod{n}$.
3. The (m, y) pair is then made public.

Bob can then verify that Alice really signed the message as:

1. Bob downloads Alice's (e_A, n) which is made public.
2. Bob then downloads computer $z \equiv (mod n)$.

If $z = m$, then Bob accepts the signature as valid, otherwise the signature is invalid.

If an intruder Eve wants to attack Alice's signature to another message m_1 , Eve cannot simply use the pair (m_1, y) ,

Since $y^{e_A} \equiv m_1 (\text{mod } n)$

Therefore, Eve needs to compute y_1 with $y_1^{e_A} \equiv m_1 (\text{mod } n)$. This is very hard and the complexity is identical to that of the decryption of the RSA ciphertext to obtain plaintext. In another case Eve chooses y_1 first then lets the message be $m_1 = y_1^{e_A} (\text{mod } n)$. It does not appear that Alice can deny having signed the message m_1 under the present scheme. Here the signature will assure that the message has come from an authorized sender Alice.

10.4.3 ElGamal Signature Scheme

The ElGamal signature scheme is a digital signature scheme that is based on the difficulty of computing discrete logarithms. It is different from the ElGamal encryption technique. The ElGamal signature algorithm described here is rarely used in practice. A variant developed at the NSA and known as the Digital Signature Algorithm is much more widely used. The ElGamal signature scheme allows that a verifier can confirm the authenticity of a message m sent by the signer, sent to him over an insecure channel.

Parameters Shared Between Users in the ElGamal Signature Scheme

H – Collision resistant hash function

p – Large prime, such that computing discrete logarithm modulo p is difficult
 $g < p$ is a randomly chosen generator of a multiplicative group of integers modulo p Z_p^*

Secret Key and Public Key Generation

User A chooses a secret key $X_A \in \{0, \dots, (p - 1)\}$.

Compute $y_A = g^{X_A} (\text{mod } p)$.

A publishes his public key as (p, g, y_A) .

These steps are performed once by the signer.

Computation of signature: To sign a message m the user A performs the following steps:

- The hash $h = H(m)$ where $0 \leq h < (p - 1)$
- To compute a signature on m , user A proceeds as follows:
- First choose a random value K such that $0 < K < p - 1$ and $\gcd(K, (p - 1)) = 1$
- Compute temporary key $r \equiv g^k \pmod{p}$
- Compute K^{-1} the $K \text{ mod } (p - 1)$
- Compute the value $s \equiv (h - X_A r)K^{-1} \pmod{p - 1}$
- The signature is (r, s)

Then the pair (r, s) is the digital signature of message m . The signer (A) repeat these steps for every signature.

Verification of a signature: Any user B knowing the public key y_A can verify the signature by computing:

$$V_1 = g^h \pmod{p} \text{ and } V_2 = y_A^r r^s \pmod{p}$$

The signature is valid if $V_1 = V_2$

The verifier accepts a signature if all conditions are satisfied and rejects it otherwise.

Example: Consider $p = 547$, $g = 9$, $X_A = 23$

$$\begin{aligned} \text{Compute } y_A &= g^{x_A} \pmod{p} \\ &= 9^{23} \pmod{547} \\ &= 81 \end{aligned}$$

$$\begin{aligned} p - 1 &= 547 - 1 \\ &= 546 \end{aligned}$$

Alice's key pair is $(23, 81)$

Take a message $m = 100$ and $K = 125$ for the signature of this message.

Here we notice that $(125, 546) = 1$

We compute $125^{-1} \pmod{546} = 83$

Having these parameters, we can start to calculate the signature of Alice on the message m , which is represented by the pair (r, s)

$$\begin{aligned} r &= g^k \bmod p \\ &= 9^{125} \bmod 547 \\ &= 304 \\ s &= \frac{m - (X_A \times r)}{K} \bmod p \\ &= (100 - 23 \times 304) \times 83 \bmod 546 \\ &= 172 \end{aligned}$$

Verification Steps

$$\begin{aligned} g^m &= y_A^r \times r^s \bmod p \\ 81^{304} \times 304^{172} &\equiv 81 \pmod{547} \\ 9^{100} &\equiv 81 \pmod{547} \end{aligned}$$

The verification confirms that the signature is valid.

10.4.4 Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm is used for authenticating a message sent by the signer.

For example, consider Alice and Bob want to exchange a message. To authenticate a message sent by Alice, she has to sign the message using her private key. She then sends the message and the signature to the receiver Bob in an unsecure channel. Bob, after receiving the message, can verify the signature only by using Alice's public key. Since Bob knows Alice's public key, he can verify whether the message is indeed sent by Alice or not.

ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups. For sending a signed message from Alice to Bob, both have to agree on elliptic curve domain parameters. Sender Alice has a key pair consisting of a private key d_A (a randomly selected integer less than n , where n is the order of the curve, an elliptic curve domain parameter) and a public key $Q_A = d_A \times G$ (where G is the generator point, an elliptic curve domain parameter). The following steps give an overview of ECDSA process.

Signature Generation

1. For signing a message m Alice uses her private key d_A .
2. Calculate $h = H(m)$, where H is a cryptographic hash function, such as SHA-1.
3. Select a random integer k from $(1, n - i)$.
4. Calculate $r = x_1 \pmod{n}$, where $(x_1, y_1) = K \times G$. If $r = 0$, go to step 3.
5. Calculate $S = K - 1(h + d_A r) \pmod{n}$. If $s = 0$, go to step 3.
6. The signature is the pair (r, s) .

Signature Verification

For Bob to authenticate Alice's signature, he must have Alice's public key Q_A .

1. Verify that r and s are integers in $(1, n - 1)$. If not, the signature is invalid.
2. Calculate $h = H(m)$, where H is the same function used in the signature generation.
3. Calculate $w = s - 1 \pmod{n}$.
4. Calculate $u_1 = hw \pmod{n}$ and $u_2 = rw \pmod{n}$.
5. Calculate $(x_1, y_1) = u_1 G + u_2 Q_A$.
6. The signature is valid if $x_1 = r \pmod{n}$, invalid otherwise.

SUMMARY

- A digital signature of a document is a piece of information based on the document and the signer's private key.
- The characteristics of a conventional signature are: relative ease of establishing that the signature is authentic, the difficulties of forging a signature, the non-transferability of the signature, the difficulty of altering the signature, and the non-repudiation of the signature to ensure that the signer cannot deny signing.
- In practice instead of using the whole message, a hash function is applied to the message to obtain the message digest.

- The two broad techniques used in digital signature computation are symmetric key cryptography and public key cryptography.
- A digital envelope is a secure electronic data container that is used to protect a message through encryption and data authentication.
- DSS is a standard and DSA is an algorithm. The DSA algorithm can be understood in two phases, as public and private key generation and signature generation and signature verification.

REVIEW QUESTIONS

1. What is the use of a digital signature? What are the requirements of a digital signature scheme?
2. Write short notes on the Digital Signature Algorithm.
3. What are the differences between a digital signature and a digital certificate?
4. Explain the concept of a digital signature.
5. What is a digital signature? Explain the characteristics of a digital signature.
6. What is a digital envelope? Explain the signing process in a digital envelope.
7. What is the Digital Signature Algorithm (DSA)? Explain the signature generation and signature verification process.
8. Explain the process of the Elliptic Curve Digital Signature Algorithm (ECDSA).
9. Explain the ElGamal signature scheme.
10. Describe the RSA digital signature.

MULTIPLE CHOICE QUESTIONS

1. What is an advantage of RSA over DSS?
 - (a) It can provide digital signature and encryption functionality
 - (b) It uses fewer resources and encrypts quicker because it uses symmetric keys
 - (c) It is a block cipher versus a stream cipher
 - (d) It employs a one-time encryption pad
2. When public key cryptography is used to calculate a digital signature, the sender encrypts the _____ of the document with his or her own private key.

(a) key	(b) digital fingerprint
(c) ciphertext	(d) none of the above
3. When two messages hash to the same message digest it is called

(a) collision	(b) attack
(c) intrusion	(d) none of the above
4. A digital signature is basically a way to ensure that an electronic document is

(a) secure	(b) authentic
(c) verified	(d) none of the above
5. To increase the speed of encryption of the best solution is to consider a public and _____ key in order to get both the security advantages.

(a) secret key	(b) private key
(c) session key	(d) none of the above
6. DSS uses the _____ hash function for the signature generation process.

(a) Secure Hash Standard	(b) MD5
(c) SHA-1	(d) SSL
7. A digital signature needs a

(a) Public key system	(b) Private key system
(c) Public and private key system	(d) none

8. A Digital envelope is a secure electronic data container that is used to protect messages through _____ and data _____.
 - (a) Encryption; Authentication
 - (b) Encryption; Integrity
 - (c) Authentication; Integrity
 - (d) Authentication; Non-repudiation
9. Hash collision means:
 - (a) Two keys for one message
 - (b) One key for two messages
 - (c) Two different keys for different messages
 - (d) Always the same key
10. In the digital signature technique, the sender of the message uses _____ to create ciphertext:

(a) Own symmetric key	(b) Own private key
(c) The receiver's private key	(d) The receiver's public key

CHAPTER

11

ENTITY AUTHENTICATION

11.1 INTRODUCTION

Authentication provides a means of reliability identification of an entity. The most common verification technique is to check whether the plaintiff possesses information or characteristics that a genuine entity should possess. In a network a computer can authenticate humans through:

- Passwords that authenticate what the user *knows*.
- A smart card and physical key that authenticate what the user *has*.
- A biometric device such as retinal scanners, fingerprint analyzer, and voice recognition systems that authenticate *who* the user is.

These components used for authentication are cheap and convenient. The password has become the most popular technique for authenticating users trying to access confidential data stored in computers. However, password-based authentication is vulnerable to several forms of attacks. People generally select short, easily memorable passwords to log in to a server without considering that password-based authentication methods are susceptible to attacks if used on unsecure communication channels like the Internet. Meanwhile, complex passwords might get lost or stolen when users write them down, defeating the purpose of constructing a secure password-based authentication scheme in the first place.

11.2 ENTITY AUTHENTICATION

Entity authentication is defined as the process whereby one party is assured of the identity of a second party involved in a protocol and that the second has actually participated. The outcome of an entity authentication protocol is either acceptance of the claimant's identity as authentic or termination without acceptance. Entity authentication typically involves no meaningful message other than the claim of being a particular entity.

Identification Process

Entity authentication techniques may be divided into three main categories, with security based on one of the following:

1. **Something known:** in this process the secret is known only by the claimant and it can be verified by the verifier.

Example: standard passwords, personal identification numbers (PIN), private keys, etc.

2. **Something possessed:** this is typically a physical accessory which can prove the identity of the claimants.

Example: Passport, magnetic strip cards, credit cards, smart cards, etc.

3. **Something inherent:** this process includes a method that makes use of human physical characteristics and involuntary actions.

Example: Handwritten signature, biometrics, fingerprints, voice recognition, iris pattern, hand geometric, dynamic keyboarding characteristics.

11.3 PASSWORD AUTHENTICATION

A password is a set of secret characters or words utilized to gain access to a computer, webpage, network resource, or database. Passwords help to ensure that computers or data are accessed only by the authorized users who have been granted the right to view or access them.

Strong password: term used to describe a password that is an effective password that would be difficult to break. Often a strong password has six to ten characters, numbers, special symbols, and both uppercase and lowercase characters in combination.

Weak password: A password that is not an effective password, because it's easy to remember. Names, birth dates, and phone numbers are easily guessable. They are considered to be weak passwords.

11.3.1 Password Attacks

The three most common ways of guessing passwords are *dictionary attacks*, *brute-force attacks*, and *key logger attacks*.

Dictionary Attacks: A dictionary attack uses a file containing words, phrases, common passwords, and other strings that are likely to be used as a password. Passwords found in any online or available list of words may be uncovered by an adversary who tries all words in this list using a dictionary attack. Each word in the password file is hashed, and its hash is compared to the password hash. If they match, that word is the password. These dictionary files are constructed by extracting words from large groups of text, and even from real databases of passwords. Further processing is often applied to dictionary files, such as replacing words with their equivalents ("hello" becomes "h3l10"), to make them more effective.

A brute-force attack: A brute-force attack tries every possible combination of characters up to a given length. These attacks are very computationally expensive, and are usually the least efficient in terms of hashes cracked per processor time, but they will always eventually find the password. Passwords should be long enough that searching through all possible character strings to find it will take too long to be worthwhile. There is no way to prevent dictionary attacks or brute-force attacks. They can be made less effective, but the only way to secure the password from dictionary or brute-force attack is by having a robust password hashing system.

Key Logger Attack: A hacker uses a program to track all of a user's keystrokes. Most of the user's activities such as their login IDs and passwords have been recorded. A key logger attack is different from a brute force or dictionary attack in many ways. It captures every key pressed on the keyboard and stores it down in a file or memory bank that can be viewed by the person performing the monitoring in real time.

Password Sniffing

Password sniffing is a process of monitoring a network to gather information that may be useful in an attack. One of the things that can be observed through this sniffing process is a password. With proper tools, a hacker can monitor the network packet to obtain passwords or IP addresses. Password sniffing is particularly a threat for users logging into a system over a network.

using TELNET, RLOGIN, FTP, or a terminal emulator. The system passes the clear text password through the password encryption algorithm and compares it to the value stored in the password file. If they match then the user is authenticated and allowed to access the system.

Generally programs such as TELNET, RLOGIN, and terminal emulators do not encrypt passwords entered at login for transmission to the system. As a result when a user enters his or her password, it is transmitted in the clear text form, meaning anyone monitoring the network with a sniffer can read the password. Password sniffing threats can be reduced or eliminated by employing a virtual private network (VPN) connection and using a program like Secure Socket Shell (SSH). SSH communication is encrypted using IDEA, DES 3DES, or RC4 algorithms. Encryption keys are exchanged using the RSA key exchange process. With this high level of encryption in a network, SSH can protect against IP spoofing, IP source routing, DNS spoofing, and interception of clear text passwords and other data by an intermediate host. Another countermeasure to password sniffing is to use *one-time passwords* (OTP).

The most widely implemented scheme employs smart cards or token cards. One of the best known products is RSA's secure ID, which uses a time-based token card. The card displays a number that is synchronized with a login server. To access a system employing secure ID, it is necessary to enter the synchronized number. The number changes constantly and is never the same twice.

Social Engineering

Social engineering is the art of manipulating people so they give up confidential information. In this method the hackers posing as system administrators, calling end users or posing as end users who called the IP support line, have been very successful in their attempts to gain passwords to systems. When individuals are targeted the criminals are usually trying to trick you into giving them your passwords or bank information, or trying to access your computer to secretly install malicious software that will give them access to your passwords and bank information as well as give them control over your computer. This type of problem is one of the most difficult to control, because it requires modifying people's behavior, and there is no technology that you can implement to prevent it. Most people are trusting by nature and are not on guard for this type of operation. The only way to prevent this type of ploy from being successful is through educating the users.

Common Social Engineering Attacks

- **E-mail from a friend:** If a criminal manages to hack or socially engineer one person's e-mail password, they have access to that person's contact list—and because most people use one password everywhere, they probably have access to that person's social networking contacts as well. Once the criminal has that e-mail account under their control, they send e-mails to all the person's contacts or leave messages on all their friend's social pages, and possibly on the pages of the person's friend's friends.
- **Phishing attempts:** Typically, a phisher sends an e-mail, instant message, comment, or text message that appears to come from a legitimate, popular company, bank, school, or institution.
- **Fictitious Competition:** The social engineer manipulates a group of users to participate in some fake competition for a jackpot prize, with the ultimate purpose of eventually extracting confidential information about network and password security.
- **The Helpful Help Desk:** The help desk gets a call from the social engineer impersonating a user reporting a forgotten password. In many cases the help desk will change the user's password over the phone. The hacker now has a legitimate user name and password to work with. To avoid problems from the original user, the social engineer will then call the user who was impersonated and say something like "This is Alice from the MIS department. We had some problems with security today, so we have changed your password. Your new password is *abc123*."
- **Baiting scenarios:** These social engineering schemes know that if you dangle something people want, many people will take the bait. These schemes are often found on peer-to-peer sites offering a download of something like a hot new movie or music. But the schemes are also found on social networking sites, malicious Websites you find through search results, and so on.
- **Response to a question you never had:** Criminals may pretend to be responding to your "request for help" from a company while also offering more help. They pick companies that millions of people use like a software company or bank. If you don't use the product or service, you will ignore the e-mail, phone call or message, but if you do happen to use the service, there is a good chance you will respond because you probably do want help with a problem.

11.3.2 How to Store Passwords Safely

Storing a password much more safely is more difficult. The following are popular methods used to store a password in a database:

1. **Store password unencrypted:** The most evident approach is for the system to store user passwords in the form of clear text in a system password file, which is both read and write protected (using the operating system access control privilege). To access any account upon password entry by the user, the system compares the entered password to the password file entry for the corresponding user ID. If it is a valid password, it is accepted; otherwise, it is rejected. The advantage of using this technique is that if someone forgets their password, the system administrator can just look it up and tell them what it is. A drawback of this method is they get a glimpse into the sort of password that each user seems to favor for other accounts belongs to that user. Storage of a password file on a database is also a security concern. Since the file contains clear text passwords, neither the user nor the system administrator should be able to look up a user's password.
2. **Encrypt the password in the database:** In this method the passwords are encrypted before storing them in the database. There is no chance of accidentally viewing them in the database, and if the password data should get stolen, it would just be shredded garbage to the crooks. Using symmetric key encryption offers an advantage, because it can automatically re-encrypt every password in the database if ever there is a change in the encryption key.

Password Hashing: Hash algorithms are one-way functions. They turn any amount of data into a fixed-length “fingerprint” that cannot be reversed. They also have the property that if the input changes by even a tiny bit, the resulting hash is completely different. This technique is used to store passwords in a form that protects them even if the password file itself is compromised, but at the same time, we need to be able to verify that a user's password is correct. The general workflow for account registration and authentication in a hash-based account system is as follows:

1. The user creates an account.
2. Their password is hashed and stored in the database. At no point is the plaintext (unencrypted) password ever written to the hard drive.

3. When the user attempts to login, the hash of the password they entered is verified against the hash of their real password retrieved from the database.
4. If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
5. Steps 3 and 4 repeat every time someone tries to login to their account.

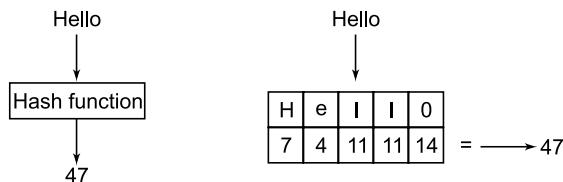


FIGURE 11.1 Password hashing.

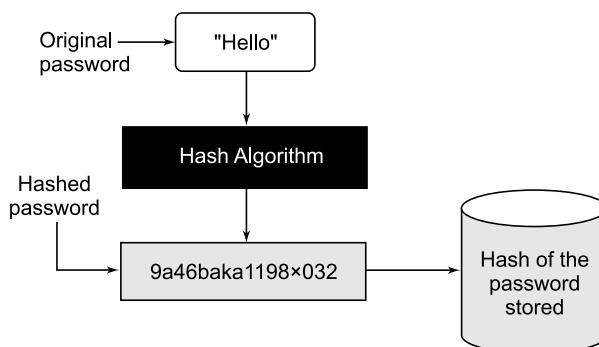


FIGURE 11.2 How a password is stored using a hash.

In step 4, never tell the user if it was the username or password they got wrong. Always display a generic message like “*Invalid username or password.*” This prevents attackers from enumerating valid usernames without knowing their passwords. There is a difference between the hash functions used to protect passwords and the hash functions used in data structures. The hash functions used to implement data structures such as hash tables are designed to be fast, not secure. Only *cryptographic hash functions* may be used to implement password hashing. Hash functions like SHA256, SHA512, RipeMD, and WHIRLPOOL are cryptographic hash functions.

How Hashes are Cracked: The simplest way to crack a hash is to try to guess the password, hashing each guess, and check if the guess's hash equals the hash being cracked. If the hashes are equal, the guess is the password.

Lookup Tables: Lookup tables are an extremely effective method for cracking many hashes of the same type very quickly. The general idea is to pre-compute the hashes of the passwords in a password dictionary and store them, and their corresponding password, in a lookup table data structure. A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

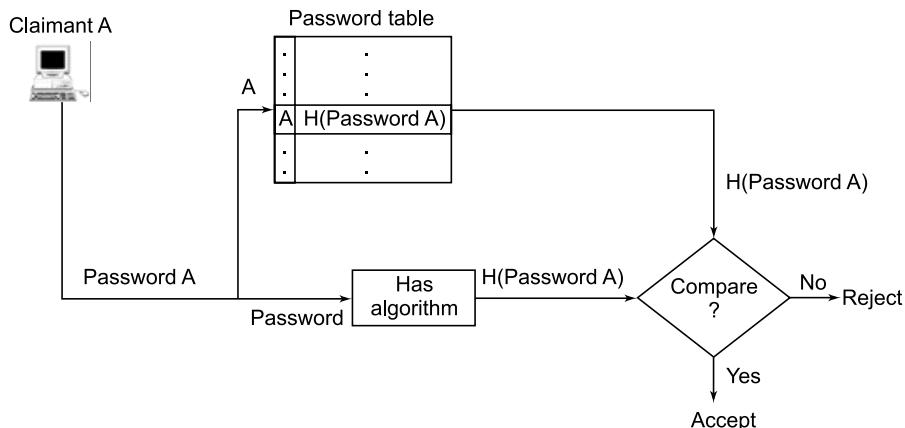


FIGURE 11.3 One-way hash function for password verification.

Reverse Lookup Tables: In this method an attacker applies a dictionary or brute-force attack to many hashes at the same time, without having to pre-compute a lookup table. First, the attacker creates a lookup table that maps each password hash from the compromised user account database to a list of users who had that hash. The attacker then hashes each password guess and uses the lookup table to get a list of users whose password was the attacker's guess. This attack is especially effective because it is common for many users to have the same password.

Rainbow Tables: Rainbow tables are a time-memory trade-off technique. They are like lookup tables, except that they sacrifice hash cracking speed to make the lookup tables smaller. Because they are smaller, the solutions to more hashes can be stored in the same amount of space, making them more effective. Rainbow tables exist that can crack any MD 5 hash of a password up to 8 characters long.

Adding Salt

Lookup tables and rainbow tables only work because each password is hashed the exact same way. If two users have the same password, they'll have the same password hashes. We can prevent these attacks by randomizing each hash, so that when the same password is hashed twice, the hashes are not the same. We can randomize the hashes by appending or prepending a random string, called a *salt*, to the password before hashing. As shown in Figure 11.4, this makes the same password hash into a completely different string every time. To check if a password is correct, we need the salt, so it is usually stored in the user account database along with the hash, or as part of the hash string itself. The salt does not need to be secret. Just by randomizing the hashes, lookup tables, reverse lookup tables, and rainbow tables become ineffective. An attacker won't know in advance what the salt will be, so they can't pre-compute a lookup table or rainbow table. If each user's password is hashed with a different salt, the reverse lookup table attack won't work either. The wrong way of incorrectly using salt is reusing the same salt in multiple hashes, or using a salt that is too short.

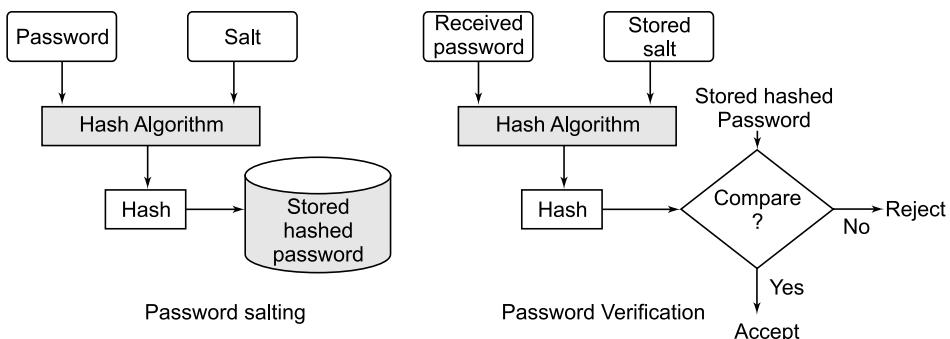


FIGURE 11.4 Password salting verification.

Salt Reuse

A common mistake is to use the same salt in each hash. Either the salt is hardcoded into the program or is generated randomly once. This is ineffective because if two users have the same password, they'll still have the same hash. An attacker can still use a reverse lookup table attack to run a dictionary attack on every hash at the same time. They just have to apply the salt to each password guess before they hash it. If the salt is hardcoded into a

popular product, lookup tables and rainbow tables can be built for that salt to make it easier to crack hashes generated by the product. A new random salt must be generated each time a user creates an account or changes their password.

Short Salt

If the salt is too short, an attacker can build a lookup table for every possible salt. For example, if the salt is only three ASCII characters, there are only $95 \times 95 \times 95 = 857,375$ possible salts. That may seem like a lot, but if each lookup table contains only 1MB of the most common passwords, collectively they will be only 837GB, which is not a lot considering 1000GB hard drives can be bought for under \$100 today. For the same reason, the username shouldn't be used as a salt. Usernames may be unique to a single service, but they are predictable and often reused for accounts on other services. An attacker can build lookup tables for common usernames and use them to crack username-salted hashes. To make it impossible for an attacker to create a lookup table for every possible salt, the salt must be long. A good rule of thumb is to use a salt that is the same size as the output of the hash function. For example, the output of SHA256 is 256 bits (32 bytes), so the salt should be at least 32 random bytes.

TABLE 11.1 Cryptographically Secure Pseudorandom Number Generator (CSPRNG).

Platform	CSPRNG
PHP	<i>mcrypt_create_iv, openssl_random_pseudo_bytes</i>
Java	<i>java.security.SecureRandom</i>
Dot NET (C#, VB)	<i>System.Security.Cryptography</i>
Ruby	<i>SecureRandom</i>
Python	<i>os.urandom</i>
Perl	<i>Math::Random::Secure</i>
C/C++ (Windows API)	<i>CryptGenRandom</i>
Any language on GNU/Linux or Unix	<i>Read from /dev/random or /dev/urandom</i>

Security Challenges in Hash Functions

This section explains some of the common password hashing misconception—wacky combinations of hash algorithms. It's easy to get carried away and try to

combine different hash functions, hoping that the result will be more secure. In practice, though, there is very little benefit to doing it. All it does is create interoperability problems, and it can sometimes even make the hashes less secure. Using a standard hashing function or using multiple hash functions makes the process of computing the hash slower, so cracking is slower.

Hash Collision

Hash functions map arbitrary amounts of data to fixed-length strings. There must be some inputs that hash into the same string. Cryptographic hash functions are designed to make these collisions incredibly difficult to find. From time to time, cryptographers find “attacks” on hash functions that make finding collisions easier. Collision attacks are a sign that it may be more likely for a string other than the user’s password to have the same hash. However, finding collisions in even a weak hash function like MD5 requires a lot of dedicated computing power, so it is very unlikely that these collisions will happen “by accident” in practice. A password hashed using MD5 and salt is, for all practical purposes, just as secure as if it were hashed with SHA256 and salt. Nevertheless, it is a good idea to use a more secure hash function like SHA256, SHA512, RipeMD, or WHIRLPOOL if possible.

Hashing with Salt

In the previous section, it is clear how malicious hackers can crack plain hashes very quickly using lookup tables and rainbow tables. Randomizing the hashing using salt is one of the solutions for this problem. But how do we generate the salt, and how do we apply it to the password?

The salt should be generated using a Cryptographically Secure Pseudorandom Number Generator (CSPRNG). CSPRNGs are very different from ordinary pseudorandom number generators like the “C” language’s `rand()` function. As the name suggests, CSPRNGs are designed to be cryptographically secure, meaning they provide a high level of randomness and are completely unpredictable. We don’t want our salts to be predictable, so we must use a CSPRNG. Table 11.1 lists some CSPRNGs that exist for some popular programming platforms.

The salt needs to be unique per-user per-password. Every time a user creates an account or changes their password, the password should be hashed using a new random salt. Never reuse a salt. The salt also needs to be long, so that there are many possible salts. As a rule of thumb, select the salt as at

least as long as the hash function's output. The salt should be stored in the user account table alongside the hash.

To Store a Password

1. Generate a long random salt using a CSPRNG.
2. Prepend the salt to the password and hash it with a standard cryptographic hash function such as SHA256.
3. Save both the salt and the hash in the user's database record.

To Validate a Password

1. Retrieve the user's salt and hash from the database.
2. Prepend the salt to the given password and hash it using the same hash function.
3. Compare the hash of the given password with the hash from the database. If they match, the password is correct. Otherwise, the password is incorrect.

11.3.3 Making Password Cracking Harder

Salt ensures that attackers can't use specialized attacks like lookup tables and rainbow tables to crack large collections of hashes quickly, but it doesn't prevent them from running dictionary or brute-force attacks on each hash individually. High-performing computing systems and custom hardware can compute billions of hashes per second, so these attacks are still very effective. To make these attacks less effective, we can use a technique known as *key stretching*.

The idea is to make the hash function very slowly, so that even with a faster computer or custom hardware, dictionary and brute-force attacks are too slow to be worthwhile. The goal is to make the hash function slow enough to obstruct attacks, but still fast enough to not cause a noticeable delay for the user. Key stretching is implemented using a special type of CPU-intensive hash function. These algorithms take a security factor or iteration count as an argument. This value determines how slow the hash function will be. For desktop software or smartphone apps, the best way to choose this parameter is to run a short benchmark on the device to find the value that makes the hash take about half a second. This way, the applications

running in the computers can be as secure as possible without affecting the user experience.

One-Time Password (OTP)

One of the major concerns of a fixed password scheme is eavesdropping and subsequent relay of the password. A common solution for this is the use of a one-time password. In OTP the password is used only once. Such schemes are safe from passive adversaries who eavesdrop and later attempt a masquerade attack. The biggest advantage addressed by OTP is that in contrast to static passwords, they are not vulnerable to replay attacks.

Types of one-time passwords:

1. Shared list of OTPs
2. Sequentially updated OTPs
3. OTP sequences based on a one way-function

Shared list of OTPs: In this method the user and the system use a set of secret passwords ($t_1, t_2 \dots t_n$). This password list is pre-shared between the users. Passwords in the list are valid for a single authentication. It is a non-cryptographic technique that involves the use of a challenge-response table, whereby the users and the system share a table of matching challenge-response pairs. To accept the password, each pair has to be valid at most once.

Sequentially updated OTPs: In this authentication process, using password i , the user creates and transmits to the system a new password (password $i+1$) encrypted under a key derived from password i . This method becomes difficult if communication failures occur.

OTP sequences based on a one way-function: This method is more efficient than sequentially updated OTPs. It is viewed as a challenge-response protocol, where the challenge is implicitly defined by the current position within the password sequence.

11.4 CHALLENGE-RESPONSE IDENTIFICATION

To secure against passive eavesdropping, some techniques known as challenge-response protocols are developed. The idea of the challenge-response protocol is that one entity (the plaintiff) “proves” its identity to another entity (verifier) by demonstrating knowledge of a secret known to

be associated with the entity, without revealing the secret itself to the verifier during the protocol. To initiate a challenge-response protocol, an entity A sends a message containing A's identity to entity B. Then B sends a random number called a challenge. A uses the challenge and its password to perform some computation and sends the result, called a response to B. Then B uses A's stored password to perform the same computation and verify the response. Since B chooses a different challenge for every run of the protocol, an adversary can't simply eavesdrop on recorded messages and resend them at a later time (a replay attack) to impersonate an entity. If the communication time is monitored, the response from one execution of the identification protocol should not provide an adversary with useful information for a subsequent identification, as subsequent challenges will differ.

Time Variant Parameters

To counteract replay and interleaving attacks, time variant parameters may be used in identification protocols. This provides a uniqueness or timeliness guarantee and to prevent certain chosen text attacks. The different time-variant parameters which serve to distinguish one protocol instance from another are sometime called *nonce*, unique numbers, or non-repeating values. A nonce is a value used no more than once for the same purpose. It typically serves to prevent (undetectable) replay. Nonce is generally referred to as a "random" number in a challenge-response protocol, but the required randomness properties vary. Three main types of time-variant parameters are:

1. Random numbers
2. Sequence numbers
3. Time stamps

Random Numbers: In a challenge-response mechanism a random number provides uniqueness and timeless assurance prevents certain replay and interleaving attacks. The random number uses for the identification and authentication process are pseudorandom numbers which are unpredictable to an adversary. In a challenge-response protocol, a random number is used as follows.

One entity includes a random number in an outgoing message. An incoming message subsequently received whose construction required knowledge of this nonce and to which this nonce is inseparably bound is then deemed to be fresh, based on the reasoning that the random number links the two messages. The non-tampering binding is required to prevent appending a

nonce to an old message. The random numbers used in this manner serve to fix a relative point in time for the parties involved analogous to a standard time clock. The maximum allowed time between protocol messages is typically constrained by time-out period, enforced using local, independent countdown timers.

Sequence Numbers: A sequence number is identical to that of a serial number. It serves as a unique identification for a message. A sequence number of a message can be normally used to detect message replay. The sequence number is specific to a particular pair of entities and must explicitly or implicitly be associated with the both originator and recipient of a message. A separate sequence number is necessary for both entities A and B. They follow a predefined policy for message numbering. A message is accepted only if the sequence number therein has not been used previously and satisfies the agreed upon policy. The simplest policy is that a sequence number that starts at zero is incremented sequentially and each successive message has a number one greater than the previous one received. The key limitation of sequence numbers is the use of a sequence number may add an additional overhead to the message.

Time stamp: A time stamp is another approach used as a challenge-response identification protocol. It may be used to provide timeliness and uniqueness guarantees to detect message replay. It may also be used to implement time-limited access privilege and to detect forced delay. The party A originating the message generates the time stamp from its local machine. This is cryptographically attached to the message to be sent to B. Upon receiving the time-stamped message, B also obtains the current time stamp from its own machine. It then finds the difference between the two time stamps. The received message is accepted under the following conditions:

1. If the time stamp difference is within the acceptance time interval (10msec or 20msec.)
2. The receiving machine verifies the time stamp received from each source entity within the acceptable time interval.

If no message with an identical time stamp has been previously received from the same originator, then the message is considered as valid. The security of time stamp-based verification depends on the use of a common time reference between the two machines. A proper synchronization is necessary to counter clock drift and must be appropriate to accommodate the accepted time interval used.

11.4.1 Challenge-Response by Symmetric Key Cartography

In this method, two parties use a shared symmetric key for challenge-response. The symmetric key technique often involves the use of a trusted online server to which each party shares a key.

A challenge-response technique using secret key cryptography can be described as follows:

Let n denote a random number.

$Q(K)$ – a secret key cryptographic function parameterized by a shared key between the users A and B .

m - an optional message used to prevent a replay attack.

The authentication of A starts by having the verifier B send a random number n . Now A computes $Q(K, (n, m))$ and sends it back to B . The challenge-response process is described as follows:

$$B \rightarrow A : n \quad A \rightarrow B : \phi(K, (n, m))$$

The verifier decrypts the received random number and checks whether it is the random number that was provided before. The previous process may include a one-way function to provide a more efficient challenge-response mechanism. The steps for the mutual authentication include the following three messages:

$$B \rightarrow A : n \quad A \rightarrow B : (n^1, \phi(K, h[n, n^1, m]))$$

$$A \rightarrow B : (\phi(K, h[n, n^1, m]))$$

In mutual authentication:

- n – random number generated by B
- n^1 – random number generated by A
- m and m^1 – are optional messages
- h – is the one-way function

The well-known Kerberos protocol and the Needham-Schroeder shared key protocol provide entity authentication based on symmetric encryption and involve the use of an online trusted third party.

11.4.2 Challenge-Response by Public Key Technique

The challenge response mechanism using the public key technique is based on a secret/public key pair. It uses both a public and private key. For user authentication the public key technique can be used so that the plaintiff (claimant) demonstrates knowledge of its private key. This is done in the following two ways:

1. The claimant decrypts a challenge which the verifier has encrypted using the claimant's public key.
2. The claimant digitally signs the challenge and the verifier checks the signature.

By preventing the use of the key pair for another purpose, this mechanism can secure better, because combined usage may compromise security. Also the use of the public key mechanism should not be susceptible to chosen ciphertext attacks. An adversary can attempt to extract information by stealing the role of the verifier and choosing strategic rather than random challenges.

Challenge-Response Based on Public Key Decryption

The challenge-response based on a public key scheme operates as follows:

The public key encryption algorithm (e.g., RSA) is denoted as ϕP_A and a one-way hash function as h .

1. $A \leftarrow B : h(n), B, \phi(n, B)A \leftarrow B : h(r), B, P_A(r, B)$
2. $A \leftarrow B : nA \leftarrow B : r$

B chooses a random number n , computes the witness $x = h(n)$ (x demonstrates knowledge of n without disclosing it), and computes the challenge $e = \phi(n, B)$ $e = P_A(r, B)$.

B sends (1) to A .

A decrypts e to recover n^1 and B^1 .

Computes $x^1 = h(n^1)$ and quits if $x^1 \neq x$ (also $n^1 \neq n$) or if $B^1 \neq B$.

Otherwise, A sends $n \# n^1$ to B .

B succeeds with (unilateral) entity authentication of A upon verifying the received n agrees with what was sent earlier. The user of the witness precludes chosen-text attacks.

11.4.3 Challenge–Response by Zero Knowledge Authentication

Zero-knowledge authentication is another challenge-response protocol, however it does not use the cryptographic method. The secret key and public key challenge-response mechanism might reveal part of the secret information covered by the user wishing to be authenticated by a verifier. A malicious verifier, for example, may be able to submit a specific challenge to obtain a

response capable of recovering part of such information. The zero-knowledge technique discussed here mitigates such limitations by allowing a user to prove knowledge of secret information of some interest of the verifier.

The general form of the zero-knowledge scheme is represented as follows. The basic version of this is the Fiat-Shamir algorithm. More efficient versions of this algorithm are now in use within various solutions. In this form of authentication, user A proves to B knowledge of a secret s in executions of a three-phase process.

Phase I: Secret Generation

- A trusted third party selects two large prime numbers p and q . He publishes $n = pq$ while keeping p and q secret.
- User A wishing to be authenticated by B selects a secret s relatively prime to n , $1 < s < e$ and computes a public key K by $K = s^2 \bmod e$.
Then A registers the public key with the trusted third party.

Phase II: Exchanging Message

User A generates a random number n , $0 < n < e$ and performs the following three actions:

$$A \rightarrow B : x = n^2 \bmod e$$

$$B \rightarrow A : \text{random Boolean number } b$$

$$A \rightarrow B : y = n \cdot s^b \bmod e$$

The previous actions are repeated t times ($t < e$). At the end of the t rounds:

- (i) A number of $3t$ messages have been exchanged
- (ii) User A has selected t random values $n_1 \dots n_t$
- (iii) A has computed t values $x_1 \dots x_t$ and determined (t numbers) $y_1 \dots y_5$, while B has selected t random Boolean values $b_1 \dots b_t$ and received all x_i and y_i for $i = 1 \dots t$,
such that $x_i = n_i^2 \bmod e$ and $y_i = n_i \cdot s^{b_i} \bmod e$

Phase III: Verification

User B accepts the proof if for all $i \leq t$ then $y_i^2 = x_i \cdot K^b \bmod e$ is satisfied that y is not 0. Both terms of the equality take the form $r_i^2 \cdot s_i^{2b_i}$ in case of success. By sending challenge b , the verifier aims to first check whether A is also to demonstrate (t times) that he has knowledge of the secret s , and to deny

actions performed by an adversary impersonating A , such as selecting any random number n and sending $x = \frac{n^2}{K}$ to the verifier B .

On receiving $b = 1$, for example, the adversary will only answer by sending n , which is enough to satisfy the previous equality.

However, this will not work for $b = 1$; the response $y = n$ (response for challenge $b = 0$) is independent of the secret s while the response $y = n.s$ mode (made when $b = 1$) provides no information about s because n is a random number.

11.4.4 Challenge-Response Based on One-Way Functions

A one-way function or non-reversible function is also considered as a challenge-response mechanism. This technique may be preferable in situations where encryption algorithms are otherwise unavailable or undesirable.

Handheld Passcode Generator

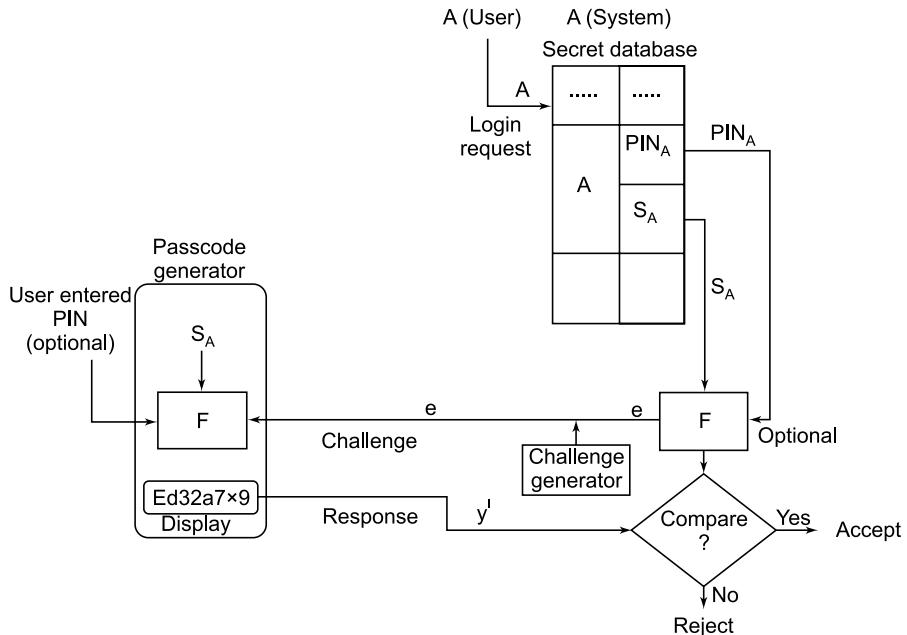


FIGURE 11.5 Passcode generator.

In challenge-response some kind of secure storage for long-term keying resources is needed. These smart devices are in the form of chip cards, which

can be used for both the key storage and response computation. Figure 11.5 shows the generation of a passcode using a simple passcode generator. The device contains a passcode generator, which uses user specific secret S_A , and a one-way function f . When a user inputs a challenge to the generator, it computes a response and shows it on the small display attached to it. A PIN code is used to secure the authentication. Now the user inputs the response from the display to the system. The verifier then checks this response using information stored inside the system. The response is computed from the challenge and secret key. The PIN can be verified locally or the response can depend on it.

Modified Needham-Schroeder Public Key Identification Protocol

This method provides mutual authentication and key transport of distinct keys K_1, K_2 from A to B and from B to A respectively. If the key establishment feature is not required, K_1 and K_2 may be omitted. With P_B denoting the public key encryption algorithm for B (e.g., RSA), the messages in the modified protocol for user authentications are then as follows:

1. $A \rightarrow B: P_B(r_1, A)$
2. $A \rightarrow B: P_A(r_1, r_2)$
3. $A \rightarrow B: r_2$

Challenge Response Based on Digital Signature

X.509 mechanisms based on digital signatures;

The ITU-T X.509 two-and three-way strong authentication protocols specify identification techniques based on digital signatures and, respectively, time stamps and random number challenges. Optionally, these protocols can be used for transporting keys in addition to entity authentication.

X.509 Strong Two-Way Protocol

Let r_A and r_B denote never reused numbers, and $cert_A$ and $cert_B$ denote certificates binding parties A and B to public keys which are suitable for both encryption and signature verification. $S_A(x)$ denotes the result of applying A 's signature private key to x (respectively to B). Let's assume that both parties have their public key pairs for signatures and encryption. Also, A must have B 's (authenticated) public key. With these markings and assumptions, the next messages are sent during authentication:

1. $A \rightarrow B: cert_A, D_A, S_A(D_A)$
 2. $A \leftarrow B: cert_B, D_B, S_B(D_B)$, Where, $D_A = (t_A, r_A, B)$, $D_B = (t_B, r_B, A, r_A)$
- A obtains a time stamp t_A indicating an expiry time, generates r_A , and sends a message to B .
 - After B has received the message, it verifies the authenticity of $cert_A$, extracts A 's signature public key and verifies A 's signature on the data block D_A , then B checks that the identifier in message (1) specifies itself as the intended recipient and that the time stamp is valid, and checks that r_A hasn't been relayed.
 - If these all checks succeed, B declares the authentication of A successful.
 - Now B obtains time stamp t_B , generates r_B , and sends a message to A (2).
 - When A has received the message, it carries out analogous actions as B did. If all checks succeed, A declares the authentication of B successful.
 - Now B obtains time stamp t_B , generates r_B , and sends a message to A (2).
 - When A has received the message, it carries out analogous actions as B did. If all checks succeeds, A declares the authentication of B successful.

X.509 Strong Three-Way Protocol

1. $A \rightarrow B: cert_A, D_A, S_A(D_A)$
2. $A \leftarrow B: cert_B, D_B, S_B(D_B)$
3. $A \rightarrow B: (r_B, B), SA(r_B, B)$

Three-way protocol differs from two-way protocol, as time stamps may be set to zero and it is not necessary to check them.

When receiving (2), A checks the received r_A matches that of message (1). When receiving (3), B verifies the signature matches the received plaintext, that plaintext identifier B is correct, and that the plaintext r_B received matches that of (2).

ISO/IEC 9798-3 mechanisms: This method includes three challenge-response identification mechanisms based on digital signatures. These mechanisms are equivalent to the ISO/IEC 9798-2 techniques which are based on symmetric keys. In this scheme r_A and t_A , respectively, denote a random number and time stamp generated by A . S_A denotes A 's signature mechanism (e.g., DSA). If this mechanism provides message recovery, some of the cleartext fields mentioned as follows are redundant and can be omitted. $cert_A$ denotes the public key certificate containing A 's signature public key.

Unilateral authentication with time stamps: $A \rightarrow B: cert_A, t_A, B, S_A(t_A, B)$

When B has received the message, it verifies that the time stamp is acceptable, the received identifier B is its own, and (using A 's public key extracted from $cert_A$ after verifying the latter) checks that the signature over these two fields is correct.

Unilateral authentication with random numbers: In this protocol reliance on time stamps is replaced by a random number, at the cost of an additional message.

1. $A \leftarrow B: r_B$
2. $A \rightarrow B: cert_A, r_A, B, S_A(r_A, r_B, B)$

B verifies that the cleartext identifier is its own, and using a valid signature public key from A (e.g., from $cert_A$), verifies that A 's signature is valid over the cleartext random number r_A , the same number r_B as sent in (1), and this identifier. The signed r_A explicitly prevents chosen-text attacks.

Mutual authentication with random numbers:

1. $A \leftarrow B: r_B$
2. $A \rightarrow B: cert_A, r_A, B, S_A(r_A, r_B, B)$
3. $A \leftarrow B: cert_B, A, S_B(r_B, r_A, A)$

Processing of (1) and (2) as previously; (3) is passed equivalent to (2).

11.5 BIOMETRIC SECURITY SYSTEMS

Humans have used body characteristics such as face, voice, gait, and so on for thousands of years to recognize each other. One emerging technology that has become more widespread and popular in industries is biometrics. The term biometrics comes from the Greek words *bios* (life) and *metrikos* (measure). To make a personal recognition, biometrics relies on who you are or what you do as opposed to what you have (such as a password or ID cards). Every individual biological measurement qualifies as biometrics. A human physiological or behavioral trait can serve as a biometric characteristic as long as it satisfies the following requirements:

- **Universality:** each person should have the characteristics.
- **Distinctiveness:** Any two people should be different in terms of characteristics.
- **Performance:** The characteristic should be sufficiently invariant (with respect to the matching criterion) over a period of time.
- **Collectability:** The characteristic should be quantitatively measurable.

However, for a particular biometric system, we must also consider issues of performance, acceptability, and circumvention. In other words, a practical system must meet accuracy, speed, and resource requirements and it must be harmless to the users, accepted by the intended users, and sufficiently robust against various fraudulent methods and attacks.

Application Fields of Biometric Technology

Biometric technology is applicable for both physical access control and logical access control.

Physical access control: Physical access control covers identity authentication processes which require users to provide physical characteristics. It is used in high security locations such as military research laboratories, airports, hospitals, and police stations. This application is confidential and important and is assigned with a high level of security. The physical access control system helps to eliminate the process of identifying long and complex passwords with different processes.

Logical access control: The logical access control process refers to access control over data files or computer programs. These contains personal or private information of many different users. This type of access control is used in the restricted access of military and government data with high security systems using biometric technology.

Biometric Systems

Biometric systems are pattern recognition systems that operate by acquiring biometric data from an individual extracting a feature set from the acquired data and comparing this feature set against the set in the database. Depending on the application context, a biometric system may operate either in verification mode or identification mode.

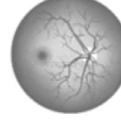
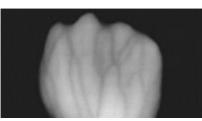
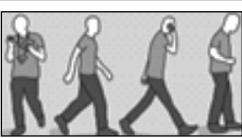
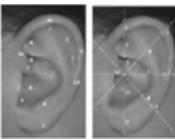
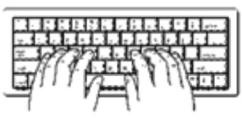
Biometric Types			Biometric Types		
1	Facial recognition detector		8	Hand Thermogram	 
2	Finger print reader		9	Hand geometry	
3	Voice recognition		10	Palm print	
4	Iris Scanner and Recognition		11	Retina	
5	Vein recognition		12	Signature	
6	DNA Biometric System		13	Gait	
7	Ear		14	Keystroke	
8	Face thermogram			—	—

FIGURE 11.6 Types of biometric techniques.

- **Verification mode:** In the verification mode, the biometrics system validates a person's identity by comparing the captured biometric data with his/her own biometric template stored in the system database. In such a system an individual who desires to be recognized claims an identity, usually via a Personal Identification Number (PIN), a user name, a smart card, and so on. The system conducts a one-to-one comparison to determine whether the claim is valid or not. The purpose of identity verification is to prevent multiple people from using the same identity.
- **Identification mode:** In identification mode, the system recognizes an individual by searching the templates of all the other users in the database for a match. Here the system conducts a one-to-many comparison to establish an individual's identity without the subject having to claim an identity. Identification is a critical component in negative recognition applications where the system establishes whether the person is who he (implicitly or explicitly) denies to be. The purpose of negative recognition is to prevent a single person from using multiple identities. Identification can also be used in positive recognition for convenience (because the user is not required to claim an identity). While the traditional methods of personal recognition such as passwords, PINs, keys, and tokens work for positive recognition, only biometrics can be used for negative recognition.

Biometric Solution

The different types of biometric techniques used to identify each human and to authenticate are shown in Figure 11.6.

Facial Recognition Detector: Face recognition is very popular and the easiest biometric security system. It is more widely used because it does not require any kind of physical contact between the users and the device. In a facial recognition system, cameras are working as input devices; they scan the user's face and match it to a database for verification. A biometric face recognition system will collect data from the user's face and store them in a database for future use. It will measure the overall structure, shape, and proportion of features on the user's face such as distance between eyes, dimension of the nose, mouth, jaws, size of the eyes, ears, and other expressions. Facial expressions such as smiling, crying, and wrinkles on the face are also considered as important factors to change during user's facial recognition process.

Face and hand vein infrared thermogram: In face thermograms, the pattern of the heat radiated by the human body of an individual is

considered as a characteristic. This can be captured by an infrared camera in an unobtrusive way, similar to that of a regular photograph. The technology could be used for convert recognition. A thermogram system also does not require any kind of physical contact between the user and the device, but image acquisition is challenging in an uncontrolled, where heat emanating surfaces (e.g., other heating bodies like room heaters, electric lamps, or vehicle exhaust pipes) are present in the vicinity of the body. A related technology using near infrared imaging is used to scan the back of a clenched fist to determine hand vein structure.

Hand and finger geometry: Hand geometry recognition systems are based on a number of measurements taken from the human hand, including its shape, size of palm, and length and width of the fingers. The hand geometry technique is very simple, relatively easy to use, and inexpensive. Similar to the hands, fingerprints are made of a number of ridges and valleys on the surface of the finger that are unique to each human. The ridges are the upper skin layer segments of the finger and valleys are the lower segments. The ridges form two minutiae points: ridge endings, where the ridges end, and ridge bifurcation, where the ridges split in two. Fingerprints are captured for the identification of users using new tools and techniques such as optical and ultrasound. The two algorithms which are used to recognize fingerprints are minutiae matching and pattern matching. Minutiae matching will compare the details of the extract minutiae to identify the difference between one user's fingerprint as compared to others; when the user registers with the system, they will record images of minutiae location and direction on the finger surface. When using the fingerprint recognition system to verify their identification, a minutiae image is brought out and compared with the one which was provided at the time of access. Pattern matching will compare all the surfaces of the finger instead of one particular point. It will concentrate more in thickness, curvature, and density of the finger's surface. The image of the finger surface for this method will contain the area around a minutiae point, areas with low curvature radius, or areas with an unusual combination of ridges.

Benefits of Fingerprint Recognition Systems

- This system is easy to use and install.
- It requires low cost equipment which generally has low power consumption.

Some Important Constraints of Hand and Finger Geometry Recognition Systems are

- The physical size of a hand geometry based system is large and it cannot be embedded in certain devices like laptops.
- There are verification systems available that are based on measurements of only a few fingers (typically index and middle) instead of the entire hand. These devices are smaller than those used for hand geometry but still much larger than those used in some other biometrics (e.g., finger-print, face, voice).

Iris: The human iris is a thin annular region of the eye bound by the pupil and the sclera on either side. It also controls the amount of light which is allowed through to the retina in order to protect the retina. The complex iris structure carries very distinctive information useful for personal recognition. Iris color is also a variable different for each person depending on their genes. The iris also has its own patterns which vary from person to person, which will create uniqueness for each individual.

Iris recognition systems will scan the iris in different ways. It will analyze the different characteristics of the iris after recording data from each individual. This information is stored in the database for future use in comparing it every time a user wants to access the system. Iris recognition security systems are considered as one of the most accurate security systems in use. It is extremely difficult to surgically tamper with the texture of the iris. During the verification process, if the user is wearing accessories such as glasses and contact lenses, the system will work as normal because it does not change any characteristics of the user's iris. The accuracy and speed of currently deployed iris based recognition systems is promising and point to the feasibility of large scale identification systems based on iris information.

Gait: Gait is a particular way one walks and is a complex spatio-temporal biometric. Gait is not supposed to be very distinctive but sufficiently discriminatory to allow verification in some low security applications. Gait is a behavioral biometric and may not remain invariant, especially over a long period of time, due to fluctuation in body weight, major injuries involving joints or brain, or due to inebriety. Acquisition of gait is similar to acquiring a facial picture and hence may be an acceptable biometric.

Vein recognition: A vein recognition system is one of the most recent methods of biometric technology. Veins are blood vessels that carry blood to the heart. An individual's veins have unique physical and behavioral traits. This is considered as a unique biometric characteristic to identify users. Vein recognition systems mainly focus on the veins in the user's hands. The

recognition system will capture images of the vein patterns inside of a user's fingers by applying light transmission to each finger. The system has a higher level of security which can protect information or access control much better. The level of accuracy used in vein recognition systems is very impressive and reliable by the comparison of the record database to that of the current database. Further, it also has low cost on installation and equipment.

Keystroke: It is hypothesized that each person types on a keyboard in a characteristic way. This behavioral biometric is not expected to be unique to each individual, but it offers sufficient discriminatory information to permit identify verification. Keystroke dynamics is a behavioral biometric; for some individuals, one may expect to observe large variations in typical typing patterns. Further, the keystrokes of a person using a system could be monitored unobtrusively as that person is keying in information.

DNA – Biometric system: The DNA pattern of an individual has unique biometric characteristics, because each person's DNA is unique. Each person's DNA contains some trait from his/her parents. The cells in the human body contain a copy of this DNA. DNA profiling will divide the amount of VNTR (variable number of tandem repeat) which repeats at a number of distinctive loci. These amount of VNTR will make up an individual's DNA profile. The DNA biometric recognition system goes through several complex steps:

- It needs to collect the DNA from a physical sample of each user such as blood, saliva, hair, semen, tissue, and others.
- Break down the sample into small fragments which contain VNTR.
- The size of each DNA fragment will be measured and sorted before it is compared to different samples.

DNA biometric technology is highly unique, and the chance of two individuals having the exact same DNA profile is impossible. The technology has not become popular due to the expensive equipment needed to break down the DNA successfully and analyze the unique features of the DNA and create a DNA profile.

Retinal Scan: Retinal vasculature represents the characteristics of an individual in each eye. It is proved that retinal scan is one of the most secure biometrics, since it is not easy to change or replicate. The data acquisition process requires a person to peer into an eyepiece and focus on a specific spot in the visual field, so that a predetermined part of the retinal vasculature can be imaged. In retinal scanning image acquisition (or data acquisition) is a tedious process. This involves cooperation of the subject, entails

contact with the eyepiece, and requires a conscious effort on the part of the user. All these factors adversely affect the public acceptability of retinal biometrics. Other factors such as hypertension deter public acceptance of retinal scan-based biometrics.

Signature: The way a person signs represents him. Signatures are a behavioral biometric. It requires contact with the writing instrument and an effort on the part of the user. The signature also changes over a period of time and is influenced by physical and emotional conditions of the signatories.

Voice Recognition: Voice is a combination of physical and behavioral biometrics. The features of an individual voice are based on the shape and size of the vocal tract, mouth, nasal cavity, and lips that are used in the synthesis of the sound. These physiological characteristics of the human voice are invariant for any individual, but the behavioral part of the voice of a person changes over time due to age, emotional state, and medical condition. The behavioral component of the voice is also known as voice accent. By combining both these factors, it is almost impossible to imitate another person's voice exactly. Taking advantage of these characteristics, biometric technology developed voice recognition systems in order to verify each person's identification only by their voice. Voice recognition systems are easy to install and require a minimal amount of equipment.

The commonly used equipment includes microphones, telephones, and PC microphones. The accuracy of the system depends on the quality of the voice recorded and stored in the database and comparison process. To analyze the voice the users are asked to repeat a short passphrase or a sequence of numbers and/or sentences so that the system can analyze the user's voice more accurately. To prevent the risk of unauthorized access via recording devices, voice recognition systems will ask user to repeat random phrases which are provided by the system during the verification process.

Advantages and Disadvantages of Biometric Systems

Advantages

1. Biometric identification can provide extremely accurate, secured access to information; fingerprint, retinal, and iris scans produce absolutely unique data sets when done properly.
2. Using a biometric security system, each individual's identification will be the single most effective identification for that user.
3. The highly secure way of identification of a user makes users less likely to share access to highly sensitive data using this technology. This is

because each trait used during identification is a single property of the user.

4. Biometric security systems are hard or impossible to duplicate.
5. The identification of users using biometrics cannot be lost, stolen, or forgotten.

Disadvantages

Despite the many advantages of biometric security systems, they still have the following disadvantages:

1. Each biometric application method has weaknesses which can cause problems for its user (finger or hands lost by accidents).
2. Voice recognition systems may not be able to get correct voice samples during illness such as strep throat. The other factors that influence voice recognition systems are continuous aging of its user, and surrounding noise where voice recognition is used to identify its users.
3. Iris and retinal scanning users may find it very intrusive. The user may also have concerns for the safety of their eyes during the iris or retinal scan.
4. A large database system may be required to store identification data.
5. Some of the biometric systems require expensive equipment.

SUMMARY

- Authentication provides a means of reliable identification of an entity.
- In a network a computer can authenticate a human through: passwords that authenticate what the user *knows*; A smart card and physical key that authenticates what the user *has*; and a biometric device such as retinal scanners, fingerprint analyzers and voice recognition systems that authenticate *who* the user is.
- Entity authentication is defined as the process whereby one party is assured of the identity of a second party involved in a protocol and that the second has actually participated.
- Authentication technique may be divided into three main categories: something known; something possessed; something inherent.

- A password is a set of secret characters or words utilized to gain access to a computer, Webpage, network resource, or database.
- The three most common ways of guessing passwords are *dictionary attacks*, *brute-force attacks*, and *key logger attacks*.
- Password sniffing is a process of monitoring a network to gather information that may be useful in an attack.
- Social engineering is the art of manipulating people so they give up confidential information.
- Hash algorithms are one-way functions. They turn any amount of data into a fixed-length “fingerprint” that cannot be reversed.
- Salt makes the same password hash into a completely different string every time.
- In OTP, the password is used only once. Such schemes are safe from passive adversaries who eavesdrop and later attempt a masquerade attack.
- To secure against passive eavesdropping, some techniques known as challenge-response protocols are developed.
- A human physiological or behavioral trait can serve as a biometric characteristic as long as it satisfies requirements such as: *universality*, *distinctiveness*, *performance*, and *collectability*.

REVIEW QUESTIONS

1. What factors contribute to a biometric’s development?
2. A biometric is an identification technology widely used in physical access control, and forensic cryptography is the science of information security. People generally agree that biometric information needs to be protected with cryptography. Discuss the benefits of biometrics and cryptography.
3. Using an example distinguish between positive and negative identification.
4. Identify two reasons for biometric characteristic variation over time.
5. Discuss the measures that reflect the effectiveness of a biometric authentication system.
6. Explain about physiological biometric technologies.
7. With suitable examples and diagrams, explain DNA, signature, and handwriting technologies.

8. Describe the benefits of biometrics over traditional authentication systems.
 9. Write a note on DNA biometrics.
 10. Write notes on keyboard/keystroke dynamics.
 11. What is a salted password?
 12. What are zero-knowledge proof systems?
 13. Draw a simple sketch for shared key authentication.
 14. What is a dictionary attack?
 15. List any four problems with using password authentication.
 16. Discuss handprint biometrics.

MULTIPLE CHOICE QUESTIONS

CHAPTER 12

AUTHENTICATION AND KEY MANAGEMENT APPLICATIONS

12.1 INTRODUCTION

Message encryption and access control are the two important requirements for ensuring data confidentiality and authentication in data communication. Data encryption is used as a data confidentiality technique, whereas the management of cryptographic keys is a critical and challenging security management function, especially in large enterprise data centers. This function becomes more complex due to the distributed nature of IT resources, as well as the distributed nature of their control.

The Following are General Secured Key Management Requirements

1. Parties performing key management functions are properly authenticated and their authorizations to perform the key management functions for a given key are properly verified.
2. All key management commands and associated data are protected from spoofing; that is, source authentication is performed prior to executing a command.
3. All key management commands and associated data are protected from undetected, unauthorized modifications; that is, integrity protection is provided.
4. Secret and private keys are protected from unauthorized disclosure.

5. All keys and metadata are protected from spoofing; that is, source authentication is performed prior to accessing keys and metadata.
6. All keys and metadata are protected from undetected, unauthorized modifications; that is, integrity protection is provided.
7. When cryptography is used as a protection mechanism for any of the previous functions, the security strength of the cryptographic mechanism used is at least as strong as the security strength required for the keys being managed.

Using public key cryptography, the one with public key can encrypt the message and the other with private key can decrypt it. The same procedure being applied to each side, sender and receiver both could immediately engage in private and secure communication. Here the public key cryptography solves the problem of public key security, but not the problem of public key acquisition, recognition, distribution, redistribution, validation and, most importantly, key binding to an identifier and key authentication to a real-world entity. Communication can be verified neither for origin authentication nor for data integrity—communication can be private but not secure. For a tamperproof binding between the public key and some desired attribute, usually the entity name and entity real-world confirmed identity, a certification is needed.

ITU-T (International Telecommunication Union-Telecommunication Standardization) recommended X.509, which is a part of the X.500 series that defines the dictionary service. The dictionary is a server or distributed set of servers that maintains a database of information about users. The information includes mapping from username to network address as well as other attributes and information about the user. X.509 is the most well-known public key certificate. The standard was defined in 1988, modified in 1993 (version 2.0), and extended in 1995 (version 3.0).

X.509 describes two levels of authentication: *single authentication*, using a password as verification of claimed identity, and *strong authentication*, involving credentials from cryptographic techniques. The dictionary is implemented by Certificate Authority (CA), which is governed by Certificate Practice Statement (CPS). The CPS is internally defined by each CA within broad limits and lies outside the scope of X.509, even though X.509 refers several subjects to be defined in the CPS. The three main entities which can be outwardly recognized in an X.509 certificate procedure are Certificate Authority (CA), subscriber, and user. In a nutshell, a certificate is a statement of a third party, claiming that the owner of the private key

corresponding with the public key in the certificate holds the attributes in the certificate.

12.2 PUBLIC KEY DISTRIBUTION

Public key distribution is the problem of distributing to the parties, in a trustworthy manner, the public key of those with whom they wish to communicate. This problem must be pursued in a wide range of settings, each with its own requirements, constraints, and opportunities. The solutions developed for public key distribution must meet the following requirements:

1. **Flexibility as a degree of assurance:** Users in different environments will have different security requirements. When chatting with her mom Alice might not care much if the public key is authentic. When arranging a high value money transfer, she will care greatly. A one size fits all solution will be too burdensome in some cases yet too lax in others. Achieving a low assurance level must be easy, and achieving a high one must be possible.
2. **Flexibility as means of assurance:** Users in different environments will find different means of gaining assurance more natural. When users have frequent real world contacts and are not part of any overarching security domain, manual exchange of fingerprints is the most feasible approach. When users are part of the same security domain, a trusted authority for this domain should be able to introduce them to each other. When users belong to different domains, it should be possible to link these domains through some sort of delegations, and in this fashion assemble a large scale infrastructure.
3. **Simplicity:** If we expect security analysis to review and approve of the solution, programmers to implement it, administrate to deploy and manage it, and users to understand and use it, it must be incredibly simple.

Public key encryption has its own difficulties, in particular the problem of obtaining someone's true public key. Problems determining a shared key for symmetric key cryptography and securely obtaining the public key for public key cryptography can be solved using a trusted intermediary.

For symmetric key cryptography the trusted intermediary is called a *key distribution center* (KDC), which is a single trusted network entity with whom one has established a shared secret key. The KDC has to provide the

shared keys needed to communicate securely with all other network entities. For public key cryptography the trusted intermediary is called a *certificate authority* (CA). A certificate authority certifies that a public key belongs to a particular entity for a certified public key; if one can safely trust the CA that certified the key, then one can be sure about to whom the public key belongs. Once a public key is certified, then it can be distributed from just about anywhere, including a public key server or a personal Webpage.

12.2.1 Trusted Centers

As in the previous section, a certificate authority (CA) is a trusted entity that issues a digital certificate and a public-private key pair. The role of the CA is to certify a public key belong to a particular entity. The CA verifies that the owner of the certificate is who says he is. The CA is a trusted third party which is responsible for physically verifying the legitimacy of the identity of an individual or an organization before issuing a digital certificate.

12.2.2 Certificate Authority

Certificate Authorities or CAs issue digital certificates. Digital certificates are verifiable small data files that contain identity credentials to help Websites, users, and devices represent their authentic online identity. CAs play a critical role in how the Internet operates and how transparent trusted transactions can take place online. For example, an SSL certificate is a popular type of digital certificate that binds the ownership details of a Web server to cryptographic keys. These keys are used in the SSL/TLS protocol to activate a secure session between a browser and the Web server hosting the SSL certificate.

A general designation for any entity that controls authentication services and management of certificates is also called the issuer. The CA can be public (a bank that issues certificates to allow its clients to access their bank accounts); commercial (a service provider that sells certificates to other parties such as Verisign); or private (a company that issues certificates to allow its employees to perform job duties). It is upon the relying party to trust if the third party verified this correctly. The trusted third party called a Certificate Authority vouches for the correctness of the information in the certificate.

Subscriber: Is an entity that supplies to the CA the information that is to be included in the entity's own certificate signed by the CA. The CA copies the subscriber information to the certificate, but neither denies nor confirms it.

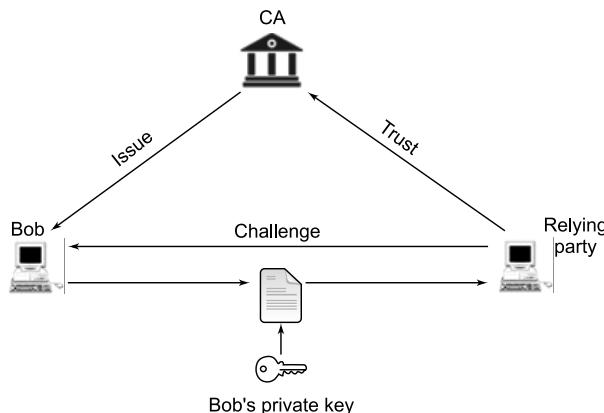


FIGURE 12.1 Bob holds a certificate issued by the CA and used to authenticate a relying party.

User: Any entity which relies upon the certificate issued by the CA in order to obtain information on the subscriber. Also called the verifier. Users may use any CA or any number of CAs, depending on their location and ease of access. The user should be central to the decision process in all steps.

In the Figure 12.1, a user (Bob) authenticates using his certificate, issued by CA, toward a relying party. In order to authenticate the relying party sends a random challenge, which Bob signs using his private key. The resulting signature is sent to the relying party, which can now verify that the signature indeed corresponds to the public key contained in the certificate. In addition the relying party must verify the certificate. Among other things, it checks that the certificate is not expired and that the certificate was created by a trusted CA. The relying party therefore fetches the public key of the issuing CA from his list of trusted CAs and verifies that the signature of the issuer is valid. If the signature is invalid or the public key was not found in the list of trusted CAs, the relying party does not trust the authentication.

12.2.3 Why Are Certificate Authorities a Vital Part of PKI?

As we already know, a PKI is a complex system for governing and managing digital certificates. It helps to facilitate encryption while also verifying the owners of the public keys themselves. The use of Certificate Authorities are so important for authentication. If the certificate of authority is removed from the PKI, you essentially have a large, unverified group of digital certificates, many of which are likely viable but some of which could also be used maliciously, given that there's no way to verify ownership of them. For a user,

this means that someone could essentially misrepresent ownership of a given key and then steal encrypted data or manipulate it. So as a result, Certificate Authorities are in place to help with authentication. Authentication proves the ownership of a message over a given certificate, and by extension that certificate's key. The CAs are trusted for a user. These infrastructures are robust in operations and are capable of verifying identities and issuing digital certificates properly.

12.2.4 How Does a Certificate Authority Work?

The process of actually authenticating and issuing a digital certificate issued by a certificate authority is explained in Figure 12.2. After the certificate is requested, depending on the level of validation required, the CA starts verifying the identity of the applicant. If it's simply a Domain Validation certificate, the CA just checks ownership over the domain and then, once this is satisfied, issues the certificate. For Organization Validation and Extended Validation, also known as business validation, the Certificate Authority will use business registration and credit reports to check the organization applying. Once it is complete, the certificate can then be issued and will contain critical details about the business itself. All of this is essential, especially for a PKI, as it allows the true owner of the keys being managed to be verified and makes the entire effort safer and more reliable.

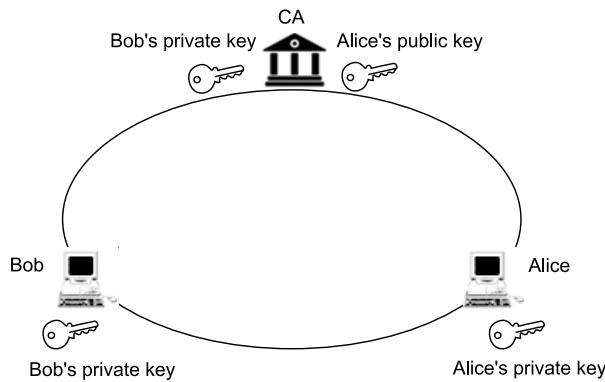


FIGURE 12.2 Certificate authority and PKI.

Digital Certificates: The design of making the public key available to anyone who requests it is an important solution, but it still has some limitations. The fact is an attacker can still make mischief with the public key. The

attacker might be able to decrypt digital signatures or even read passwords encrypted with the user's private key. It is safer to provide some kind of security system for ensuring who gets access to a public key.

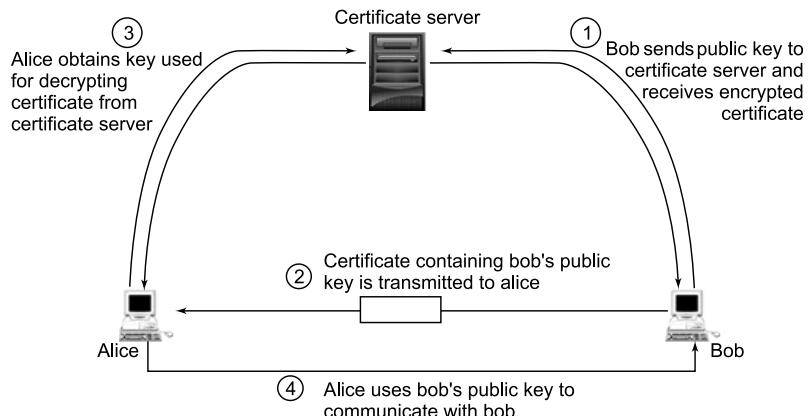


FIGURE 12.3 Authentication using digital certificates.

One of the solutions for this problem is what is called a *digital certificate*. A digital certificate is essentially an encrypted copy of the public key. Public keys are stored within digital certificates along with other relevant information (user information, expiration date, usage, who issued the certificate, etc.). The CA enters the information contained within the certificate when it is issued and this information cannot be changed. Since the certificate is digitally signed and all the information in it is intended to be publicly available, there is no need to prevent access to reading it, although you should prevent other users from corrupting, deleting, or replacing it. The certificate process is shown in Figure 12.3. This process requires a third-party certificate server (certificate authority) that has a secure relationship with both the parties that want to communicate. Several companies provide certificate services for the Internet. One major certificate authority is VeriSign Corp. Some large organizations provide their own certificate services. The certificate process varies among the various vendors. The digital certificate process is designed to serve a community of users. This process is for a safe distribution of any keys necessary for communicating with the certificate server. This might seem like simply transferring the problem. However, the fact that the protected communication channel is limited to a single certificate server makes it much more feasible to impose the overhead of additional safeguards necessary for

ensuring a secure exchange. A rough schematic description of the process shown in Figure 12.3 is as follows:

1. User Bob sends a copy of his public key to the certificate server through a secure communication.
2. The certificate server encrypts Bob's public key (along with other user parameters) using a different key. This newly encrypted package is called the certificate. Included with the certificate is the digital signature of the certificate server.
3. The certificate server returns the certificate to Bob.
4. User Alice needs to obtain Bob's public key. Computer A asks Computer B for a copy of Bob's certificate.
5. Computer A obtains a copy of the key used to encrypt the certificate through a secure communication with the certificate server.
6. Computer A decrypts the certificate using the key obtained from the certificate server and extracts Bob's public key. Computer A also checks the digital signature of the certificate server (step 2) to ensure that the certificate is authentic.

The best known standard for the certification process is the X.509 standard, which is described in several RFCs. X.509 version 3 is described in RFC 2459.

12.2.5 X.509 Certificate Format

The X.509 public key infrastructure (PKI) standard identifies the requirements for robust public key certificates. A certificate is a signed data structure that binds a public key to a person, computer, or organization. Certificates are issued by certification authorities. All who are party to secure communications that make use of a public key rely on the CA to adequately verify the identities of the individuals, systems, or entities to which it issues certificates. The level of verification typically depends on the level of security required for the transaction. If the CA can suitably verify the identity of the requester, it signs (encrypts), encodes, and issues the certificate.

The X.509 certificate accommodates information related to key identifiers, key usage, certificate policy, alternate names and name attributes, certification path constraints, and enhancement for certificate revocation including

revocation reason and Certificate Revocation List (CRL) partitioning. A typical X.509 certificate has the following structure as shown in Figure 12.4.

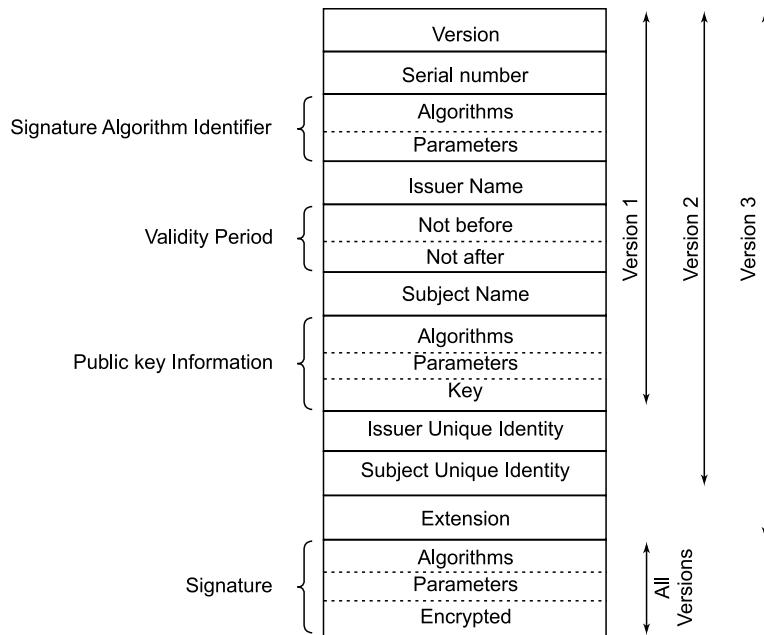


FIGURE 12.4 Structure of a typical X.509 certificate.

Version: Specifies the version number of the encoded certificate. Currently, the possible values of this field are 1, 2, or 3, but this may be expanded in the future.

Serial Number: Contains a positive, unique integer assigned by the certification authority (CA) to the certificate. Provides a unique identifier for each certificate that a CA issues.

Signature: The algorithm used to sign the certificate, together with any associated parameter; this has little utility due to a repeat of this information in the signature field.

Issuer Name: Provides a distinguished name for the CA that issued the certificate. The issuer name is commonly represented by using an X.500 or LDAP format.

Validity Period: Specifies the time interval during which the certificate is valid. Dates through the end of 2049 use the Coordinated Universal Time (Greenwich Mean Time) format (*yyyymmddhhmmssz*). Dates beginning with January 1, 2050 use the generalized time format (*yyyyyyymddhhmmssz*).

- **Valid From:** Provides the date and time when the certificate becomes valid.
- **Valid To:** Provides the date and time when the certificate is no longer considered valid. The date when an application or service evaluates the certificate must fall between the Valid From and Valid To (Not before and Not after) fields of the certificate for the certificate to be considered valid.

Subject Name: Provides the name of the computer, user, network device, or service that the CA issues the certificate to. The subject name is commonly represented by using an X.500 or Lightweight Directory Access Protocol (LDAP) format.

- **Subject:** Contains an X.500 distinguished name of the entity associated with the public key contained in the certificate.
- **Subject alternative name:** A subject can be presented in many different formats. For example, if the certificate must include a user's account name in the format of an LDAP distinguished name, e-mail name, and a user principal name (UPN), you can include the e-mail name or UPN in a certificate by adding a subject alternative name extension that includes these additional name formats.

Public Key Information: The public key of the subject plus an identifier of the algorithm for which the key is to be used together with any associated parameters.

Issuer Unique Identifier: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

Subject Unique Identifier: An optional bit string field to identify uniquely the subject in the event the X.500 name has been reused for different events.

Extension: A set of one or more extension fields; these were added in version 3. It provides additional functionality and features to the certificate. These extensions are optional and are not necessarily included in each certificate that the CA issues.

Signature: Covers all of the other fields of the certificate. It contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.

CRL distribution points (CDP): When a user, service, or computer presents a certificate, an application or service must determine whether the certificate has been revoked before its validity period has expired. The CDP

extension provides one or more URLs where the application or service can retrieve the certificate revocation list (CRL).

Authority Information Access (AIA): After an application or service validates a certificate, the certificate of the CA that issued the certificate, also referred to as the parent, must also be evaluated for revocation and validity. The AIA extension provides one or more URLs from where an application or service can retrieve the issuing CA certificate.

Enhanced Key Usage (EKU): This attribute includes an object identifier (OID) for each application or service a certificate can be used for. Each OID is a unique sequence of numbers from a worldwide registry.

Certificate policies: Describes what measures an organization takes to validate the identity of a certificate requestor before it issues a certificate. An OID is used to represent the validation process and can include a policy-qualified URL that fully describes the measures taken to validate the identity.

12.2.6 Certificate Authority Hierarchy

Digital certificates created by a Public Key Infrastructure (PKI) Certificate Authority are verified using a chain of trust. The trust anchor for the digital certificate is the *Root Certificate Authority* (Root CA), and any Certificate Authority which comes under root certificate authority is known as a *subordinate Certificate Authority*. Figure 12.5 shows the Certificate Authority Hierarchy.

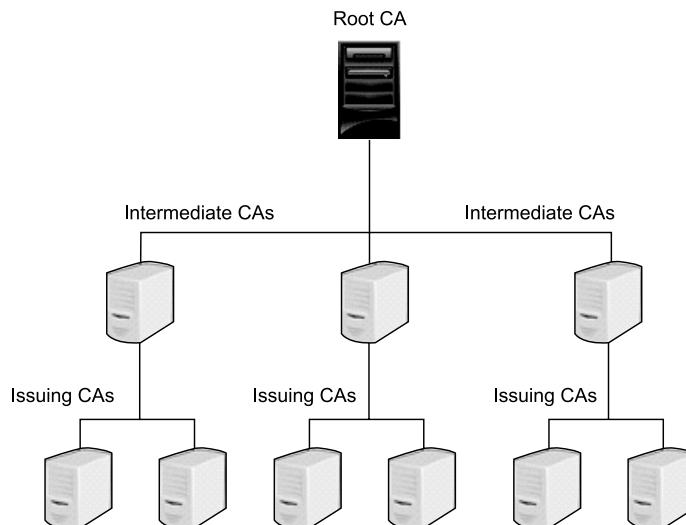


FIGURE 12.5 Certificate authority hierarchy.

Root CA: A Root CA is the topmost Certificate Authority in a Certificate Authority hierarchy. Each Certificate Authority hierarchy begins with the Root CA, and multiple CAs branch from this Root CA in a parent-child relationship. All child CAs must be certified by the corresponding parent CA back to the Root CA. The Root CA is kept in a secure area and it is usually a standalone offline CA (to make it the topmost secure Certificate Authority). The root CA provides certificates for intermediate CAs. The certificates can be revoked if they are compromised.

Intermediate CAs: An intermediate Certificate Authority is a CA that is subordinate to another CA (Root CA or another intermediate CA) and issues certificates to other CAs in the CA hierarchy. Intermediate CAs are usually standalone offline CAs like root CAs.

Issuing CAs: Issuing CAs are used to provide certificates to users, computers, and other services. There can be multiple issuing CAs, and one issuing CA can be used for generating computer certificates and another can be used for generating user certificates.

12.3 PUBLIC KEY INFRASTRUCTURE (PKI)

Public key infrastructure is a security architecture that has been introduced to provide an increased level of confidence for exchanging information over an unsecured channel. The term PKI on one hand may mean the methods, technologies, and techniques that together provide a secure infrastructure. On the other hand PKI means the use of a public and private key pair for authentication and proof of content. These features of PKI are delivered using a mathematical technique called *public key cryptography* that uses a pair of related cryptographic keys to verify the identity of the sender and to ensure privacy. PKI can also be used to deliver cryptographic keys between users securely and to facilitate other cryptographically delivered secret services. An individual who intends to communicate securely with others can distribute the public key but must keep the private key secret. Content encrypted by using one of the keys can be decrypted by using the other. For example, Bob wants to send a secure e-mail message to Alice. This can be accomplished in the following manner:

1. Both Bob and Alice have their own key pairs. They have kept their private keys securely to themselves and have sent their public keys directly to each other.
2. Bob uses Alice's public key to encrypt the message and sends it to her.
3. Alice uses her private key to decrypt the message.

This example highlights at least one obvious concern Bob must have about the public key he used to encrypt the message. That is, Bob cannot know with certainty that the key he used for encryption actually belonged to Alice. It is possible that another party monitoring the communication channel between Bob and Alice substituted a different key. To solve this problem the public key infrastructure (PKI) concept was proposed. This consists of software and hardware elements that a trusted third party can use to establish the integrity and ownership of a public key. The trusted party, called a certification authority, typically accomplishes this by issuing signed (encrypted) binary certificates that affirm the identity of the certificate subject and bind that identity to the public key contained in the certificate. The CA signs the certificate by using its private key. It issues the corresponding public key to all interested parties in a self-signed CA certificate. When a CA is used, the confidentiality and authentication process in the previous example can be explained as follows:

1. Assume that the CA has issued a signed digital certificate that contains its public key. The CA self-signs this certificate by using the private key that corresponds to the public key in the certificate.
2. Alice and Bob agree to use the CA to verify their identities.
3. Alice requests a public key certificate from the CA.
4. The CA verifies her identity, computes a hash of the content that will make up her certificate, signs the hash by using the private key that corresponds to the public key in the published CA certificate, creates a new certificate by concatenating the certificate content and the signed hash, and makes the new certificate publicly available.
5. Bob retrieves the certificate, decrypts the signed hash by using the public key of the CA, computes a new hash of the certificate content, and compares the two hashes. If the hashes match, the signature is verified and Bob can assume that the public key in the certificate does indeed belong to Alice.
6. Bob uses Alice's verified public key to encrypt a message to her.
7. Alice uses her private key to decrypt the message from Bob.

The certificate signing process enables Bob to verify that the public key was not tampered with or corrupted during transit. Before issuing a certificate, the CA hashes the contents, signs (encrypts) the hash by using its own private key, and includes the encrypted hash in the issued certificate. Bob

verifies the certificate contents by decrypting the hash with the CA public key, performing a separate hash of the certificate contents, and comparing the two hashes. If they match, Bob can be reasonably certain that the certificate and the public key it contains have not been modified.

Components of PKI

- **Certification Authority (CA):** It serves as the root of trust that authenticates the identity of individuals, computers, and other entities in the network.
- **Registration Authority (RA):** Is certified by a root CA to issue certificates for specific uses permitted by the root.
- **Certificate Database:** It saves the details of certificate requests and issued and revoked certificates and certificate requests on the CA or RA.
- **Certificate Store:** It saves issued certificates and pending or rejected certificate requests on the local computer.
- **Key Archival Server:** It saves encrypted private keys in a certificate database for disaster recovery purposes in case the Certificate Database is lost.

The Certificate Enrollment API enables you to submit certificate and key archival requests to certification and registration authorities and install the issued certificate on a local computer. It does not enable you to directly manipulate the certificate database or certificate store.

12.4 KERBEROS

Kerberos is a network authentication protocol developed as part of a project at MIT. Kerberos uses a trusted third party or calls a middle man sever for authentication. It is mainly based upon another protocol called Needham-Schroeder. Kerberos is designed to provide strong authentication for client-server applications by using secret key cryptography. It has the following characteristics:

1. It is highly secured; it never sends a password unless it is encrypted.
2. Only a single login is required per session. Credentials defined at login are then passed between resources without the need for additional logins.

3. The concept depends on a trusted third party, a key distribution center (KDC). The KDC is aware of all systems in the network and it is trusted by all of them.
4. It performs mutual authentication, where a client proves its identity to a server and a server proves its identity to the client.

Under Kerberos, a client (generally either a user or a service) sends a request for a ticket to the KDC. The KDC creates a ticket-granting-ticket (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT using its password. If the client successfully decrypts the TGT, it keeps the decrypted TGT, which indicates proof of the client identity.

12.4.1 Working of Kerberos Authentication

The following key entities are mainly considered in the Kerberos authentication process, as shown in Figure 12.6.

1. Generic Security Service Application Program Interface (GSS-API): Is an API for the program to access security services to be compatible with Kerberos.
2. Key Distribution Center (KDC): This is the server which provides tickets and temporary session keys to users and computers for communication.
3. REALM: A Kerberos network identified by a name, the KDC authenticates users only within its realm. Users and services in different realms would seem not to be able to authenticate each other.
4. Ticket Granting Server (TGS): Is the logical KDC component that is used by the Kerberos protocol as a trusted third party. A TGS validates the use of a ticket for a specified purpose such as network service access.
5. Ticket-Granting-Ticket (TGT): A special ticket which contains the session key for communication between the client machine and the central KDC server.
6. Authentication Server (AS): verifies client during login.

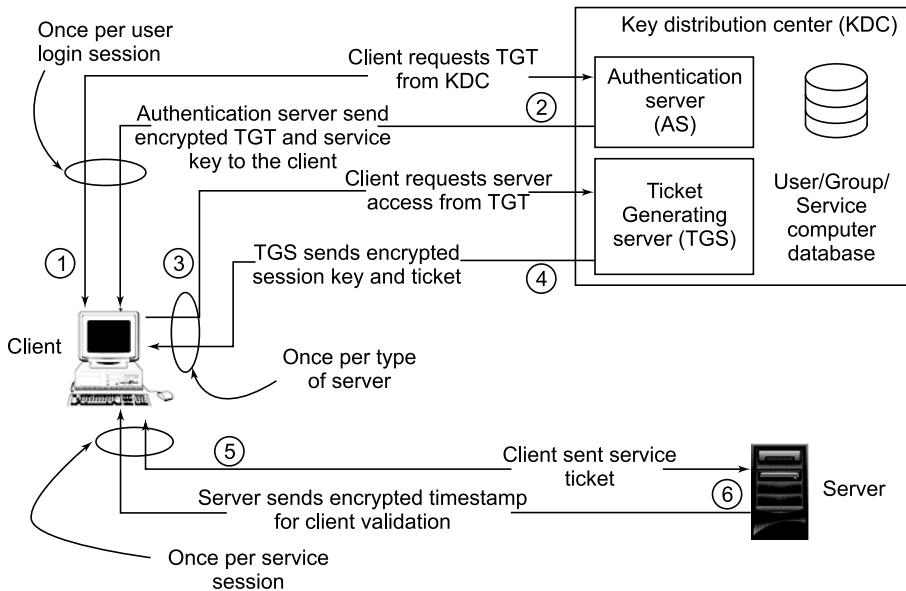


FIGURE 12.6 Kerberos authentication.

Figure 12.7 shows the sequence of events required for a client to gain access to a service using Kerberos authentication.

Step 1: Client requests TGT from KDC: The user logs on to the client computer and sends a Kerberos authentication service request to the KDC. The request contains the user name, the service information for which the TGT is requested, and a time stamp that is encrypted using the user's long-term key or password.

Step 2: Authentication Server (AS) sends encrypted TGT and session key: The AS verifies the user's access rights in the user database and creates a TGT and session key. The AS encrypts the result using a key derived from the user's password and sends a message back to the client. The client computer prompts the user for a password and uses the password to decrypt the incoming message. When the decryption is successful, the user will be able to use TGT to request the service ticket.

Step 3: Client requests server access from TGT: When the user wants to access to a service, the client application sends a request to the TGS containing the client's name, realm name, and time stamp. The user proves his identity by sending an authenticator encrypted with the session key received in step 2.

Step 4: TGS sends session key and ticket: The TGS decrypts the ticket and the authenticator verifies the request and creates a ticket for the

requested server. The ticket contains the client name and optionally the client IP address. It also contains the realm name and the ticket life-span. The TGS returns the ticket to the client. The returned message contains two copies of a server session key, one encrypted with a client password and one encrypted by the service password.

Step 5: Client sends service request: The client application now sends a service request to the server containing the ticket received in step 4 and an authenticator. The service authenticates the request by decrypting the session key. The server verifies that the ticket and authenticator match, and then grants access to the service.

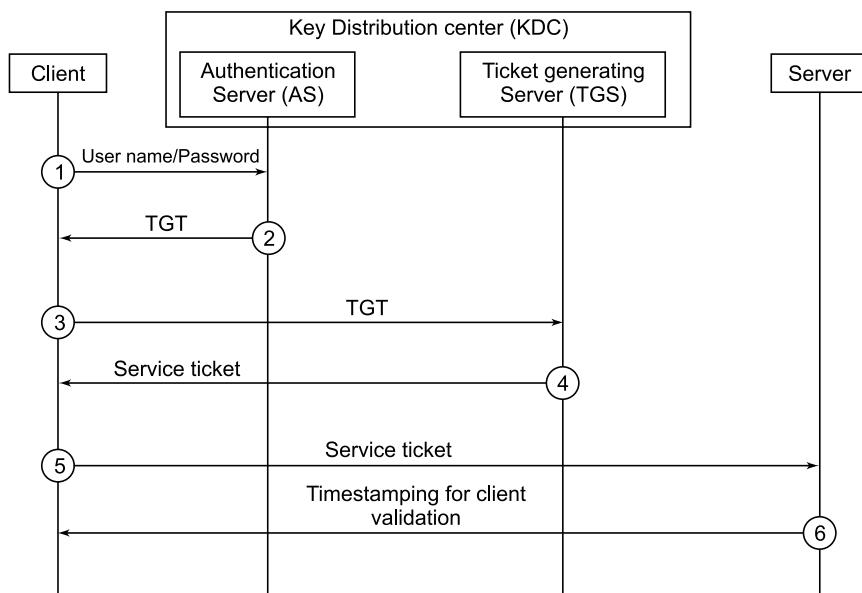


FIGURE 12.7 Sequence of events in Kerberos authentication.

Step 6: Server sends encrypted time stamp for client validation: The server decrypts the service ticket and extracts the service session key. Using the service session key, the server decrypts the authenticator message (time stamp) and evaluates it. If the authenticator passes the test, the server encrypts the authenticator (time stamp) using the service session key and then passes the authenticator back to the client. The client decrypts the time stamp, and if it is the same as the original, the service is genuine and the client proceeds with the connection.

Advantages of Kerberos

- 1. Secured password transmission:** A network that is connected to the Internet can no longer be assumed to be secure. Any attacker who gains access to the network can use a simple packet analyzer, such as packet sniffer, and can intercept the user name and passwords, compromising the user or account and the integrity of the entire security infrastructure. The primary design goal of Kerberos is to eliminate the transmission of unencrypted passwords across the network.
- 2. To prevent replay attack:** Kerberos uses time stamps as part of its protocol. Here both the computers need to be set to the same time and date. If the difference between a client's clock and server clock is less than the maximum time difference in the set policy by the administrator, any time stamp used in a session between the two computers will be considered authentic.
- 3. Client/Server Authentication:** Within Kerberos the client and server must each authenticate each other. Communication breaks down if each side is not able to authenticate the counterpart.
- 4. Durability and Reusability:** Authentication using the Kerberos protocol is durable and reusable. Once a user has authenticated using the protocol, the authentication is reusable for the lifetime of the ticket. In other words it is possible to remain authenticated through the Kerberos protocol without having to re-enter a user name and password across the network.

Disadvantages

1. Kerberos has only partial compatibility with the pluggable authentication module (PAM) system used by most Red Hat Enterprise Linux servers.
2. The primary goal of Kerberos is to prevent unencrypted passwords from being sent across a network. However, if anyone other than the proper user has access to the one host that issues tickets used for authentication by the KDC, this puts the entire Kerberos authentication system at risk.
3. Kerberos may not be compatible for all applications. For an application to use Kerberos, its source must be modified to make the appropriate calls into the Kerberos libraries. Applications modified in this way are considered to be Kerberized. For some applications this can be quite problematic due to the size of the application or its design.

4. Once Kerberos is used on the network, any unencrypted passwords transferred to a non-kerberized service are at risk.
5. Migrating user passwords from a standard UNIX password database to a Kerberos password database can be tedious, as there is no automated mechanism to perform this task.

12.5 PLUGGABLE AUTHENTICATION MODULE (PAM)

The Pluggable Authentication Module (PAM) is a generalized API for authentication related service. This allows a system administrator to add new PAM modules and to modify authentication policies by editing configuration files. PAM offers the following advantages:

1. A common authentication scheme that can be used with a wide variety of applications.
2. Significant flexibility and control over authentication for both system administrators and application developers.
3. A single document library which allows developers to write programs without having to create their own authentication scheme.

12.6 KEY DISTRIBUTION CENTER (KDC)

The KDC is a server that shares a unique symmetric secret key with each registered user. The KDC knows the secret key of each user, and each user can communicate securely with the KDC using this key.

Key Distribution: Suppose there are n entities and they all want to be able to communicate securely with each other. A security system would have to have a shared secret key for each pair of entities. This results in 2^n (which is $\mathcal{O}(n^2)$) shared secrets or keys. Managing such a large number of keys becomes tedious. Moreover, whenever a new entity joins a network, it has to contact all other entities in the network to establish a shared secret with them.

A more efficient alternative consists of providing every group member with a single key for securely communicating with a key distribution center (KDC). This key would be called a *master key*. When “A” wants to establish a secure communication link with “B,” A requests a session key from KDC

for communication with B. When two different parties want to communicate with each other, then they can request the KDC to generate a shared key for them. The basic steps present in a protocol that utilizes a KDC are as follows.

Suppose Bob and Alice want to communicate using symmetric key cryptography; how can they now agree on a secret key, given that they can communicate with each other over the network? The solution often adopted in practice is to use the trusted KDC.

If Alice and Bob are users of the KDC, they know only their individual keys K_{A-KDC} and K_{B-KDC} respectively for communicating securely with KDC as in Figure 12.8.

1. Alice using K_{A-KDC} wants to encrypt her message with the KDC, and she sends her message to the KDC saying she (A) wants to communicate with (B).

This message is denoted as: $K_{A-KDC}(A, B)$.

2. The KDC knowing K_{A-KDC} decrypts $K_{A-KDC}(A, B)$. The KDC then generates a random number R_1 . This is shared key value that Alice and Bob will use to perform symmetric encryption when they communicate with each other. This key is known as a *one-time session key*, because Alice and Bob will use this key for only this one session. The KDC now needs to inform Alice and Bob of the value of R_1 . The KDC thus sends back an encrypted message to Alice using K_{A-KDC} containing the following:

- R_1 – the one-time session key that Alice and Bob will use to communicate.
- A pair of values A and R_1 encrypted by the KDC using Bob's key K_{B-KDC} . This is represented as: $K_{B-KDC}(A, R_1)$.

The KDC provides not only the value of R_1 for her own use, but also an encrypted version of R_1 and Alice's name encrypted using Bob's key. The KDC puts these items in a message, encrypts them using Alice's shared key, and sends them to Alice. The message from the KDC to Alice is thus $K_{A-KDC}(R_1, K_{B-KDC}(A, R_1))$.

3. Alice receives this message from the KDC, decrypts it, and extracts R_1 from the message. She also extracts $K_{B-KDC}(A, R_1)$ using K_{B-KDC} and extracts A and R_1 . Bob now gets the one-time session key R_1 and the person with whom he is sharing this key.

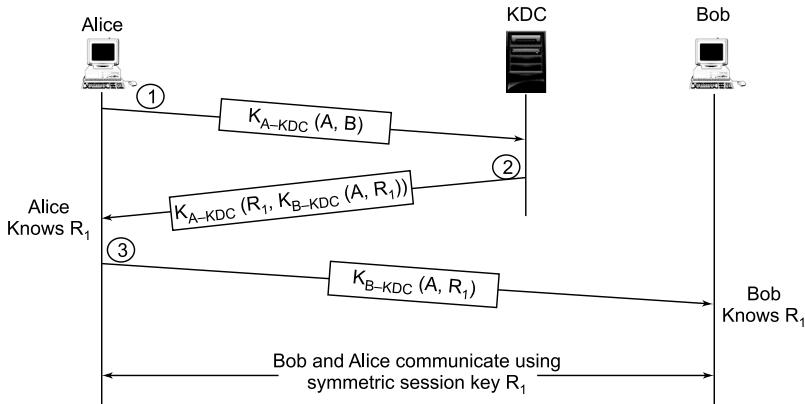


FIGURE 12.8 Setting up a one-time session key using a key distribution center.

Security Handshake Pitfalls

Security in message communication always includes an initial authentication handshake along with integrity protection and/or encryption of data. If Alice and Bob want to communicate, they need to know some information about themselves and about the other party. In practice the security protocols are designed and verified for all the limitations one can imagine and fixes them.

During the handshake process there are various anomalies. These anomalies are known as *security handshake pitfalls*. The two approaches by which the handshake process are carried out are:

1. One-way authentication
2. Mutual authentication

One-way Authentication: One-way authentication is a single protocol exchange to authenticate one party to another. If there are two users, Alice and Bob, to be involved in the communication process, Bob authenticates Alice, but Alice does not authenticate Bob. Hence it is called *one-way authentication*. The one-way authentication can be implemented by login only, shared secret, and one-way public key methods as represented in Figure 12.9.

1. **Login only:** This method of authentication is very simple to implement and understand. The authentication in this method includes the following steps:
 - (i) Alice (A) sends her name and password across the network to Bob (B).

- (ii) Bob verifies the name and password as in Figure 12.9. If the name and password are valid, communication occurs, with no further authentication of the message or integrity check for secured communication.

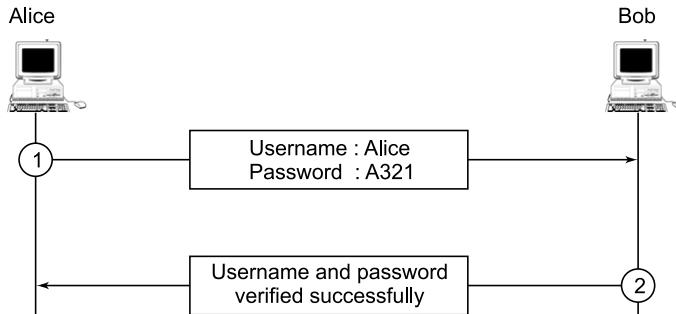


FIGURE 12.9 Login only authentication.

3. **Shared Secret:** In this method of authentication both Alice and Bob have agreed on a shared symmetric key K_{A-B} before starting the communication. In Figure 12.10 $K_{A-B}\{R\}$ means that a random challenge (R) (is different every time) is encrypted by using the shared secret key K_{A-B} . This could be done by using K_{A-B} as a secret key in some algorithm such as DES or IDEA. The K_{A-B} is used to encrypt R . These operations are represented in the following steps:

- (i) Alice sends her name and password to Bob.
- (ii) Bob now creates a random challenge R and sends it to Alice.
- (iii) Alice encrypts this random challenge R using the shared secret K_{A-B} or by hashing R and K_{A-B} , for instance by concatenating R and K_{A-B} and computing a message digest of the results. Alice then sends the encrypted R to Bob. At the receiving end Bob also encrypts the original random challenge (R) with the same shared secret K_{A-B} and compares it with the received encrypted R . If they are the same, Bob considers Alice to be authentic.
- (iv) The shared secret is a big improvement over the login only method. And eavesdropper Eve cannot impersonate Alice based on overhearing the shared secret key exchange. However, there are some limitations of this method listed as follows:

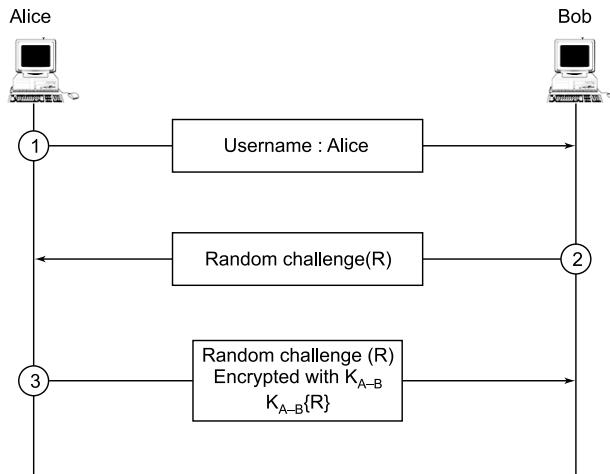


FIGURE 12.10 Shared secret \Rightarrow Bob authenticates Alice based on a shared secret K_{A-B} .

- a. The shared secret authentication method is not mutual. Bob authenticates Alice, but Alice does not authenticate Bob. If Eve can receive packets transmitted to Bob's network address and respond with Bob's network (or by some other means convince Alice that Eve's address is Bob's), then Alice's will be misled into assuming Eve is Bob. Eve does not need to know Alice secret in order to impersonate Bob—she just needs to send any old number R to Alice and ignore Alice's response.
- b. With the absence of the cryptographic function intruder Eve can hijack the conversation after the initial exchange, assuming she can generate packets with Alice's source address. It's also useful to Eve, but not absolutely essential, that she be able to receive packets transmitted to Alice's network layer address.
- c. An intruder could attempt an off-line password guessing attack (assuming K_{A-B} is derived from a password), knowing R and $K_{A-B}\{R\}$.
- d. Besides these drawbacks, if there are limited resources available for addressing security, an enhancement can be done as shown in Figure 12.11.
- e. Here Bob chooses a random challenge (R), encrypts it, and transmits the result to Alice. Alice then decrypts the received message, using the secret key K_{A-B} to get R, and sends R to Bob.

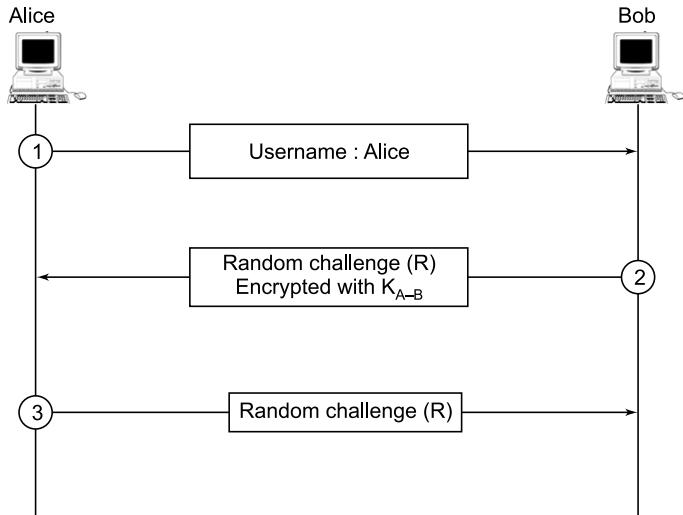


FIGURE 12.11 Shared secret enhancement: Bob authenticates Alice based on a shared secret key K_{A-B} .

This method requires reversible cryptography; that is, Alice has to be able to reverse what Bob has sent to retrieve R. Sometimes there is a performance advantage to being able to use one of the message digest functions rather than having to use high-end encryption algorithms like DES.

Yet there may be export issues involved in having code for encryption available, even if it is only used for authentication, whereas using a message digest function would be less likely to create an export problem. If R is a recognizable message, a 32 bit random number, for instance, then Eve can, without eavesdropping, attack by the password guessing technique and send the message to Bob to obtain $K_{A-B}\{R\}$. If Eve is eavesdropping however and sees both R and $K_{A-B}\{R\}$, she can perform a password guessing attack with either protocol.

If R is a recognizable quantity with a limited lifetime, such as a random number concatenated with a time stamp, Alice authenticates Bob on the grounds that only someone knowing K_{A-B} could generate $K_{A-B}\{R\}$. Another alternative for the protocol shown in Figure 12.12 is to shorten the handshake to a single message by having Alice use a time stamp instead of an R that Bob supplies. The basic requirement of this protocol is that both Alice and Bob have a reasonably synchronized clock. Alice encrypts the current time. Bob decrypts the result and makes sure the result is acceptable. The effect of this enhancement is described as follows:

1. The enhancement can be implemented very easily to a protocol designed for sending cleartext passwords. It does not add any additional messages. The cleartext password is replaced by the encrypted time stamp in the first message transmitted by Alice to Bob.
2. This protocol is more efficient. It can be added as a request-response protocol. In the request Alice merely adds the encrypted time stamp; Bob can authenticate the request and generate a reply.
3. An eavesdropper can use Alice's transmitted $K_{A-B}\{timestamp\}$.

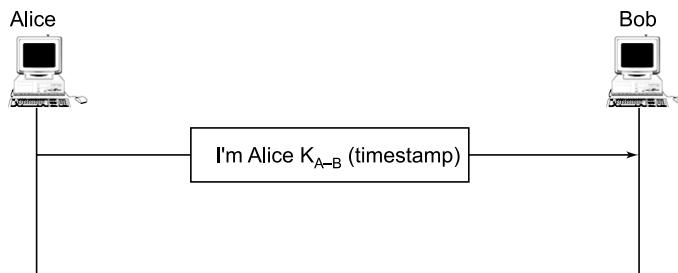


FIGURE 12.12 Shorten the handshake to a single message.

SUMMARY

- Message encryption and access control are the two important requirements for ensuring data confidentiality and authentication in data communication.
- Using public key cryptography, the one with public key can encrypt the message and the other with the private key can decrypt it.
- ITU-T (International Telecommunication Union-Telecommunication Standardization) recommends X.509, which is a part of the X.500 series that defines the dictionary service.
- X.509 describes two levels of authentication: *single authentication*, using a password as verification of claimed identity, and *strong authentication*, using credentials from cryptographic techniques.
- The use of Certificate Authorities are so important for authentication.
- A digital certificate is essentially an encrypted copy of the public key. Public keys are stored within digital certificates along with other relevant information.

- The X.509 public key infrastructure (PKI) standard identifies the requirements for robust public key certificates.
- Public key infrastructure is a security architecture that has been introduced to provide an increased level of confidence for exchanging information over an unsecured channel.
- Kerberos is designed to provide strong authentication for client-server application by using secret key cryptography.
- The Pluggable Authentication Module (PAM) is a generalized API for authentication related service.
- The KDC is a server that shares a unique symmetric secret key with each registered user.
- Security in message communication always includes an initial authentication handshake along with integrity protection and/or encryption of data.

REVIEW QUESTIONS

1. List out the requirements of Kerberos.
2. Does PKI use symmetric or asymmetric encryption? Explain your answer.
3. How are public key certificates generated in the X.509 authentication service? What are the forward and reverse certificate in X.509?
4. Illustrate the configuration of Kerberos.
5. Client machine C wants to communicate with server S. Explain how can it be achieved through the Kerberos procedure.
6. Write a note on replicated KDCs.
7. Describe Kerberos as a key distribution center and an authentication protocol.
8. Explain the Kerberos protocol and give a sketch of the simplified overview of the actions taken by the TGS.
9. What is a Ticket Granting Server?

MULTIPLE CHOICE QUESTIONS

1. _____ is a well-known public key certificate process.
(a) SHA 1 (b) X.509 (c) SSL (d) LDAP
2. For symmetric key cryptography the trusted intermediary is:
(a) Certificate Authority (b) Key Distribution Center
(c) Pluggable Authentication Module (d) None of the above
3. The trusted anchor for the digital certificate is:
(a) certificate Authority (b) authentication Server
(c) root Certificate Authority (d) all of the above
4. _____ provides a certificate to the user.
(a) Issuing CA (b) Intermediate CA
(c) Root Certificate Authority (d) None of the above
5. In Kerberos, _____ verifies the client during login.
(a) ticket Granting Server (b) certificate Authority
(c) authentication Server (c) none of the above
6. For each _____ the Kerberos Key Distribution Center (KDC) maintains a database of the realm's principal and the principal's associated "secret keys."
(a) key (b) realm
(c) document (d) none of the above
7. For a client-server authentication, the client requests from the KDC a _____ for access to a specific asset.
(a) ticket (b) local (c) token (d) user
8. A special ticket which contains the session key for communication between the client machine and the KDC server is:
(a) AS (b) TGS
(c) TGT (d) none of the above

- 9.** Which of the following factors must be considered when implementing Kerberos authentication?

 - (a) Kerberos can be susceptible to the ticket issued for authentication by the KDC for anyone other than the proper user having access to any one of the hosts.
 - (b) Kerberos tickets can be spoofed using replay attacks to network resources.
 - (c) Kerberos requires a centrally managed database of all user and resource passwords.
 - (d) Kerberos uses clear text passwords.
- 10.** Which of the following must be deployed for Kerberos to function correctly?

 - (a) Dynamic IP (Internet Protocol) routing protocols for routers and servers.
 - (b) Separate network segments for the realms.
 - (c) Token authentication devices.
 - (d) Time synchronization services for clients and servers.

PART

3

NETWORK SECURITY APPLICATIONS

CHAPTER 13

NETWORK SECURITY AND PROTOCOLS

13.1 INTRODUCTION

A computer network consists of two or more computing devices that are connected in order to share the resources in a network and information stored. The most basic computer network can expand and become more usable when additional computers join and add their resources to those being shared. As more and more computers are connected to a network and share their resources, the network becomes a more powerful tool, because users using a network with more information and more capability are able to accomplish more through those added computers. Generally computer networks are classified as wired and wireless networks. *Wired* as a term refers to any physical medium consisting of cables. The cables can be copper wire, twisted pair, or fiber optics. A wired network is used to carry different forms of electrical signals from one end to the other. *Wireless* as a term refers to a medium made of electromagnetic waves or infrared waves. All the wireless devices will have antenna and sensors. This network does not use wires for data or voice communication; it uses radio frequency waves as mentioned previously. The common examples of wireless networks are wireless LAN (WiFi), Bluetooth, Zigbee, ZWave, IrDA WiMAX, LTE, GSM, and CDMA.

13.2 INTERNET

The Internet is the world's largest network of networks. This interconnects millions of computers, providing a global communication, storage, and computation infrastructure. Moreover, the Internet is currently being integrated

with mobile and wireless technology, ushering in an impressive array of new applications. It typically provides two services to its distributed applications; a connection-oriented reliable service and a connectionless unreliable service. The connection-oriented reliable service guarantees the data transmitted from a sender to a receiver will eventually be delivered to the receiver in order and in its entirety. The connectionless unreliable service does not make any guarantee about eventual delivery. Typically a distributed application makes use of one or other of these two services.

13.3 VULNERABILITY AND ATTACKS IN NETWORKS

All kinds of computer networks face different types of security problems. Threats to any computer network arise from both internal and external entities. External threats include unauthorized access by outsiders such as hackers, virus attacks, and so on. Among the internal threats is exploitation of the network by its users, knowingly or unknowingly. Internal threats arise due to malicious intentions and/or ignorance of the users of networks. Some attacks are passive, meaning information is monitored; others are active, meaning the information is altered with intent to corrupt or destroy the data or network itself. The network and data are vulnerable to any of the different types of attacks explained in Chapter 14 if there are no security plans in place.

TABLE 13.1 Comparison Between Wired Networks and Wireless Networks.

Specifications	Wired Network	Wireless Network
Speed of operation	Higher	Lower compared to wired networks, but advanced wireless technologies such as LTE, LTE-A, and WLAN-11ad will make it possible to achieve speed equivalent to wired network
System Bandwidth	High	Low, as Frequency Spectrum is a very scarce resource
Cost	Less as cables are not expensive	More as wireless subscriber stations, wireless routers, wireless access points, and adapters are expensive
Installation	Wired network installation is cumbersome and it requires more time	Wireless network installation is easy and it requires less time

Specifications	Wired Network	Wireless Network
Mobility	Limited, as it operates in the area covered by connected systems with the wired network	Not limited, as it operates in the entire wireless network coverage
Transmission medium	copper wires, optical fiber cables, Ethernet	Electromagnetic waves or radiowaves or infrared
Network coverage extension	requires hubs and switches for network coverage limit extension	More area is covered by wireless base stations which are connected to one another
Applications	LAN (Ethernet), MAN	WLAN, WPAN (Zigbee, Bluetooth), Infrared, Cellular (GSM, CDMA, LTE)
Channel Interference and signal power loss	Interference is less as one wired network will not affect the other	Interference is higher due to obstacles between wireless transmitters and receivers, e.g., weather conditions, reflection from walls, etc.
QoS (Quality of Service)	Better	Poor due to high value of jitter and delay in connection setup
Reliability	High compared to wireless counterpart, as manufactured cables have higher performance due to existence of wired technology for years	Reasonably high, this is because failure of the router will affect the entire network

13.4 NETWORK COMMUNICATION ARCHITECTURE AND PROTOCOLS

A network architecture is an organization of the complete communication network that provides the framework and technology foundation for designing, building, and managing the communication network. For better understanding of the functionality network communication, the architecture typically has a layered structure. Layering is a modern network design principle that divides the communication tasks into a number of smaller parts, each part accomplishing a particular sub-task and interacting with the other parts. Layering also allows the other parts in a small number of well-defined

and tested cases without a combinatorial explosion of cases, keeping each design relatively simple.

If the network architecture is open, no single vendor owns the technology and controls its definition and development. Anyone is free to design hardware and software based on the network architecture. The TCP/IP network architecture, which the Internet is based on, is such an open network architecture, and it is adopted as a worldwide network standard and widely deployed in local area networks (LAN), wide area networks (WAN), small and large enterprises, and also mainly the Internet.

Network Protocols: A protocol defines the format and the order of messages exchanged between two or more communicating entities as well as the actions taken on the transmission and/or receipt of messages or other events. The Internet and computer networks in general make extensive use of protocols. Different protocols are used to accomplish different communication tasks.

13.4.1 ISO-OSI Model

Open system interconnection (OSI) network architecture developed by the International Standard Organization (ISO) is an open standard for communication in the network across different applications. OSI is a seven-layer network architecture that is widely considered as the primary network architecture model for LAN and the Internet.

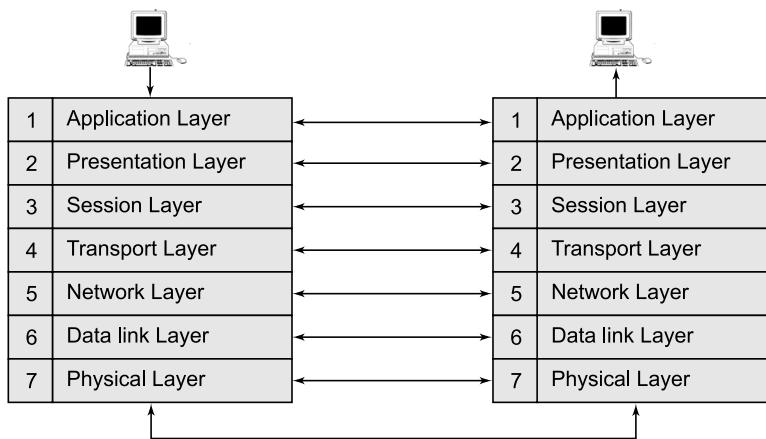


FIGURE 13.1 Seven-layer ISO-OSI model.

In addition to the OSI network architecture model, there exist other network architecture models by many vendors, such as IBM SNA (system network architecture), Digital Equipment Corporation (DEC now with HP)

Digital Network Architecture (DNA), Apple Computer's Apple Talk, and Novell's Netware. In an OSI seven-layer model these layers are divided into two groups: layers 7, 6, and 5 as lower layers, and 4, 3, 2, and 1 as upper layers. The upper layer of the OSI model deals with application issues and generally is implemented only in software. The lower layers of the OSI model handle data transport issues. The Physical and Data link layers are implemented in hardware and software. The lowest layer is the Physical layer, which is closest to the physical network medium communication between computers in a network, as shown in Figure 13.1.

TABLE 13.2 OSI Layer Functions.

Layer		Layer Functions	
1	Application Layer	-	Defines interface to user processes for communication and transfer in network
		-	Provides standardized services such as virtual terminal, file and job transfer, and operations
2	Presentation Layer	-	Masks the differences of data formats between dissimilar systems
		-	Specifies architecture independent data transfer format
		-	Encodes and decodes data; encrypts and decrypts data; compresses and decompresses data
3	Session Layer	-	Manages user sessions and dialogues
		-	Controls establishment and termination of logical link between users
		-	Reports upper layer errors
4	Transport Layer	-	Manages end-to-end message delivery in networks
		-	Provide reliable and sequential packet delivery through error recovery and flow control mechanism
		-	Provides connection oriented packet delivery
5	Network Layer	-	Determines how the data is transferred between network devices
		-	Routes packets according to unique network device addresses
		-	Provides flow and congestion control to prevent network resource depletion
6	Data link Layer	-	Defines procedure for operating the communication links
		-	Frames packets
		-	Detects and corrects packet transmit errors

(continued)

Layer		Layer Functions
7	Physical Layer	<ul style="list-style-type: none"> - Defines physical means of sending data over network devices
		<ul style="list-style-type: none"> - Interfaces between network medium and devices
		<ul style="list-style-type: none"> - Defines optical, electrical, and mechanical characteristics

13.4.2 TCP/IP Four-Layer Architecture Model

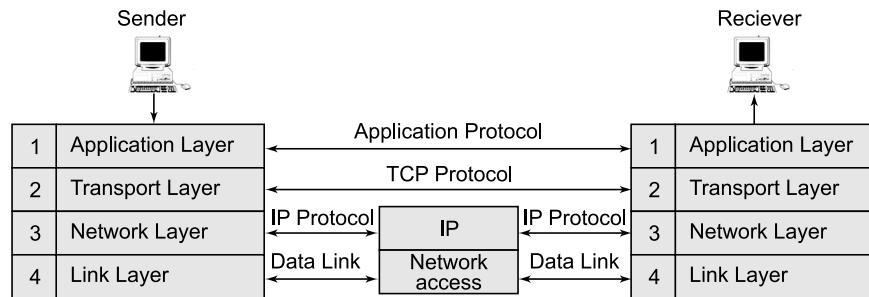


FIGURE 13.2 TCP/IP communication model.

TABLE 13.3 TCP/IP Layer Functions.

Layer		Layer Functions
1	Application Layer	<ul style="list-style-type: none"> - Any layer above the transport layer is called application layer. - It groups the function of OSI application, presentation layer and session layer. - In TCP/IP socket and port are used to describe the path over which applications communicate - Most application level protocols are associated with one or more port number.
2	Transport Layer	<ul style="list-style-type: none"> - There are two transport layer protocols; transmission control protocol (TCP) and user datagram protocol (UDP) - Both protocols are useful for different applications
3	Network Layer	<ul style="list-style-type: none"> - The internet protocol (IP) is the primary protocol in the TCP/IP network layer. - All the upper and lower layer communication must travel through IP as they are passed through TCP/IP protocol stack. - ICMP is another protocol used in network layer which manages the routing process.

	Layer	Layer Functions
4	Link Layer	<ul style="list-style-type: none"> - It is also known as network access layer - In TCP/IP architecture the data link layer and physical layer are normally grouped to become the link layer - The existing data link protocols are Ethernet, Token Ring, FDDI, HSSI and ATM. - The physical layer which defines the hardware communication properties.

The layer architecture of the TCP/IP and OSI models are not exactly the same. TCP/IP has fewer layers than the OSI model. The four-layer structure of TCP/IP is built as information is passed down from application to physical layer as shown in Figure 13.2.

The TCP/IP protocol stack four-layer model with data encapsulation is shown in Figure 13.3, and Table 13.4 depicts the TCP/IP protocol suite.

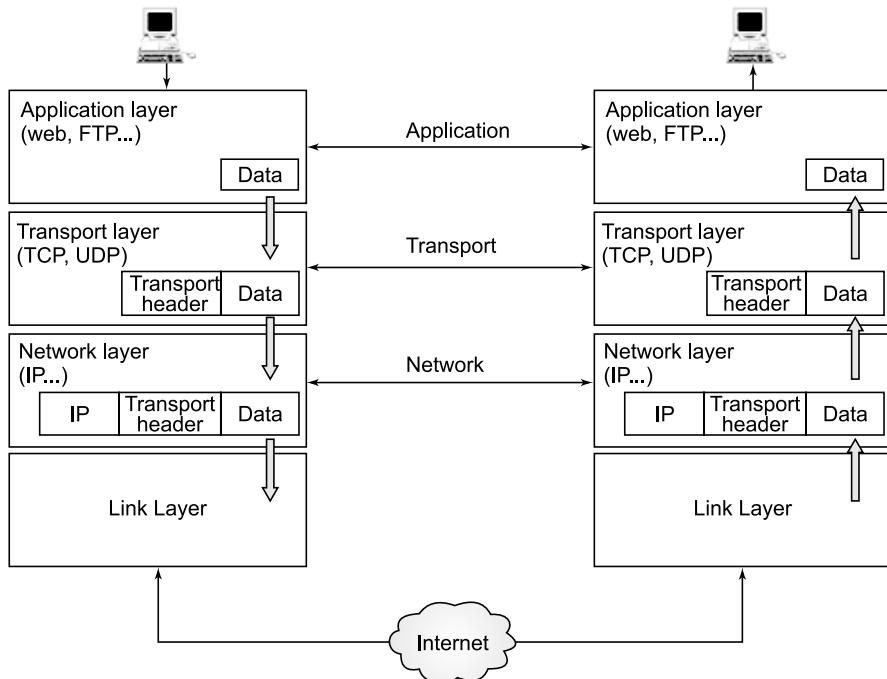


FIGURE 13.3 TCP/IP header.

TABLE 13.4 TCP/IP Protocol Suite.

Layer	Protocols
Application Layer	BOOTP: Bootstrap Protocol DCAP: Data Link Switching Client Access Protocol DHCP: Dynamic Host Configuration Protocol DNS: Domain Name Systems FTP: File Transfer Protocol Finger: User Information Protocol HTTP: Hypertext Transfer Protocol S-HTTP: Secure Hypertext Transfer Protocol (S-HTTP) IMAP & IMAP4: Internet Message Access Protocol IPDC: IP Device Control IRCP (IRC): Internet Relay Chat Protocol LDAP: Lightweight Directory Access Protocol MIME (S-MIME): Multipurpose Internet Mail Extensions (Secure MIME) NAT: Network Address Translation NNTP: Network News Transfer Protocol NTP: Network Time Protocol POP & POP3: Post Office Protocol (version 3) RLOGIN: Remote Login in Unix
	RMON: Remote Monitoring MIBs in SNMP SLP: Service Location Protocol SMTP: Simple Mail Transfer Protocol SNMP: Simple Network Management Protocol SNTP: Simple Network Time Protocol TELNET: TCP/IP Terminal Emulation Protocol TFTP: Trivial File Transfer Protocol URL: Uniform Resource Locator X-Window: X Window or X Protocol or X System
Presentation Layer	LPP: Lightweight Presentation Protocol
Session Layer	RPC: Remote Procedure Call protocol
Transport Layer	ITOT: ISO Transport Over TCP/IP RDP: Reliable Data Protocol RUDP: Reliable UDP TALI: Transport Adapter Layer Interface TCP: Transmission Control Protocol UDP: User Datagram Protocol Van Jacobson: Compressed TCP

Layer	Protocols
Network Layer	<p>Routing</p> <p>BGP/BGP4: Border Gateway Protocol EGP: Exterior Gateway Protocol IP: Internet Protocol IPv6: Internet Protocol version 6 ICMP/ICMPv6: Internet Control Message Protocol IRDP: ICMP Router Discovery Protocol Mobile IP: IP Mobility Support Protocol for IPv4 & IPv6 NARP: NBMA Address Resolution Protocol NHRP: Next Hop Resolution Protocol OSPF: Open Shortest Path First RIP (RIP2): Routing Information Protocol RIPng: RIP for IPv6 RSVP: Resource Reservation Protocol VRRP: Virtual Router Redundancy Protocol</p>
	<p>Multicast</p> <p>BGMP: Border Gateway Multicast Protocol DVMRP: Distance Vector Multicast Routing Protocol IGMP: Internet Group Management Protocol MARS: Multicast Address Resolution Server MBGP: Multiprotocol BGP MOSPF: Multicast OSPF MSDP: Multicast Source Discovery Protocol MZAP: Multicast-Scope Zone Announcement Protocol PGM: Pragmatic General Multicast Protocol PIM-DM: Protocol Independent Multicast - Dense Mode PIM-SM: Protocol Independent Multicast - Sparse Mode</p>
	<p>MPLS Protocols</p> <p>MPLS: Multi-Protocol Label Switching CR-LDP: Constraint-Based Label Distribution Protocol LDP: Label Distribution Protocol RSVP-TE: Resource Reservation Protocol-Traffic Engineering</p>
Data Link Layer	<p>ARP and InARP: Address Resolution Protocol and Inverse ARP IPCP and IPv6CP: IP Control Protocol and IPv6 Control Protocol RARP: Reverse Address Resolution Protocol SLIP: Serial Line IP</p>

13.4.3 Domain Name System (DNS)

IP addresses are hard to remember (nearly impossible in IPv6). The domain name system, or DNS, comes to the rescue by creating a way to convert hierarchical text names to IP addresses. Thus, for example, one can type *www.google.com* instead of 173. 194. 67.102. Virtually all Internet software uses the same basic library calls to convert DNS names to actual addresses. DNS is hierarchical and distributed. In looking up *www.google.com*, different DNS servers may be queried. Searching a hierarchy can be cumbersome, so DNS search results are normally cached locally. If a name is not found in the cache, the lookup may take a couple of seconds to search. The DNS hierarchy need have nothing to do with the IP-address hierarchy.

DNS Resolution: During the normal DNS resolution process, clients are provided with correct IP addresses for requested sites. The steps involved in DNS resolution is shown in Figure 13.4.

1. Client queries DNS for the IP address of *www.alliance.edu.in*.
2. DNS replies to client with IP address of *www.alliance.edu.in*; 192.0.32.10.
3. Client connects to 192.0.32.10; the IP address of *www.alliance.edu.in*.

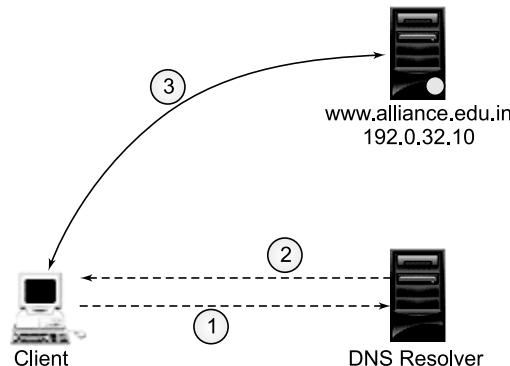


FIGURE 13.4 The normal DNS resolution process.

DNS Caching: The Internet doesn't just have a single DNS server, as that would be extremely inefficient. The Internet Service Provider (ISP) runs its own DNS servers, which cache information from other DNS servers. The router connected to the network also functions as a DNS server, which caches information from the ISP's DNS servers. Each entry in a DNS cache has a time limit set, depending upon operating systems and accuracy of DNS resolutions. After the period expires, the computer or server containing the DNS cache will contact the DNS server and update the entry so that the

information is correct. However, there are malicious users who can poison the DNS cache for criminal activity.

DNS resolvers are typically configured to query upstream counterparts if they do not have an entry cached for the requested domain name. This is known as *recursion*, or caching. Recursion improves response times and performance by caching replies similar to the way in which history is cached by a web browser. Entries will remain in a DNS resolver's cache depending on the time to live (TTL) value in the returned record. A common TTL value for DNS is 86400 seconds (24 hours).

16	21					28	32 bit	
ID	Q	Query	A	T	R	V	B	Recode
Question Count	Answer Count							
Authority Count	Additional Count							

FIGURE 13.5 DNS protocol frame format.

- ID: 16-bit field used to correlate queries and responses.
- Q: 1-bit field that identifies the message as a query or response.
- Query: 4-bit field that describes the type of message: 0 Standard query (name to address); 1 Inverse query; 2 Server status request.
- A: Authoritative Answer. 1-bit field. When set to 1, identifies the response as one made by an authoritative name server.
- T: Truncation. 1-bit field. When set to 1, indicates the message has been truncated.
- R: 1-bit field. Set to 1 by the resolve to request recursive service by the name server.
- V: 1-bit field. Signals the availability of recursive service by the name server.
- B: 3-bit field. Reserved for future use. Must be set to 0.
- Recode: Response Code. 4-bit field that is set by the name server to identify the status of the query.
- Question count: 16-bit field that defines the number of entries in the question section.
- Answer count: 16-bit field that defines the number of resource records in the answer section.
- Authority count: 16-bit field that defines the number of name server resource records in the authority section.
- Additional l count: 16-bit field that defines the number of resource records in the additional records section.

DNS Protocol: The DNS protocol works when a computer sends out a DNS query to a name server to resolve a domain. For example, you type “www.google.com” in the web browser; this triggers a DNS request, which your computer sends to a DNS server in order to get the website’s IP Address. The DNS protocol utilizes Port 53 for its service. This means that a DNS server listens on Port 53 and expects any client wishing to use the service to use the same port. There are, however, cases where you might need to use a different port, something possible depending on the operating system and DNS server you are running. The frame format of the DNS protocol is shown in Figure 13.5.

Attack on DNS

DNS has no authentication mechanisms included by default. The lack of authentication increases the risk of falsified DNS information being stored on your agency’s DNS resolver by hosts with no authority to do so. These activities are known as DNS spoofing and DNS cache poisoning.

DNS spoofing and cache poisoning can permit a cyber-actor to map the internal network of the organization based on queries from the internal DNS resolver to upstream DNS resolvers. DNS cache poisoning can disrupt client connections to provide false information, facilitating installation of malicious code or the extraction of sensitive information.

DNS Cache Poisoning and Spoofing

DNS Cache Poisoning: DNS cache poisoning, also known as DNS spoofing, is a type of attack that exploits vulnerabilities in the DNS to divert Internet traffic away from the legitimate server and toward a fake one. One of the reasons DNS poisoning is very dangerous is because it can spread from DNS server to DNS server.

Poisoning the cache means changing the real values of URLs. For example, cybercriminals can create a Website that looks like, say, *www.abc.com* and enter its DNS record in the DNS cache. Thus, when we type *www.abc.com* in the address bar of the browser, the latter will pick up the IP address of the fake website and take the connection to it, instead of the real Website. This is called *pharming*. Using this method, cyber criminals can phish out login credentials and other information such as card details, social security number, phone numbers, and more for identity theft. DNS poisoning is also done to inject malware into the computer or network. Once we land on a fake Website using a poisoned DNS cache, the criminals can do anything they want.

Sometimes, instead of the local cache, criminals can also set up fake DNS servers so that when queried, they can give out fake IP addresses. This is high-level DNS poisoning and corrupts most of the DNS caches in a particular area, thereby affecting many more users. A DNS cache poisoning attack subverts the normal DNS resolution process as follows:

1. Cyber-actor queries a DNS resolver for the IP address of a malicious site.
2. DNS resolver does not have the IP address and queries a malicious DNS resolver which has already established a relationship with the DNS resolver.
3. Malicious DNS resolver provides requested IP address (203.6.114.66) along with falsified IP addresses for additional sites (e.g., *www.abc.com*). (IP addresses may be different from that of the malicious DNS resolver.)

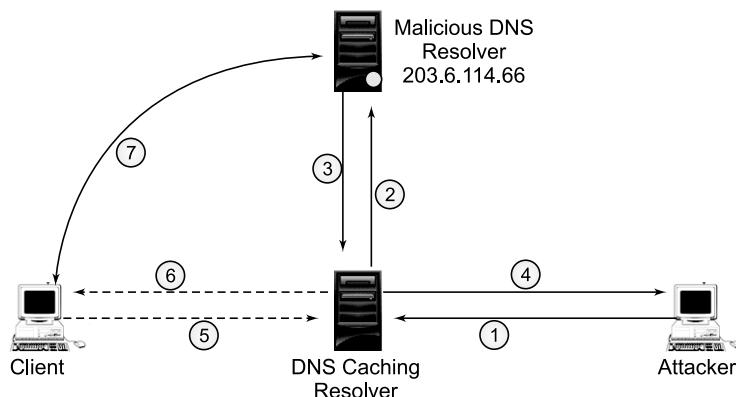


FIGURE 13.6 The normal DNS resolution process altered by DNS cache poisoning.

4. DNS resolver replies to cyber-actor and caches false IP addresses.
5. Client queries DNS for the IP address of *www.abc.com*.
6. DNS resolver replies to client with (cached) IP address of 203.6.114.66.
7. Client connects to 203.6.114.66 expecting it to be the genuine *www.abc.com* Website.

DNS Cache Poisoning with Flooding: A DNS cache poisoning attack with flooding subverts the normal DNS resolution process as follows:

1. Cyber-actor queries DNS caching resolver for IP address of *www.abc.com*.

2. The DNS caching resolver queries the authoritative DNS server for *www.abc.com*.
3. The DNS caching resolver will accept the first response that matches the transaction ID and source port of its query to the authoritative server. The attacker floods the caching DNS resolver with fraudulent responses containing many different transaction IDs and source ports, hoping one of these will match.
4. One of the attacker's fraudulent responses is accepted. The DNS caching resolver responds to the attacker's original query with the poisoned result.
5. The authoritative DNS resolver responds to the DNS caching resolver. This response is ignored since the caching server already accepted a fraudulent response from the attacker.
6. A client tries to reach *www.alliance.edu.in* and looks up the IP address of the site by querying the DNS caching server.
7. The DNS caching server returns a result from its cache, which is the poisoned result provided by the attacker in 3. This poisoned result will direct the client to a malicious site.

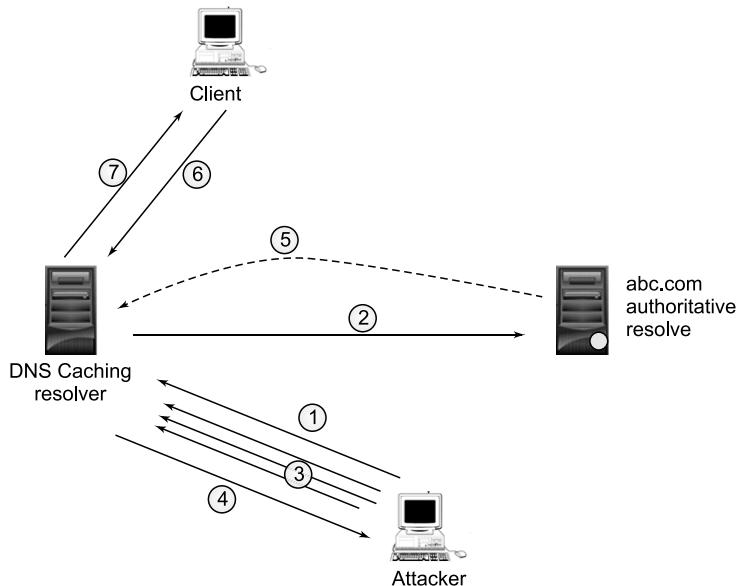


FIGURE 13.7 The normal DNS resolution process altered by DNS cache poisoning (with flooding).

DNS Spoofing: A DNS spoofing attack subverts the normal DNS resolution process as follows:

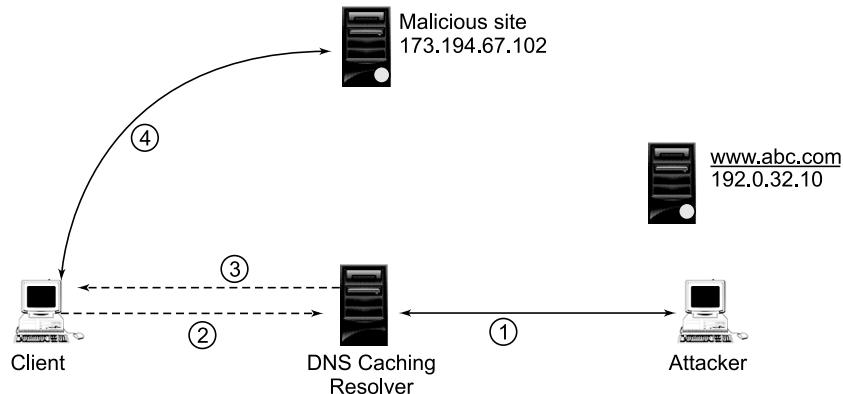


FIGURE 13.8 The normal DNS resolution process altered by DNS spoofing.

1. Cyber-actor adds or alters DNS record for *www.alliance.edu.in* on DNS resolver to point to 203.6.114.66 instead of 192.0.32.10.
2. Client queries DNS for the IP address of *www.alliance.edu.in*.
3. DNS replies to client with IP address of 203.6.114.66.
4. Client connects to malicious site 203.6.114.66, expecting it to be the genuine site for *www.alliance.edu.in*.

DNS Amplification for DDoS: DNS amplification attacks are not threats against the DNS systems. Instead, they exploit the open nature of DNS services to strengthen the force of distributed denial of service (DDoS) attacks. In DDoS attacks the attacker uses a network of malware-infected computers to send large amounts of traffic to a target, such as a server. The goal is to overload the target and slow or crash it. Amplification attacks add more strokes. Rather than sending traffic directly from a botnet to a victim, the botnet sends requests to other systems. Those systems respond by sending even greater volumes of traffic to the victim. DNS amplification attacks are a perfect example. Attackers use a botnet to send thousands of lookup requests to open DNS servers. The requests have a spoofed source address and are configured to maximize the amount of data returned by each DNS server.

DNS Attacked by DDoS: DDoS attacks can be used against many different types of systems. This includes DNS servers. A successful DDoS attack against a DNS server can cause it to crash, rendering the users who rely on the server unable to browse the Web. The users will still likely be able to reach websites they've visited recently, assuming the DNS record is saved in a local cache.

Reflected attacks: Reflect attacks send thousands of requests with the victim's name as the source address. When recipients answer, all replies converge on the official sender, whose infrastructures are then affected.

Fast flux: In addition to falsifying their IP address, attackers can hide their identity by using this technique, which relies on fast-changing location-related information to conceal where the attack is coming from. Variants exist, such as single flux (constantly changing the address of the Web server) and double flux (constantly changing the address of the Web server and the names of the DNS servers).

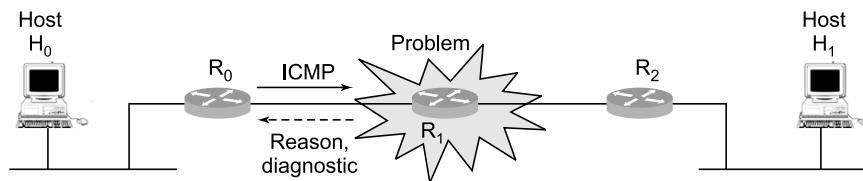


FIGURE 13.9 An ICMP message consisting of 4 bytes of PCI and an optional message payload.

Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) is a classic example of a client server application. The ICMP server executes on all IP end system computers and all IP intermediate systems (routers). The protocol is used to report problems with the delivery of IP datagrams within an IP network. It is used to show when a particular end system is not responding, when an IP network is not reachable, when a node is overloaded, when an error occurs in the IP header information, and so on. The protocol is also frequently used by Internet managers to verify correct operations of an end system and to check that routers are correctly routing packets to the specified destination address. ICMP messages are generated by router R_1 in response to a message sent by H_0 to H_1 and forwarded by R_0 , as in Figure 13.9. This message could, for instance, be generated if the maximum transmission unit (MTU) of the link between R_0 and R_1 was smaller than the size of the IP packet, and

the packet had the Don't Fragment (DF) bit set in the IP packet header. The ICMP message is returned to H_0 , since this is the source address specified in the IP packet that suffered the problem. A modern version of Path MTU Discovery provides a mechanism to verify the Path MTU.

0	7 8	15 16	31
8-bit Type	8-bit Code	16 bit Checksum	
Contents depend on type and code			

FIGURE 13.10 ICMP frame format.

The format of an ICMP message is shown in Figure 13.10. The 8-bit type code identifies the type of message. This is followed by at least the first 28 bytes of the packet that resulted in generation of the error message (i.e., the network-layer header and first 8 bytes of the transport header). This payload is, for instance, used by a sender that receives the ICMP message to perform Path MTU Discovery so that it may determine the IP destination address of the packet that resulted in the error. Longer payloads are also encouraged (which can help better identify the reason why the ICMP message was generated and which program generated the original packet). Figure 13.11 shows the encapsulation of ICMP over an Ethernet LAN using an IP network layer header, and a MAC link layer header and trailer containing the 32-bit checksum:

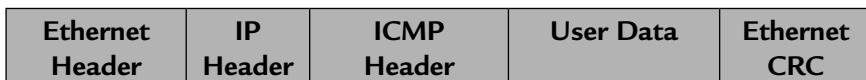


FIGURE 13.11 Encapsulation for a complete ICMP packet.

It is the responsibility of the network layer (IP) protocol to ensure that the ICMP message is sent to the correct destination. This is achieved by setting the destination address of the IP packet carrying the ICMP message. The source address is set to the address of the computer that generated the IP packet (carried in the IP source address field), and the IP protocol type is set to “ICMP” to indicate that the packet is to be handled by the remote end system’s ICMP client interface.

The Ping Application

The “ping” program contains a client interface to ICMP. It may be used by a user to verify that an end-to-end Internet Path is operational. The ping

program also collects performance statistics (i.e., the measured round trip time and the number of times the remote server fails to reply). Each time an ICMP echo reply message is received, the ping program displays a single line of text. The text printed by the ping shows the received sequence number and the measured round trip time (in milliseconds). Each ICMP Echo message contains a sequence number (starting at 0) that is incremented after each transmission, and a time stamp value indicating the transmission time.

The operation of ICMP is illustrated in the frame transition diagram shown previously. In this case there is only one Intermediate System (IS) (i.e., IP router). In this case two types of messages are involved: the ECHO request (sent by the client) and the ECHO reply (the response by the server). Each message may contain some optional data. When data are sent by a server, the server returns the data in the reply which is generated. ICMP packets are encapsulated in IP for transmission across the Internet.

The Internet Control Message Protocol, or ICMP, is a protocol for sending IP-layer error and status messages; it is defined in RFC 792. ICMP is, like IP, host-to-host, so it is never delivered to a specific port, even if it is sent in response to an error related to something sent from that port. In other words, individual UDP and TCP connections do not receive ICMP messages, even when it would be helpful to get them. ICMP messages are identified by an 8-bit type field, followed by an 8-bit subtype, or code. Here are the more common ICMP types, with subtypes listed in the description.

```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-2001.

C:\DOCUME~1\CHAD\DESKTOP>ping www.youtube.com

Pinging youtube-ui.l.google.com [74.125.127.113] with 32 bytes of data:
Reply from 74.125.127.113: bytes=32 time=53ms TTL=247
Reply from 74.125.127.113: bytes=32 time=55ms TTL=247
Reply from 74.125.127.113: bytes=32 time=54ms TTL=247
Reply from 74.125.127.113: bytes=32 time=53ms TTL=247

Ping statistics for 74.125.127.113:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 53ms, Maximum = 55ms, Average = 53ms

C:\DOCUME~1\CHAD\DESKTOP>
```

FIGURE 13.12 Use of the ping program to test whether a particular computer is operational.

Attacks Using ICMP Messages

ICMP facilitates sending a one-way informational message to a host and informs the source host about errors in datagram processing. These two operations are heavily exploited by the attackers to launch the following attacks:

ICMP Sweep: An ICMP sweep is not a direct attack on a network, but it is a definite threat to security. By using a sweep, attackers can determine active hosts and perform more direct targeted attacks specific to those hosts. By sending a series of ICMP “echo request” packets to every IP on a network segment, an attacker will receive ICMP replies confirming that a host is alive. This process is fairly “noisy,” as the attackers are broadcasting across a whole network range.

Inverse Mapping: Networks are protected by filtering devices such as firewalls and gateways that prevent internal hosts from being reached externally. An attacker uses inverse mapping to obtain a map of an internal network. ICMP reply messages are sent to internal routers about the hosts nearby and get the information about the network. This is accomplished without the filtering devices knowing.

TABLE 13.5 ICMP.

Type	Description
Echo Request	ping queries
Echo Reply	ping responses
Destination Unreachable	Destination network unreachable
	Destination host unreachable
	Destination port unreachable
	Fragmentation required but DF flag set
	Network administratively prohibited
Source Quench	Congestion control
Redirect Message	Redirect datagram for the network
	Redirect datagram for the host
	Redirect for TOS and network
	Redirect for TOS and host
Router Solicitation	Router discovery/selection/solicitation
Time Exceeded	TTL expired in transit
	Fragment reassembly time exceeded

(continued)

Type	Description
Bad IP Header or Parameter	Pointer indicates the error
	Missing a required option
	Bad length
Time stamp Reply	Like ping, but requesting a time stamp from the destination

Traceroute Network Mapping: Microsoft Windows and all Linux derivatives include a network tool known as *traceroute* that provides a mechanism for tracking the path of packets flowing between the host and a destination host. It achieves this by utilizing the IP protocols TTL (time to live) field, where it attempts to elicit an ICMP “*time exceeded*” response from each gateway/router along the path to some host. By default the Linux version of the *traceroute* application uses UDP to perform its tracing, but it also provides an argument (-I) that allows the tool to use ICMP instead.

OS Fingerprinting: Often an attacker will need to identify what system they are about to attack before they can exploit a vulnerability. In this technique, the attacker relies upon the operating system manufacturer to have built their communications system slightly differently from other operating systems, and the steps to recreate this technique are: The attacker sends malformed ICMP packets to the destination. The destination host will respond with numerous answers to the given requests. Each operating system will send slightly different results back to the host. The installed operating system is determined by a process of elimination by evaluating the responses. This flaw in the development of the operating system allows specially designed tools to examine the structure of the returned ICMP data and determine the likely operating system.

ICMP Route Redirect: One of the main functions of ICMP is to provide the ability to redirect routing. For example, another route has been found to be more efficient or a route failure been discovered. The route redirect technique exploits this function by allowing a false ICMP packet to be transmitted, telling a target host that they must route information through a new gateway, the “*attacker*.” After the traffic is rerouted through the attacker, it can be monitored using a packet sniffing application.

Ping of Death: The attacker sends excessively large ICMP messages to a target host. Exploiting the weakness in the operating system’s implementation of the TCP/IP specification, the attacker can send an ICMP packet greater than the maximum of 65535 octets allowed. The host may become unavailable as a buffer overflow may be created based on the operating

system. The computer may crash, force a reboot, or make the host hang. A similar attack can be achieved by sending multiple fragmented ICMP packets that require the operating system to restructure the data on arrival. On examination, the operating system discovers that the packets are not the size they say they are, and as a result it forces the machine to hang or reboot.

ICMP Smurf Attack: This attack exploits the weakness in the ICMP and IP protocols by forging the original source address of the packet with the address of the machine to be attacked. This “spoofing” hides the attacker and begins a chain reaction of network disruption.

ICMP Nuke Attack: A nuke is an old denial-of-service attack against computer networks consisting of fragmented or otherwise invalid ICMP packets sent to the target. It can be achieved by using a modified ping utility to repeatedly send this corrupt data, thus slowing down the affected computer until it comes to a complete stop. Nukes send a packet of information that the target OS can't handle, which causes the system to crash. A specific example of a nuke attack that gained some prominence is the *WinNuke*, which exploited the vulnerability in the NetBIOS handler in Windows 95. A string of out-of-band data was sent to TCP port 139 of the victim's machine, causing it to lock up and display a Blue Screen of Death (BSOD).

Attack Using Source Quench: ICMP source quench messages are generated when a gateway device runs out of buffer space. It is an informational message that is generated in an attempt to inform the remote host generating the traffic to limit the speed at which it is sending network traffic to the remote host. Denial of service attackers could potentially use ICMP source quench datagrams to rate limit a remote host that listens to unsolicited ICMP source quench datagrams. The attacker needs to guess the sequence number and also match the same with that of the connection. The need to guess the connection source port will lead to many Source Quench messages which do not correspond to an existing connection.

SUMMARY

- A computer network consists of two or more computing devices that are connected in order to share the resources in a network and information stored.
- The Internet is the world's largest network of networks. This interconnects millions of computers, providing a global communication, storage, and computational infrastructure.

- The threat to any computer network arises from both internal and external entities.
- A network architecture is an organization of the completer communication network, which provides the framework and technology foundation for designing, building, and managing the communication network.
- The OSI is a seven-layer network architecture that is widely considered as the primary network architecture model for LAN and the Internet.
- The domain name system (DNS) converts hierarchical text names to IP addresses.
- The Internet Service Provider (ISP) runs its own DNS servers, which cache information from other DNS servers.
- DNS has no authentication mechanisms included by default. These activities are known as DNS spoofing and DNS cache poisoning.
- DNS cache poisoning, also known as DNS spoofing, is a type of attack that exploits vulnerabilities in DNS to divert Internet traffic away from a legitimate server and toward a fake one.
- DNS amplification attacks are not threats against the DNS systems. Instead, they exploit the open nature of DNS services to strengthen the force of distributed denial of service (DDoS) attacks.
- ICMP is used to show when a particular end system is not responding, when an IP network is not reachable, when a node is overloaded, when an error occurs in the IP header information, and so forth. ICMP is a protocol for sending IP-layer error and status messages.
- The “*ping*” program contains a client interface to ICMP.
- ICMP facilitates sending one-way informational messages to a host and informs the source host about errors in datagram processing.

REVIEW QUESTIONS

1. Describe the comparison between the OSI and TCP/IP layer architecture model.
2. Explain the DNS resolution process.
3. Explain DNS cache poisoning and spoofing.
4. Explain the attacks using ICMP messages.
5. Describe the applications of ping.

MULTIPLE CHOICE QUESTIONS

9. The _____ is code that recognizes some special sequence of input or is triggered by being run from a certain user ID by an unlikely sequence of events.
- (a) trap doors (b) Trojan horse (c) logic bomb (d) virus
10. The _____ is code embedded in some legitimate program that is set to “explode” when certain conditions are met.
- (a) trap doors (b) trojan horse (c) logic bomb (d) virus
11. Which of the following malicious programs does not replicate automatically?
- (a) Trojan Horse (b) Virus (c) Worm (d) Zombie
12. _____ programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.
- (a) Zombie (b) Worm (c) Trojan Horses (d) Logic Bomb
13. State whether true or false.
- (i) A worm mails a copy of itself to other systems.
(ii) A worm executes a copy of itself on another system.
- (a) True, False (b) False, True
(c) True, True (d) False, False
14. A _____ is a program that can infect other programs by modifying them; the modification includes a copy of the virus program, which can go on to infect other programs.
- (a) worm (b) virus (c) zombie (d) trap doors
15. This is a series of messages sent by someone attempting to break into a computer to learn which computer network services the computer provides.
- (a) Bit robbing (b) Web services description language (WSDL)
(c) Port Scan (d) Service Profile Identification

NETWORK ATTACKS AND SECURITY THREATS

14.1 INTRODUCTION

Network security has become more important to personal computer users, organizations, and the military. With the advent of the Internet, security became a major concern. Systems and network technology is a key technology for a wide variety of applications. Although security is a critical requirement in emerging networks, there is a significant lack of security methods that can be easily implemented. Network design is well-developed process that is based on the open system interconnection (OSI) model. The OSI model has several advantages when designing networks. It offers modularity, flexibility, ease of use, and standardization of protocols. The protocols of different layers can be easily combined to create stacks which allow modular development.

When considering network security, it must be emphasized that the whole network is secure. Network security does not concern the security in the computer at each end of the communication chain. When transmitting data the communication channel should not be vulnerable to attack. A possible hacker could target the communication channel, obtain the data, decrypt it, and reinsert a false message. Securing the network is just as important as security in the computers and encryption of the message. For developing a secure network model, the following needs to be considered:

- **Access:** Only authorized users are provided the means to communicate to and from a particular network.
- **Confidentiality:** Information in the network remains private.
- **Authentication:** Ensure the users of the network are who they say they are.
- **Integrity:** Ensure the message has not been modified in transit.
- **Non-repudiation:** Ensure the user does not refuse that he used the network.

An effective network security plan is developed with the understanding of security issues, potential attacks, needed level of security, and factors that make a network vulnerable to attack.

14.2 DIFFERENT TYPES OF NETWORK ATTACKS AND SECURITY THREATS

A network attack or security threat is defined as a threat, intrusion, denial of service (DoS), or other attack on a network infrastructure. The attacks eventually gain access to the network and cause your network to crash or to become corrupted. In many cases the attacker might also try to obtain unauthorized access to network devices. The attacks on different layers of the OSI model are as depicted in Figure 14.1. There are at least eight types of network attacks:

1. Spoofing
2. Sniffing
3. Mapping
4. Session Hijacking
5. Trojans
6. Denial of service (DoS) and Distributed Denial of Service (DDoS)
7. Smurf Attack
8. SYN Floods
9. Social Engineering
10. Password Based Attack
11. Compromised Key Attack
12. Application Layer Attack
13. FTP Bounce

OSI-Layers	Protocols	Attacks
Application	DNS, DHCP, HTTP, FTP, TFTP, LDAP, NTP, Radius, SMTP, SNMP, Telnet, POP3, IMAP	DNS Poisoning, Phishing, SQL Injection, SPAM, SCAM
Presentation	XML, XDR, ASN.1, SMB, AFP	
Session	SSH, ISO 8327, CCITT, X.225, RPC, NetBIOS, ASP	
Transport	TCP, UDP, RTP, SCTP, ATP	TCP Attacks, Routing Attacks, SYN flooding, sniffing
Network	IPv4, IPv6, ICMP, IGMP, IPSec, CLNP	Ping/ICMP flooding
Data Link	ARP, Token Ring, Ethernet, wireless	ARP Spoofing, MAC Flooding
Physical		

FIGURE 14.1 Attacks on different layers of the OSI model.

14.2.1 Spoofing (IP Spoofing)

Any Internet-connected device necessarily sends IP datagrams into the network, and such Internet data packets carry the sender's IP address as well as the application layer data. Spoofing means having the address of the computer copy the address of a trusted computer in order to gain access to other computers. The identity of the intruder is hidden by different means, making detection and prevention difficult with the current IP-technology. IP-spoofed packets cannot be eliminated. With a spoofed-source IP address on a datagram, it is difficult to find the host that actually sent the datagram. Some of the popular attacks launched by IP spoofing are:

- **Blind spoofing:** In this type of attack a cracker outside the perimeter of the local network transmits multiple packets to his intended target to receive a series of sequence numbers, which are generally used to assemble packets in the order in which they were intended. The cracker is blind to how transmissions take place on his network, so he needs to coax the machine into responding to his own requests so he can analyze

the sequence numbers. By taking advantage of knowing the sequence number, the cracker can falsify his identity by injecting data into the stream of packets without having to have authenticated himself when the connection was first established.

- **Non-blind spoofing:** In this type of attack the cracker resides on the same subnet as his intended target, so by sniffing the wire for existing transmissions, he can understand an entire sequence/acknowledgement cycle between his target and another host (hence, the cracker isn't "blind" to the sequence number). Once the sequence number is known, the attacker can hijack sessions that have already been built by concealing himself as another machine, bypassing any sort of authentication that was previously conducted on that connection.

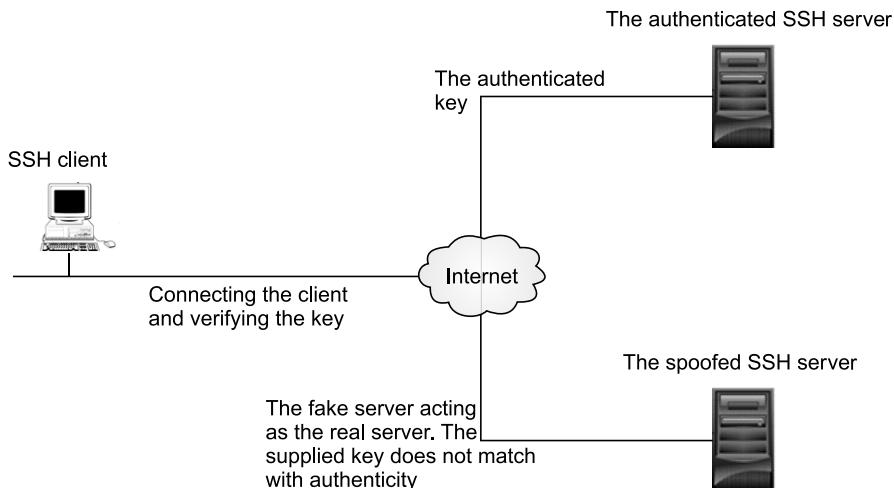


FIGURE 14.2 IP spoofing.

- **DoS attack:** To keep a large-scale attack on a machine or group of machines from being detected, spoofing is often used by the malefactors responsible for the event to disguise the source of the attacks and make it difficult to shut it off. Spoofing takes on a whole new level of severity when multiple hosts are sending contrast streams of packets to the DoS target. In that case all the transmissions are generally spoofed, making it very difficult to track down the source of the storm.
- **Man-in-the-middle attack:** In a man-in-the-middle attack, a malicious machine intercepts the packets sent between two machines that are in the process of data transmission. The attacker alters these packets and then sends them on to the intended destination, with the originating and

receiving machines unaware their communication has been tampered with. Typically this type of attack is used to get targets to reveal secure information and continue such transmissions for a period of time, all the while unaware that the machine in the middle of the transmission is eavesdropping the whole time.

The following techniques help to prevent IP spoofing and its related attacks from affecting the networks.

- a. Use authentication based on the key exchange between the machines connected in a network or on the Internet. The use of *IPsec* will significantly cut down on the risk of spoofing.
- b. Use an access control list to deny private IP addresses in the downstream interface.
- c. Implement filtering of both inbound and outbound traffic.
- d. Configure the routers and switches, if they support such configurations, to reject packets originating from outside the local network that claim to originate from within.
- e. Enable encryption sessions in a router so that trusted hosts that are outside of the network can securely communicate with the local hosts.

14.2.2 Sniffing

Packet sniffing is the interception of data packets traversing a network. A sniffer program works at the Ethernet layer in combination with the network interface card (NIC) to capture all traffic traveling to and from the Internet host site as in Figure 14.3. In addition, if any of the Ethernet NIC cards are in promiscuous mode, the sniffer program will pick up all communication packets floating by anywhere near the Internet host site.

A sniffer placed on any backbone device internetwork link or network aggregation point will therefore be able to monitor a whole lot of traffic. Most of the packet sniffers are passive and they listen all data link layer frames passing by the device network interface. Sniffing can be detected in two ways:

- (i) **Host-based:** Software commands exist that can be run on individual host machines to tell if the NIC is running in promiscuous mode.
- (ii) **Network-based:** Solutions exist that check for the presence of running processes and log files. A sophisticated intruder almost always hides their tracks by disguising the process of cleaning up the log files.

Some of the popular packet sniffing tools are:

- Coin and abel
- Carnivore
- Dsniff
- Ethercap
- Fiddler
- Tcpdump
- Wireshark

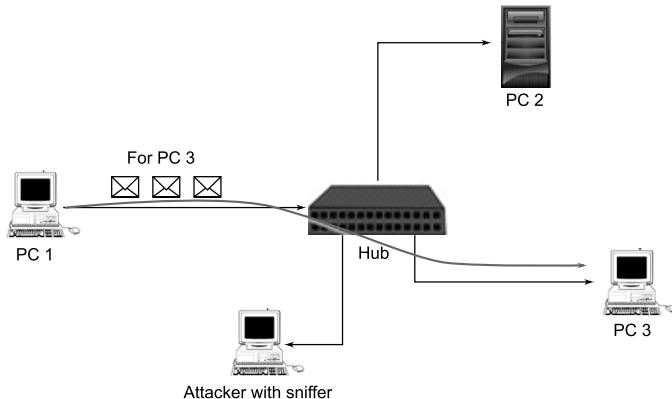


FIGURE 14.3 Attacker sniffs the packets traversing between PC1 and PC3.

14.2.3 Mapping (Eavesdropping)

This type of network attack occurs when an attacker monitors or listens to network traffic in transit, then interprets all unprotected data. Before attacking a network, attackers would like to know the IP address of machines on the network, the operating systems they use, and the services that they offer. With this information, their attacks can be more focused and are less likely to cause alarm. The process of gathering this information is known as *mapping*.

In general, the majority of network communications occur in an unsecured or “clear text” format, which allows an attacker who has gained access to data paths in a network to “listen in” or interpret the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators look for in an enterprise. A user needs specialized equipment and access to Internet switching facilities to eavesdrop on message communication. This type of attack is basically due to TCP/IP being an open architecture that transmits unencrypted data over the network.

A few methods of preventing intruders from eavesdropping on the network are:

- Implement Internet Protocol Security (IPSec) to secure and encrypt IP data before it is sent over the network.
- Implement security policies and procedures to prevent attackers from attaching a sniffer on the network.
- Install antivirus software to protect the corporate network from Trojans. Trojans are typically used to discover and capture sensitive, valuable information such as user credentials.

14.2.4 Session Hijacking

Session hijacking is a means of either gaining total or partial control over an established TCP/IP connection. This attack relies on a trusted connection between two computers to be in place, and then works either to modify the packet traveling between the machines or take the place of one of the two computers. Session hijacking is an effective means of gaining control over a machine or process that otherwise would not be accessible. Most authentication takes place during the initiation of a connection. After authentication, the conversation is considered trusted. This is the point at which attackers want session hijacking to take place. The most common and effective form of session hijacking is called a *man-in-the-middle attack* and works by placing a computer in the middle of an established connection. Both ends of a connection must be convinced that to speak to either side, they need to go through the attacking computer.

Session hijacks are usually waged against users that are members of large networks containing a substantial number of open sessions. Network protocols such as FTP, Telenet, and rlogin are especially attractive to the attackers because of the session-oriented nature of their connections and the length of their communication sessions. Additionally, FTP, Telnet, and rlogin do not implement any security during login, authentication, or data transmission. In fact, data sent using these protocols is sent in clear text, which can be easily be viewed by anyone monitoring the network. A successful session hijack attack also allows the attacker to issue commands to the server over the network. This is later used to create user accounts and to access the resources.

Types of Session Hijack Attacks

Session hijack attacks can be classified into three different types: active, passive, and hybrid.

- (i) **Active hijack attack:** The active attack is when the attacker hijacks a session on the network. The attacker takes control of one of the client positions in a client-server network when there is a session between

them, as in Figure 14.4. An attacker also gains control over the networks to issue commands on the network, making it possible to create new user accounts on the network. This account can be later used to intrude and to perform malicious operations like session hijack attacks.

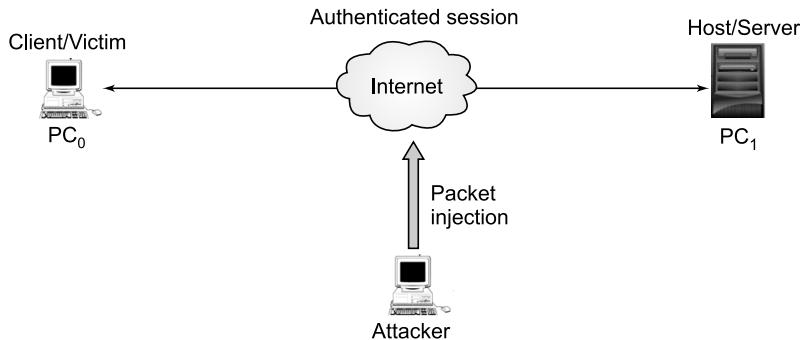


FIGURE 14.4 Active hijacking.

- (ii) **Passive session hijack:** Passive session hijacking is performed by traffic analysis. The attacker monitors the traffic between the client and server network as in Figure 14.5. The primary objective of passive attack is it provides the attackers with the ability to monitor network traffic and potentially discover valuable data and other confidential information.
- (iii) **Hybrid session hijack:** This attack is a combination of both an active and a passive attack. It allows the attacker to listen to network traffic until the expected result is obtained. The attacker can then modify the attack by isolating the user computer from the session and assuring their identity. A session hijack attack involves the following steps:

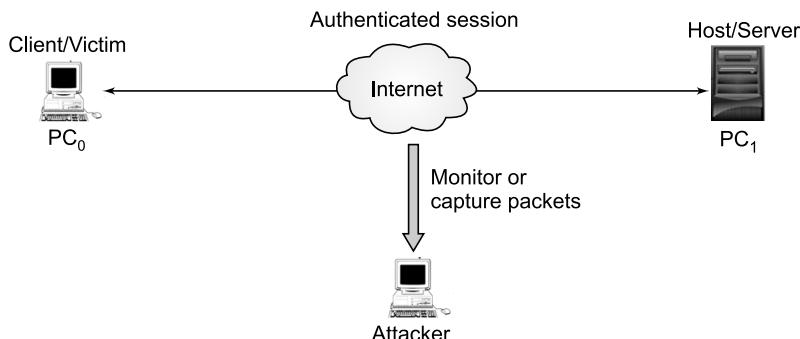


FIGURE 14.5 Passive hijacking.

- a. **Locating a target:** In this step the attacker identifies the target user. The two main requirements identified by the attacker before the start of the attack are:
 - (i) Consider networks which have a very high level of utilization. This provides a healthy supply of users to choose from, which also helps the attack remain anonymous.
 - (ii) The attacker usually targets networks in which users frequently use insecure network protocols such as Telnet, rlogin, and FTP, due to their inherently insecure design. Packet sniffing software can be used to sniff network traffic for the purpose of locating vulnerable protocols like FTP, Telnet, and rlogin. To identify the active Telnet, rlogin, and FTP ports in the server, port scanning software can also be used.
- b. **Find an active session:** Session hijack attacks are usually against servers with a large amount of activities. This is done for the following two reasons:
 - (i) The high network utilization provides an environment containing adequate sessions which can be exploited.
 - (ii) The high usage on the server helps hide the disruption caused by the attack. Attackers generally target session-oriented protocols like Telnet, rlogin, and FTP, which provide prolonged connections to other computers. Some of the commonly used session hijack software are Wireshark, T-sight, or Juggernaut.
- c. **Perform sequence number prediction:** After deciding the target the attacker thinks of a session hijacking process by session number prediction. This process involves guessing the next sequence number that the server is expecting from the client in a client-server communication process. The sequence number prediction is a critical step, because failing to predict the correct sequence number will result in the server sending reset packets and terminating the connection attempts. With a continuous attempt of guessing the wrong sequence number repeatedly, there is a chance of detecting the attack. The sequence number can be accurately predicted by using some of the software tools like Juggernaut, Hunt, and T-sight. These software tools help the skilled attacker to predict the next session number.

- d. **Take one of the parties off-line:** Once the location is targeted and the session number is predicted accurately, the next step is to block the client (user) computer in the network. This is generally done with the help of denial of service or any other identical techniques that render the computer unable to communicate on the network. The attacker must ensure that the client computer remains off-line for the duration of the attack. If not the client computer will begin transmitting data on the network, repeatedly causing client-server synchronization in the network. This may result in a situation known as an ACK storm. Isolating the client computer completely prohibits the attacker from examining the communication between the client-server computer in the network.
- e. **Take over the session and maintain the connection:** This is the final face of the session hijack attack, where the attacker starts communicating with the server using his workstation. The attacker will spoof this client IP address to avoid detection, and will include a sequence number that was predicted earlier. If the server accepts this information, the attacker has successfully hijacked the session. At this point in the attack, full access to the network is limited only by the permissions of the compromised user or computer; providing that the TCP/IP session is maintained, the attacker will not have to repeat the hijack process for the duration of the connection.

TCP Session Hijack

TCP hijacks are meant to intercept the already established TCP session in a client and server network and then pretend to be one of them, finally redirecting the TCP traffic to it by injecting the spoofed IP packet so that the attacker's commands are processed on behalf of the authenticated host of the session. It desynchronizes the session between the actual communicating parties and inserts itself in between.

As authentication is only required at the time of establishing connection, an already established connection can be easily hijacked without going through any sort of authentication or security measures. TCP session hijacks can be implemented in two different ways:

- (i) Man-in-the-middle attack
- (ii) Blind attack

The man-in-the-middle attack is a packet sniffer to intercept communication between client-server networks. Once the attacker reads the TCP header,

he can know the sequence number expected by the server, the acknowledgement number, the port, and the protocol numbers so that the attacker can forge the packet and send it to the server before the client does so.

An alternative approach is by changing the default gateway of the client machine so that it will route its packets via the attacker's machine. This can be done by ARP spoofing. In case of blind attack, the attacker may be able to sniff the packet and guess the correct sequence number expected by the server and implement it.

(i) Hijacking (man-in-the-middle attack)

In a man-in-the-middle attack, the malicious attacker intercepts a conversation between two parties and gains access to the information that the two parties are trying to send to each other. This is a technique that takes advantage of a weakness in the TCP/IP protocol stack and the way headers are constructed. In hijacking, the malicious attacker actively monitors, captures, and controls the message communication between the two parties.

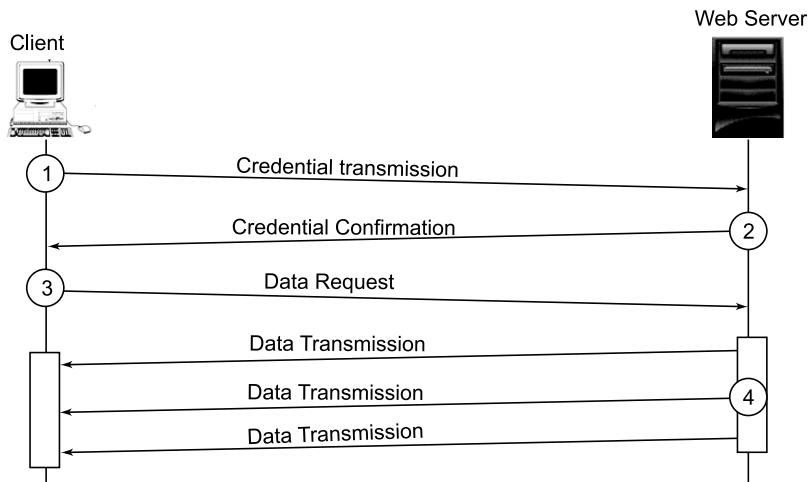


FIGURE 14.6 A normal session.

An MITM attack involves the exploitation of a session between devices; it is also referred to as session hijacking. When we refer to a session, we are talking about a connection between devices in which there is a state. That is, there is an established dialogue in which a connection has been formally set up, the connection is maintained, and a defined process must be used to terminate the connection.

One of the best examples of session hijacking is through cookie stealing, which involves HTTP sessions. When we visit a Website that requires login credentials, that is a great example of a session-oriented connection. It is authenticated by the Website with a username and password to formally set up the session. The Website maintains some form of session tracking to ensure that the user is still logged in and is allowed to access resources (often done with a cookie), and when the session is ending, the credentials are cleared and the session ends. In this example the sessions are occurring constantly and most communications rely on some form of session- or state-based activity.

The principle behind most forms of session hijacking is that if the attacker can intercept certain portions of the session establishment, he can use that data to impersonate one of the parties involved in the communication so that he may access session information. In the example explained previously, this means that if we were to capture the cookie that is used to maintain the session state between your browser and the Website you are logging into, it represents that cookie to the Web server and impersonates the connection as shown in Figure 14.7.

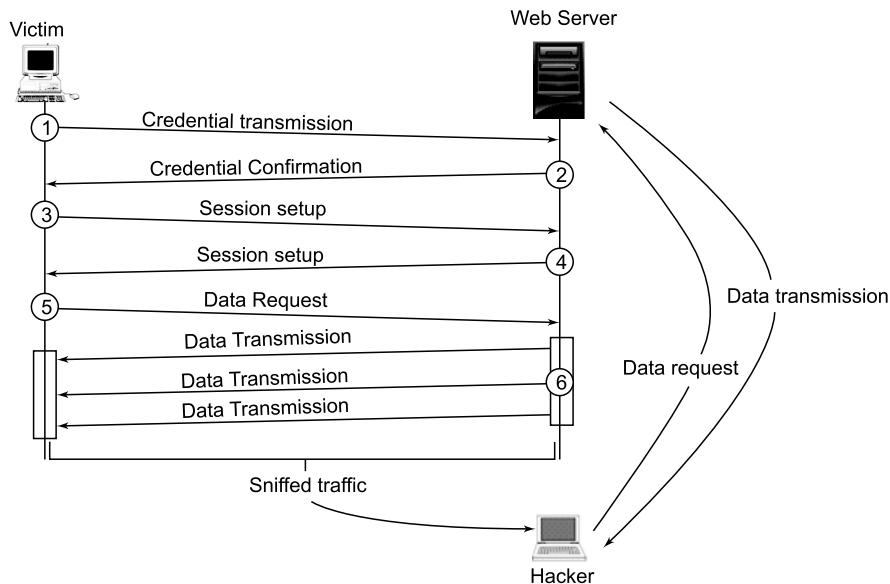


FIGURE 14.7 Session hijacking.

Important concepts of a man-in-the-middle (MITM) attack: MITM is a type of eavesdropping attack that occurs when a malicious actor

inserts himself as a relay/proxy into a communication session between people or systems, as shown in Figure 14.8.

A MITM attack exploits the real-time processing of transactions, conversations, or transfers of other data. MITM allows attackers to intercept, send, and receive data not meant for them without either outside party knowing until it is too late.

In Figure 14.8, the hacker is impersonating both sides of the conversation to gain access to funds. This example holds true for a conversation with a client and server as well as person-to-person conversations. The attacker intercepts a public key and with that can transpose his own credentials to trick the people on either end into believing they are talking to one another securely.

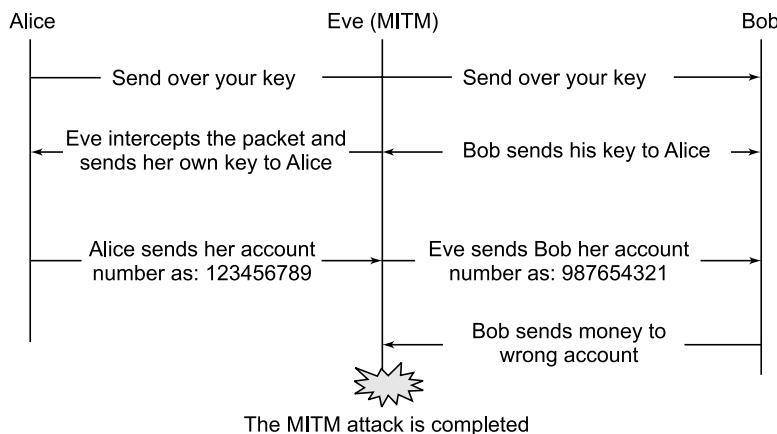


FIGURE 14.8 Man-in-the-middle attack.

(ii) Blind Attack

Detection of session hijack attacks: Session hijacking can be detected by two commonly used techniques, such as packet sniffing software and intrusion detection systems (IDS) and intrusion prevention systems (IPS). The second method provides a more automated method of detection; also it provides additional details to the security administrator for further analysis.

UDP session hijacking: Since UDP does not use packet sequencing and synchronizing, it is easier to hijack a UDP session than a TCP session. The hijacker has simply to forge a server reply to a client UDP request before the server can respond. If sniffing is used, then it will be easier to control the traffic generating from the side of the server, thus restricting the server's reply to the client in the first place.

14.2.5 Trojans

A Trojan horse, often shortened to Trojan, is a type of malware designed to provide unauthorized, remote access to a user's computer. Trojan horses do not have the ability to replicate themselves like viruses; however, they can lead to viruses being installed on a machine, since they allow the computer to be controlled by the Trojan creator. Trojan horses are one of the most common methods a criminal uses to infect the computer and collect personal information from your computer.

The term gets its name from the Greek story of the Trojan War, when the Greeks offered the Trojans a peace offering in the form of a large wooden horse, as in Figure 14.9. However, once the Trojans wheeled the horse behind their closed gates and night fell, the soldiers hidden inside the horse climbed out and opened the city gates, allowing the Greek army to infiltrate Troy and capture the city. Trojan horse software operates the same way, where Troy is your computer and the horse is the benign-seeming application. Trojan horses can assist an attacker in turning a user's computer into a zombie computer, stealing various data such as credit card information, installing more malware, keylogging, and various other malicious activities. Also, it is possible for other crackers to control the compromised computer simply by searching for computers on a network using a port scanner and finding ones that have already been infected with a Trojan horse.



FIGURE 14.9 Trojan horse.

Some of the ways in which the Trojan horse program could get inside a computer is through embedding in an otherwise genuine program, through

e-mail attachments, executable web content such as *ActiveX* controls, and so forth. One of the most notorious Trojan horse programs of recent times was the *Love Bug*, which originated someplace in the Philippines and infected innumerable computer systems around the globe. Actually this horse contained a worm program which caused the damage of nearly 6 billion U.S. dollars, and even organizations such as the CIA and the Pentagon had to shut down their systems temporarily to get rid of it.

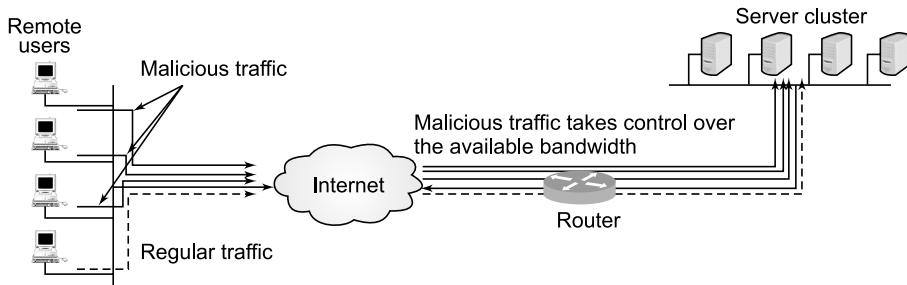
Trojan Horse Attack Mitigation Techniques

Trojan horse attacks can be kept under control by the use of proper precautions such as the usage of proper antivirus software. The following are a few steps which should ensure that attacks are kept at the minimum threat level, even if not totally eliminated:

- A large variety of antivirus software applications are available as stand-alone software or embedded with a large array of applications known as the Internet protection suite. It is important to use some tested and effective antivirus software in order to keep viruses and Trojans at bay.
- Installation of the appropriate software application is necessary, but it is certainly not sufficient in order to keep malicious code at bay. Unless these applications are constantly upgraded on a continuous basis and care is taken to keep them in order, they will not be effective against the latest threats and attacks.
- Deploying appropriate intrusion detection and prevention systems is also effective in warding off such dangers.

14.2.6 Denial-of-Service Attacks (DoS) and Distributed-Denial-of-Service (DDoS)

Denial of service is a very common type of attack. The attacker prevents a server from providing services. The denial may occur at the source by preventing the server from obtaining the resources needed to perform its function. At the destination it is done by blocking communication from the server, as in Figure 14.10, or along the intermediate path by discarding messages from either the client or the server, or both. The DoS results in an infinite delay. It is a very common type of attack. The attacker prevents a server from providing services. One common type of cyber threat by a denial of service attack is, as its name implies, it renders Websites and other online resources unavailable to intended users.

**FIGURE 14.10** DoS attack.

Both DoS and DDoS attacks are used as part of sophisticated organized cybercrime. Cybercriminals use easy-to-employ DoS attacks to distract the victim, draw his attention away, and silently conduct a hidden highly sophisticated attack in the background against key systems or databases.

DoS Types of Attacks

DoS attacks can be divided into two general categories:

1. **Application layer attacks:** It is also known as a layer 7 attack. It can be either DoS or DDoS threats that seek to overload a server by sending a large number of requests requiring resource-intensive handling and processing. Among other attack vectors, this category includes HTTP floods, slow attacks, and DNS query flood attacks.
2. **Network layer attacks:** These types of attacks are also known as layer 3 or layer 4 attacks. Most of these attacks are DDoS assaults set up to clog the “pipelines” connecting the network. Attack vectors in this category include UDP flood, SYN flood, Network Time Protocol (NTP) flood, UDP flood, SYN flood, and NTP amplification attacks.

Any of these can be used to prevent access to the servers, while also causing severe operational damages, such as account suspension and massive overage changes. A DoS attack can be perpetrated in a number of ways; the basic three types are:

1. Consumption of computational resources, such as bandwidth, disk space, or CPU time.
2. Disruption of configuration information, such as routing information.
3. Disruption of physical network components.

The consequences of a DoS attack are the following:

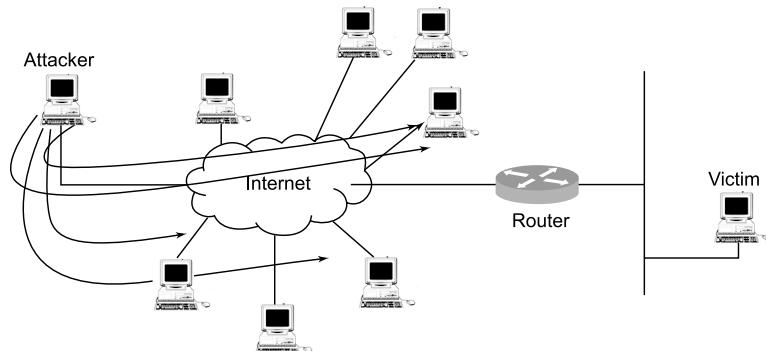
- Unusually slow network performance.
- Unavailability of a particular Website.
- Inability to access any Website.
- Dramatic increase in the amount of spam you receive in your account.

Distributed Denial of Service Attack (DDoS)

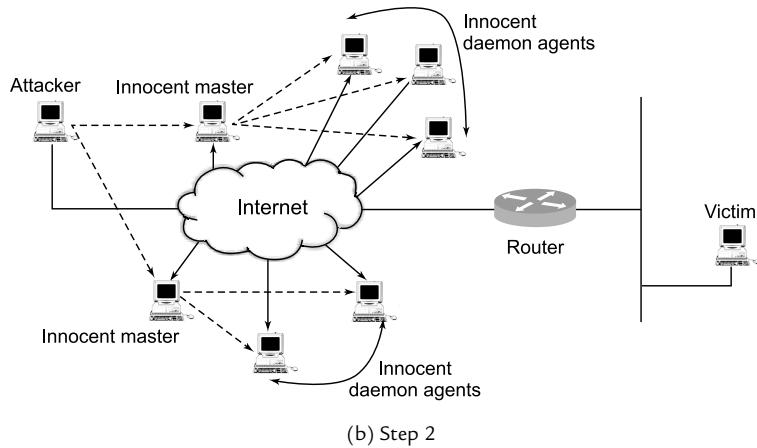
A distributed denial of service attack (DDoS) occurs when multiple compromised systems or multiple attackers flood the bandwidth or resources of a targeted system with useless traffic. These systems are compromised by attackers using a variety of methods.

In DDoS attacks, the attacker first gains access to user accounts on numerous hosts across the Internet. The attacker then installs and runs a slave program at each compromised site that quietly waits for commands from a running master program; the master program then contacts the slave programs, instructing each of them to launch a denial of service attack directed at the same target host. The resulting coordinated attack is particularly devastating, since it comes from so many attacking hosts at the same time. DDOS attacks are explained in the following three steps as in Figure 14.11.

Step 1: Find the vulnerable hosts. The attacker uses reconnaissance tools to locate vulnerable hosts to be used as master and daemons.



(a) Step 1



Step 2: Install software on master and agents. Use master and agent program on all cracked hosts. Create a hierarchical covert control channel using innocent looking ICMP packets whose payload contains DDoS commands. Some DDoS further encrypt the payload.

Step 3: Launch the attack.

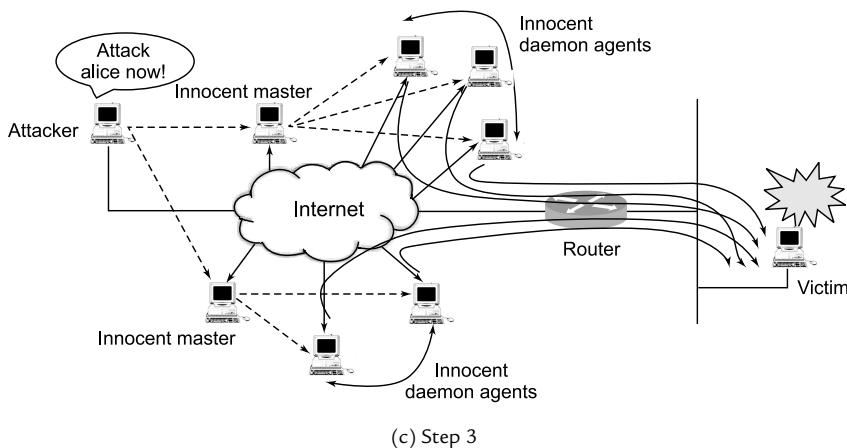


FIGURE 14.11 DDoS attack.

The most infamous DDoS attack consists of sending many requests until the server gets overloaded and fails to respond to the attacker but above all to any legitimate customer. The proliferation of this attack is caused by the fact that it is relatively easy to launch, does not involve many technical skills, can be used against any server, and has a huge impact.

Comparison Between DoS and DDoS

The differences between DoS and DDoS are substantive and worth noting. Some of the important differences between DoS and DDoS are given as follows:

Denial of Service (DoS)	Distributed Denial of Service (DDoS)
<ul style="list-style-type: none"> In a DoS attack, a perpetrator uses a single Internet connection to either exploit a software vulnerability or flood a target with fake requests—usually in an attempt to exhaust server resources (e.g., RAM and CPU). 	<ul style="list-style-type: none"> DDoS attacks are launched from multiple connected devices that are distributed across the Internet. These multi-person, multi-device barrages are generally harder to deflect, mostly due to the sheer volume of devices involved.
<ul style="list-style-type: none"> A DoS attack is generated from a single source. 	<ul style="list-style-type: none"> DDoS assaults tend to target the network infrastructure in an attempt to saturate it with huge volumes of traffic.
<ul style="list-style-type: none"> DoS attacks are launched using homebrewed scripts or DoS tools (e.g., Low Orbit Ion Canon). 	<ul style="list-style-type: none"> DDoS attacks also differ in the manner of their execution. They are launched from botnets — large clusters of connected devices (e.g., cell phones, PCs, or routers) infected with malware that allow remote control by an attacker.

Denial of service attack types: Common forms of denial of service attacks are: buffer overflow attack, Smurf attack, and SYN flood attack.

(a) Buffer overflow attack

Buffer overflow attacks have been the most common form of security vulnerability. It is a programming error that throws a memory access exception. It occurs when a process attempts to store data beyond the boundaries of a fixed length buffer, overwriting adjacent locations of memory, including some programs “flow data” and causing the process termination with a segmentation fault error.

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold. One of the reasons for buffer overflow is due to poorly constructed software programs. These programs may have multiple deficiencies such as stack overflow, heap corruption format string bugs, and race conditions. These attacks are also referred to as simply buffer overflow. A buffer is a contiguous allocated chunk of memory, such as an array or a pointer in the C programming language.

For example:

```
Int main ( )
{
    Int buffer [10];
    Buffer [20] = [10];
}
```

In this program snippet, the program attempts to write beyond the allocated memory for the buffer which might result in unexpected behavior.

Types of Buffer Overflow

Buffer overflow can be divided into two categories as *stack overflow* and *heap overflow*. The stack is a region in a program memory space that is only accessible from the top. There are two operations, *PUSH* and *POP*, for a stack. The process of inserting an element into a stack is known as a *PUSH* operation and extracting the content from the stack is *POP*.

In order to insert an element into the stack, first we have to check whether free space is available in the stack or not. If the stack is full, we cannot not insert an element into the stack. If the value of the TOP variable is greater than or equal to SIZE-1, then the stack is full. This condition is known as *overflow*. If the stack is not full, then we can insert an element into the stack using the following steps. First we have to increment the value of variable TOP by one and then insert an element into the stack.

Step 1: IF TOP > = SIZE -1

THEN

Write “stack overflow”

Step 2: TOP = TOP + 1

Step 3: STACK [TOP] = X

In order to remove an element, first we have to check whether the stack is empty or not. If the stack is empty, we cannot remove any element from the stack. This condition is known as *underflow*. If the value of the TOP variable is -1, then the stack is empty, so we cannot remove an element from the stack.

After removing the topmost element from the stack, we have to decrement the value of the TOP by one so that it can point to the next topmost element in the stack.

Step1: IF TOP = -1

THEN

Write “Stack is Underflow”

Step 2: Return Stack [TOP]

Step 3: TOP = TOP-1

Every process has its own memory space as a stack region and a heap region. The stack is used heavily to store local variables and return addresses of a function.

For example, assume that we have a function:

Void foo(constant char* input)

```
{
    Char buf[10];
    Print ("Hello world\n");
}
```

When the function is called from another function, for example, main,

```
int main(int argc, char* argv[ ])
{
    foo(argv[1]);
    return 0;
}
```

When the code is executed, the calling function pushes the return address that is the address of the return statement onto the stack. Then the called function pushes zeroes on the stack to store its local variable. Since foo has a variable buf, this means there will be space allowed for 10 characters. The stack will look like Figure 14.12.

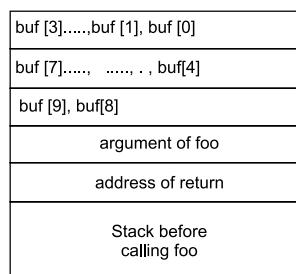


FIGURE 14.12 Buffer allocation for 10 characters.

The stack holds the return address, the arguments, and the local variable.

Inject Attack Code

The attacker provides an input string that is actually executable binary code native to the machine being attacked. Typically this code is simple and does something similar to the exec("sh") to produce a root shell.

Change the return address: There is a stack frame for a currently active function above the buffer being attacked on the stack. The buffer overflow changes the return address to point to the attack code. When the function returns, instead of jumping back to where it was called from, it jumps to the attack code. The programs that are attacked using this technique are usually privileged daemon programs that run under the user ID of "root" to perform some service. The injected attack code is usually a short sequence of instructions that spawns a shell, also under the user ID of "root." The effect is to give the attacker a shell with root privileges.

If the input to the program is provided from a local running process, then this class of vulnerability may allow any user with a local account to become "root." If the program input comes from a network connection, this class of vulnerability may allow any user anywhere on the network the ability to become root on the local host.

The general technique to achieve this is as follows:

- The location of the return address can be approximated by simply repeating the desired return address several times in the approximate region of the return address.
- The offset to the attack code can be approximated by prepending the attack code with an arbitrary number of "NOP" instructions. The overwritten return address need only jump onto the field of "NOP" to hit the target.

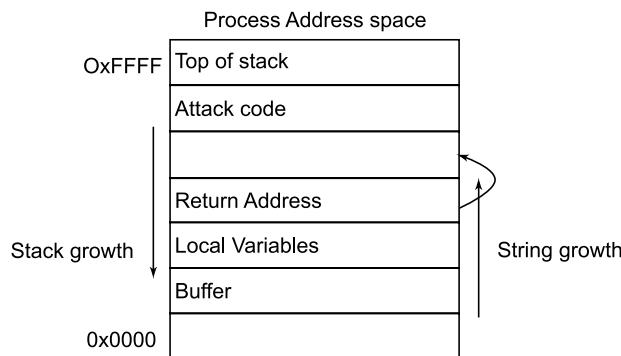


FIGURE 14.13 Stack smashing buffer overflow attack.

Buffer overflow is probably the best known form of software security vulnerability. A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code.

In a classic buffer overflow attack, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function's return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to the malicious code contained in the attacker's data.

Although this type of stack buffer overflow is still common on some platforms, there are other types of buffer overflow attacks, which include *heap buffer overflow* and *format string attack*.

At the code level, buffer overflow vulnerabilities usually involve the violation of a programmer's assumptions. Many memory manipulation functions in C and C++ do not perform bounds checking and can easily overwrite the allocated bounds of the buffers they operate upon. Even bounded functions, such as `strncpy()`, can cause vulnerabilities when used incorrectly. The combination of memory manipulation and mistaken assumptions about the size or makeup of a piece of data is the root cause of most buffer overflows.

Buffer Overflow and Web Applications

Attackers use buffer overflows to corrupt the execution stack of a Web application. By sending carefully crafted input to a Web application, an attacker can cause the Web application to execute arbitrary code — effectively taking over the machine.

Buffer overflow flaws can be present in both the Web server or application server products that serve the static and dynamic aspects of the site, or the Web application itself. Buffer overflows found in widely used server products are likely to become widely known and can pose a significant risk to users of these products. When Web applications use libraries such as a graphics library to generate images, they open themselves to potential buffer overflow attacks.

Buffer overflows can also be found in custom Web application code, and may even be more likely given the lack of scrutiny that Web applications typically go through. Buffer overflow flaws in custom Web applications are less likely to be detected because there will normally be far fewer hackers trying to find and exploit such flaws in a specific application. If discovered in a custom application, the ability to exploit the flaw (other than to crash

the application) is significantly reduced by the fact that the source code and detailed error messages for the application are normally not available to the hacker.

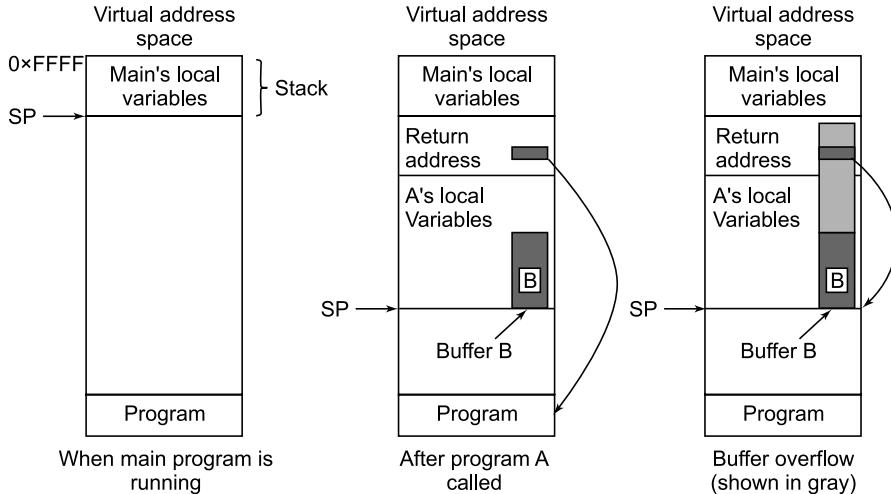


FIGURE 14.14 Buffer overflow.

Consequences

- **Availability:** Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.
- **Access control (instruction processing):** Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy.
- **Other:** When the consequence is arbitrary code execution, this can often be used to subvert any other security service.

(b) Smurf attack

A Smurf attack is a form of distributed denial of service attack that renders computer networks inoperable. In a Smurf attack, an attacker creates lots of ICMP packets with the intended victim's IP address as source IP and broadcasts those packets in a computer network using an IP broadcast address.

The Smurf attack process is usually explained in the following five steps:

1. Hacker identifies a victim IP address.
2. Hacker identifies an intermediate site that will amplify the attack.
3. Hacker sends a large amount of ICMP traffic at the broadcast address of the intermediary site. These packets have the source IP address spoofed to point toward the victim.
4. Intermediaries deliver the broadcast at layer 2 to all the hosts on their subnet.
5. Hosts reply to the victim network.

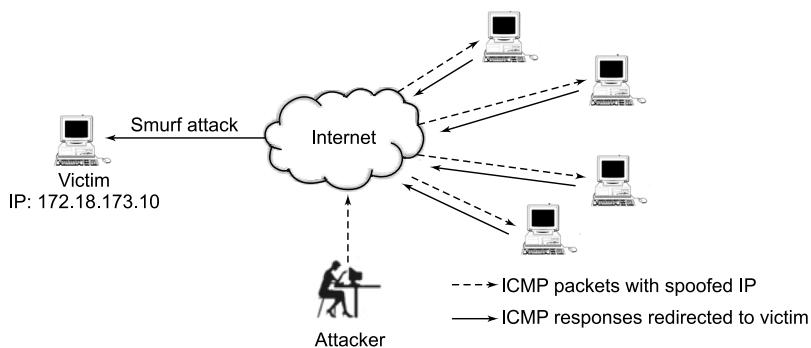


FIGURE 14.15 Smurf attack.

In this attack, the attacker sends an IP ping request to a receiving site. The ping packet specifies that it has been broadcast to a number of hosts within the receiving site's local network. The packet also indicates that the request is from another site, which is the target site that is to receive the denial of service attack. The result will be lots of ping replies flooding back to the innocent, spoofed host as in Figure 14.15. If the flood is great enough, the spoofed host will no longer be able to receive or distinguish real traffic.

(c) SYN floods

In order to understand a SYN flood attack, we first need to understand the TCP/IP handshake. Normally when a client attempts to start a TCP connection to a server, the client and server exchange a series of messages which normally run like the three steps mentioned as follows:

1. The client requests a connection by sending a SYN (synchronize) message to the server.

2. The server acknowledges this request by sending synchronization acknowledged (SYN-ACK) back to the client.
3. The client responds with an ACK, and the connection is established. This is called the *TCP three-way handshake*, as shown in Figure 14.16, and is the foundation for every connection established using the TCP protocol.

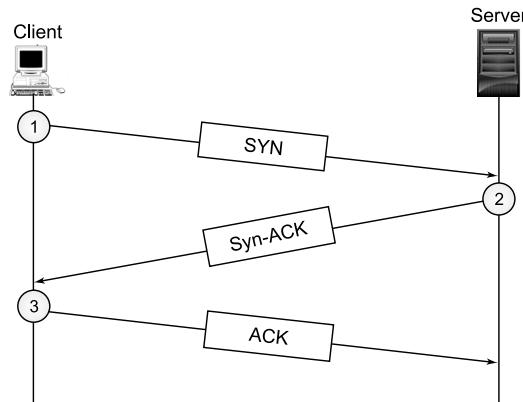


FIGURE 14.16 Three-way handshake.

SYN Flood

A SYN flood DDoS attack exploits a known weakness in the TCP connection sequence (as explained in the three-way handshake) wherein a SYN request to initiate a TCP connection with a host must be answered by a SYN-ACK response from that host, and then confirmed by an ACK response from the requester. In a SYN flood scenario, the requester sends multiple SYN requests, but either does not respond to the host's SYN-ACK response, or sends the SYN requests from a spoofed IP address. Either way, the host system continues to wait for acknowledgement for each of the requests, binding resources until no new connections can be made, and ultimately resulting in denial of service. The targeted server keeps each of these false connections open. This eventually overflows the maximum concurrent connection pool, and leads to denial of additional connections from legitimate clients.

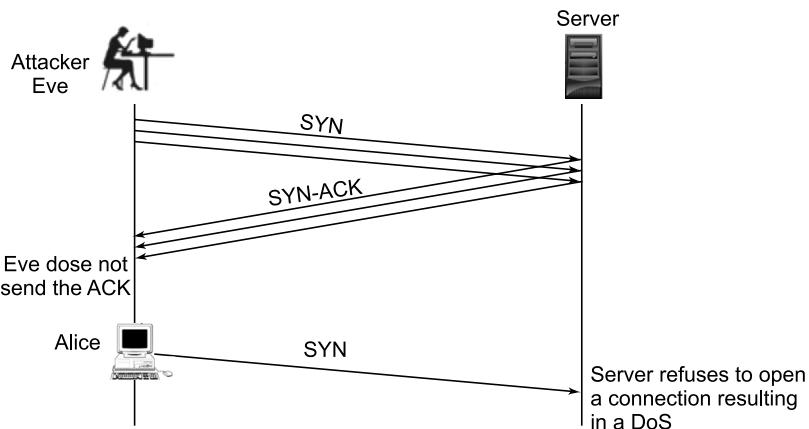


FIGURE 14.17 SYN flood.

In Figure 14.17, the attacker (Eve) sends several SYN packets but does not send the “ACK” back to the server for the SYN-ACK response. The connections are hence half-opened and consuming server resources. Alice, a legitimate user, tries to connect but the server refuses to open a connection, resulting in a denial of service.

SYN flooding is a method that the user of a hostile client program can use to conduct a DoS attack on a computer server. The hostile client repeatedly sends SYN packets to every port on the server, using fake IP addresses. When an attack begins, the server sees the equivalent of multiple attempts to establish communications. The server responds to each attempt with a SYN-ACK packet from each open port, and with a RST (reset) packet from each closed port. In a normal three-way handshake, the client would return an ACK packet to confirm that the server’s SYN-ACK packet was received, and communications would then commence. However, in a SYN flood, the ACK packet is never sent back by the hostile client. Instead, the hostile client program sends repeated SYN requests to all the server’s ports. The hostile client makes the SYN requests all appear valid, but because the IP addresses are fake ones, it is impossible for the server to close down the connection by sending RST packets back to the hostile client. Instead, the connection stays open. Before a time-out can occur, another SYN packet arrives from the hostile client. A connection of this type is called a *half-open connection*. Under these conditions, the server becomes completely or almost completely busy with the hostile client. Communications with legitimate clients is difficult or impossible. A hostile client can exploit half-open connections and possibly

get access to server files. The transmission by a hostile client of SYN packets for the purpose of finding open ports and hacking into one or more of them is called *SYN scanning*. A hostile client always knows a port is open when the server responds with a SYN-ACK packet.

Social Engineering

Intruders are always on the lookout for ways to gain access to valuable resources such as computer systems or corporate or personal information on them that can be used maliciously for the attackers' personal gain. Social engineering is the use of dishonesty to gain access to information systems. The medium is usually a telephone or e-mail message. The attacker usually pretends to be a director or manager in the company traveling on business with a deadline to get some important data left on their network drive. They pressure the help desk to give them the toll-free number of the RAS server to dial and sometimes get their password reset. The main purpose behind social engineering is to place the human element in the network-breaching loop and use it as a weapon. The human element has been referred to as the weakest link in network security.

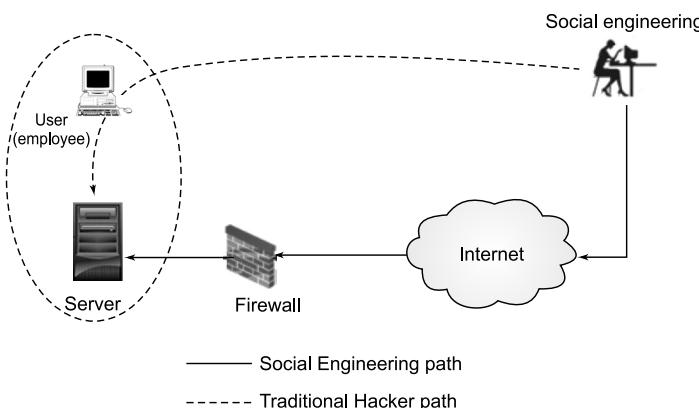


FIGURE 14.18 Social engineering.

- It includes extensive research information (legal and illicit) about an enterprise, which is gathered and used to exploit people.
- Successful social engineering results in partial or complete circumvention of an enterprise's security systems. The best firewall is useless if the person behind it gives away either the access code or the information it is installed to protect.

- Social engineering principally involves the manipulation of people rather than technology to breach security.
- Different types of social engineering attacks are discussed in Chapter 11.

Password-Based Attacks

A common denominator of most operating system and network security plans is password-based access control. This means the access rights to your computer and network resources are determined by who you are, that is, your user name and your password. Older applications do not always protect identity information as it is passed through the network for validation. This might allow an eavesdropper to gain access to the network by posing as a valid user. When an attacker finds a valid user account, the attacker has the same rights as the real user. Therefore, if the user has administrator-level rights, the attacker also can create accounts for subsequent access at a later time. After gaining access to the network with a valid account, an attacker can do any of the following:

- Obtain lists of valid user and computer names and network information.
- Modify server and network configurations, including access controls and routing tables.
- Modify, reroute, or delete your data.

Compromised-Key Attack

A key is a secret code or number necessary to interpret secured information. Although obtaining a key is a difficult and resource-intensive process for an attacker, it is possible. After an attacker obtains a key, that key is referred to as a *compromised key*.

An attacker uses the compromised key to gain access to a secured communication without the sender or receiver being aware of the attack. With the compromised key, the attacker can decrypt or modify data and try to use the compromised key to compute additional keys, which might allow the attacker access to other secured communications.

Application-Layer Attack

An application-layer attack targets application servers by deliberately causing a fault in a server's operating system or applications. This results in the attacker gaining the ability to bypass normal access controls. The attacker takes advantage of this situation, gaining control of user applications and systems. The attacker may also:

- Read, add, delete, or modify the data or operating system.
- Introduce a virus program that uses the computers and software applications to copy viruses throughout the network.
- Introduce a sniffer program to analyze the network and gain information that can eventually be used to crash or to corrupt systems and networks.
- Abnormally terminate the data applications or operating systems.
- Disable other security controls to enable future attacks.

FTP Bounce

An FTP bounce attack is a legacy attack that will not work well on the FTP software. It uses the port command to indirectly request access through a victim's machine. Once in a port, an attacker can gain information or else disrupt network communication.

Network Security Model (NSM)

A network security model is a seven-layer model that divides the daunting task of securing a network infrastructure into seven manageable sections. When an attack on a network has succeeded, it is much easier to locate the underlying issue and fix it with the use of an NSM. A general NSM which follows a top-down approach is shown in Figure 14.19.

1	Physical Layer
2	Virtual Local Area Network (VLAN) Layer
3	Access Control List (ACL) Layer
4	Software Layer
5	User Layer
6	Administrative Layer
7	IT Department Layer

FIGURE 14.19 Network security model.

If a failure in any layer is found, it also determines that all of the layers above this layer have also failed. The security professional will be able to quickly determine if other possible computers have been compromised with the breach of the layer and how to secure it against the same attack in the future. A well structured NSM will give the security community a way to

study, implement, and maintain a network in a way that can be applied to any network.

NSM Seven-Layer Model

1. **Physical layer:** This layer's primary focus is on physical security. Physical security is applied to prevent attackers from accessing a facility to gain data stored on server computers or other sources. Physical security is the first chosen layer, because it is a breaking point for any network. If the physical layer security is attacked, providing other security devices like firewalls will not help to provide the required security. From this it is clear that if the layer below the physical layer fails, the physical layer has failed as well, because the attacker would be able to manipulate data as if they had breached the facility. The different forms of physical security systems that can be planned and implemented are:
 - a. Site selection and design
 - b. Implementing security devices like fencing, barbed wires, warning signs, metal and concrete barriers, flood lights
 - c. Access control devices including gates, door locks (mechanical or electronic)
 - d. Security alarms
 - e. Camera

Hiring a security guard is the only form of physical security that can be considered both an access control and monitoring measure.

2. **Virtual Local Area Network (VLAN) layer:** Creation and maintenance of VLAN segments the network into multiple networks. The purpose of VLAN is to group together common hosts to meet security requirements. The security level and access policies for different VLANs are decided based on their functionalities. For the implementation of VLAN first we have to determine whether the network is public or private. Public networks are those networks with connections to the external network, which includes Web servers, external FTP servers, and external DNS servers. The internal networks are the networks within the organization with internally connected network devices. Break down the internal user and server VLANs based on department access permission and data grouping.

3. Access Control List (ACL) layer: ACLs are network filters utilized by the routers and firewalls to allow and restrict dataflow into and out of the network interface. When the ACL is configured on an interface, the network device analyzes data passing through the interface, compares it to certain data described in the ACL, and either permits the data to follow or prohibits it. This makes it absolutely indispensable in the area of network security. Creating and maintaining a strong ACL, a network security professional can stop many attacks before they begin—for a security professional, placement of his defense is critical to protecting the network, its assets, and its data.

Implementation of ACL security: ACLs should be placed on external routers to filter traffic against less desirable networks and known vulnerable protocols. One of the most commonly used architectures implemented with two separate network devices is as shown in Figure 14.20.

The most exterior router provides access to all outside network connections. This router usually has a less restrictive access control list. It also provides large protection access blocks to areas of the global routing table that the security professional wishes to restrict. This router also protects against well-known protocols that are not allowed access into or out of the network. They will be configured to allow or block basic Web activity related to port 80 (HTTP), port 443 (HTTPS), and port 53 (DNS). The internal router in the network contains more restrictive ACLs designed to protect the internal network from more defined threats. ACLs here are often configured with explicit permit and deny statements for specific addresses and protocol services.

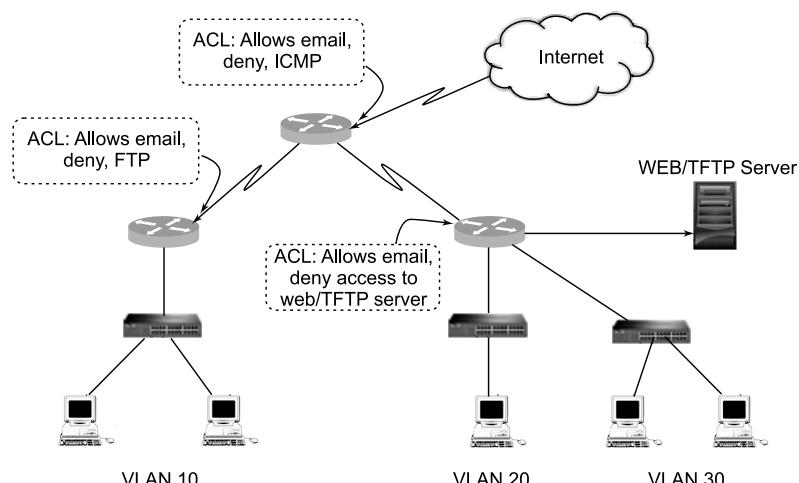


FIGURE 14.20 Access control list.

4. **Software layer:** The software layer is an important layer in the NSM, because if the software layer is compromised, then an attacker can potentially get access to that host. The function of the software layer is to maintain and keep the software up to date with upgrades in order to mitigate software related vulnerabilities. Network security professionals remove any unwanted software and also know what vulnerabilities currently exist or have existed recently. They should also implement patches and upgrades. This reduces the amount of exploits and vulnerabilities.
5. **User layer:** The function of this layer is mainly user's training and creating an awareness about network security. Training with network security can prevent users from potentially compromising a host.
6. **Administrative layer:** The administrative layer deals with the roles of the administrative members in an organization. The members should be trained to identify problems and different applications to be installed in the host to provide the required security.
7. **IT department layer:** The IT department layer contains all of the network security professionals, network technicians, architects, and support specialists. The function of the IT department layer is like the administrative layer, except the IT department has accounts to access any device on the network. For example, an IT department user can have read, write, and modify access to a database table structure, where an administrator or user only has read, write, and modify access to records within the table structure. The IT department is responsible for the implementation and maintenance of all network layers, including the physical layer, VLAN layer, ACL layer, software layer, user layer, and the administrative layer. A mapping between the NSM and OSI model is given in Table 14.1.

TABLE 14.1 Mapping the Functions of the Network Security Model and the OSI Model.

NSM	OSI Model	Remark
Physical	Physical	Deals with physical security of the resources
VLAN	Data link	Deals with MAC addressing and VLANs
ACL	Network	Deals with IP addressing and LANs
Software	Transport	Deals with actual connections on the network from host to host
User	Session	Deals with utilization of the local machine

(continued)

NSM	OSI Model	Remark
Administrative	Presentation	Deals with administration functions
IT Department	Application	Deals with the maintenance of all layers to make sure that the entire network works correctly from the NSM mode and all layers of the OSI model

SUMMARY

- When considering the network security, it must be emphasized that the whole network is secure.
- For developing a secure network model, access, confidentiality, authentication, integrity, and non-repudiation need to be considered.
- Spoofing means have the address of the computer copy the address of a trusted computer in order to gain access to the other computer.
- Packet sniffing is the interception of data packets traversing a network.
- Before attacking a network, the attacker would like to know the IP address of machines on the network, the operating systems they use, and the services that they offer. With this information their attacks can be more focused and are less likely to cause alarm. The process of gathering this information is known as mapping.
- Session hijacking is a means of either gaining total or partial control over an established TCP/IP connection.
- TCP hijacks are meant to intercept the already established TCP session in a client and server network and pretend to be one of them, finally redirecting the TCP traffic to it by injecting a spoofed IP packet so that the attacker's commands are processed on behalf of the authenticated host of the session.
- A Trojan is a type of malware designed to provide unauthorized remote access to a user's computer.
- In DoS, the attacker prevents a server from providing services. DDoS occurs when multiple compromised systems or multiple attackers flood the bandwidth or resources of a targeted system with useless traffic.
- Buffer overflow is the most common form of security vulnerability. It is a programming error that throws memory access exceptions.
- Social engineering is the use of dishonesty to gain access to an information system.
- The network security model (NSM) is a seven-layer model that divides the daunting task of securing a network infrastructure into seven manageable sections.

REVIEW QUESTIONS

1. List the different types of attacks on networks.
 2. Write a note on network vulnerability.
 3. What is meant by a man-in-the-middle attack?
 4. What do you mean by a Trojan horse?
 5. Discuss a three-way security threat.
 6. How does DoS differ from DDoS? Which is potentially more dangerous and devastating? Why?
 7. What is social engineering?
 8. What are packet sniffers? Name two of the well-known packet sniffers.
 9. Explain the important aspects of packet sniffers and their use.
 10. What is IP Spoofing? Explain the different types of IP spoofing.
 11. Explain session hijacking.
 12. Explain a buffer overflow attack.
 13. Discuss synchronous flooding.
 14. What is a network security model (NSM)? Explain the functions of different layers of the NSM.
 15. What is access control? Briefly enumerate the types of access controls.

MULTIPLE CHOICE QUESTIONS

1. Which of the following is an independent malicious program that need not have any host program?
 - (a) Trap doors
 - (b) Trojan horse
 - (c) Virus
 - (d) Worm
 2. What is malware?
 - (a) A virus or worm
 - (b) A Trojan horse
 - (c) A hacker tool
 - (d) A corrupted program

3. Which of the following is a type of program that either pretends to have, or is described as having, a set of useful or desirable features, but actually contains damaging code.

(a) Trojans (b) Viruses (c) Worms
(d) Adware (e) Bots

4. Which of the following is the type of software that has self-replicating software that causes damage to files and systems?

(a) Viruses (b) Trojan horses
(c) Bots (d) Worms

5. Which of the following is a program capable of continually replicating with little or no user intervention?

(a) Virus (b) Trojan horses
(c) Rootkit (d) Worms

6. Which of the following is a software that, once installed on your computer, tracks your Internet browsing habits and sends you popups containing advertisements related to the sites and topics you've visited?

(a) Backdoors (b) Adware
(c) Malware (d) Bots

7. What is the software called that's designed to exploit a computer user and is a broad term covering computer viruses, worms, Trojans, adware, etc.?

(a) Backdoors (b) Key-logger
(c) Malware (d) Spyware

8. What is the software called which when it gets downloaded on a computer scans your hard drive for personal information and your Internet browsing habits?

(a) Key-logger (b) Malware
(c) Antiware (d) Spyware

CHAPTER 15

APPLICATION LAYER SECURITY

15.1 INTRODUCTION

The application layer, which accommodates the user interface and other key functions, is the closest layer of the OSI model to the user end. This layer provides the hacker with the widest attack surface. When exploited, the entire application can be manipulated, user data can be stolen, or in some cases the network can be shut down completely (denial of service attack). For the purpose of information security, the application layer can be considered as the realm where user interaction is obtained and high-level functions operate above the network layer. These high level functions access the network from either a client or server perspective, with peer-based systems filling both functions simultaneously. Poor application code integrity and design flaws can cause a wide range of problems—from performance/stability issues to application layer vulnerabilities that can be exploited by hackers. Traditional security methodologies are no longer effective as stand-alone solutions due to their inherited deficiencies.

In the application layer, end-to-end security is provided at a user level by encryption of applications at client workstations and server hosts. The different application layer protocols and application layer security mechanisms like PEM (privacy enhanced mail), S-MIME (secure-multipurpose internet mail extensions), and PGP (pretty good privacy) are discussed in this chapter.

15.2 E-MAIL SECURITY

E-mail has become one of the most widely used Internet applications. E-mail message communication that takes place on the Internet is vulnerable to many security threats. The needs of e-mail security are based on e-mail access modes, e-mail infrastructure, and e-mail protocols.

15.2.1 Mail Access Modes

To reach its final destination the message should be handled by an Internet mail server. An Internet mail server is the software responsible for transmitting and receiving e-mail across the Internet. Some mail servers store mail only until the user retrieves it, whereas the others store user mail permanently. An e-mail user typically uses a mail client program to interact with the mail server. The communication between the mail client and mail server is regulated by the mail access protocol, a standardized set of transmitted commands and responses sent over many different types of network connections. These mail access protocols operate in three common modes.

1. **Offline mode:** In this mode the e-mail message is downloaded from temporary storage on the mail server to the user computer. After download, the mail is deleted from the server.
2. **Online mode:** A user's e-mail, his or her inbox, and all filed mail remain permanently on the mail server. By connecting to the server and establishing an e-mail session, the user can download a temporary copy of his or her e-mail and read it, or send e-mail. Once the connection is terminated, the copy is erased from the user's computer and only the original mail remains on the server.
3. **Disconnected or resynchronization mode:** It is the combination of both offline and online modes. A copy of the user's e-mail is downloaded to his or her computer, and the original message remains on the mail server. The user can change a local copy of his or her e-mail on any computer then synchronize all copies, including the original e-mail message on the server and copies on additional computers.

15.2.2 E-Mail Infrastructure

The simplest way of sending an e-mail would be by sending a message directly from the sender's computer to the recipient's computer. To perform this it is essential for both computers to be running on the network simultaneously.

However, this setup is impractical, as users may occasionally connect their computers to the network to check their mail messages in the inbox or to send the mail. Hence, it is essential to set up the e-mail server. In this setup the mail server maintains a database. The mail sent to the mail server is permanently available on the Internet. When the recipient's computer connects to the network, it reads the mail from the mail server.

In general the e-mail infrastructure consists of a mesh connection of mail servers also called message transfer agents (MTA) and client computers running an e-mail server comprising user agents (UA) and local MTAs as in Figure 15.1. As the user sends an e-mail the message gets forwarded from the UA, goes through the mesh of MTA, and finally reaches the UA on the recipient's computer.

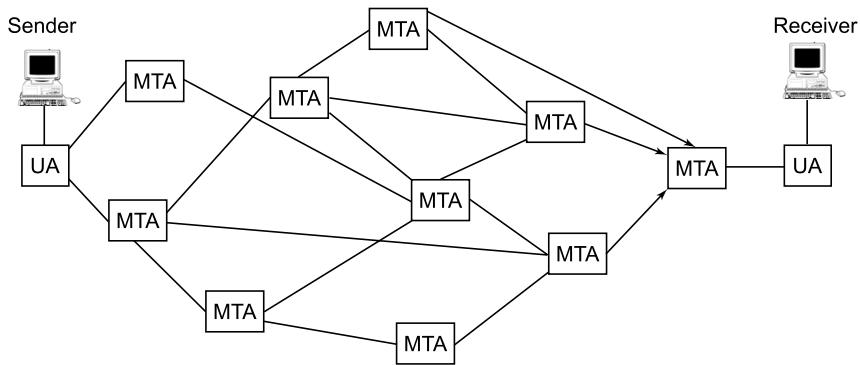


FIGURE 15.1 E-mail infrastructure.

15.2.3 E-mail Protocols

The popularly used protocols for e-mail are:

1. Simple mail transfer protocol (SMTP)
 2. Post office protocol (PoP)
 3. Internet message access protocol (IMPA)

Simple Mail Transfer Protocol (SMTP)

SMTP is one of the basic protocols for transmitting messages between computers connected on the Internet. It describes the allowed sequence of control messages passed between two computers to transfer a mail message.

There are provisions for verifying that the two computers know that they are correctly connected for identifying the message sender, for negotiating a set of recipients, and for forwarding an e-mail message.

SMTP protocol is a mechanism for communication between two MTAs across a single TCP connection. The RFC 822 standard specifies the format of the e-mail message that is transmitted using the SMTP protocol between the two MTAs. As a result of this the sender requests his SMTP to establish a two-way connection with a receiver SMTP. The receiver SMTP can be either the ultimate destination or an intermediate one. The sender SMTP will generate commands, which are replied to by the receiver SMTP as in Figure 15.2. Both the SMTP client and server should have two basic components, a UA and a local MTA.

In the SMTP message communication process the sender UA prepares the message, creates the envelope, and packs message in the envelope. The MTA transfers the mail across the network to TCP port 25 of the receiver MTA.

User Agent (UA) is a user interface application program. The user reads and composes mail with a user interface application program. It provides extensive mail management functions to the users. These functions are not required or even defined in the protocol specification. Nonetheless most users have available a mailbox that collects incoming messages along with functions that aid in reviewing the contents of the mailbox.

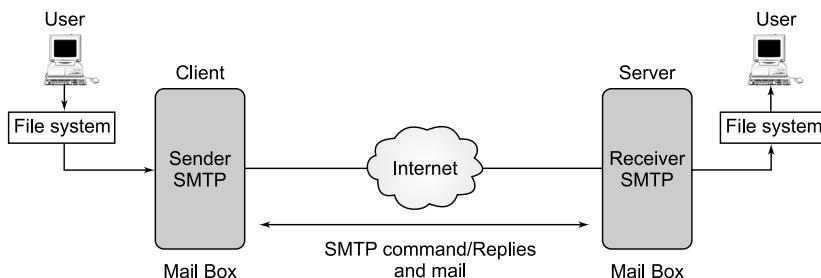


FIGURE 15.2 Model for SMTP.

The user agents for Windows include Microsoft Outlook or Outlook Express and Netscape or Mozilla Communicator. The most common MTA for UNIX is Sendmail, and MTA for Windows is Microsoft Exchange 2000/2003. In addition to stable host based e-mail servers, Microsoft Corporation has developed LDAP or Active Directory servers and B2B servers that enhance mail delivery practices. Users normally do not deal with the MTA.

POP3

Post Office Protocol (POP) is the oldest and most recognized Internet e-mail protocol. Its current widespread implementation is POP3. POP was built around the off-line mail delivery model. This means that the end user connects to an e-mail server, then reads e-mail while off-line. In other words POP was designed to collect mail for a single e-mail client. POP messages are stored on the mail server until it is downloaded to the client. They are then stored on the client computer and deleted from the server. POP2 (RFC937) became a standard in the mid-1980s and requires SMTP to send messages. The newer version POP3 (RFC 1723) can be used with or without SMTP.

Benefits of POP

1. Local storage: When not connected, the user can still access and read downloaded e-mail.
2. Server saving: POP frees server storage space, because it downloads e-mails and attachments then deletes them from the server.
3. Legacy systems: There are almost no interoperability issues between the POP server and mail client; the user can use any POP mail client with any POP server. All ISPs support and use POP.

Limitations of POP

1. POP is primarily designed for use with a single e-mail client on a single computer.
2. When implemented the leave-mail-on server feature forces the downloading of the same e-mail multiple times. This consumes more bandwidth, server resources, and client storage space.

Internet Message Access Protocol (IMAP)

IMAP is the most recent e-mail protocol for retrieving e-mail messages. Its current implementation is IMAP4. It is similar to POP3 but supports some additional features. By using IMAP4, the user can search through his or her e-mail messages for keywords while the messages are still on the mail server. The user can then choose which messages to download to his or her computer.

IMAP uses SMTP for a transport mechanism. In this the IMAP server acts as a post office, whereas SMTP acts like the postal carriers. For a reliable data transmission IMAP also uses TCP.

Benefits of IMAP

1. Multiple client support: messages can be viewed on any computer with an IMAP client.
2. Public and group folders: Because they are on the server everyone can see and use them.
3. Confidentiality: It has multiple options based on all three main delivery modes.

Limitations of IMAP

1. Too many options.
2. It is server intensive. Consumes more server CPU time and storage resources.

15.3 E-MAIL SECURITY SERVICE

Using e-mail across the Internet or other untrusted network imposes security risks. These risks must be analyzed to ensure that proper e-mail security services are implemented to minimize these risks. Certain fundamental security services are:

- **Confidentiality:** Confidentiality is associated with sending e-mail to another person through the Internet. This e-mail passes through many systems before it reaches the intended recipients. If the message is not encrypted a hacker can intercept and read the e-mail at any point along the delivery route.
- **Authentication:** An e-mail recipient can be sure of the identity of the sender of the e-mail. E-mail authentication helps to determine if something is spam or is worth blocking completely to protect users.
- **Integrity:** Assures the recipient that the e-mail message has not been altered since it was transmitted by the sender.
- **Non-repudiation of origin:** The e-mail receiver must be given a way to prove that a given e-mail indeed originated from the specified sender.
- **Non-repudiation of recipient:** The sender must have a way to prove that the intended recipient indeed received the e-mail.

E-mail security services such as confidentiality, authentication, integrity, and non-repudiation are usually provided by using public key cryptography.

For the implementation of security services, consider the three different scenarios of e-mail communication as:

- (i) One-to-one e-mail
- (ii) One-to-multiple recipient e-mail
- (iii) One-to-distribution list of e-mail

15.3.1 One-to-One E-Mail

In this scenario of e-mail communication, the sender forwards an e-mail message to only one recipient. This is accomplished by using two MTAs as shown in Figure 15.3.

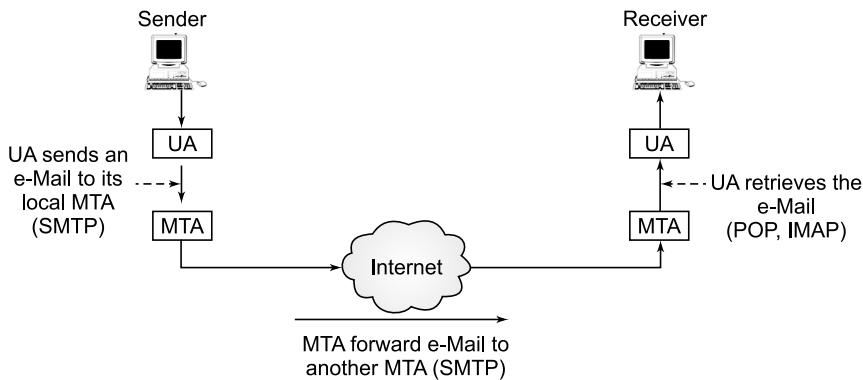
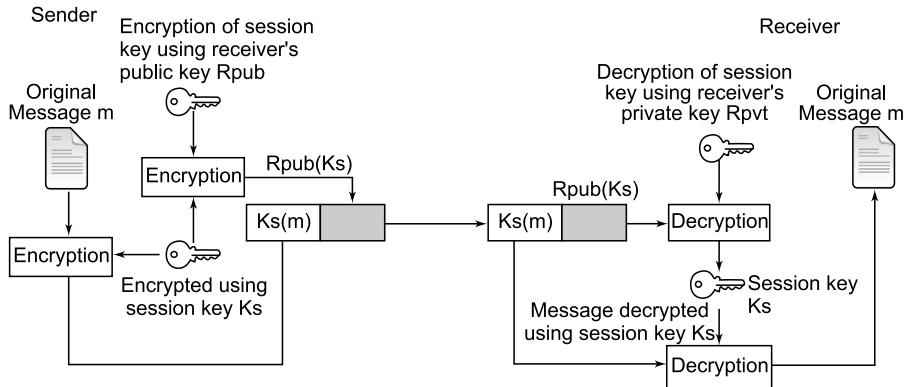


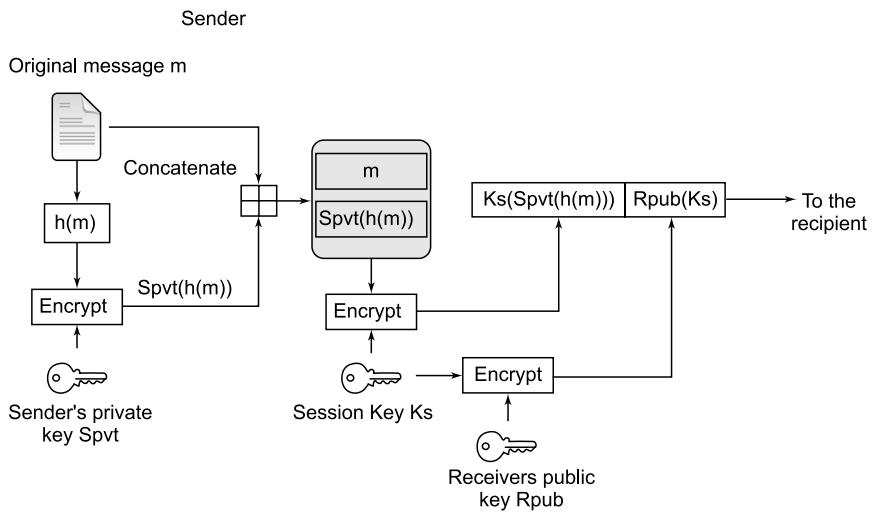
FIGURE 15.3 One-to-one e-mail.

The provision for confidentiality is illustrated in Figure 15.4 and is explained in the following steps:

1. Both sender and receiver have their private and public keys as (S_{pvt} , S_{pub}) and (R_{pvt} and R_{pub}) respectively.
2. The sender generates a session key K_s for encryption. The session key is used to encrypt the message.
3. The sender then encrypts the session key K_s using the public (R_{pub}) key of the recipient.
4. The encrypted message and the encrypted session key K_s are sent to the recipient.
5. The recipient then decrypts the session key K_s using his private key and then decrypts the message by using K_s .



The other security services such as message integrity, authentication, and non-repudiation can be added to the previous process as described in Figure 15.5.



Sender

1. The sender produces a hash of the message and digitally signs this hash with his private key S_{priv} .
2. The signed hash with other components is sent to the recipient.

Recipient

1. The recipient uses the sender's public key S_{pub} and extracts the hash received with the sender's signature.
2. The recipient now generates the hash out of the decrypted message and then compares the two hash values. If they are same, message integrity is considered to be achieved.
3. Also, the authentication is proved. The recipient is sure that the message is sent by the sender. The sender cannot deny that he sent the message (non-repudiation).

15.3.2 One-to-Multiple Recipient E-Mail

In the e-mail communication process the sender sends an e-mail message to two or more recipients. All the recipients get the same e-mail message. The list of all the recipients is managed by the sender's e-mail program, which includes the UA and local MTA. The sender can provide the confidentiality of the e-mail message by adding the following steps. For providing authentication, integrity, and non-repudiation, the same steps are to be followed as mentioned in the one-to-one e-mail scenario.

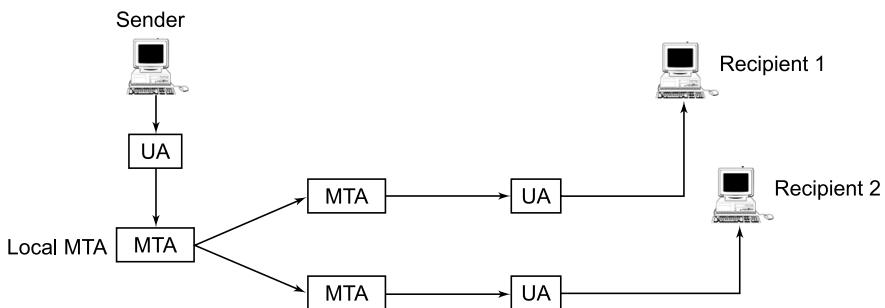


FIGURE 15.6 One-to-multiple recipient e-mail.

1. The sender and all recipients have their own pair of private-public keys.
2. The sender generates a session key K_s to encrypt the message.
3. Once the message is encrypted the sender then encrypts K_s multiple times with public keys $R_{1pub}(K_s), R_{2pub}(K_s), \dots, R_{npub}(K_s)$ of the recipients R_1, R_2, \dots, R_n respectively.
4. The encrypted message and the encrypted session key are sent to the respective recipient.

5. The recipients first extract the session key K_s s by decryption using their private key ($R_{1pvt}, R_{2pvt} \dots, R_{npvt}$).

15.3.3 One-to-Distribution List E-Mail

In this type of e-mail communication, the sender sends an e-mail message to two or more recipients. The list of recipients is not maintained locally by the sender. The mailing list is maintained by the e-mail server (MTA). The sender sends an e-mail to this MTA as in Figure 15.7. The e-mail message confidentiality is ensured to the multiple recipients of the mailing list (R_1, R_2, \dots, R_n) as follows:

1. The sender and all the recipients have their own pair of private-public keys. The Exploder server has a pair of private-public keys for each mailing list ($List_{pub}, List_{pvt}$) maintained by it.
2. The sender generates session key K_s s and then encrypts the message with this key.
3. The session key K_s s is then encrypted with the public key associated with the list obtained $List_{pub}(K_s)$.
4. The sender sends the encrypted message and $List_{pub}(K_s)$ using $List_{pvt}$ and obtains K_s s.
5. The Exploder encrypts K_s s using the public key $L_{pub}(K_s)$ of the recipients present in the list.
6. The Exploder forwards the received encrypted message from the sender and corresponding encrypted key K_s s to the respective recipients in the list.

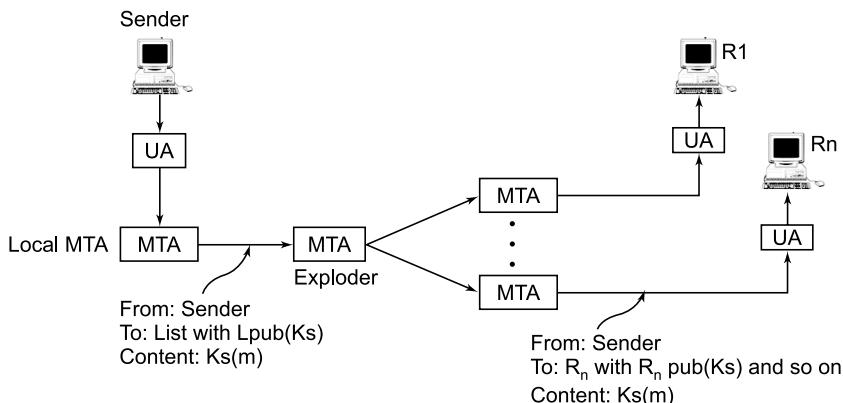


FIGURE 15.7 One-to-distribution list e-mail.

The additional security services such as authentication, integrity, and non-repudiation are provided as explained in the scenario for one-to-one e-mail.

15.4 E-MAIL SECURITY MECHANISMS

The three major security mechanisms to provide e-mail security are:

1. Privacy Enhanced Mail (PEM)
2. Pretty Good Privacy (PGP)
3. Secure-Multipurpose Internet Mail Extension (s-MIME)

Among these popular e-mail security schemes, Pretty Good Privacy (PGP) and s-MIME follow the aforementioned security mechanisms.

15.4.1 Privacy Enhanced Mail (PEM)

Extending the RFC 822 Internet mail message standard, the Internet Activities Board's Privacy Task Force developed a new standard known as Privacy Enhanced Mail (PEM) embodied in RFC 989, which was issued in 1987, with two revisions in 1993.

The PEM specification consists of four parts:

- Message format (in addition to RFC 2822)
- Certificate (as defined in X.509), a certification authority (CA) hierarchy, and certificate revocation list (CRLs)
- Cryptographic algorithms to be used (e.g., DES-CBC, RSA, MD5...)
- Formats of control messages

PEM defined two major features for ASCII message protection: signed message and signed and encrypted message. A message is encrypted using symmetric key encryption (DES) and signed using RSA public key encryption. Users need to establish their RSA public keys' digital certificates conforming to the X.509 CCITT standard. The certificates require to be signed using the private RSA of a trusted certifying authority (CA). The CA's public key itself needs to be replaced in another certificate, which itself could be signed by another CA and so on, thus forming a chain of certificates with a single trusted root.

The major problem with PEM was the requirement that the user's public keys needed to be certified by their local CAs. These local CAs needed

to be registered to the root CA named IPRA (Internet Policy Registration Authority). Nonexistence of such an infrastructure resulted in inertia in establishing a PEM-based e-mail system. RIPEM (Riordan's Internet Privacy Enhanced Mail) is a public domain implementation of PEM.

15.4.2 Pretty Good Privacy (PGP)

PGP is an e-mail security scheme. It has become the de facto standard for providing security services for e-mail communication. The actual operation of PGP provides the five following security services:

1. Authentication
2. Confidentiality
3. Message integrity
4. Non-repudiation
5. Compression

These security services can be achieved by the use of public key cryptography, symmetric key cryptography, hash functions, and digital signatures:

- Public key encryption: RSA, DSS, Diffie-Hellman
- Symmetric key: CAST-128, IDEA, 3DES
- Hash coding: SHA-1

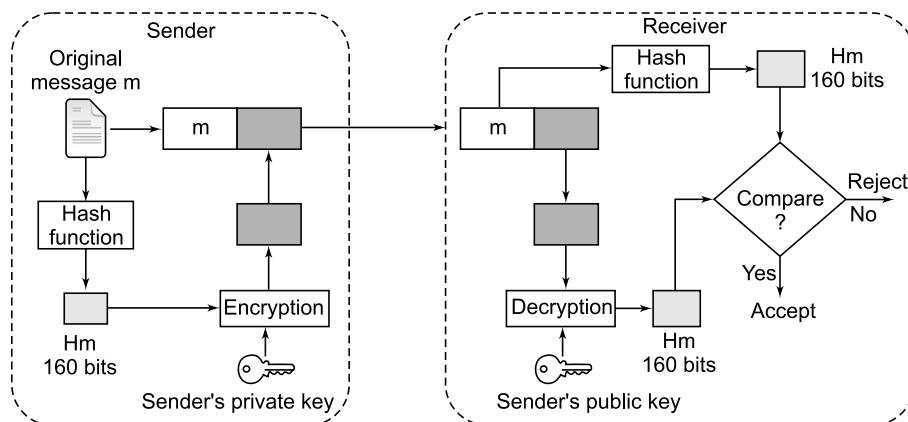


FIGURE 15.8 PGP authentication.

Working of PGP: PGP operation – Authentication: The authentication process in PGP is a digital signature authentication.

- Sender creates message.
- SHA-1 is used to generate a 160 bit hash message.
- The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message.
- The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
- The receiver also generates a new hash code for the message and compares it with the decrypted hash code.

PGP Confidentiality: PGP provides encryption for messages sent or stored as files.

Sender

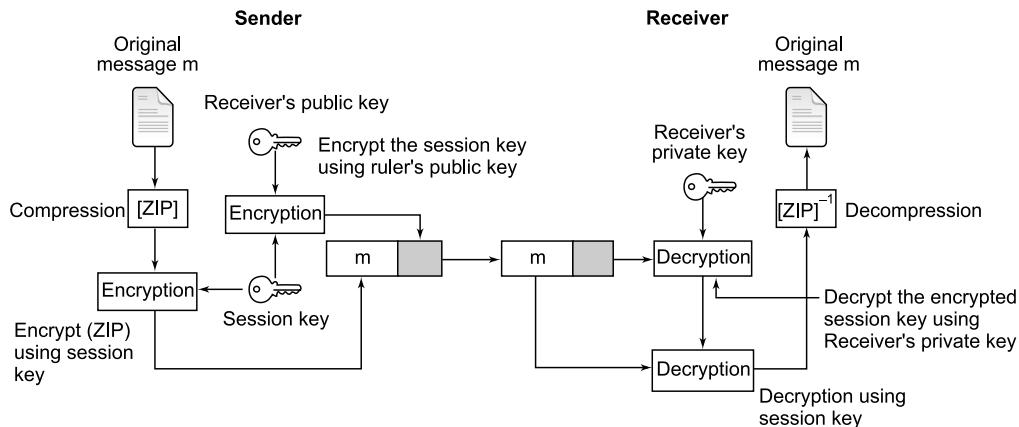
- The sender generates a message and a random number of a 128 bit session key.
- The message is encrypted using CAST – 128 (or IDEA or 3DES) with the session key.
- The session key is encrypted with RSA using the recipient's public key and prepended to the message.

Receiver

- The receiver uses RSA with his private key to decrypt and recover the session key.
- The session key is used to decrypt the message.

The different algorithms used are:

- Symmetric key: CAST, IDEA, 3DES
- Asymmetric key: RSA, ElGamal

**FIGURE 15.9** PGP confidentiality.

Authentication and Confidentiality: Both authentication and confidentiality may be combined for a given message, because confidentiality service provides no assurance to the receiver as to the identity of sender (*i.e.*, no authentication). The combination of these two is called authenticated confidentiality.

- The sender generates a signature for the plaintext message and prepends it to the message.
- The plaintext message plus signature is encrypted.
- The session key is encrypted using RSA (receiver public key) and prepended to the message.

PGP Compression: As a default, PGP compresses the message, using the compression algorithm ZIP, after applying the signature and before encryption. This has the benefit of saving bandwidth both for e-mail transmission and for file storage.

- It is preferable to sign an uncompressed message so that the signature does not depend on the compression algorithm.
- Encryption after compression strengthens the encryption, since compression reduces redundancy in the message.

E-Mail Compatibility: PGP always involves encryption of the message. The encrypted text contains 8 bit octets. However, many e-mail systems only permit the use of ASCII text. To accommodate their restriction, PGP provides the service of converting the raw 8 bit binary stream to a stream of

printable ASCII characters. It uses Radix-64 conversion to map groups of three octets into four ASCII characters. It also appends a CRC for data error checking. The use of Radix-64 expands the message by 33%, but an overall compression of about one-third can still be achieved.

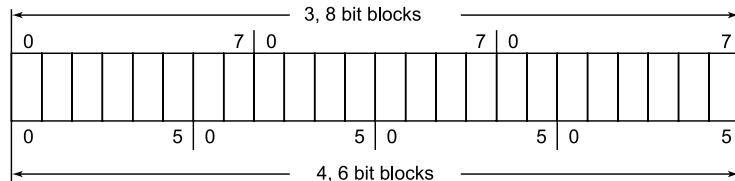


FIGURE 15.10 Radix-64.

Segmentation and Reassembly: E-mail facilities are often restricted to a maximum message length. A longer message must be divided into segments, which are mailed separately. PGP automatically segments messages that are too large. The segmentation is done after all four steps (authentication, confidentiality, compression, and e-mail compatibility) discussed previously, including Radix-64 conversions. Thus signature and session keys appear only once. Reassembly of the e-mail message received at the receiving end is required before verifying signature and decryption. PGP strips off the mail header.

15.4.3 PGP Key Management

PGP makes use of four types of keys:

- One-time session symmetric keys
- Public keys
- Private keys
- Passphrase-based symmetric keys

Keys are stored in encrypted form. PGP stores the keys in two files on the system storage; one for public keys and one for private keys. These files are called key rings. The public key of the recipient is added to the public key ring and private keys are saved in private key rings. A user who misplaces a private key will be unable to decrypt any information encrypted to keys on the ring. In PGP, each entity must maintain a file of its own public/private key pairs as well as the public keys of others.

TABLE 15.1 Radix-64 Table.

6 bit Value	Character Encoding						
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

(Pad) =

A session key is generated on the fly for symmetric encryption of a single message. Each session is associated with a single message and used only once. Encryption is done with CAST-128, IDEA (128 bit keys), or 3DES (168 bit keys). CAST-128 is used to generate the key from a previous session key and two 64 bit blocks generated based on the users keystrokes, including keystroke timing. The two 64 bit blocks are encrypted using CAST-128 and the previous key and concatenated to form the new key.

Managing key pairs: PGP maintains a file of public/private key pairs as well as the public keys of others. Given the desire to allow one user to have multiple public/private key pairs, PGP was used to encrypt a message. The important challenges for maintaining multiple keys are:

- Send the public key along with a message. It is inefficient if the length of the key is hundreds of bits.
- Associate a unique ID with each key pair and send that with the message. The sender must know the mapping of key IDs for all recipients.

- Generate an ID likely to be unique for a given user. This is PGP's solution. It uses the least significant 64 bits of the key as the ID.

The receiver uses this ID to verify that he has such a key on his "key ring." The associated private key is used for the decryption.

Private Key Ring	Public Key Ring
<p>Each user maintains two key ring data structures – a private key ring for his own public/private key pairs and a public key ring for the public keys of correspondents. The private key ring is a table with following details:</p> <p>Time stamp: when the key pair was generated.</p> <p>Key-ID: 64 least significant digits of the public key.</p> <p>Public key: the public portion of the key.</p> <p>Private Key: the private portion encrypted using a passphrase.</p> <p>User ID: usually the user's e-mail address. It may be different for different key pairs.</p>	<p>The public keys of other users are stored on a user's public key ring. The public key ring is a table of rows containing:</p> <p>Time stamp: When the entry was generated.</p> <p>Key-ID: 64 least significant digits of this entry.</p> <p>Public key: the public key for the entry.</p> <p>User ID: identifier for the owner of the key. Multiple IDs may be associated with a single public key.</p>

15.4.4 PGP Message Format

A PGP message consists of three components represented as in Figure 15.11.

- (i) Message component
- (ii) Signature component
- (iii) Session component

Message component: This component of PGP messages includes the actual data to be stored or transmitted, the file name, and a time stamp that specifies the time of creation.

Signature component: The signature component includes the following:

- a. Time stamp: This indicates the time at which the signature was made.
- b. Key ID of Ksnd: It is the key ID of the sender's public key. It identifies the message digest and hence identifies the private key that was used to encrypt the message digest.

- c. Leading two octets of the hash: This section is to enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets of the decrypted digest. These octets act as a 16 bit frame check sequence for the message.
- d. Message digest: A message digest is created by using 160 bit SHA-1 hashing, encrypted with the sender's private signature key.

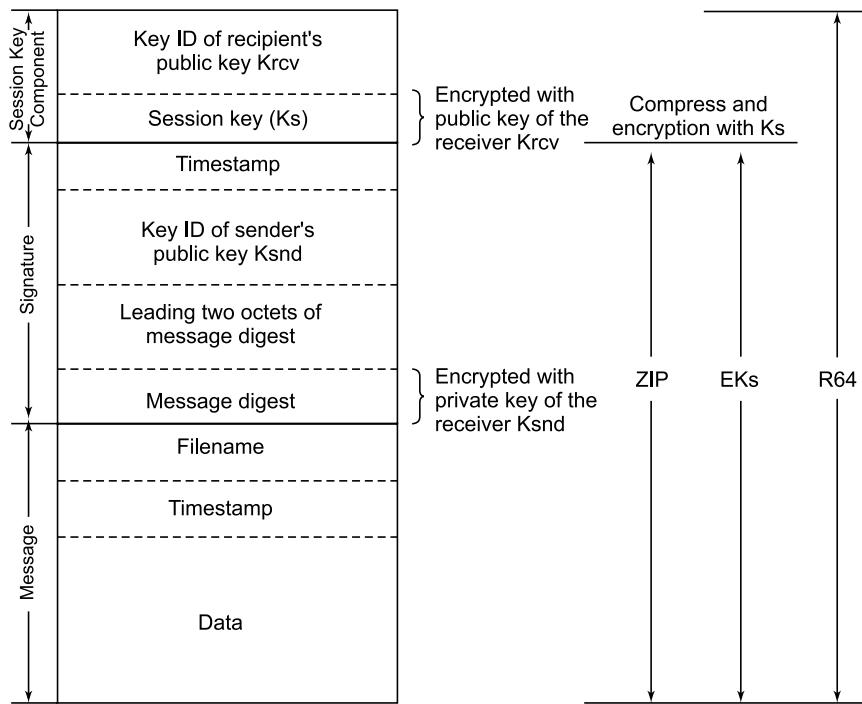


FIGURE 15.11 PCP message format.

Session key component: This component includes the session key and key ID of the recipient's public key Krcv that was used by the sender to encrypt the session key Ks. The entire message block is usually encoded with Radix-64 (R64) encoding.

15.4.5 PGP Message Transmission and Reception

PGP is a hybrid cryptosystem. It encompasses both conventional and public key cryptography. When a user encrypts plaintext with PGP, it first

compresses the plaintext. This will have multiple advantages such as saving modem transmission time and reducing the storage space. More importantly, it strengthens cryptographic security by enhancing resistance to cryptanalysis. PGP then creates a session key, which is a one-time only secret key. This key is a random number generated from the random movement of the user mouse and the keystrokes typed by him. This generated session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext message. After encryption the message and the session key are then encrypted by using the recipient's public key.

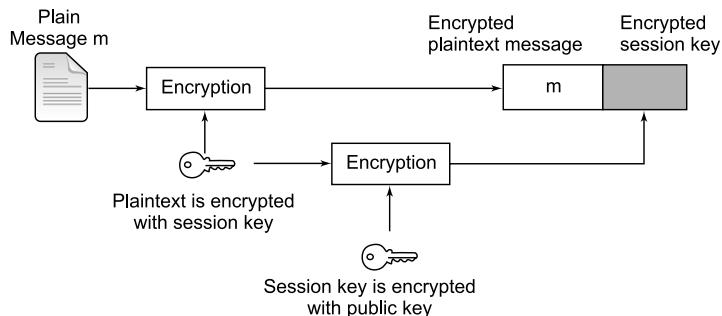


FIGURE 15.12 PGP encryption.

The ciphertext and the public key encrypted session key are transmitted to the recipient. In the decryption process, the recipient uses his or her private key to recover the temporary session key. The PGP then uses this session key to decrypt the ciphertext.

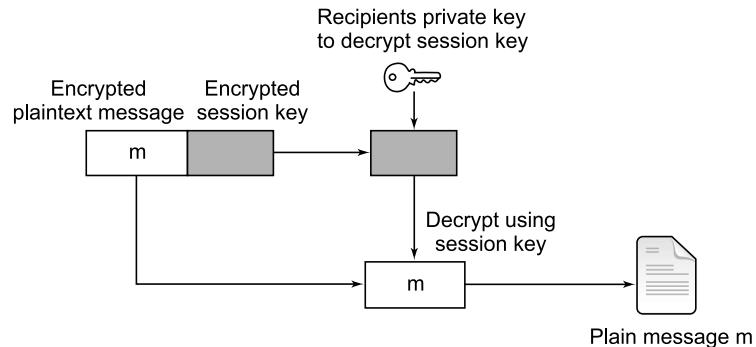


FIGURE 15.13 PGP decryption.

15.4.6 Secure-Multipurpose Internet Mail Extension (s-MIME)

Both PEM and PGP aimed to provide textual e-mail message security. However, with the increase of power in terms of computation and networking capabilities, users' need for using non-textual objects such as image, audio, and video in e-mail become a fair demand. To support diversity of content, multiple message structures, and non-English text, the Multipurpose Internet Mail Extension (MIME) was proposed by IETF in 1992. s-MIME (secure-Multipurpose Internet Mail Extension) is an enhancement of MIME to provide cryptographic security for MIME-based e-mail.

MIME

MIME stands for multipurpose internet mail extension. As the name indicates, it is an extension to the Internet e-mail protocol that allows its users to exchange different types of data files over the Internet such as image, audio, and video. The MIME is required if text is in character sets other than ASCII.

E-mail in its simplest form sends messages only in 7 bit NVT ASCII format. In other words it has some limitations. MIME is a supplementary protocol that allows non-ASCII data (images, audio, and video) to be sent through e-mail. MIME transforms non-ASCII data at the sender site to 7 bit NVT ASCII data and delivers it to the e-mail system (client MTA) to be sent through the Internet. The message at the receiving site is transformed back to the original data as shown in Figure 15.14.

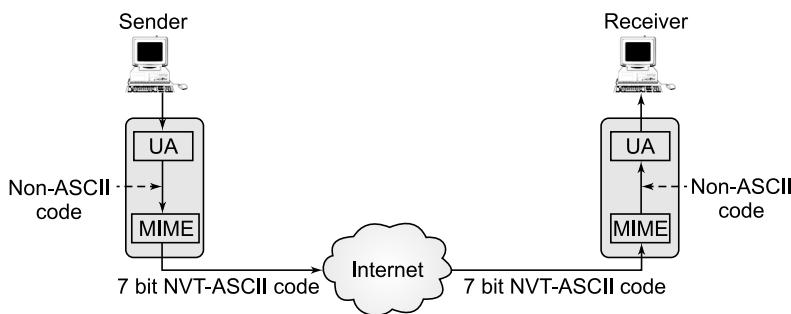


FIGURE 15.14 MIME.

MIME defines five headers mentioned as follows, which can be added to the original e-mail header section to define the content transformation parameters.

1. MIME version
2. Content type
3. Content Transfer Encoding
4. Content-ID
5. Content Description

MIME version: The MIME version header is used to indicate that the message is MIME formatted. Currently only version 1.0 exists.

Content type: This header is used to describe the content types of the message. The content type's header includes the type and sub-type parts. MIME supports:

- Simple text message using text/plain – default value of content type.
- Text and attachments – includes text/plain and other non-text parts.
- Alternative content, such as a message sent in both plaintext and other formats such as HTML.
- Image, audio, video, and other applications.

MIME allows different types of data as listed in Table 15.2.

TABLE 15.2 MIME Content Types and Sub-Types.

Type	Sub-Type	Details
Text	Plain	Unformatted
	HTML	HTML format
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as mixed but order not defined
	Digest	Similar to mixed but the default is message /RFC822
	Alternative	Parts are different version of the same message
Message	RFC 822	Body is an encapsulated message
	Partial	Body is a fragment of a large message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video in MPEG format
Audio	Basic	Single channel encoding of voice at 8KHz
Application	Postscript	Adobe postscript
	Octal-stream	General binary data (8bit bytes)

Content-Transfer Encoding: MIME has defined a set of methods for representing binary data in formats other than ASCII text format. To transfer binary data, MIME offers five encoding formats which can be used in the header transfer encoding as shown in Table 15.3.

Content-ID: It uniquely identifies the entire message in a multiple message environment.

Content Description: It defines whether the body of the message is image, audio, or video.

Secure Multipurpose Internet Mail Extension (s-MIME): s-MIME is an enhancement of MIME to provide cryptographic security for MIME-based e-mail. To secure MIME-based e-mail communication, the s-MIME adds some additional content types to it. The first version of s-MIME was developed in 1995 and the second version in 1998.

Two IETF RFCs that make up s-MIME version 2 are RFC 2311, which establishes the standard for messages, and RFC 2312, which established the standard for certificate handling. With s-MIME version 2, s-MIME emerges as the standard for message security. In 1999, s-MIME version 3 was proposed by IETF to enhance s-MIME capability. RFC 2632 built on the work of RFC 2311 in specifying the standard for s-MIME messages, and RFC 2633 enhanced the RFC 2312 specification of certificate handling. RFC 2634 extended overall capabilities by adding additional services to s-MIME, such as secure recipients, triple wrapping, and security labels. s-MIME version 3 has achieved wide acceptance as the standard for e-mail message security. s-MIME provides two major security services:

- Digital signature
- Message encryption

These two services are the core s-MIME-based message security. All other concepts related to message security support these two services.

TABLE 15.3 MIME Content Transfer Encoding.

Type	Details
7 bit	7 bit text format
8 bit	8 bit text format
Quoted-printable	Non-ASCII characters are encoded as an equal sign followed by an ASCII code
Base 64	Recommended for sending binary files as alternating
Binary	Non-ASCII characters with unlimited length line

Digital Signature

Digital signatures are a more commonly used service of s-MIME. This provides the following security capabilities.

Authentication: A signature serves to validate an identity. The authentication in digital signatures solves the problem by allowing a recipient to know that a message was sent by the person who claims to have sent the message.

Integrity: An additional service that digital signatures provide is data integrity. Data integrity is a result of the specific operations that make digital signatures possible. With data integrity service, when the recipient of a digitally signed e-mail message validates the digital signature, the recipient is assured that the e-mail message that is received is in fact the same message that was signed and sent and has not been altered while in transit. Any alteration of the message while in transit after it has been signed invalidates the signature.

Non-repudiation: The uniqueness of a signature prevents the owner of the signature from disowning the signature. This capability is called non-repudiation. Thus the authentication that a signature provides allows the message to enforce non-repudiation.

Functions of s-MIME

s-MIME provides the following functions:

1. **Signed data:** This process is to form a digital signature by taking the message digest of the content to be signed. The digest is then encrypted by using the private key of the signer. The content plus signature are then encoded using base 64 encoding.
2. **Enveloped data:** The enveloped data is used to provide privacy for the message. This consists of encrypted content of any type and encrypted content keys for one or more recipients. For each recipient a copy of the session key is encrypted with the public key of each recipient. The content is encrypted using the defined algorithm and the session key created.
3. **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case only the digital signature is encoded using base 64. As a result, a recipient without s-MIME capability can view the message content, although they cannot verify the signature.

- 4. Signed and enveloped data:** Signed-only and encrypted-only entities may be signed and signed data or clear-signed data may be encrypted.

Digitally Signed Mail: In general, a message could be signed by the sender by encrypting the message digest using his private key. This process is known as signing. The receiver can decrypt the digest using the sender's public key. If he succeeds, he can be sure that the message is authentic and has not been altered, because a message that can be decrypted using a sender's public key must have been encrypted by using his private key. However, for the sake of performance, s-MIME does signing a bit differently.

- Only a message digest is encrypted, which is faster than encrypting the entire message.
- Therefore, a copy of the original unsigned message must be included with the mail.

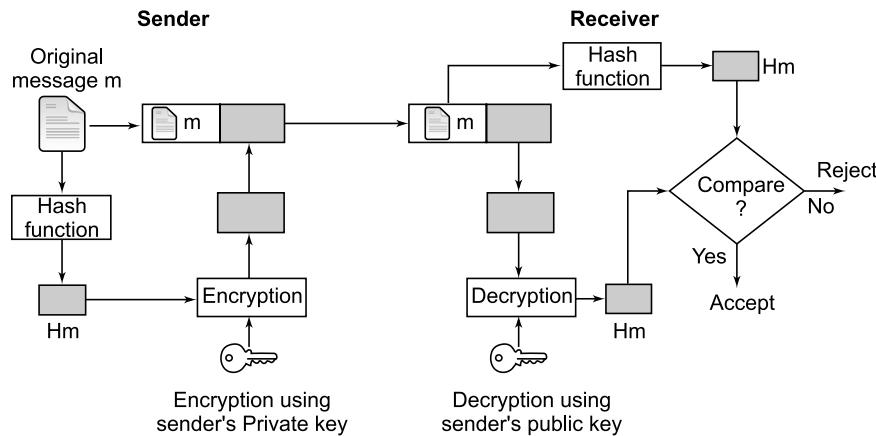


FIGURE 15.15 Signed mail.

The following steps are taken in order to create a signed message:

1. A sender writes the message as clear text.
2. The message digest is computed (algorithms used are SHA-1 or MD5).
3. The message digest is encrypted using the signer's private key (algorithms used are DSS or RSA).

Message Encryption: Encrypted Mail

For secure mail communication the sender encrypts the message and forwards it to the receiver. It can only be read by the receiver. This is ensured by encrypting the message using the receiver's public key, which is available to everyone. However, only the receiver can decrypt the message, because only he owns his private key.

To enhance the performance, s-MIME implements something slightly different:

- The message is not encrypted using the receiver's public key but instead using a randomly generated symmetric session key. Symmetric encryption/decryption is faster than asymmetric algorithms.
- This temporary session key is encrypted using the receiver's public key. Therefore, only the receiver can retrieve the session key and thus decrypt the original message.

The following steps are taken in order to create an encrypted message:

1. The user writes the message as clear text.
2. A random session key is generated (algorithm used: Triple DES or RC2).
3. The message is encrypted using the random session key.
4. For every recipient the session key is encrypted using the recipient's public key (algorithm used: Diffie-Hellman or RSA).

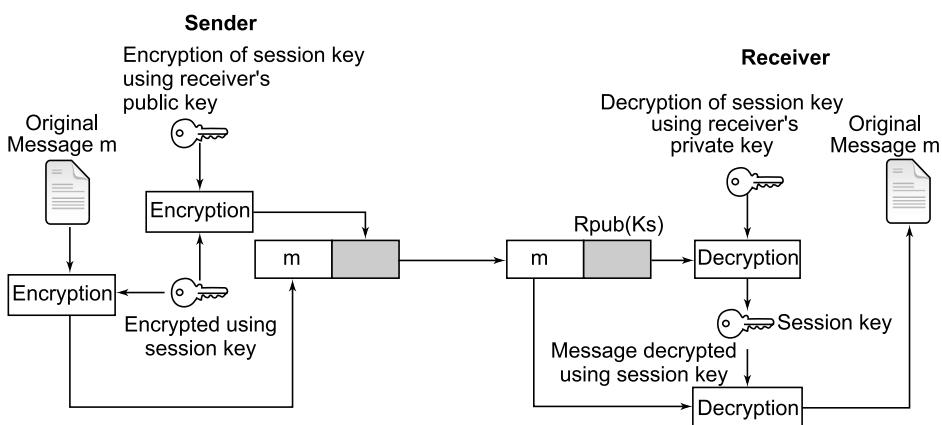


FIGURE 15.16 Encrypted mail.

Cryptographic Algorithms Used in s-MIME

Table 15.4 defines several cryptographic algorithms used in s-MIME and uses the terms “must” and “should.” This is taken from RFC 2119 to specify the requirement level. “Must” defines an absolute requirement of specification; the term “should” means a recommendation that an implementation include that feature or function.

s-MIME incorporates three public key algorithms:

- DSS is the preferred algorithm for digital signature.
- Diffie-Hellman is the preferred algorithm for encrypting session keys; in fact, s-MIME uses a variant of Diffie-Hellman that does provide encryption/decryption known as ElGamal. As an alternative RSA can be used for both signature and session key encryption.
- For the hash function used to create a digital signature, the specifications require the 160 bit SHA-1 but recommend receiver support for 128 bit MD5 for backward compatibility with older versions of s-MIME.

The sending and receiving agents have different capabilities. A sending agent has no knowledge about the decryption capabilities of the intended recipient, and is not willing to risk that the recipient may not be able to decrypt the message. Many may not be able to decrypt the message. The sending agent has two decisions to make:

- The sending agent must determine if the receiving agent is capable of decrypting using a given encryption algorithm.
- If the receiving agent is only capable of accepting weakly encrypted content, the sending agent must decide if it is acceptable to send using weak encryption.

TABLE 15.4 Cryptographic Algorithms used in s-MIME.

Algorithm (Functions)	Sender Must Support	Receiver Must Support	Sender Should Support	Receiver Should Support
Encrypted message digest to form digital signature	DSS	DSS	RSA	1. MD5 2. RSA (512 bits to 1024 bits)

Algorithm (Functions)	Sender Must Support	Receiver Must Support	Sender Should Support	Receiver Should Support
Encrypt session key for transmission with message	RSA (512 bits – 1024 bits)	RSA (512 bits – 1024 bits)	Diffie-Hellman	Diffie-Hellman
Encrypt message for transmission with one-time session key	DES	DES	---	1. AES 2. RC2/4
Create message authentication code	---	HMAC with SHA-1	---	HMAC with SHA-1
Hash function	SHA -1	SHA-1	---	MD5

In order to accomplish this, a sending agent may announce its decrypting capabilities needed in order of preference in any message that it sends out. The receiving agent may store this information for future use. A sending agent has to send a message to multiple receivers. It cannot select a common encryption algorithm for all receivers. The sending agent will need to select multiple algorithms.

SUMMARY

- The application layer, which accommodates the user interface and other key functions, is the closest OSI model layer to the user end. This layer provides the hacker with a wider attack surface.
- E-mail security has become one of the most widely used Internet applications. The needs of e-mail security are based on e-mail access modes, e-mail infrastructure, and e-mail protocols.
- The three major security mechanisms to provide e-mail security are: Privacy Enhanced Mail (PEM), Pretty Good Privacy (PGP), and Secure-Multipurpose Internet Mail Extension (s-MIME).

- PEM defined two major features for ASCII message protection: signed messages and signed and encrypted messages.
- PGP is an e-mail security mechanism. It has become the de facto standard for providing security services for e-mail communication.
- PGP provides textual e-mail security.
- s-MIME provides cryptographic security for MIME-based e-mail for securing non-textual objects like image, audio, and video in e-mail.

REVIEW QUESTIONS

1. Explain e-mail protocols.
2. Describe e-mail security services.
3. What are the three scenarios to be considered for the implementation of e-mail security services? Explain.
4. What is a fingerprint in a PGP system? How can you import a public key of a person into your key ring?
5. Write a note about Privacy Enhanced Mail (PEM).
6. (Explanation for Exercises 7–11) Pretty Good Privacy (PGP) uses the following approach for the generation of session keys that are used in encrypting the body of the e-mail message. A random secret key is generated by the sender, which is used for encrypting the message body. The sender then encrypts this secret key using receiver's public key, and the result is appended to the encrypted message. An alternative approach would be to use an iterative protocol, such a Diffie-Hellman exchange, to decide the session keys.
7. Explain Pretty Good Privacy in detail.
8. List out the services provided by PGP.
9. How is authentication achieved in Pretty Good Privacy?
10. Explain PGP message generation and reception.
11. Explain clearly with relevant illustration how authentication is addressed in PGP.
12. s-MIME (Secure-Multipurpose Internet Mail Extensions) is a standard for public key encryption and signing of e-mail encapsulated in MIME. Which are the cryptographic security services for electronic messaging applications provided by s-MIME?

13. s-MIME allows messages to be signed and encrypted. Should information be signed or encrypted first? What would be the difference?
14. Compare and contrast s-MIME and PGP protocols.
15. Distinguish between signed data and clear-signed data in the context of S-MIME.

MULTIPLE CHOICE QUESTIONS

1. Pretty good privacy (PGP) is used in:
(a) browser security (b) E-mail security
(c) FTP security (d) none of the above
2. PGP encrypts data by using a block cipher called:
(a) international data encryption algorithm
(b) private data encryption algorithm
(c) internet data encryption algorithm
(d) none of the above
3. The protocol used to provide security to e-mails is:
(a) POP (b) PGP
(c) SNMP (d) HTTP
4. PGP uses _____ security systems.
(a) private key cryptosystem (b) public key cryptosystem
(c) public and private key cryptosystem (d) only session key
5. PGP offers _____ block ciphers for message encryption.
(a) 3DES (b) CAST
(c) IDEA (d) all of these
6. The key size allowed in PGP is _____ bits.
(a) 1024-1056 (b) 1024-4056
(c) 1024-4096 (d) 1024-2048

- 7.** The cryptographic algorithms used in s-MIME are:

(a) IDEA	(b) RSA, DES-3
(c) RC4	(d) RC5
- 8.** Which algorithms for digital signatures are used in s-MIME?

(a) DSS and RSA	(b) ECC and RSA
(c) ECC and RSA	(d) ECC and Diffie-Hellman
- 9.** PGP uses the following public key algorithm:

(a) DSS or RSA	(b) RSA or ECC
(c) DSS or ECC	(d) DSS or Diffie-Hellman
- 10.** PGP key management relies on:

(a) X.509 certificate	(b) Kerberos Server
(c) OSI reference model	(d) Web of trustw

CHAPTER 16

TRANSPORT LAYER SECURITY

16.1 INTRODUCTION

One of the problems of publically accessible information repositories or dissemination systems that contain sensitive but unclassified data is to ensure sensitive data is protected appropriate to the risk and magnitude of the harm that would result for the loss, misuse, unauthorized access to, or modification of such data. Given the nature of interconnected networks and the use of the Internet to share information, protection of sensitive data can become difficult if proper mechanisms are not employed to protect the data. Transport Layer Security (TLS) provides such a mechanism to protect sensitive data during message communication across the Internet.

TLS is a protocol developed to provide security services such as confidentiality, authentication, and integrity between two communicating applications. TLS is based on a predecessor protocol called Secure Socket Layer Version 3.0 (SSL 3.0). SSL was developed by Netscape Communications Corporation in 1994 to secure transactions over the World Wide Web. Soon after that IETF began to work to develop a standard protocol that provides the same functionality. They used SSL 3.0 as the basis for that work that became the TLS protocol.

TLS and SSL are most widely recognized as the protocols that provide secure HTTP (HTTPS) for Internet transactions between web browsers and web servers. TLS/SSL can also be used for other application level protocols, such as file transfer protocol (FTP), Lightweight Directory Access Protocol (LDAP), and SMTP.

16.2 SECURE SOCKET LAYER (SSL)

The SSL Protocol was adopted by Netscape in 1994 as a response to the growing concerns over Internet security. The objective is to create a secured data path between a client and a server. SSL is designed to provide security for web traffic, including confidentiality, message integrity, and authentication. These security services are achieved by the use of cryptography and properly authenticated digital signature.

The SSL protocol is a protocol which may be placed between reliable connections-oriented network layer protocols (e.g., TCP/IP) and the application layer protocol (e.g., HTTP). The SSL provides secure communication between client and server by allowing mutual authentication, the use of digital signature for integrity, and encryption for privacy. The protocol is designed to support a range of choices for specific algorithms used for cryptography, digests, and signatures. Table 16.1 identifies the key establishment, confidentiality, digital signature, and hash mechanisms used in both TLS and SSL 3.0 protocols. SSL certificates are therefore critical for the user to trust a Website operating from a server before sending private information to the server. But encryption is only one part of the trust question that SSL delivers.

Generally SSL provides three levels of security:

- Privacy – connection through encryption.
- Identity Authentication - Identification through certifications.
- Reliability – Dependable maintenance of the server connection through message integrity checking.

TABLE 16.1 Mapping the Security Parts of TLS and SSL.

Mechanism	SSL (3.0)	TLS (1.0)
Key establishment	RSA DH – RSA DH-DSS DHE-RSA DHE-DSS DH-Anon Fortezza-KEA	RSA DH-RSA DH-DSS DHE-RSA DHE-DSS DH-Anon

Mechanism	SSL (3.0)	TLS (1.0)
Confidentiality	IDEA-CBC RC4-128bit 3DES – EDE-CBC Fortezza-CBC	IDEA – CBC RC4-128 3DES-EDE-CBC Kerberos AES
Signature	RSA DSA	RSA DSA Elliptic Curve (EC)
Hash	MD5 SHA-1	MD5 SHA-1

A common application of SSL with a Web system is on an online store, where client computers are sending requests to a merchant's server. Since the SSL protocol is integrated into most Web browsers, and those browsers are normally used to access Web applications, no further configuration is required from the client side of the SSL connection.

16.2.1 Working of the SSL Protocol

SSL has two different entities, server and client. The client is the entity that initiates the transaction, whereas the server is the entity that responds to the client and negotiates which cipher suites are used for encryption. In SSL, the Web browser acts as the client and the Website server as the server. Three protocols that lie within SSL are the Handshake Protocol, the Record Protocol, and the Alert Protocol. The client authenticates the server during the Handshake Protocol. When the session is initiated and the handshake is complete, the data transfer is encrypted during the Record Protocol phase. If there are any alarms at any point during the session, the alert is attached to the questionable packet and handled according to the Alert Protocol.

Handshake Protocol

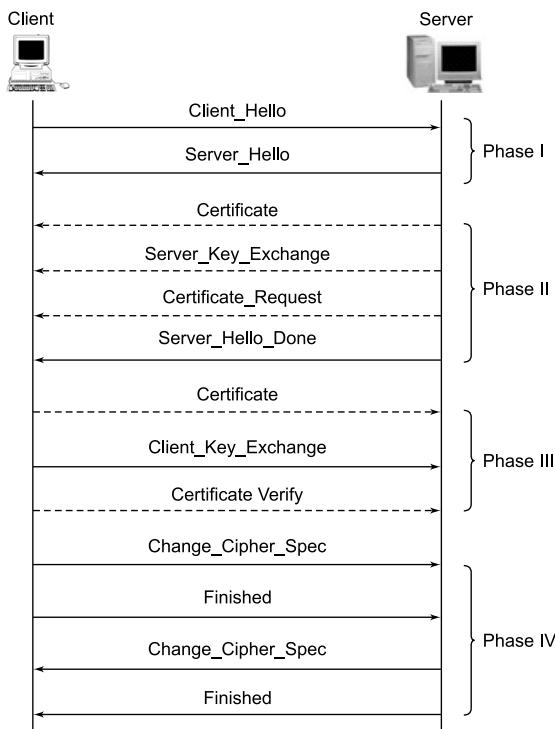


FIGURE 16.1 Handshaking process between the client and server.

The client always authenticates the server, and the server has the option of also authenticating the client. In general, Web servers do not authenticate the client during the Handshake Protocol because the server has other ways to verify the client other than SSL. For e-commerce, the website server can verify the credit card number externally from the SSL session. In this way, the server can reserve precious processing resources for encrypted transactions. Messages are exchanged between the client and server to establish a handshake that begins a secure connection. The handshaking is done in four phases as shown in Figure 16.1 and involves the following steps:

Client Hello, Server Hello, Server key exchange, Server Hello Done, Client Key Exchange, ChangeCipherSpec Finished, ChangeCipherSpec Finished.

Phase I: Creating Security Competence

It includes negotiation of the session ID, key exchange algorithm, MAC algorithm, encryption algorithm, and exchange of the initial random number.

Hello Messages

Client Message to Server:	Server Response to Client:
<p>The client initiates a session by sending a Client Hello message to the server. It contains:</p> <ul style="list-style-type: none"> • Version Number - The client sends the version number corresponding to the highest version it supports. Version 2 is used for SSL 2.0, version 3 for SSL 3.0, and version 3.1 for TLS. Although the IETF RFC for TLS is TLS version 1.0, the protocol uses 3.1 in the version field to indicate that it is a higher level (newer and with more functionality) than SSL 3.0. • Randomly Generated Data - Client Random[32 bit], the random value, is a 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number that will ultimately be used with the server random value to generate a master secret from which the encryption keys will be derived. • Session Identification (if any) - The session ID is included to enable the client to resume a previous session. Resuming a previous session can be useful, because creating a new session requires processor-intensive public key operations that can be avoided by resuming an existing session with its established session keys. Previous session information, identified by the sessionID, is stored in the respective client and server session caches. • Cipher Suite - The list of cipher suites available on the client. An example of a cipher suite is TLS_RSA_WITH_DES_CBC_SHA. 	<p>Server Hello - The server responds with a Server Hello message. The Server Hello message includes:</p> <ul style="list-style-type: none"> • Version Number. The server sends the highest version number supported by both sides. This is the lower number of the highest version number the server supports and the version sent in the Client Hello message. • Randomly Generated Data - Server Random[32 bit], the Random Value, is a 4-byte number of the server's date and time plus a 28-byte randomly generated number that will be ultimately used with the client random value to generate a master secret from which the encryption keys will be derived. • Session Identification (if any) - This can be one of three choices. <ul style="list-style-type: none"> – New session ID - The client did not indicate a session to resume so a new ID is generated. A new session ID is also generated when the client indicates a session to resume but the server can't or won't resume that session. This latter case also results in a new session ID. – Resumed Session ID - The ID is the same as indicated in the client hello. The client indicated a session ID to resume and the server is willing to resume that session. – Null - this is a new session, but the server is not willing to resume it at a later time so no ID is returned. • Cipher Suite - The server will choose the strongest cipher that both the client and server support. If there are no cipher suites that both parties support, the session is ended with a "handshake failure" alert.

(continued)

<ul style="list-style-type: none"> Where TLS is the protocol version, RSA is the algorithm that will be used for the key exchange, DES_CBC is the encryption algorithm (using a 56-bit key in CBC mode), and SHA is the hash function. <p>Compression Algorithm - The requested compression algorithm (none currently supported).</p> <p>The following is an example of a Client Hello message:</p> <ul style="list-style-type: none"> ClientVersion 3,1 ClientRandom[32] SessionID: None (new session) <p>Suggested Cipher Suites:</p> <ul style="list-style-type: none"> TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH DES_CBC_SHA <p>Suggested Compression Algorithm: NONE</p>	<ul style="list-style-type: none"> Compression Algorithm - Specifies the compression algorithm to use (none currently supported). <p>The following is an example of a Server Hello Message:</p> <ul style="list-style-type: none"> Version 3,1 ServerRandom[32] <p>SessionID: bd608869foc629767ea7e3ebf7a63bdc ffb0ef58b1b941e6b0c044acb6820a77</p> <p>Use Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA</p> <p>Compression Algorithm: NONE</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Phase II: *Server Authentication*

In this phase, the server may send its certificate and key exchange message, and it may request the client to send a certificate; the server signals the end of the Hello process.

Server Certificate: The server sends its certificate to the client. The server certificate contains the server's public key. The client will use this key to authenticate the server and to encrypt the premaster secret. The client also checks the name of the server in the certificate to verify that it matches the name the client used to connect. The browser will warn the user if these names do not match.

Server Key Exchange: This is an optional step in which the server creates and sends a temporary key to the client. This key can be used by the client to encrypt the Client_Key_Exchange message later in the process. The step is only required when the public key algorithm does not provide the key material necessary to encrypt the *Client_Key_Exchange* message, such as when the server's certificate does not contain a public key.

Client Certificate Request: This is an optional step in which the server requests authentication of the client. This step might be used for websites (such as a banking websites) where the server must confirm the identity of the client before providing sensitive information.

Server Hello Done: This message indicates that the server is finished and awaiting a response from the client.

Phase III: Client Authentication

The client sends the certificate if requested and may send an explicit certificate verification message; the client always sends its key exchange message.

Client Response to Server

Client Certificate: If the server sent a Client Certificate Request, the client sends its certificate to the server for client authentication. The client's certificate contains the client's public key.

Client Key Exchange: The client sends a Client_Key_Exchange message after computing the premaster secret using both random values. The premaster secret is encrypted by the public key from the server's certificate before being transmitted to the server. Both parties will compute the master secret locally and derive the session key from it.

If the server can decrypt this data and complete the protocol, the client is assured that the server has the correct private key. This step is crucial to prove the authenticity of the server. Only the server with the private key that matches the public key in the certificate can decrypt this data and continue the protocol negotiation.

This message will also include the protocol version. The server will verify that it matches the original value sent in the client hello message. This measure guards against rollback attacks. Rollback attacks work by manipulating the message in order to cause the server and the client to use a less secure, earlier version of the protocol.

Certificate Verify: This message is sent only if the client previously sent a Client Certificate message. The client is authenticated by using its private key to sign a hash of all the messages up to this point. The recipient verifies the signature using the public key of the signer, thus ensuring it was signed with the client's private key.

Change Cipher Spec: This message notifies the server that all messages that follow the Client Finished message will be encrypted using the keys and algorithms just negotiated.

Client Finished: This message is a hash of the entire conversation to provide further authentication of the client. This message is the first message that the record layer encrypts and hashes.

Phase IV: Finishing the Handshake

Change-Cipher_Spec and Finished handshake.

Server Final Response to Client

Change_Cipher_Spec Message: This message notifies the client that the server will begin encrypting messages with the keys just negotiated.

Server Finished Message: This message is a hash of the entire exchange to this point using the session key and the MAC secret. If the client is able to successfully decrypt this message and validate the contained hashes, it is assured that the SSL/TLS handshake was successful, and the keys computed on the client machine match those computed on the server.

Handshake Sequence Protocol: The handshake sequence uses three protocols:

- **The SSL Handshake Protocol:** for performing the client and server SSL session establishment.
- **The SSL Change Cipher Spec Protocol:** for actually establishing agreement on the Cipher Suite for the session.
- **The SSL Alert Protocol:** for conveying SSL error messages between client and server.

These protocols, as well as application protocol data, are encapsulated in the SSL Record Protocol, as shown in Figure 16.2. An encapsulated protocol is transferred as data by the lower layer protocol, which does not examine the data. The encapsulated protocol has no knowledge of the underlying protocol.

The encapsulation of SSL control protocols by the record protocol means that if an active session is renegotiated, the control protocols will be transmitted securely. If there was no previous session, the Null cipher suite is used, which means there will be no encryption and messages will have no integrity digests until the session has been established.

SSL Record Protocol: The encryption for all messaging in SSL is handled in the Record Protocol. It formats the alert, Change_Cipher_Spec, Handshake, and application protocol message. This formatting provides a header for each message and a hash generated from a message authentication code (MAC) at the end. The five byte header of the record layer includes: 1-byte protocol definition, 2-byte protocol version, and the 2-byte length. The message that follows the header is of length 16,384 bytes. SSL records consist of the encapsulated data, digital signature, message type, version, and length. SSL records are 8 bytes long. Because the record length is fixed, encrypted messages sometimes include padding and pad length in the frame, as shown in Figure 16.3.

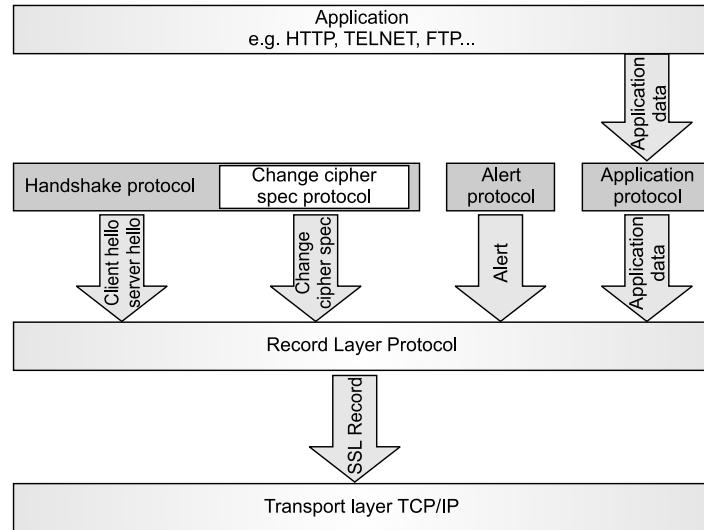


FIGURE 16.2 SSL protocol stack.

The *Change_Cipher_Spec* layer is composed of one message that signals the beginning of secure communication between the client and server. This message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated *Cipher_Spec* and keys.

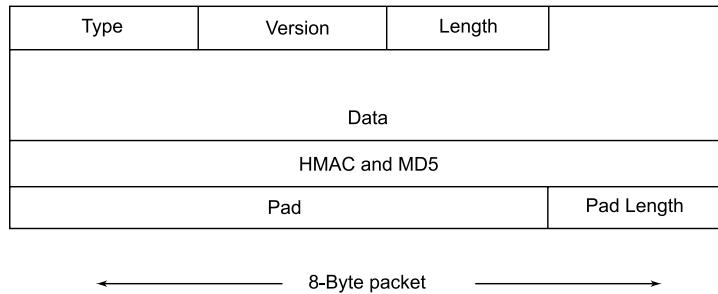


FIGURE 16.3 SSL record protocol format.

The functions of the record protocol are:

- It receives the data from the application layer.
- It fragments the data into blocks or reassembles fragmented data into its original structure.

- It numbers the sequence of data blocks in the message to protect against attacks that attempt to reorder data.
- It compresses or decompresses the data using the compression algorithm negotiated in the handshake protocol.
- It encrypts or decrypts the data using the encryption keys and cryptographic algorithm negotiated during the handshake protocol.
- It applies an HMAC (or a MAC for SSL 3.0) to outgoing data. It computes the HMAC and verifies that it is identical to the value transmitted in order to check data integrity when a message is received.

Once the record protocol has completed its operations on the data, it sends the data to the TCP/IP transport layer for transmission. If the data is incoming, it is sent to the appropriate process for reception.

Data Transfer

The SSL Record Protocol, shown in Figure 16.4, is used to transfer application and SSL Control data between the client and server. If necessary it fragments this data into smaller units or combines multiple higher level protocol data messages into single units. It may compress, attach digest signatures, and encrypt these units before transmitting them using the underlying reliable transport protocol.

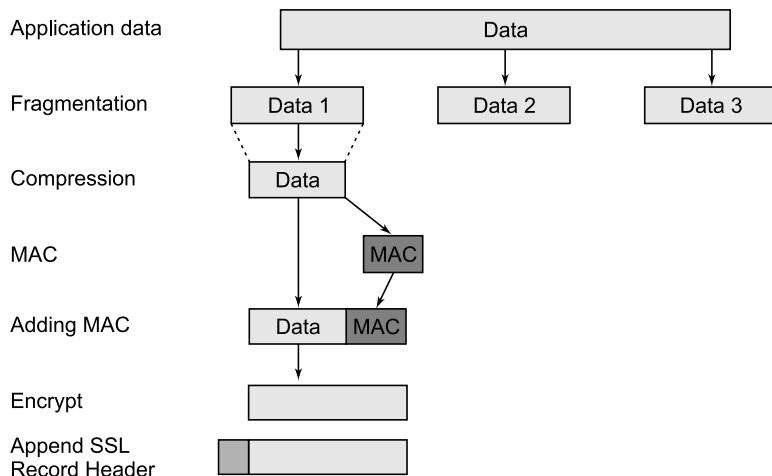


FIGURE 16.4 SSL record protocol.

Alert Protocol: The alert protocol is a component of the Handshake protocol that includes event-driven alert messages that can be sent from

either party. It sends errors, problems, or warnings about the connection between the clients and server. Table 16.2 lists the alert messages and their descriptions from either end. These alerts are defined in the TLS specification in RFC 2246.

TABLE 16.2 Event-Driven Alert Messages from the Alert Sub Protocol.

Alert Message	Description	Fatal?
unexpected_message	Inappropriate message	Yes
bad_record_mac	Incorrect MAC	Yes
decryption_failed	Unable to decrypt TLSCiphertext correctly	Yes
record_overflow	Record is more than 214+1024 bytes	Yes
handshake_failure	Unacceptable security parameters	Yes
bad_certificate	There is a problem with the Certificate	Yes
unsupported_certificate	Certificate is unsupported	No
certificate_revoked	Certificate has been revoked	No
certificate_expired	Certificate has expired	No
certificate_unknown	Certificate is unknown	No
illegal_parameter	Security parameters violated	Yes
unknown_ca	CA unknown	Yes
access_denied	Sender does not want to negotiate	Yes
decode_error	Unable to decode message	Yes
decrypt_error	Handshake cryptographic operation failed	Yes
export_restriction	Not in compliance with export regulations	Yes
protocol_version	Protocol version not supported by both parties	Yes
insufficient_security	Security requirements not met	Yes
internal_error	Error not related to protocol	Yes
user_canceled	Not related to protocol failure	Yes
no_renegotiation	Negotiation request rejected	No

16.3 TRANSPORT LAYER SECURITY (TLS)

Netscape developed SSL protocol to enable e-commerce transaction security on the Web. E-commerce requires encryption to protect a customer's confidential and private data, and also to provide authentication and integrity.

Its main objective is to ensure a safe transaction. The combination of these features the overall view of online security. To achieve this the SSL protocol is implemented at the application layer, directly on top of TCP as shown in Figure 16.5. When IETF standardized the SSL protocol, it was renamed the Transport Layer Protocol (TLS), but technically they are different protocols, since each describes a different version of the protocols.

The purpose of both TLS and SSL is to provide a secure transaction between the client (browser) and the server. The terms SSL and TLS are often used interchangeably or in conjunction with each other (TLS/SSL). The SSL 3.0 served as the basis for TLS 1.0. It used to be believed that TLS 1.0 was marginally more secure than SSL 3.0. The subsequent versions of TLS 1.1 and 1.2 are significantly more secure and fix many vulnerabilities in SSL 3.0 and TLS 1.0. For example, the BEAST attack can completely break websites running on the SSL 3.0 and TLS 1.0 protocols. The new TLS version is properly configured to prevent the BEAST and other attacks. TSL protocol is designed to provide three essential services: confidentiality, authentication, and data integrity. Along with this, interoperability, extensibility, and higher efficiency are the other objectives. These objectives are achieved through implementation of TLS protocol on two levels.

- (i) TLS record protocol
- (ii) TLS handshake protocol

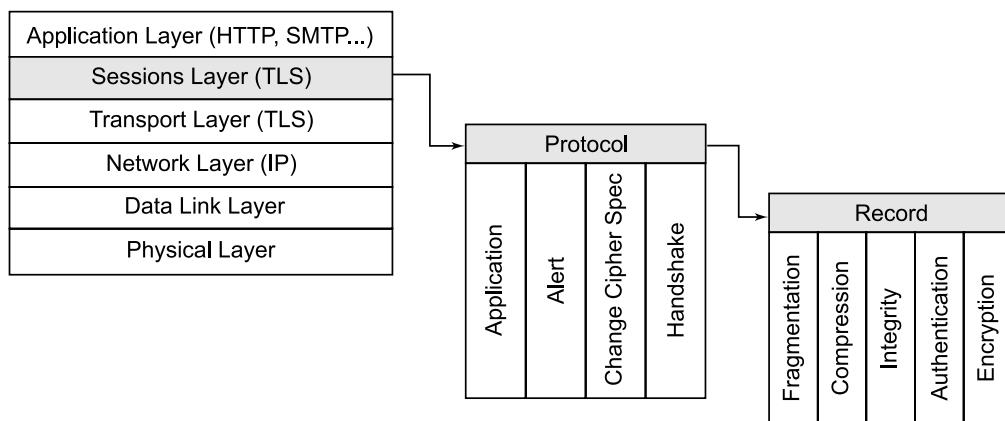


FIGURE 16.5 Transport layer security.

16.3.1 TLS Record Protocol

The TLS record protocol negotiates a private reliable connection between the client and server. Though the record protocol can be used without encryption, it uses a symmetric cryptography key to ensure a private connection. This connection is secured through the use of a hash function generated by using a message authentication code.

16.3.2 TLS Handshake Protocol

TLS handshake protocol enables the client and server to begin exchanging data over TLS. The client and server negotiate to agree on the version of the TLS protocol and the selection of the cipher suite, and verify the certificate if necessary. TLS protocol runs over reliable TCP protocol using a TCP connection in place, and the client sends a number of specifications in plaintext, such as the version of the TLS protocol it is running, the list of supported cipher suites, and other TLS options it may want to use. The server selects the TLS protocol version for further communication, decides on a cipher suite from the list provided to the client, attaches its certificate, and sends the response back to the client. Also the server sends a request for the client certificate and parameters for other TLS extensions.

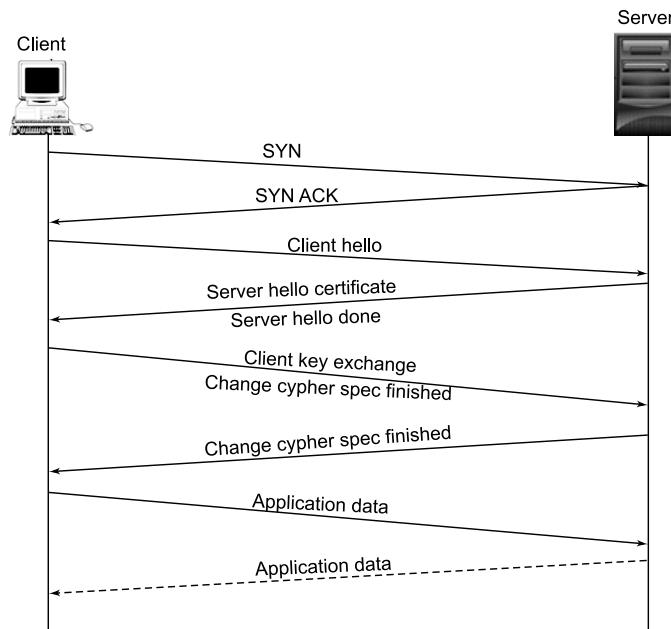


FIGURE 16.6 TLS Handshake protocol.

The client initiates either the RSA or Diffie-Hellman key exchange, which is used to establish the symmetric key for guaranteeing the session. The server processes the key exchange using the parameters sent by the clients, checks message integrity by verifying the MAC, and returns an encrypted “Finished” back to the client. The client decrypts the message with the negotiated symmetric key and verifies the MAC, and if all is well the tunnel is established and application data can now be sent. The different security algorithms used in TLS are given in Table 16.3.

TABLE 16.3 Data Encryption Options.

Integrity				
Key size	144	160		
Algorithm options	HMAC	HMA		
	MD5	SHA		
Confidentiality				
Key size	40	56	128	168
Algorithm options	RC4-40 RC2-CBC-40 DES-CBC-40	DES-CBC	RC4 IDEA-CBC	3DES-EDE-CBC

16.3.3 TLS Confidentiality, Authentication, and Integrity

TLS uses a combination of symmetric and asymmetric encryption to ensure privacy. During a TLS handshake, the TLS client and server agree on an encryption algorithm and a shared secret key to be used for one session only. All messages transmitted between the TLS client and server are encrypted using that algorithm and key, ensuring that the message remains private even if it is encrypted. The following symmetric key encryption algorithms are used in various cipher suits of TLS to provide confidentiality.

IDEA	IDEA is a block cipher that operates on a 64 bit plaintext block. The key is 128 bits long. The same algorithm is used for encryption and decryption.
RC4	Is a stream cypher that uses a variable length key between 8 and 2048 bits long
3DES	It uses a 64 bit block and a 56 bit key
AES	Uses 128, 192, or 256 bit keys; however, cipher suites are only defined for 128 and 256 bit keys to reduce the over proliferation of cipher suites.

TLS provides data integrity by calculating a message digest. It generally uses RSA and DSA digital signature algorithms for various cipher suits.

RSA	To sign a message with the RSA algorithm, the signatory computer's message digests or hashes the message and encrypts it with the private key. To verify an RSA signed message, the verifier decrypts the message digest with the signatory's public key and compares it with a locally computed hash of the original message.
DSA	To sign a message with the DSA algorithm, the signatory computes a signature using an SHA-1 hash algorithm and the signatory's private key. To verify a DSA signature, the verifier performs a computation using the message hash, the signatory's signature, and the public key.

16.4 HYPERTEXT TRANSFER PROTOCOL SECURE (HTTPS) – HTTP OVER SSL

HTTPS (Hypertext Transfer Protocol Secure) is a communications protocol used on the Internet that has a layer of security added. It is a combination of the standard HTTP protocol and a security protocol called SSL/TLS. HTTPS is important because standard HTTP sends data over the Internet in plaintext, making it easy to intercept. The added security layer with HTTPS encrypts information traveling over the internet to prevent wiretapping, stolen credit card numbers, and other interceptions. HTTPS is often used to protect highly confidential online transactions like online banking and online shopping order forms. Web browsers such as Internet Explorer, Firefox, and Chrome also display a padlock icon in the address bar to visually indicate that a HTTPS connection is in effect.

How Does HTTPS Work?

HTTPS pages typically use one of two secure protocols to encrypt communications: SSL (Secure Sockets Layer) or TLS (Transport Layer Security). Both the TLS and SSL protocols use what is known as an “asymmetric” Public Key Infrastructure (PKI) system. An asymmetric system uses both a public key and a private key. Anything encrypted with the public key can only be decrypted by the private key and vice versa.

As the names suggest, the private key should be kept strictly protected and should only be accessible to the owner of the private key. In the case

of a Website, the private key remains securely concealed on the web server. Conversely, the public key is intended to be distributed to anybody and everybody that needs to be able to decrypt information that was encrypted with the private key.

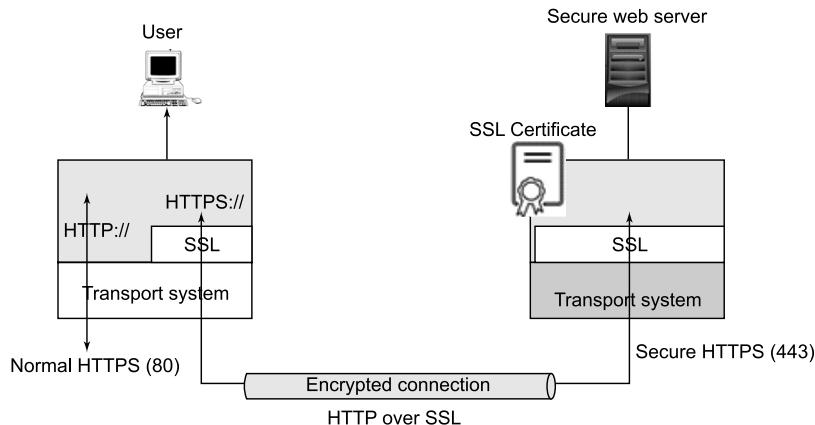


FIGURE 16.7 HTTP over SSL.

HTTPS Certificate

When an HTTPS connection is requested to a Webpage, the website will initially send its SSL certificate to the user browser. This certificate contains the public key needed to begin the secure session. Based on this initial exchange, the browser and the website then initiate the SSL handshake. The SSL handshake involves the generation of shared secrets to establish a uniquely secure connection between user system and the Website. When a trusted SSL Digital Certificate is used during an HTTPS connection, users will see a padlock icon in the browser address bar. When an Extended Validation Certificate is installed on a Website, the address bar will turn green.

16.5 SSH – SECURE SHELL

Secure Shell is a secure protocol and the most common way of safely administering remote servers.

It uses a number of encryption technologies. SSH provides a mechanism for establishing a cryptographically secure connection between two parties, authenticating each side to the other and passing commands and output back and forth. SSH is organized as three protocols that typically run on top of TCP as in Figure 16.8.

1. **Transport Layer Protocol:** It provides data confidentiality, server authentication, and data integrity with forward secrecy. The transport layer may optionally provide compression.
2. **User Authentication Protocol:** It authenticates the user to the server.
3. **Connection Protocol:** Multiplexes multiple logical communication channels over a single underlying SSH connection.

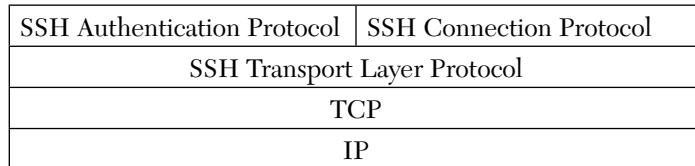


FIGURE 16.8 SSH protocol stack.

16.5.1 Transport Layer Protocol

The transport layer provides secure authentication based on the server processing a public-private key pair. A server may have multiple host keys using multiple different asymmetric encryption algorithms. Multiple hosts may share the same host key. In any case the server host key is used during the key exchange to authenticate the identity of the host. For this authentication to be possible, the client must have presumptive knowledge of the server public host key. Figure 16.9 illustrates the sequence of events in the SSH transport layer protocol.

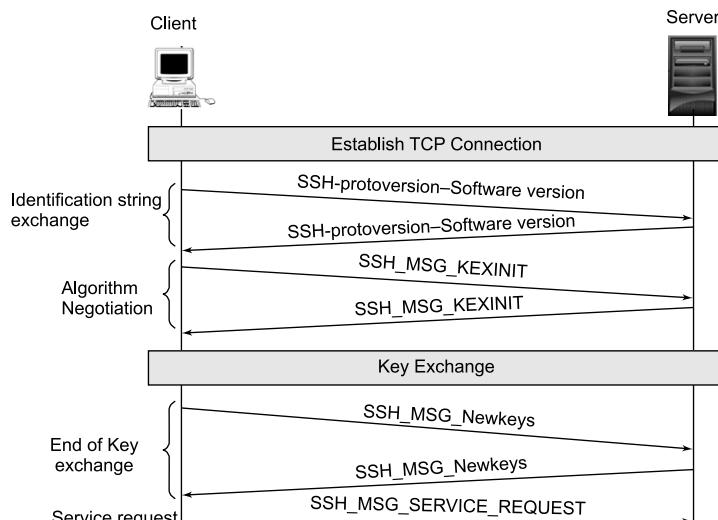


FIGURE 16.9 SSH transport layer protocol exchange.

1. The client establishes a TCP connection to the server with the TCP protocol and is not part of the transport layer protocol.
2. On establishing the connection, the client server exchanges data referred to as packets in the data field of the TCP segment.

The format of the packet exchange between the client and server is given in Figure 16.10.

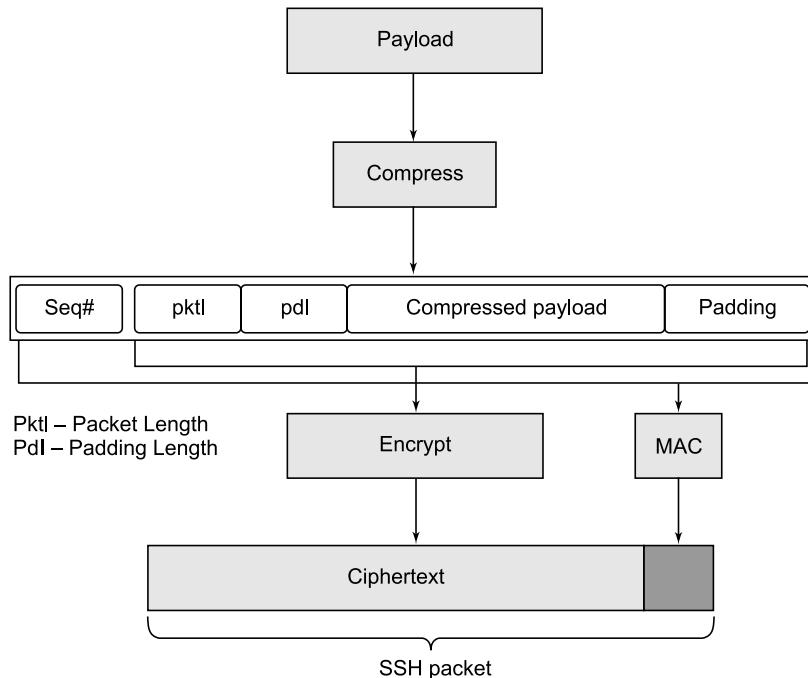


FIGURE 16.10 SSH transport layer protocol packet format.

Pktl: It is the length of the packet in bytes. This does not include the packet length and message MAC (message authentication code) field.

Pdl: Padding length is the length of the random padding field.

Payload: It constitutes the use of the contents of the packet. This portion is compressed before the algorithm negotiation. If the compression is negotiable, then in subsequent packets this field is compressed.

Random padding: This field is added after an encryption algorithm is negotiated. It contains random bytes of padding so that the total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher.

Message Authentication Code (MAC): If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.

After an encryption algorithm is negotiated, the entire packet (excluding the MAC field) is encrypted after the MAC value is calculated. Figure 16.11 represents the sequence of steps in SSH transport layer packet exchange:

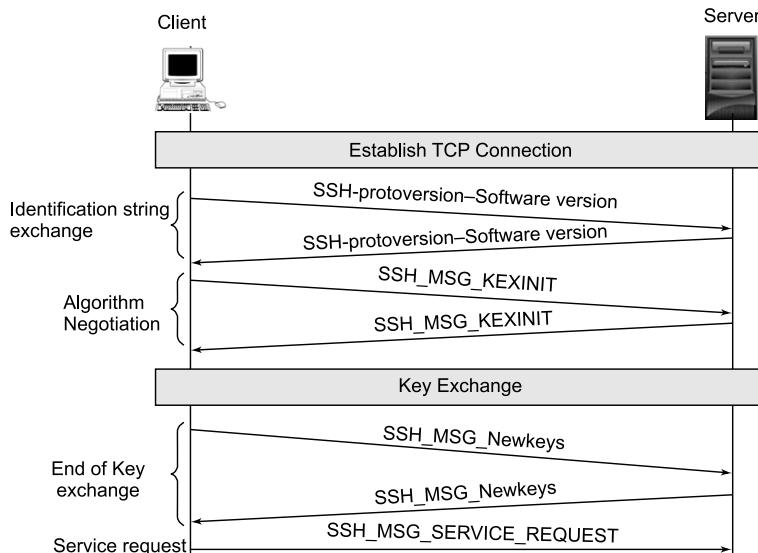


FIGURE 16.11 Transport layer packet exchange.

- **Identification string exchange:** SSH_protoversion_software version SP comments CR LF where SP: space character, CR: carriage return, and LF: line feed.
- **Algorithm negotiation (SSH-MSG-KEXINIT):** This contains a list of supported algorithms in the order of preference of the sender. The algorithm includes key exchange, encryption, MAC algorithm, and compression algorithms.
- **Key exchange:** This specification allows alternative methods of key exchange. This will consider only two versions of the Diffie-Hellman key exchange. These keys are used for encryption and MAC operations.

- **End of key exchange:** In this step, both sides may start using the key generated in the previous step.
- **Service request:** The client sends an SSH_MSG_SERVICE_REQUEST packet to request either the user authentication or the connection protocol. Following this request all data is exchanged as the payload of an SSH transport layer packet, protected by encryption and MAC.

16.5.2 SSH-User Authentication Protocol

It authenticates the client to the server. User authentication protocol always uses three types of messages. The format of an authentication request from the client is as follows:

Byte SSH-MSG_USERAUTGH_REQUEST (50)
String username
String service name
String method name
... method specific field.

Username – Is the authorization identity the client is claiming.

Service name – Is the facility to which the client is requesting access.

Method name – Is the authentication method being used in this request.

The first byte has decimal value 50, which is interpreted as SSH-MSG_USERAUTGH_REQUEST. A server may accept or reject the authentication request. In case of failure it requires one or more additional authentication.

The message exchange between the client and server involves the following steps:

1. The client sends a SSH_MSG_USERAUTH_REQUEST with a requested method of none.
2. The server checks to determine if the username is valid. If not, the server returns SSH_MSG_USERAUTH_FAILURE with the partial success value of false. If the username is valid, the server proceeds to step 3.
3. The server returns SSH_MSG_USERAUTH_FAILURE with a list of one or more authentication methods to be used.
4. The client selects one of the acceptable authentication methods and sends a SSH_MSG_USERAUTH_REQUEST with that method name and the required method-specific fields. At this point, there may be a sequence of exchanges to perform the method.

5. If the authentication succeeds and more authentication methods are required, the server proceeds to step 3, using a partial success value of true. If the authentication fails, the server proceeds to step 3, using a partial success value of false.
6. When all required authentication methods succeed, the server sends a SSH_MSG_USERAUTH_SUCCESS message, and the Authentication Protocol is over.

The server may require one or more of the following authentication methods:

- Public key:** This method depends on the public-key algorithm selected. In essence, the client sends a message to the server that contains the client's public key, with the message signed by the client's private key. When the server receives this message, it checks to see whether the supplied key is acceptable for authentication and, if so, it checks to see whether the signature is correct.
- Password:** The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol.
- Hostbased:** Authentication is performed on the client's host rather than the client itself. Thus, a host that supports multiple clients would provide authentication for all its clients. This method works by having the client send a signature created with the private key of the client host. Thus, rather than directly verifying the user's identity, the SSH server verifies the identity of the client host—and then believes the host when it says the user has already authenticated on the client side.

16.5.3 SSH – Connection Protocol

The SSH connection protocol provides multiple logical connections over a single underlying SSG connection. It runs on top of the SSH Transport Layer Protocol and user authentication protocol and assumes that a secure authentication connection is in use. That secure authentication connection, referred to as a tunnel, is used by the Connection Protocol to multiplex a number of logical channels.

All types of communication using SSH, such as a terminal session, are supported using separate channels. Either side may open a channel. For each channel, each side associates a unique channel number with a unique

identity, which need not be the same on both ends. Channels are flow-controlled using a window mechanism. No data may be sent to a channel until a message is received to indicate that window space is available. The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel.

Opening a Channel

For opening a new channel, it allocates a local number for the channel and send the message. The message format is:

byte SSH_MSG_CHANNEL_OPEN
string channel type
unit32 sender channel
unit32 initial window size
unit32 maximum packet size
...channel type specific data follow

Unit32	Unsigned 32 bit integer
Channel type	Identifies the application for this channel
Sender channel	Is the local channel number
Initial window size	It specifies how many bytes of channel data can be sent to the sender of this message without adjusting the window
Maximum packet size	Specifies the maximum packet size of an individual data packet that can be sent to the sender

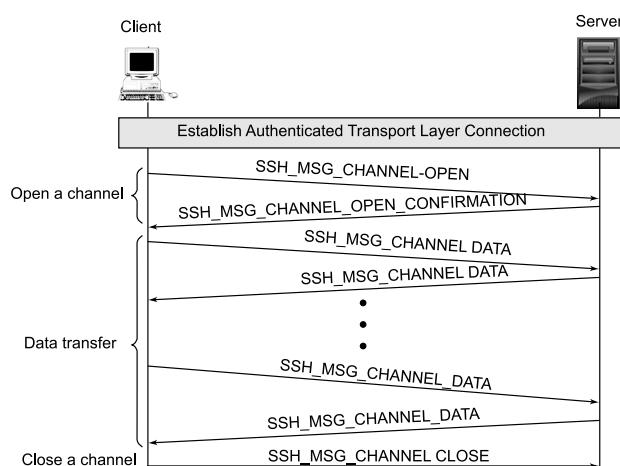


FIGURE 16.12 SSH connection protocol message exchange.

Data Transfer

The data transfer message SSH_MSG_CHANNEL DATA includes the recipient channel number and a block of data. This bidirectional data transfer also includes the recipient's channel number.

Close Channel

When both sides need to close a channel, it sends a close message as SSH_MSG_CHANNEL_CLOSE, which also includes the recipient's channel number.

16.5.4 SSH Services

The three main secured services provided by SSH are:

1. Secure Command Shell (Remote logon)
2. Secure File Transfer (SFTP)
3. Port Forwarding (SSH Tunneling)

Secure Command Shell

It is a remote login process. This service allows the user to edit files, view the content of directories, and access applications on the connected devices. The administrator can create user accounts, change file/directory permissions, and so on. Using secure command shell all the tasks that are feasible at a machine command prompt can now be performed securely for the remote computer.

Secure File Transfer

Secure file transfer is a separate protocol layered over the secure shell protocol to handle file transfer. SSH File Transfer (SFTP) encrypts both the user name/password and the file data being transferred. It uses the same port (port no. 22) as the secure shell server.

Port Forwarding (SSH Tunneling)

The port forwarding (or SSH Tunneling) service allows an unsecured TCP/IP-based application to be secured. A transport layer communication (TCP) uses a port number. The incoming TCP traffic is delivered to the appropriate application on the basis of the port number. An application can also employ multiple port numbers (e.g., FTP).

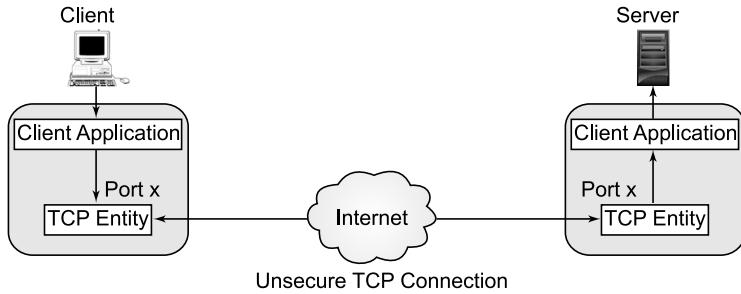


FIGURE 16.13 Unsecure TCP connection.

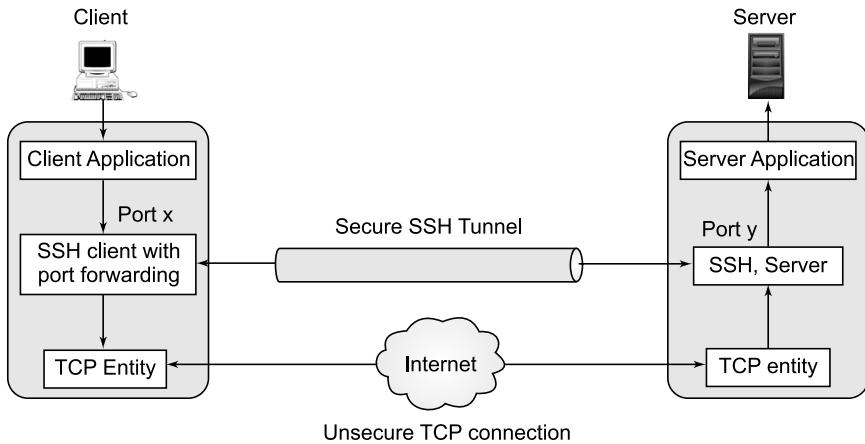


FIGURE 16.14 Communication via SSH tunnel.

As shown in Figure 16.13, a client application that is identified by the port number x and a server application identifies by the port number y . At some point the client application invokes the local TCP entity and requests a connection to the remote server on port y . The local TCP entity negotiates a TCP connection with the remote TCP entity as shown in Figure 16.14, such that the connection links local port x to the remote port y . To secure this connection SSH is configured so that SSH transport layer protocol establishes a TCP connection between the SSH client and server entities with TCP port numbers x and y respectively. A secure SSH channel is established over this TCP connection. The message from the client at port x is redirected to the local SSH entity and travels through the tunnel, where the remote SSH entity delivers the data to the secure application on port y .

Traffic in the other direction is similarly redirected. SSH provides two types of port forward:

1. **Local Forward:** It allows the client to set up a hijacker process. This process intercepts selected application-level traffic and redirects it from an unsecured TCP connection to a secure SSH tunnel.
2. **Remote Forwarding:** In remote forwarding the user's SSH client acts on the server's behalf. The client receiver's traffic with a given destination port number places the traffic on the correct port and sends it to the destination the user selects.

SUMMARY

- TLS is a protocol developed to provide security services such as confidentiality, authentication, and integrity between two communication applications.
- TLS and SSL are most widely recognized as the protocols that provide secure HTTP (HTTPS) for Internet transactions between web browsers and web servers.
- SSL provides for secure communication between client and server by allowing mutual authentication and the use of a digital signature for integrity and encryption for privacy.
- The three protocols within SSL are Handshake Protocols, Record Protocols, and Alert Protocols.
- The main security services such as confidentiality, authentication, and the data integrity of the TLS protocol are achieved through implementation of TLS protocols on two levels: TLS protocol and TLS handshake protocol.
- HTTPS is a communication protocol used on the Internet that has a layer of security added. It is a combination of the standard HTTP protocol and a security protocol called SSL/TLS.
- Secure Shell (SSH) is a secure protocol and the most common way of safely administering remote servers.
- SSH provides the three main secured services: secure command shell; secure file transfer (SFTP); and port forwarding (SSH Tunneling).

REVIEW QUESTIONS

1. In a Secured Socket Layer (SSL) connection, is the session key chosen by the client or the server?
 2. Justify the inclusion of an SSL layer in between the application layer and transport layer.
 3. Explain the handshake protocol actions of SSL.
 4. In a Secured Socket Layer (SSL), is the session key for the connection chosen by the client or the server? How is it communicated to the other party?
 5. How does SSL use TCP to provide end-to-end secure service? What is the record protocol operation in SSL?
 6. Describe SSL Architecture in detail.
 7. Mention four SSL Protocols.

MULTIPLE CHOICE QUESTIONS

- 12.** An endpoint of an inter-process communication flow across a computer network is called
 - (a) socket
 - (b) pipe
 - (c) port
 - (d) none of the above
- 13.** A socket-style API for windows is called
 - (a) wssock
 - (b) winsock
 - (c) win
 - (d) none of the above
- 14.** Which one of the following is a version of UDP with congestion control?
 - (a) Datagram congestion control protocol
 - (b) Stream control transmission protocol
 - (c) Structured stream transport
 - (d) None of the above
- 15.** A _____ is a TCP name for a transport service access point.
 - (a) port
 - (b) pipe
 - (c) node
 - (d) none of the above
- 16.** Transport layer protocols deals with
 - (a) application to application communication
 - (b) process to process communication
 - (c) node to node communication
 - (d) none of the above
- 17.** Which one of the following is a transport layer protocol?
 - (a) Stream control transmission protocol
 - (b) Internet control message protocol
 - (c) Neighbor discovery protocol
 - (d) Dynamic host configuration protocol

CHAPTER 17

NETWORK LAYER SECURITY

17.1 INTRODUCTION

The network layer is one of the most complex layers within the OSI seven-layer model. As a result, several OSI standards are required to specify transmission, routing, and internetworking functions for this layer. The ISO/IEC 8880 standard describes an overview of the network layer. Two other standards, ISO/IEC 8348 and 8648, define the network service and describe the internal organization of the network layer, respectively. The most recent standard published is ISO/IEC 11577, which describes the Network Layer Security Protocol (NLSP). The important functions of the network layer include handling protocols, switching, routing, forwarding, addressing, and error handling. It can even break up the message into smaller pieces so the whole thing will reassemble at the destination in one meaningful piece. The network layer also converts logical addresses into physical addresses.

In this chapter, the more commonly known IP Security protocol IPsec—a suite of protocols that provides security at the network layer—is examined. Before getting into the specifics of IPsec, let's consider first what it means to provide security at the network layer. The network layer would provide secrecy if all data carried by all IP datagrams were encrypted. This means that whenever a host wants to send a datagram, it encrypts the data field of the datagram before shipping it out into the network. In principle, the encryption could be done with symmetric key encryption, public key encryption, or with session keys that are negotiated using public key encryption. The data field could be a TCP segment, a UDP segment, an ICMP message, and so on. If such a network layer service were in place, all data sent by hosts, including e-mail, webpages, and control and management messages (such as

ICMP and SNMP) would be secured from intruders. Thus, such a service would provide a certain blanket coverage for all Internet traffic, thereby giving all of us a certain sense of security.

17.2 NEED OF NETWORK LAYER SECURITY

IP (Internet Protocol) security refers to the security mechanism implemented at the IP layer to ensure integrity, authentication, and confidentiality of data during transmission in the open environment. One of the weaknesses of the original IP is that it lacks any sort of general purpose mechanism for ensuring the authenticity and privacy of the data as it is passed over the internetwork. Since IP datagrams must usually be routed between two devices over unknown networks, any information in them is subjected to being intercepted and even possibly changed. The primary objective of the members in the IETF IP security (IPSec) working group is to improve the robustness of the cryptographic key-based security mechanism at the IP layer for the users who request security. IPsec is a framework of open standards for ensuring private, secure communication over IP networks through the use of cryptographic security services. IPsec is a suite of cryptographic based protection service and security protocols. Its security service includes:

- 1. Data confidentiality:** The sender can encrypt packets before transmitting them across the network.
- 2. Data Authentication:** The IPsec receiver can authenticate the source of IPsec packets sent. This service depends on data integrity.
- 3. Data integrity:** The IPsec receiver can authenticate packets sent by the IPsec sender to ensure that the data has not been altered during transmission.
- 4. Replay attack:** Protection against certain types of security attacks, such as replay attacks.
- 5. Negotiate security algorithms:** IPsec has the ability for devices to negotiate the security algorithms and key required to meet their security needs.
- 6. Security modes:** Two security modes transport and tunnel to meet different network needs.

17.3 INTERNET PROTOCOL SECURITY (IPSEC)

The technology that brings secure communication to IP is called IP security, generally abbreviated as IPsec (or IPSEC). It is not a single protocol, but rather a set of services and protocols that provide security to IP networks. It is defined by a sequence of several Internet standards. These services and protocols combine to provide various types of protection. Since IPsec works at the IP layer, it can provide these protections for any higher layers of the TCP/IP protocol suite without the need for additional security methods. This is one of the main features of IPsec. IPsec is a collection of RFCs which define the architecture, services, and different protocols as shown in Table 17.1.

17.3.1 Features of IPsec

1. IPsec is developed to work with different protocols such as TCP, UDP, ICMP, OSPF, and so on.
2. IPsec protects the entire packet presented to the IP layer, including the higher layer header.
3. The cryptanalysis in this case is very difficult, because the higher layer headers are hidden, which even includes the port number.
4. The concept of IPsec works from one network entity to another network entity, not from application process to application process. Hence security can be adopted without requiring changes to individual user computers/applications.
5. IP security protocol can also provide host-to-host security.
6. The most common use of IPsec provides a virtual private network (VPN), either between two locations (gateway-to-gateway) or between a remote user and an enterprise network (host-to-gateway).

TABLE 17.1 RFCs Representing the Architecture and Security Algorithms.

RFCs	Name	Details
2401	Security architecture for IP	It describes the architecture and general operation of the technology and shows how the different components fit together.
2402	IP Authentication Header (AH)	It defines IPsec AH protocol, which is used for ensuring data integrity and origin verification.

(continued)

RFCs	Name	Details
2403	The use of HMAC MD5-96 with ESP and AH	The encryption algorithms used by AH and Encapsulation Security Payload (ESP) protocols, MD5 and HMAC variant.
2404	HMAC, SHA1-96 within AH and ESP	Describes the IPsec ESP protocol, which provides data encapsulation for confidentiality.
2408	ISAKMP – Internet security Association and Key Management Protocol	It defines the methods for exchanging keys and negotiating security associations.
2409	Internet Key Exchange (IKE)	IKE protocol is used to negotiate security associations and exchange keys between devices for secure communication. It is based on ISAKMP and OAKLEY.
2412	OAKLEY determination protocol	Generic key exchange protocol.

17.3.2 Components of IPsec

IPsec provides security services at the IP layer of the TCP/IP model. These devices (either end-user hosts or intermediate devices such as routers or firewalls) want to engage in secure communications. They establish a secure path between themselves. That may traverse many insecure intermediate systems. This is accomplished by performing the following tasks:

1. The data communication between the two end user hosts and intermediate devices must be in a specific format and security protocols agreed upon.
2. The messages are encrypted using a specific encryption algorithm.
3. To decrypt the data the key must be exchanged between the devices.
4. Once these processes are completed, now the devices can use the specified protocol methods and keys previously agreed upon to encrypt data and send it across the network.

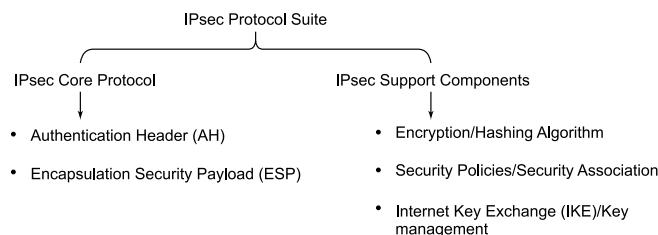


FIGURE 17.1 Components of IPsec.

Security under IPsec uses a number of different components. These components work together to provide the required security services. Among this the two main components are the IPsec core protocol and IPsec support components as shown in Figure 17.1.

17.4 IPSEC CORE PROTOCOL

The two main IPsec core protocols are Authentication Header (AH) and Encapsulation Security Payload (ESP), which provides the data encryption for confidentiality. If confidentiality is not required, the Authentication Header (AH) alone can provide security to an IP datagram. The implementation can be host-to-host, host-to-gateway, or gateway-to-gateway. But only host-to-host implementation is encouraged. The reason is that, in this case, the security gateway provides security service for the trusted hosts behind the gateway. The security attack can still arise when the trusted hosts become untrusted. In other words, the security can be violated for two communicating end users if the security (without confidentiality) does not cover completely the communicating path, but instead stops at the gateway, even though the security association (SA) is established. Certainly in any kind of implementation, the untrusted systems (*i.e.*, the systems that don't have the SA established) can't have the ability to attack data authentication (always referring to both data integrity and data origin authentication).

The IP Encapsulating Security Payload (ESP) header provides integrity, authentication, and confidentiality to IP datagrams. It can provide a mix of optional security. The ESP header can be applied alone, in combination with the IP Authentication Header (AH), or in a nested way, for example by using Tunnel-mode. The ESP header implementation can be host-to-host, host-to-gateway, or gateway-to-gateway. The ESP header is inserted after the IP header and before a higher-level protocol header (Transport mode) or the encapsulated IP header (Tunnel mode). Gateway-to-gateway ESP implementation, using encryption/decryption, is critical for building a Virtual Private Network (VPN) across an untrusted backbone in an open environment such as the Internet. Table 17.2 represents the security services provided by both AH and ESP.

TABLE 17.2 IPsec Security Services.

Security Services	AH	ESP (Confidentiality Only)	ESP (Confidentiality and Authentication)
Access Control	✓	✓	✓
Connectionless Integrity	✓		✓
Data Origin Authentication	✓		✓
Rejection of Replayed Packet	✓	✓	✓
Confidentiality		✓	✓
Limited Traffic Flow Confidentiality		✓	✓

17.4.1 Authentication Header (AH)

AH provides the authentication services for IPsec. The AH protects an IP packet and also adds an additional header to the IP packet, but it does not encrypt the packets to ensure confidentiality. It also provides protection against reply attacks, whereby a message is captured by an unauthorized user and resent.

17.4.2 Encapsulation Security Payload (ESP)

Encapsulation Security Payload provides confidentiality (encryption) and authentication (connectionless integrity and data origin authentication). ESP protects an IP packet but not the additional header that ESP adds. The AH and ESP may be applied alone or together to provide the required security services. The security they provide depends on the cryptographic algorithm they apply. Both these protocols are algorithm independent, which lets the new algorithms be added without affecting other parts of the implementation. A set of default algorithms used are specified by the RFCs as shown in table (17.1). Both AH and ESP protocols support two IPsec modes:

1. Transport mode
2. Tunnel mode

Transport Mode

IPsec transport mode is used for end-to-end communication, when communication happens between the client and server or between a workstation

and a gateway, for example in a remote desktop session from a workstation to a server.

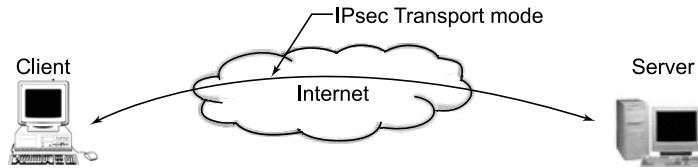


FIGURE 17.2 IPsec in transport mode.

Transport mode protects an IP load in upper layer protocols such as UDP and TCP protocols. AH and ESP intercept packets from the transport layer that are intended for the network layer protect the transport header and provide the configured security. It is a default mode for IPsec that when transport mode is used, IPsec encrypts only the IP payload. It provides the protection of an IP payload through AH and ESP headers.

Tunnel Mode

With tunnel mode the entire IP packet is protected by IPsec. This means in tunnel mode the IPsec wraps the original packet, encrypts it, adds a new IP header, and sends it to the other side of the VPN tunnel. Tunnel mode is most commonly used between gateways or at an end station to a gateway.

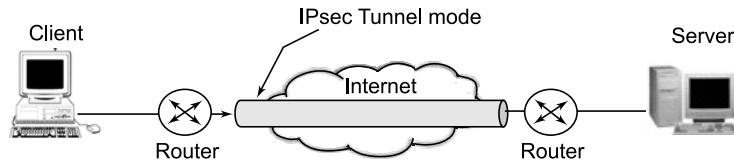


FIGURE 17.3 IPsec tunnel mode.

Tunnel mode protects data by encapsulating entire packets that are de-capsulated by a security gateway. In tunnel mode, an IPsec header (AH and ESP header) is inserted between the IP header and the upper layer protocol.

17.5 OPERATIONS OF TRANSPORT AND TUNNEL MODE

The basic functions of transport and tunnel mode are explained in the following sections. Tunnel mode protects the original IP datagram as a whole,

header and all, while transport mode does not. Thus the order of the headers in transport and tunnel mode when used with AH and ESP is:

Transport mode: IP header, IPsec header (AH and ESP), IP payload (including transport header).

Tunnel mode: New IP header, IPsec header (AH and ESP), old IP header, IP payload.

TABLE 17.3 Possible Authentication and Encryption Combinations for a Connection.

Encryption		Authentication	
Protocol	Algorithm	Protocol	Algorithm
None	None	ESP or AH	Any of the following algorithms: <ul style="list-style-type: none"> • HMAC_MD5 • HMAC_SHA1 • AES128_XCBC_96 • HMAC_SHA2_256_128 • HMAC_SHA2_384_192 • HMAC_SHA2_512_256 • AES_GMAC_128 • AES_GMAC_256
ESP	Any of the following algorithms: <ul style="list-style-type: none"> • DES • 3DES • AES_CBC KeyLength 128 • AES_CBC KeyLength 256 	ESP or AH	Any of the following algorithms: <ul style="list-style-type: none"> • HMAC_MD5 • HMAC_SHA1 • AES128_XCBC_96 • HMAC_SHA2_256_128 • HMAC_SHA2_384_192 • HMAC_SHA2_512_256
ESP	Any of the following algorithms: <ul style="list-style-type: none"> • AES_GCM_16 KeyLength 128 • AES_GCM_16 KeyLength 256 	ESP	NULL (AES_GCM provides built-in authentication)

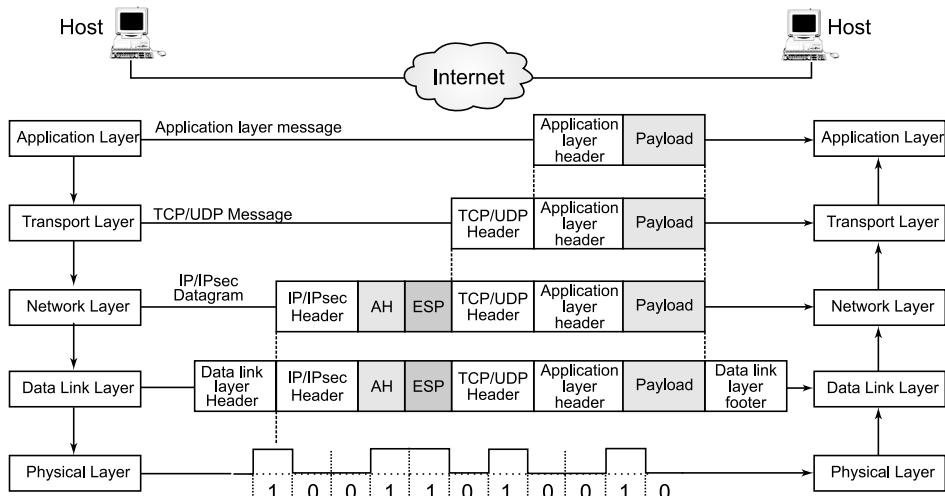
In transport mode it is integrated with IP and used to transport the layer (TCP/UDP) message directly. After processing, the datagram has just one IP header that contains the AH and ESP IPsec headers.

TABLE 17.4 Combinations of IPsec Modes.

Mode	IPsec Protocol	IP Version
Transport mode	AH	IPv4 IPv6
	ESP	IPv4 IPv6
Tunnel mode	AH	IPv4 IPv6
	ESP	IPv4 IPv6

The exact order of the header either in transport or in tunnel mode depends on which version of IP is being used (IPv4 or IPv6). IPv6 uses an extension header that must be arranged in a particular way when IPsec is used. The header position is also depends on which IPsec protocol is being used (AH and ESP). Thus there are three variables and eight basic combinations of modes as in Table 17.4.

The transport mode requires that the IPsec be integrated into IP, because AH and ESP must be applied as the original IP packaging is performed on the transport layer message. This often is the choice for implementations requiring end-to-end security with hosts that run IPsec directly.

**FIGURE 17.4** IPsec transport mode operation with AH and ESP.

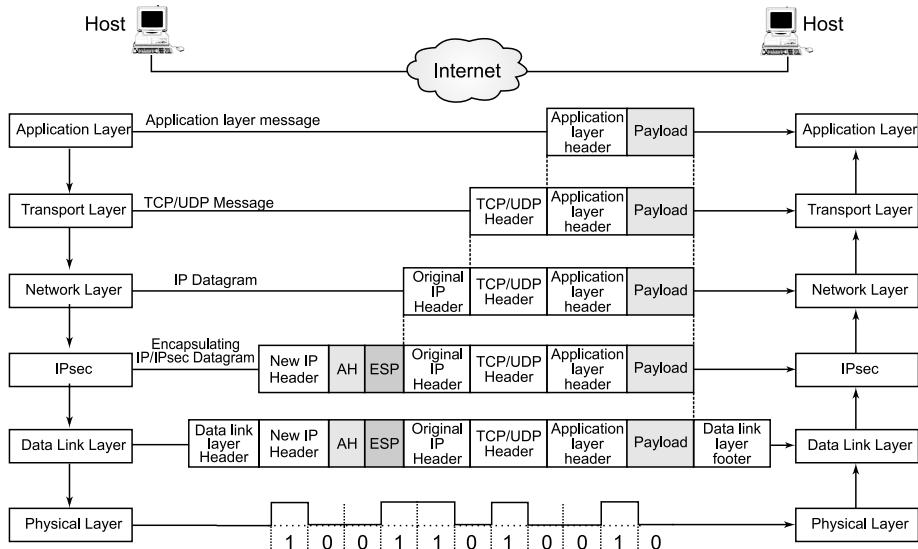


FIGURE 17.5 IPsec tunnel mode operation with AH and ESP.

The tunnel mode encapsulates the complete IP datagram, thereby forming a virtual tunnel between IPsec capable devices. This IP datagram is passed to IPsec, where a new IP header is created with the AH and ESP IPsec header added.

17.5.1 Authentication Header for IPv4 and IPv6

The AH provides support for data integrity and authentication of IP packets. Authentication is performed by computing a cryptographic hash-based message authentication code over nearly all the fields of the IP packets. The data integrity field ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network devices to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed on the Internet. The frame format of IPsec AH is as shown in Figure 17.6. The AH contains five fields with authentication data and it's injected between the original IP header and the payload.

- 1. Next Hdr. (8 bit):** This identifies the types of header immediately following this header.
- 2. Payload Length (8 bit):** This defines the length in 32-bit words of the whole AH header minus two words. For example: The default length of the authentication data field is 96 bits, or three 32 bit words, with a three

word fixed header; there are a of total six words in the header and the payload length field has a value of 4.

3. **Reserved (16 bit):** This field is reserved for future use and must be zero.
4. **SPI (Security Parameter Index – 32 bits):** It is a 32-bit value field that identifies the Security Association (SA) for this datagram, relative to the Destination IP Address contained in the IP header.
5. **Sequence Number (32 bits):** This is a monotonically increasing identifier that's used to assist in anti-reply protection. This value is included in the authentication data, so modification (during transit) is detected.
6. **Authentication Data (variable):** This is the integrity check value calculated over the entire packet. It includes most of the headers. The recipient re-computes the same hash. A mismatched value during comparison marks the packet as either damaged in transit or not having the proper secret key. These are discarded.

IPsec AH frame format: AH is one of the two main security protocols used in IPsec. It provides authentication of either all or part of the contents of a datagram through the addition of a header. The location of the header always depends on the mode (transport or tunnel) and the version of IP (IPv4 or IPv6).

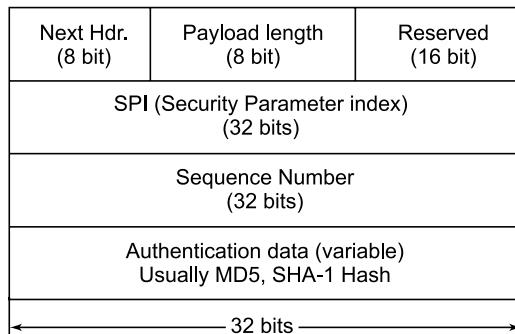


FIGURE 17.6 AH frame format.

Using the AH Header: The AH header may be used in transport mode or tunnel mode. In transport mode, the AH header is appended before the IP header of an IP datagram and is only used for end-to-end implementation. The reason is that only higher-layer protocols and selected IP header fields are protected. In transport mode, the AH header is inserted after the

IP header and before a high-layer protocol (but before other IPsec headers such as ESP Header if that header is already inserted before the higher-layer protocol). For gateway security implementation, tunnel mode is required. In this case, the AH header protects the entire inner IP packet, including the entire IP header.

Non-repudiation, referring to being able to tell if the sender denies sending data, can be provided by some authentication algorithms (e.g., asymmetric algorithms when both sender and receiver keys are used in the authentication calculation) used with the AH Header. The default authentication algorithm is keyed MD5, which, like all symmetric algorithms, cannot provide non-repudiation by itself, because the sender's key is not used in the computation. Confidentiality protection is not provided by the AH Header. The working of AH is denoted in the following steps:

1. The IP header and the data payload are hashed.
2. The hash is used to build a new AH header, which is appended to the original packet.
3. The new packet is transmitted to the IPsec peer router.
4. The peer router hashes the IP header and data payload, extracts the transmitted hash from the AH header, and compares the two hashes. The hashes must match exactly. Even if one bit is changed in the transmitted packet, the hash output on the received packet will change and the AH will not match.

IPv4 AH

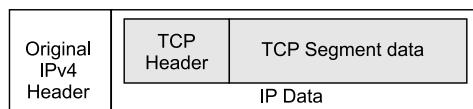


FIGURE 17.7 Original IPv4 datagram format.

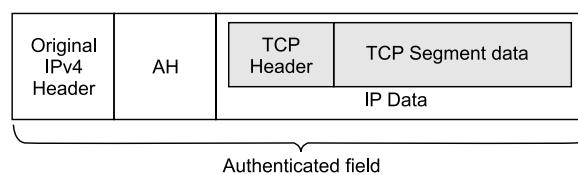


FIGURE 17.8 IPv4 AH datagram format – IPsec transport mode.

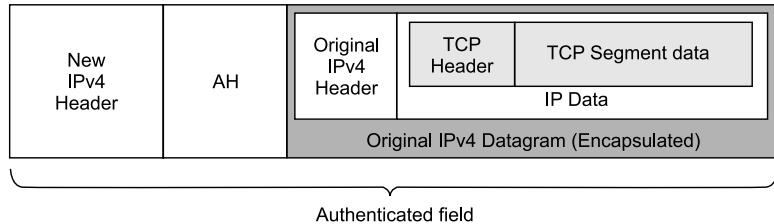


FIGURE 17.9 IPv4 AH datagram format – IPsec tunnel mode.

IPv6 AH: In the IPv6 version of the Internet Protocol, the authentication header is inserted into the IP datagram as an extension header, following the normal IPv6 rules for extension heading linking. It is linked by the previous header (extension or main), putting into its next header field the assigned value for the header. The AH header then links to the next extension header or the transport layer header using the next header field.

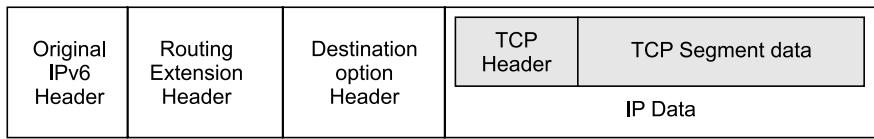


FIGURE 17.10 Original IPv6 datagram format.

When AH is applied in transport mode, it is simply added as a new extension header that goes between the routing extension header and the destination option header. In transport mode the AH is placed into the main IP header and appears before any destination option header containing the option intended for the final destination, and before an ESP header if present but after any other extension headers.

In tunnel mode it appears as an extension header of the new IP datagram that encapsulates the original one being tunneled. The entire datagram is encapsulated into the new IPv6 datagram that contains the AH. In both cases the next header fields are used to link each header one to the next.

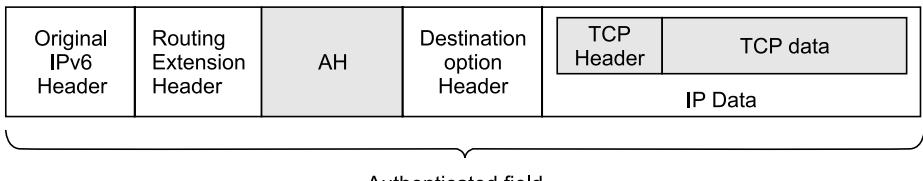


FIGURE 17.11 IPv6 AH datagram format – IPsec transport mode.

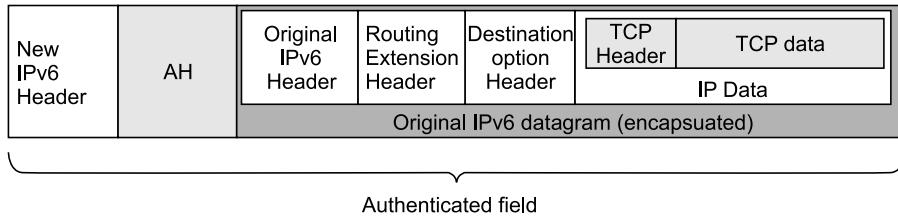


FIGURE 17.12 IPv6 AH datagram format – IPsec tunnel mode.

17.5.2 Encapsulating Security Payload (ESP) for IPv4 and IPv6

The IPsec AH protocol provides integrity and authentication services to IPsec capable devices. It can verify that messages are received intact from other devices. To meet the requirements of confidentiality, a mechanism commonly used is message encryption.

IPsec provides confidentiality service through Encapsulating Security Payload (ESP). ESP can also provide data origin authentication, connectionless integrity, and anti-replay service. Confidentiality can be selected independent of all other services. There are two modes for providing confidentiality using ESP. One is transport mode and the other is tunnel mode. Tunnel mode encapsulates an entire IP datagram within the ESP header. Transport mode encapsulates the transport layer frame inside the ESP. When incorporating the ESP into the IP system (IPv4, IPv6, or extension) the protocol header immediately preceding the ESP header will contain the value 5 in its protocol (IPv4) or the next header (IPv6) field.

Encapsulating Security Payload fields: ESP divides its fields into three components:

- ESP Header:** This contains two fields, the security parameter index (SPI) and the sequence number, and comes before the encrypted data.
- ESP Trailer:** This section is placed after the encrypted data. It contains padding that is used to align the encrypted data, through a padding and pad length field.
- ESP Authentication Data:** This field contains an Integrity Check Value (ICV), computed in a manner similar to how the AH protocol works, for when ESP's optional authentication feature is used.

ESP Header: The ESP header is designed to provide a mix of security services in IPv4 and IPv6. ESP may be applied in combination with AH

or in a nested fashion. Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host. The ESP header is inserted after the IP header and before the next layer header (transport mode) or before an encapsulated IP header (tunnel mode). Figure 17.13 represents the ESP frame format with the following fields.

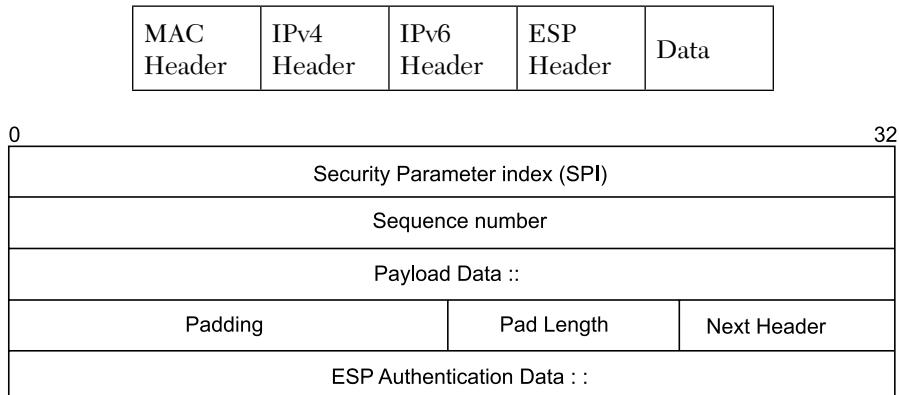


FIGURE 17.13 ESP frame format.

SPI – Security Parameter Index (32 bits): A 32-bit value that is combined with the destination address and security protocol type to identify the security association to be used for this datagram.

Sequence Number (32 bits): It is a counter field initialized to zero when a Security Association (SA) is formed between two devices, and then incremented for each datagram using that SA. This is used to provide protection against replay attacks.

Padding Data (variable): The encrypted payload data consisting of a higher level message or encapsulated IP datagram. It may also include additional information such as an initialization vector, required by certain encryption methods.

Padding (Variable 0–255 bytes): Additional padding bytes included as needed for encryption or for alignment.

Pad Length (8 bits): The number of bytes in the preceding padding field.

Next Header (8 bits): Containing the protocol number of the next header in the datagram. Used to chain together headers.

ESP Authentication Data (variable): This field contains the integrity check value (ICV) resulting from the applications of the optional ESP authentication algorithm.

Figure 17.14 depicts the security services provided by ESP. It represents the filed name, the data bit size, encryption, and authentication coverage.

ESP Operation: ESP operation includes three basic steps performed by ESP: header calculation and placement, trailer calculation and placement, and ESP authentication field calculation and placement.

Section	Field Name	Size	Encryption coverage	Authentication coverage
ESP Header	SPI	32 bits		
	Sequence Number	32 bits		
Payload	Payload Data	Variable		
	Padding	Variable (0–255 bytes)		
	Pad Length	8 bits		
ESP Trailer	Next Header	8 bits		
	ESP Authentication Data	Variable		

FIGURE 17.14 Security services provided by ESP.

Header Calculation and Placement:

IPv4 ESP

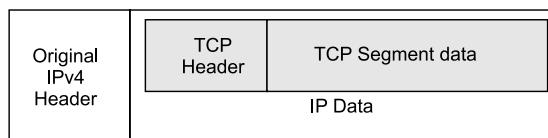


FIGURE 17.15 Original IPv4 datagram format.

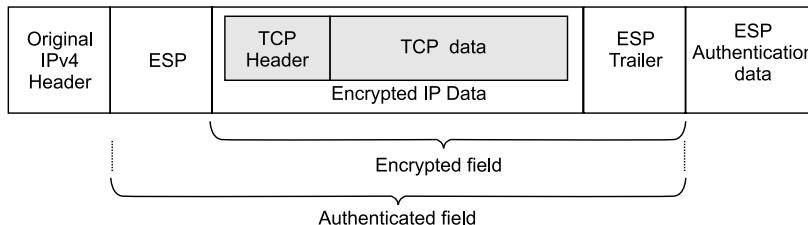


FIGURE 17.16 IPv4 ESP datagram format IPsec transport mode.

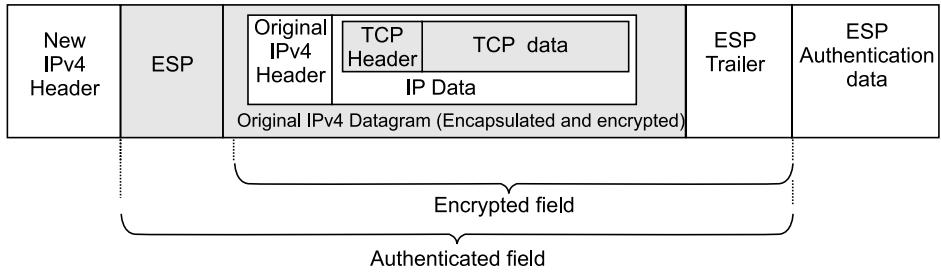


FIGURE 17.17 IPv4 ESP datagram format – IPsec tunnel mode.

IPv4: As with the authentication header, the ESP header is placed after the normal IPv4 header. In transport mode it appears after the original IPv4 header. When this datagram is processed in transport mode, the ESP header is placed between the IPv4 header and data with the ESP trailer and ESP authentication data following it. In tunnel mode, the entire original IPv4 datagram is surrounded by this ESP content, rather than just the IPv4 data.

IPv6 ESP

IPv6: The ESP is inserted into the IP datagram as an extension header, following the normal IPv6 rules for extension header linking. As in Figure 17.19 the transport mode appears before a destination option header containing options intended for the final destination, but after any other extension header, if present. In tunnel mode it appears as an extension header of the new IP datagram that encrypts the original one being tunneled, as in Figure 17.20.

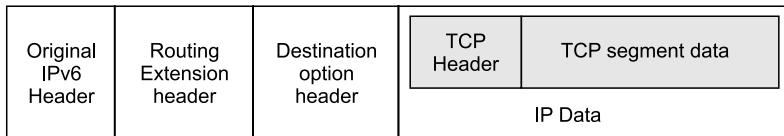


FIGURE 17.18 Original IPv6 datagram format.

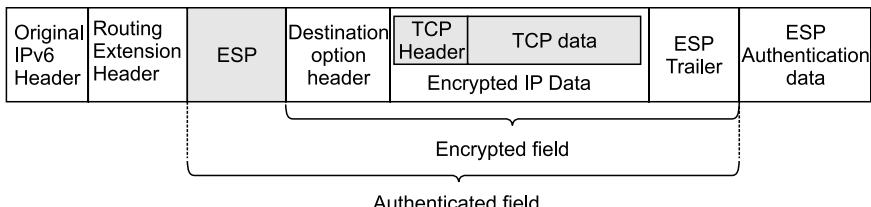


FIGURE 17.19 IPv6 ESP datagram format IPsec transport mode.

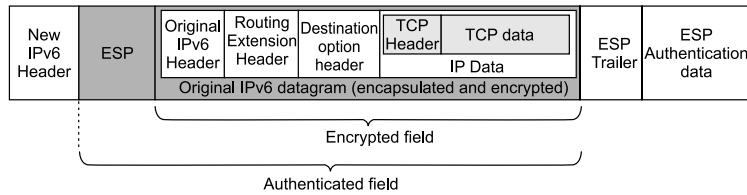


FIGURE 17.20 IPv6 ESP datagram format – IPsec tunnel mode.

In IPv6, the AH is inserted into the IP datagram as an extension header. It is linked by the previous header. In transport mode the AH is placed into the main IP header and appears before any destination option header that contains an option intended for the final destination and before the ESP header if present, but after any other extension header. In tunnel mode, it appears as an extension header of the new IP datagram that encapsulates the original one being tunneled.

Trailer Calculation and Placement

The ESP Trailer is added to the data to be encrypted. ESP then performs the encryption. The payload (TCP/UDP message or encapsulated IP datagram) and the ESP trailer are both encrypted, but the ESP Header is not. Also consider that any other IP headers that appear between the ESP header and the payload are also encrypted. In IPv6, this can include a Destination Options extension header.

ESP Authentication Field Calculation and Placement

If the optional ESP authentication feature is used, the authentication field is computed over the entire ESP datagram (except the Authentication Data field itself). This includes the ESP header, payload, and trailer. The format of the ESP sections and fields is described in Figure 17.14 and shown in Figure 17.21.

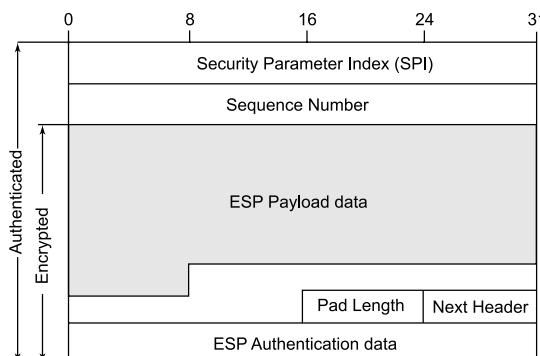


FIGURE 17.21 IPsec Encapsulating Security Payload (ESP) format.

17.6 SECURITY ASSOCIATION AND ENCRYPTION STRENGTH

17.6.1 Security Association (SA)

The mechanisms mentioned previously in IPsec use cryptography, and thus require a certain number of parameters (encryption algorithms used, keys, selected mechanisms, etc.) on which the communicating parties must agree. In order to manage these parameters, IPsec uses the Security Association (SA) concept.

A Security Association (SA) is a set of security information relating to a given network connection or set of connections. The concept of a SA is fundamental to both the IP Encapsulating Security Payload (IP ESP) and the IP Authentication Header (IP AH). The combination of a given Security Parameter Index (SPI) and destination address uniquely identifies a particular SA. This model is required by the implementation of IPsec, which may also support other options.

A Security Association normally includes the following essential parameters:

- Authentication algorithm and algorithm mode being used with the IP AH.
- Key(s) used with the authentication algorithm in use with the AH.
- Encryption algorithm, algorithm mode, and transform being used with the IP ESP.
- Key(s) used with encryption algorithm in use with the ESP.
- Presence/absence and size of a cryptographic synchronization or initialization vector field for the encryption algorithm.

It is also recommended to have:

- Authentication algorithm and mode used with the ESP transform (if any in use).
- Authentication key(s) used with the authentication algorithm that is part of the ESP transform (if any).
- Lifetime of the key or time when key change should occur.
- Source address(es) of the SA; might be a wild-card address if more than one sending system shares the same SA with the destination.
- Sensitivity level (e.g., Secret or Unclassified) of the protected data (required for all systems claiming to provide multilevel security, recommended for all other systems).

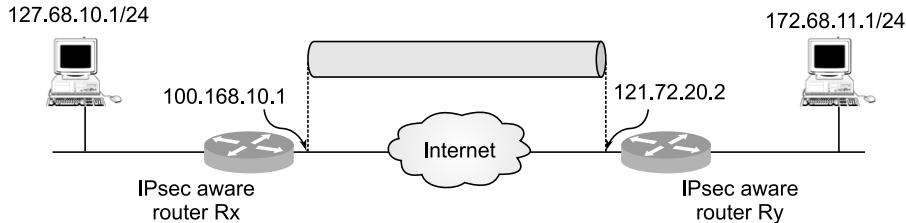


FIGURE 17.22 Security associations between two routers.

The SAs between two routers Rx and Ry involved in IPsec communication are shown in Figure 17.22. The router Rx stores the following details for SA:

- Security Parameter Index (SPI) – 32 bit identifier for SA
- Origin SA interface (100.168.10.1)
- Destination SA interface (121.72.20.2)
- Type of encryption used (e.g., 3DES with CBC)
- Encryption Key
- Type of integrity check used (e.g., HMAC with MD5)
- Authentication Key

A SA is normally one-way. An authenticated communications session between two hosts will normally have two SPIs in use (one in each direction).

17.6.2 Encryption Strength – Algorithms

The encryption and authentication algorithms used for IPsec are the heart of the system. They are directly responsible for the security the system can provide. There are however major drawbacks in this area. As the Internet is a global network, the IP should provide uniform security everywhere. Many countries, however, either restrict or forbid the use or export of encryption algorithms. This means that the IPsec must be able to balance between legal restrictions in the use of strong encryption and authentication, and places where encryption is allowed.

All hosts claiming to provide IPsec services must implement the AH with at least the MD5 algorithm using a 128-bit key as specified in the AH RFC. An implementation may support other authentication algorithms in addition to keyed MD5. All ESP implementations must support the use of the Data Encryption Standard (DES) in Cipher-Block Chaining (CBC) mode as detailed in the ESP specification. Other cryptographic

algorithms and modes may also be implemented in addition to this mandatory algorithm and mode. MD5 and DES-CBC should be set as default algorithms.

17.7 KEY DISTRIBUTION

Security Associations rely on keys for the authentication and encryption algorithms to be used with AH and ESP. A manual key distribution mechanism is a simplest method to distribute keys, but it does not scale and only works in very small, static systems. It can be used for LANs and some firewall systems, but it is a short-term approach.

17.7.1 Security Policies

The protections offered by IPsec are based on policy choices defined in the Security Policy Database (SPD). This database is established and maintained by a user, a system administrator, or an application installed by them. The SPD contains an ordered list of rules, each rule being identified by one or more selectors that define the set of IP traffic affected by this rule. The possible selectors are the set of information available in the IP and transport layer headers. The selectors define the granularity for applying the security services and directly influence the corresponding number of SAs. If security services are to be applied to the traffic corresponding to an entry, the entry includes a SA (or SA bundle) specification, listing the IPsec protocols, modes, and algorithms to be employed, including any nesting requirements.

17.7.2 Key Management

Key management is the main challenge in cryptography. The process key management covers the generation, distribution, storage, and deletion of the keys.

This section will explain some of the important concepts related to key management.

- (i) **Key Hierarchy:** Several key types can be defined depending on their role.
- (ii) **Keys Encrypting Keys:** These keys are exclusively used to encrypt other keys.

- (iii) **Master Keys:** These keys are not used to encrypt but only to generate other keys by derivation. A master key can thus be used for example to generate two keys; one for encryption and one for a signature.
- (iv) **Session key:** These keys generally have a shorter life span (the key can even be changed with each message). Session keys are used to encrypt data and are thus located at the bottom of the hierarchy. Because of their slowness, asymmetric algorithms are rarely used for encrypting data, and session keys are thus generally secret keys.

17.7.3 Key Exchange Protocol Properties

Diffie, Van Oorschot, and Wiener defined the concept of a secure authenticated key exchange protocol. A protocol is said to be secure if the two following conditions are verified in every case where the protocol is carried out and one of the two parties, for example Alice, carries out the protocol honestly and accepts the identity of the other party:

- When Alice accepts Bob's identity, the recordings of the messages exchanged by the two parties correspond (i.e., the messages were not modified during their transfer).
- It is materially impossible for anybody except Alice and Bob to find the exchanged key.

The previous property represents the minimum necessary for any key exchange protocol. However, other properties of key exchange protocols can be desirable:

- The property known as Perfect Forward Secrecy (PFS) is guaranteed if the discovery, by an opponent, of long-term secrets does not compromise the session keys that were previously generated: the past session keys cannot be recovered with the knowledge of long-term secrets. It is generally considered that this property also ensures that the discovery of a session key compromises neither the long-term secrets nor the other session keys. Perfect Forward Secrecy can be provided, for example, by generating the session keys using the Diffie-Hellman protocol, wherein the Diffie-Hellman exponentials are short-term values.
- The property known as Back Traffic Protection is provided if the generation of each session key is done in an independent way: the new keys do not depend on the preceding keys and the discovery of a session key

makes it possible neither to find the last session keys nor to deduce the future keys.

- A protocol provides direct authentication if, at the end of the protocol, the values being used to generate the shared secret are authenticated or if each party proved that he knew the session key. In contrast, authentication is said to be indirect if it is not guaranteed at the end of the protocol, but depends on the capacity of each party to use, during the exchanges that follow, the keys that were agreed upon.
- A protocol guarantees identity protection if an eavesdropper will not be able to know the identities of the communicating peers.
- Finally, the use of time stamps in order to avoid replay is subject to controversy, mainly because of its too-great dependence on synchronized clocks.

17.7.4 Authenticated Key Exchange Protocols Developed for IP

There are a number of authenticated key exchange protocols which impose different requirements and provide different properties.

For a secured key exchange using IP, it is necessary to consider the difference between the connection oriented and connectionless protocols. In case of connection oriented protocols an authenticated key establishment protocol is used before the communication. The resulting key is then used to secure the IP traffic. The disadvantage of this approach is that it requires the establishment and the management of a pseudo session layer under IP, whereas IP is a connectionless protocol. In case of a connectionless protocol, a stateless authenticated key establishment protocol is used, which does not require any connection. This is feasible through an in-band protocol, where the key that is used to encrypt the packet is transmitted with it, encrypted with the recipient's public key, for example. The disadvantage of this system is that it adds data to each transmitted packet.

17.7.5 Key Management for IPsec: ISAKMP and IKE

The security services in the IP layer manage the necessary parameters (keys, protocols, etc.) by using Security Association. ISAKMP (Internet Security Association and Key Protocol Management) is designed to negotiate, establish, modify, and delete security associations and their attributes. ISAKMP is designed to support SA for protocols of the IP level or any upper level. It

can be used to negotiate, in the form of SAs, the parameters relating to any security mechanism: IPsec, TLS, and so on.

ISAKMP was designed to:

- Work independently of the security mechanisms for the data related to key management and is transported separately.
- Define a framework to negotiate security associations, but it does not impose anything about the parameters that compose them.
- ISAKMP has two phases that allow for a clear separation between ISAKMP traffic protection and Security Association negotiation for a given protocol.
 - During the first phase, a set of security related attributes is negotiated, the identities of the parties are authenticated, and some keys are generated. These elements constitute a first security association, known as the ISAKMP SA. It will be used to secure all the following ISAKMP exchanges.
 - The second phase is used to negotiate the security parameters related to an SA to establish for a given security mechanism (AH or ESP). This provides the confidentiality and authentication during the exchange.

Exchange Types

It is a specification of the set of messages composing a given exchange type. Each exchange type is conceived to provide a certain number of security services, like anonymity, perfect forward secrecy, mutual authentication, and so on. Generally ISAKMP specification contains five exchange types.

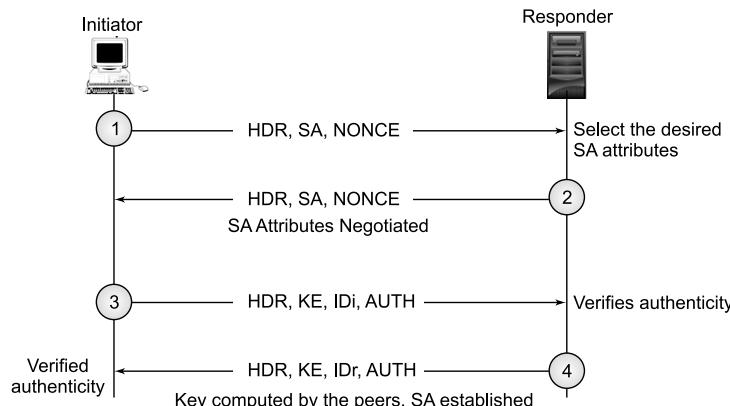
1. Base Exchange
2. Identity Protection Exchange
3. Authentication Only Exchange
4. Aggressive Exchange
5. Informational Exchange

The notations for these exchanges are given in Table 17.5; out of this only the last exchange (Informational Exchange) is mandatory.

TABLE 17.5 Notations for the ISAKMP Exchanges.

HDR	ISAKMP Header
SA	Security Association Payload
KE	Key Exchange Payload
ID	Identity Payload
AUTH	Authentication Payload (HASH or SIG)
NONCE	Nonce Payload
*	Means that the message is encrypted

- Base Exchange:** It is designed to allow the key exchange and authentication-related information to be transmitted together. The process of combining these two will reduce the number of round trips at the expense of not providing identity protection. These identities are exchanged before a shared secret has been established and, therefore, they cannot be encrypted.

**FIGURE 17.23** Base exchange.

The data exchanged during messages 3 and 4 are protected using AUTH (Authentication payload).

- Identity Protection Exchange:** It is designed to separate the key exchange information from the identity and authentication-related information. With this it is providing protection of the communicating identities along with two additional messages. These identities are exchanged under the protection of a previously established shared secret.

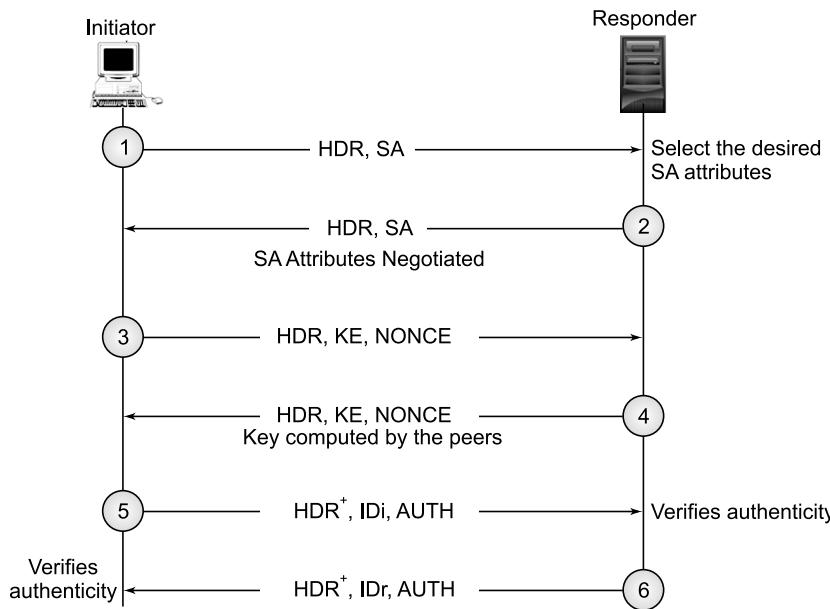


FIGURE 17.24 Identity protection exchange.

- 3. Authentication Only Exchange:** It is designed to allow only authentication-related information to be transmitted. This allows it to perform only authentication without the computational expense of computing keys. This exchange is particularly useful during the second phase, where it will be protected by the security services negotiated during the first phase.

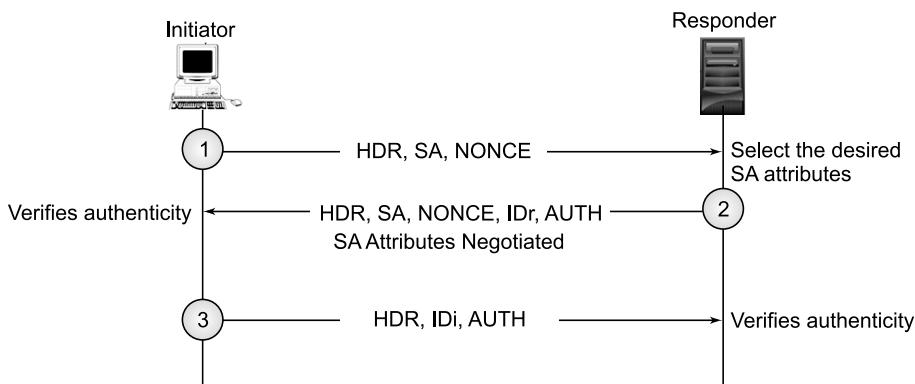


FIGURE 17.25 Authentication only exchange.

4. Aggressive Exchange: It is designed to allow the security association, key exchange, and authentication-related payloads to be transmitted together. This reduces the number of round-trips at the expense of not providing identity protection. Additionally, there can be only one Proposal and one Transform offered (i.e., no choices) in order for the aggressive exchange to work.

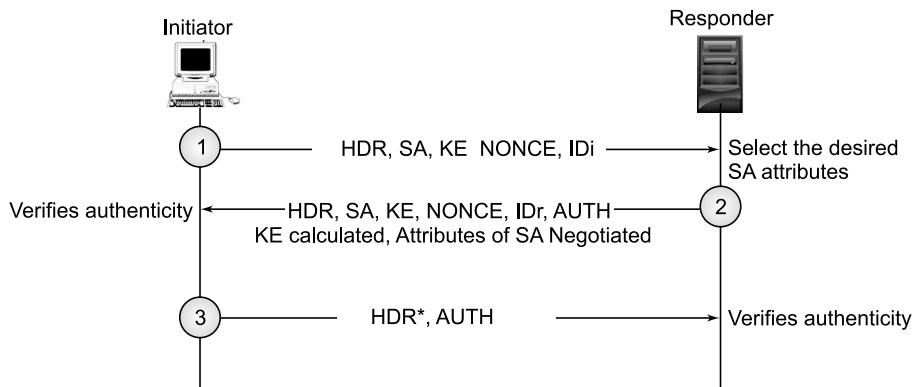


FIGURE 17.26 Aggressive exchange.

5. Informational Exchange: It is designed as a one-way transmittal of information that can be used for security association management. It uses either a Notification payload or a Delete payload. This message is only protected by the ISAKMP SA if it has been established.

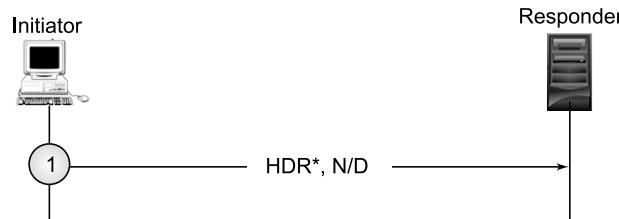


FIGURE 17.27 Informational exchange.

Internet Key Exchange (IKE): IKE is a default key management protocol used for IPsec. It is the main part of the IPSec implementation, and is used to negotiate secret key material between two parties, called the initiator and responder. The key material is the result of the protocol and is used to create the Security Associations (SAs) that define how the traffic between

the two hosts are to be protected. The IKE also provides mutual authentication by authenticating both of the parties to each other; both of the parties need to provide a digital signature in the key exchange protocol that the other party will verify.

IKE is a hybrid protocol based on the Internet Security Association and Key Management Protocol (ISAKMP), described in RFC 2408. The IKE protocol implements parts of two other key management protocols: Oakley, described in RFC 2412, and SKEME. The protection policy within SAs is negotiated and established with the help of the ISAKMP protocol, and keying material (session keys for encryption and packet authentication) is agreed on and exchanged with the use of Oakley and SKEME protocols.

ISAKMP: The Internet Security Association and Key Management Protocol is a protocol framework that defines payload formats, the mechanics of implementing a key exchange protocol, and the negotiation of a security association. The ISAKMP establishes a secure management session between IPsec peers, which is used to negotiate IPsec SAs. ISAKMP provides the means to do the following:

- Authenticate the remote peer
- Cryptographically protect the management session
- Exchange information for key exchange
- Negotiate all traffic protection parameters using configured security policies

Therefore, the goal of ISAKMP is the establishment of an independent security channel between authenticated peers in order to enable a secure key exchange and the negotiation of IPsec SAs between them.

Oakley: A key exchange protocol that defines how to derive authenticated keying material. Oakley is originally a freeform protocol that allows each party to proceed with the exchange at its own speed. IKE borrowed this idea from Oakley, and defines the mechanisms for key exchange in different modes over the IKE (ISAKMP) session. Each protocol produces a similar result—an authenticated key exchange, yielding trusted keying material used for IPsec SAs. Oakley, within IKE, determines AH and ESP keying material (authentication and encryption session keys) for each IPsec SA automatically, and by default uses an authenticated Diffie-Hellman algorithm to accomplish this.

Skeme: A key exchange protocol that defines how to derive authenticated keying material with rapid key refreshment.

Internet Key Exchange includes the following four modes:

1. Main mode
2. Aggressive mode
3. Quick mode
4. New Group mode

Where main mode and aggressive mode are used during phase 1, quick mode is a phase 2 exchange. New Group Mode is a little apart; it is neither a phase 1 exchange nor a phase 2 exchange, but it can take place only once as ISAKMP SA has been established. It is used to agree on a new group for a future Diffie-Hellman Exchange.

Phase 1: Main Mode and Aggressive Mode - Uses a main or aggressive mode exchange. Used to negotiate the IKE SA (establish a secure IKE session).

In Figure 17.28, Alice and Bob want to talk using IKE. Therefore they must agree on a common IKE protection suite. The initiator (Bob) proposes several protection suites and the responder (Alice) chooses one of the offered protection suites. The selection of security policy is made by the responder according to its priorities in the configuration. In the example, Bob proposes three protection suites, and Alice chooses the second one (based on her local policy configuration). Peers must agree exactly on the protection suite. If they do not, no common policies exist between peers, and the IKE session will be terminated.

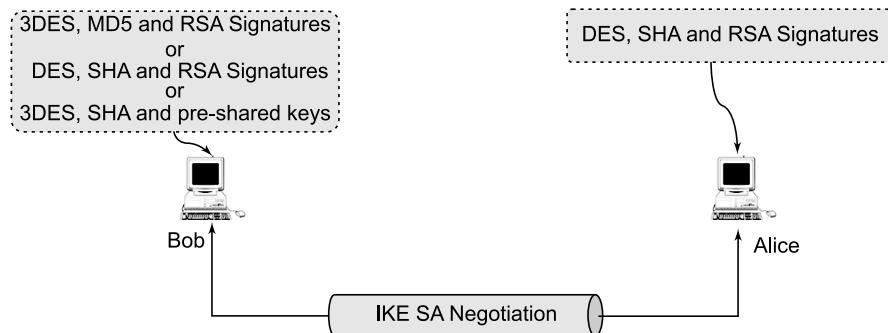


FIGURE 17.28 Phase 1 – main mode and aggressive mode.

Phase 2: Quick Mode - Uses a quick mode exchange (negotiates and creates IPsec protection policy).

IKE phase 2 is used to negotiate and establish SAs of other protocols (IPsec's AH and ESP, IP PCP (payload compression protocol), and so on). Phase 2 needs an established IKE SA (produced in IKE phase 1 to protect the IKE session) to operate, and only operates in one defined mode, the quick mode. The IKE initiator presents a list of (IPsec) policy proposals and the IKE responder chooses an acceptable proposal according to its locally defined policy. When the policy between peers is agreed upon, the keying material is agreed upon, and IPsec SAs are established. In Figure 17.29, Alice and Bob want to protect their traffic with IPsec, and an IKE SA is already established between them. The initiator (Bob) proposes several IPsec security policies, and the responder (Alice) chooses one of the offered policies. The selection of security policy is made by the responder according to its priorities in the configuration. In the example, Bob proposes two IPsec security policies and Alice chooses one of them (the one that has the highest priority in her configuration). After successful negotiation, keying material is exchanged, and the IPsec SAs are established to protect network traffic.

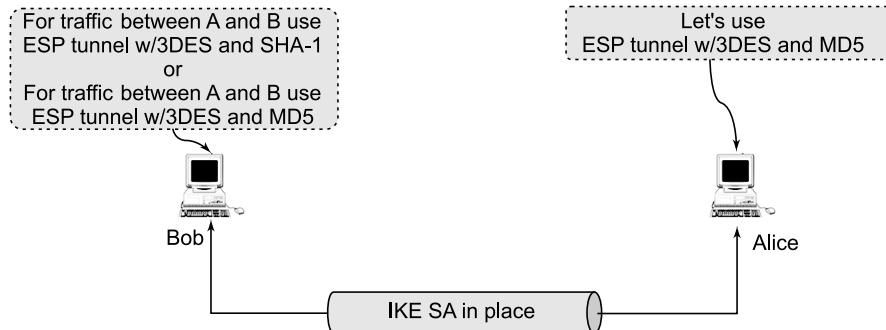


FIGURE 17.29 Phase 2 – quick mode.

IKE Peer Authentication: One of the most important factors in the IKE SA negotiation is the mutual authentication of peers. Each peer must be sure that it is talking to the correct peer before negotiating traffic protection (IPsec) policies with it. This mutual authentication is accomplished via IKE's two-way authentication methods. IKE provides three defined methods for two-way authentication:

- Authentication using a pre-shared secret
- Authentication using RSA encrypted nonces
- Authentication using RSA signatures

IKE Session Encryption: Besides peer authentication, IKE also provides privacy and integrity of the IKE session. The IKE session can be encrypted with the use of DES or 3DES. The keying material for encryption is generally derived from the initial Diffie-Hellman exchange that occurs between peers at the start of the IKE session. When IKE is negotiating in main mode, peer identity is also encrypted, whereas aggressive mode does not hide peer identities.

IKE Session Integrity: IKE uses the HMAC functions to guarantee the integrity of an IKE session. Usually, a choice between keyed SHA-1 and MD5 is available. The keying material for HMAC functions is also generally derived from the initial Diffie-Hellman exchange. IKE initially performs an ephemeral Diffie-Hellman exchange at the start of the IKE session. From that exchange, peers get shared keying material, which is then used for IKE encryption and integrity functions. The strength of that keying material can be traded off for faster performance, by choosing lower key sizes for Diffie-Hellman exchanges. The key length (strength) of Diffie-Hellman exchanges can be changed with the use of different DH groups. When an IKE session's lifetime expires, a new Diffie-Hellman exchange is performed between peers and the IKE SA is re-established.

SUMMARY

- The important functions of a network layer includes handling protocols, switching, routing, forwarding, addressing, and error handling.
- One of the weaknesses of the original IP is that it lacks any sort of general purpose mechanisms for ensuring the authenticity and privacy of the data as it is passed over the internetwork.
- IP security (IPsec) is a suite of protocols that provide security at the network layer.
- IPsec security services include Data confidentiality, Data authentication, Data integrity, Replay attack, Negotiate security algorithms, and Security modes.
- IPsec is developed to work with different protocols such as TCP, UDP, ICMP, OSPF, etc.
- The two main IPsec core protocols are AH and ESP.
- AH provides the authentication of services for IPsec.
- ESP provides confidentiality and authentication.

- Both AH and ESP protocols support two IPsec modes: Transport Mode and Tunnel Mode.
- Transport mode protects an IP load in upper layer protocols such as UDP and TCP protocols.
- With tunnel mode the entire IP packet is protected by IPsec.
- SA is a set of security information relating to a given network connection or set of connections.
- All hosts claiming to provide IPsec services must implement AH with at least an MD5 algorithm using 128 bits, and ESP with DES-CBC.
- SA rely on keys for the authentication and encryption algorithms to be used with AH and ESP.
- IKE is a default key management protocol used for IPsec. It is the main part of the IPsec implementation and is used to negotiate secret key materials between two parties called initiator and responder.

REVIEW QUESTIONS

1. What is IPSec and what are the two modes of IPSec operation? What types of security services are provided by IPSec?
2. How will you justify the need of IP security along with other security measures?
3. Discuss five benefits of IPSec as a security protocol.
4. Explain the architecture of IPSEC.
5. Discuss the various components of IPSec architecture. What is an anti-reply mechanism in context of IPSec?
6. How are transport and tunnel nodes used in IPsec Encapsulating Security Payload (ESP) service?
7. Give the IPSec ESP format.
8. AH in IPsec is responsible to prevent replay attacks?
9. What parameters identify a Security Association (SA) and what parameters characterize the nature of a particular SA?
10. Briefly explain the ESP protocol along with its different modes of operation.

11. What is the security purpose for the fields, such as sequence number, of an IPSec packet?
12. Briefly explain Security Associations in IPSec.

MULTIPLE CHOICE QUESTIONS

1. IPSec is designed to provide security at the:
(a) transport layer (b) network layer
(c) application layer (d) session layer
2. In tunnel mode, IPsec protects the:
(a) entire IP packet (b) IP header
(c) IP payload (d) none of the above
3. A network layer firewall works as a:
(a) frame filter (b) packet filter
(c) both (a) and (b) (d) none of the above
4. The Encrypted Security payload extension header is new in:
(a) IPv4 (b) IPv5 (c) IPv6 (d) IP
5. Performance, reliability, and security are criteria of an:
(a) efficient network (b) intranet
(c) ethernet (d) none of the above
6. The network layer is concerned with:
(a) bits (b) frames
(c) packets (d) none of the above
7. Which one of the following is not a function of the network layer?
(a) Routing (b) Error control
(c) Congestion control (d) None of the above

- 8.** AH provides the _____ services for IPSec.
(a) authentication (b) confidentiality
(c) integrity (d) all the above
- 9.** ESP provides _____ and _____ for IPSec.
(a) confidentiality; integrity (b) confidentiality; authentication
(c) integrity; authentication (d) none of the above
- 10.** A new IP header is added in _____ mode.
(a) transport mode (b) tunnel mode
(c) both (a) and (b) (d) none
- 11.** The _____ is a default key management protocol used for IPSec.
(a) key Distribution Center (b) AH
(c) internet Key Exchange (IKE) (d) none of the above

CHAPTER 18

DATA LINK LAYER SECURITY

18.1 INTRODUCTION

Data link layer functions are relatively different from the other layers of the OSI model. The data link layer is considered as the most trusted layer. The data link layer is used for filtering, access control lists, and authentication and application control to limit access at the higher layers of the protocol design. Yet the data link layer is highly prone to several attacks. An attacker may use different techniques to attack the data link layer. In all cases there is the attempt to compromise confidentiality, authentication, or availability of information. Some of the common data link layer attacks are:

1. ARP Spoofing
2. MAC Flooding
3. Port Stealing
4. DHCP Attacks
5. Other attacks

18.2 ARP SPOOFING

ARP spoofing is a type of attack in which an attacker sends a falsified (masquerade) ARP message over a local area network (LAN). This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is

connected to the authentic IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify, or even stop data in transit. This attack can only occur on LANs that utilize the address resolution protocol. ARP spoofing attacks are used to steal sensitive information. Beyond this, they are also often used to facilitate the other attacks such as:

- Denial of service attack
- Session hijacking
- Man-in-the-middle attack

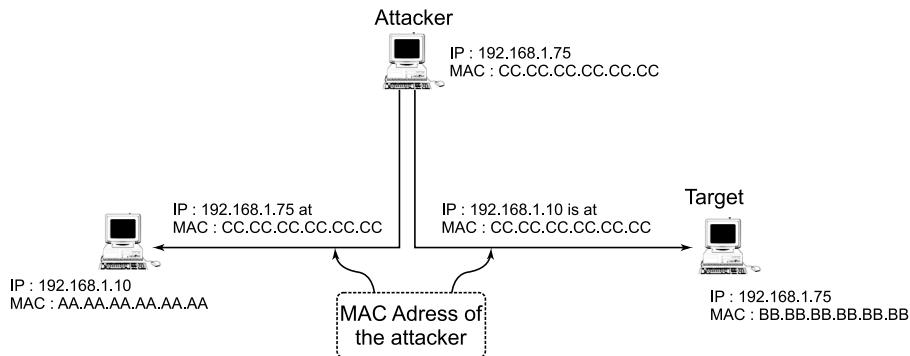


FIGURE 18.1 Man-in-the-middle attack using ARP spoofing.

18.2.1 ARP Spoofing Attack

The following steps are used by an attacker to perform an ARP spoofing attack:

- Step 1:* The attacker opens an ARP spoofing tool and sets the tool's IP address to match the IP subnet of the target.
Some of the popular ARP spoofing software are Arpspoof, Cain & Abel, Arpoison, and Ettercap.
- Step 2:* The attacker uses the ARP spoofing tool to scan for the IP and MAC addresses of hosts in the target subnet.
- Step 3:* The attacker chooses its target and begins sending ARP packets across the LAN that contains the attacker's MAC address and targets the IP address.

Step 4: As other hosts on the LAN catch the spoofed ARP packets, data that they send to the victim will go to the attacker. With this the attacker can steal data or launch a more sophisticated follow-up attack.

18.2.2 Protection against ARP Spoofing Attack

ARP spoofing can be detected and prevented, and the network users can be protected by adopting the following techniques:

1. **Packet filtering:** This method allows filtering out and blocking packets with conflicting source address information.
2. **Avoid trust relationships:** Organizations should develop protocols that rely on trust relationships a little as possible. Trust relationships rely only on IP addresses for authentication, making it significantly easier for attackers to run ARP spoofing attacks when they are in place.
3. **Using secured network protocols:** TLS-Transport Layer Security, SSH-Secure Shell, HTTPS-HTTP Secure, and other secure communication protocols strengthen ARP spoofing attack prevention by encrypting data prior to transmission and authenticating data when it is received.
4. **Use of ARP spoofing attack software:** This software works by inspecting and certifying data that appears to be spoofed before it is transmitted.

18.3 MAC FLOODING

MAC flooding is one of the most common network attacks. It is not a method of attacking any host computer in the network, but it is the method of attacking the network switches. However, the victim of the attack is a host computer in the network. The result of this attack causes the switch to enter a state called fail-open mode, in which all incoming packets are broadcasted out on all parts, instead of just down the correct port as per the normal operation. A malicious user could then use a packet sniffer running in promiscuous mode to capture sensitive data from other computers, which would not be accessible were the switch operating normally.

18.3.1 MAC Flooding Attack

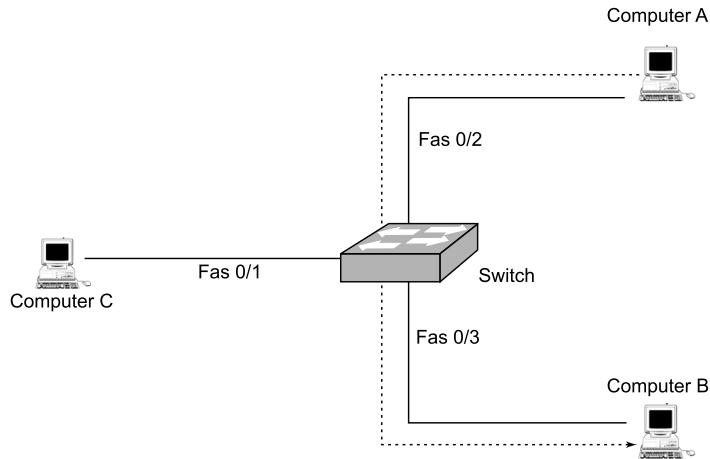


FIGURE 18.2 Two PCs communicating through a switch.

Three PC, Computer A, Computer B, and Computer C, are connected to a switch as in Figure 18.2. In a normal situation, when Computer A sends a packet to Computer B, Computer C does not view the packet sent between Computer A and Computer B. This is because these three computers are connected to a switch and not a hub.

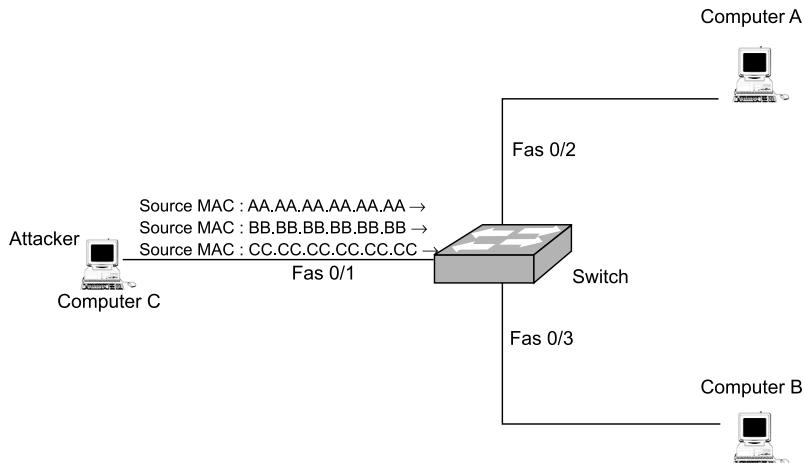


FIGURE 18.3 Malicious computer C floods the MAC address.

Under a MAC flooding attack, the switch behavior is different. The attacker floods the switch with packets, each with a different source MAC address. The memory of the switch where the MAC address is stored (content addressable memory) becomes full; the switch works like a hub, and so if Computer A sends a packet to Computer B, the packet will reach Computer C too as shown in Figure 18.4.

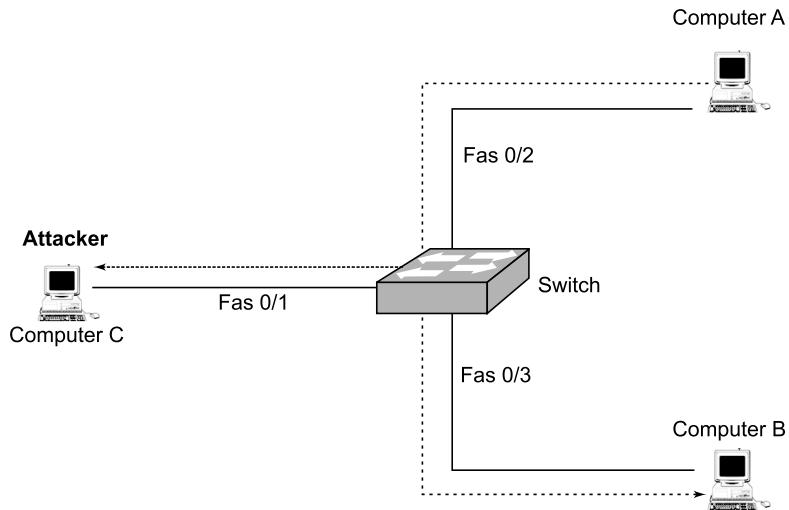


FIGURE 18.4 Packet reaching the malicious computer C.

18.3.2 Protection against MAC Flooding Attacks

The following methods are generally used to secure MAC addresses from MAC flooding attacks:

1. **Static source MAC address:** These are manually configured by using the switch port, a port security MAC address, and a MAC address interface configuration command stored in address table, and is added to the switch running configuration.
2. **Dynamic secure MAC address:** The MAC address is dynamically learned, stored only in the address table, and removed when the switch restarts.
3. **Sticky secure MAC address:** These can be dynamically learned or manually configured, stored in a database table, and added to the running configuration. If these addresses are saved in the configuration file,

the interface does not need to dynamically relearn them when the switch restarts.

18.4 PORT STEALING

This attack exploits the vulnerability of switch devices in a LAN. In this attack someone steals traffic that is directed to another port of an Ethernet switch. This allows someone to receive packets that were originally directed to another computer. It does so by making the switch believe that the attacker's port is the correct destination for the packet. Port stealing attacks exploit this ability of the switch. The attacker floods the switch with forged ARP frames with the target host's MAC address as the source address.

18.4.1 Port Stealing Attack

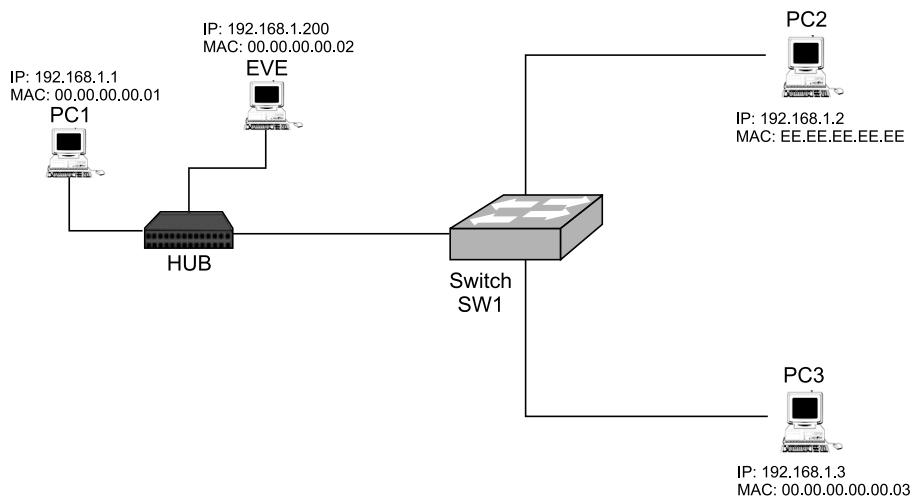


FIGURE 18.5 Port stealing attack.

Let us say that an attacker Eve (behind switch port 1) wants to steal PC2's (victim) port on the switch (port 2). For this Switch SW1 has to be deceived into thinking that PC2 is behind the same switch port as Eve (port1). Eve performs this by sending an Ethernet packet with its source MAC address EE.EE.EE.EE.EE as in Figure 18.5. That is, Eve has to spoof the victim's MAC address or in other words to forge an Ethernet packet with a

spoofed source MAC address. Eve has to send any packets (ARP, IP, ICMP, empty UDP/TCP, DNS, etc.) with the spoofed source MAC address, and the switch will update the same in the switch forwarding database (FDB) improperly.

18.5 DHCP ATTACKS

The Dynamic Host Configuration Protocol (DHCP) is used to dynamically allocate IP addresses to computers for a specific time period. By a DHCP attack it is possible to attack a DHCP server by causing denial of service in the network or by impersonating the DHCP server. There are a number of attacks related to the DHCP server. The two most important DHCP attacks discussed here are DHCP Spoofing and DHCP Starvation.

18.5.1 DHCP Spoofing

In this type of attack, the attacker inserts a rogue DHCP server in the network to sniff LAN traffic. When a client broadcasts the DHCP DISCOVERY packet, the rogue DHCP server replies before the actual or genuine DHCP server replies. The DHCP OFFER packet by the rogue DHCP server consists of an IP address and other information such that one of the attacker's machines is designated as the default gateway to the client. All the traffic from the client computer now goes to the attacker's computer before it travels any further. The attacker now sniffs all the packets to see the content. By placing a rogue DHCP server on the network, an attacker can provide clients with addresses and other network information. Because DHCP responses typically include default gateway and DNS (Domain Name Systems) server information, network attackers can supply their own systems as the default gateway and DNS server as shown in Figure 18.6, resulting in a man-in-the-middle attack.

The different steps involved in DHCP spoofing are given as follows:

- Step 1:* The attacker inserts a rogue DHCP server in the network.
- Step 2:* The client broadcast a DHCP DISCOVERY packet to get the DHCP OFFERS.
- Step 3:* The rogue DHCP server responds before the actual DHCP server is able to respond.
- Step 4:* The client accepts the invalid IP configuration.
- Step 5:* All the packets from the client go to the attacker's computer before going to the actual DHCP server.

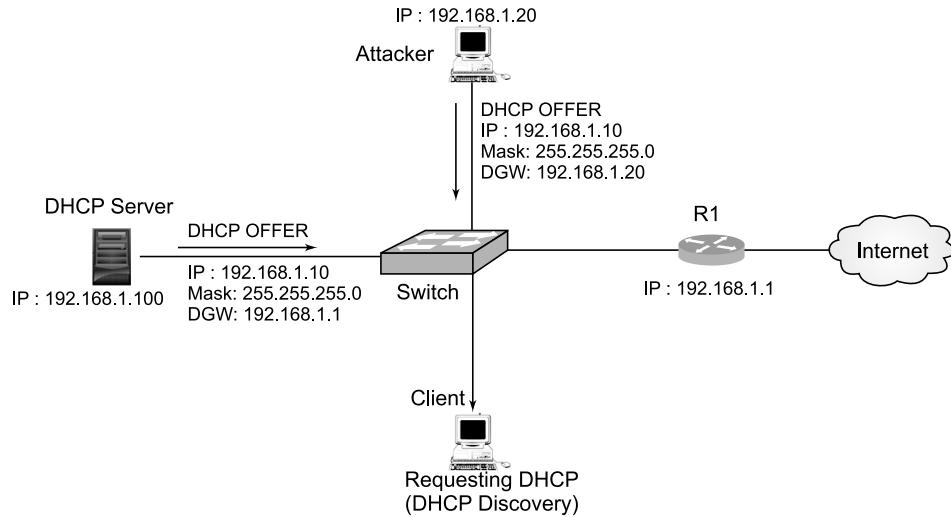


FIGURE 18.6 Rogue DHCP server attack.

18.5.2 DHCP Starvation

In a DHCP starvation attack, the attacker keeps on requesting the IP configuration from the DHCP server through different slave machines or by spoofing his MAC address until the DHCP server's entire pool of addresses is exhausted. This blocks the genuine client from getting the IP configuration from the DHCP server, and hence it cannot be connected to the network. This is easily achieved with attack tools such as "the gobbler." If enough requests are sent, the network attacker can exhaust the address space available to the DHCP server for a period of time.

18.5.3 Protection against a DHCP Attack

DHCP snooping: It is a DHCP security feature that provides network security by filtering untrusted DHCP messages and by building and maintaining a DHCP snooping binding database, which is also called a DHCP snooping binding table.

Port scanning: Port security can be used to mitigate DHCP starvation attacks by limiting the number of MAC addresses allowed on a port. The port security also allows a specific MAC address for each port or to permit a limited number of MAC addresses.

In addition to the previous attacks on the data link layer, other attacks are:

- Content Addressable Memory (CAM) overflow
- VLAN Hopping
- Spanning Tree Protocol (STP) manipulation

18.6 CAM OVERFLOW ATTACKS

A common Layer 2 or switch attack is MAC flooding, resulting in a switch's CAM table overflow, which causes flooding of regular data frames out of all switch ports. This attack can be launched for the malicious purpose of collecting a broad sample of traffic or as a denial of service (DoS) attack.

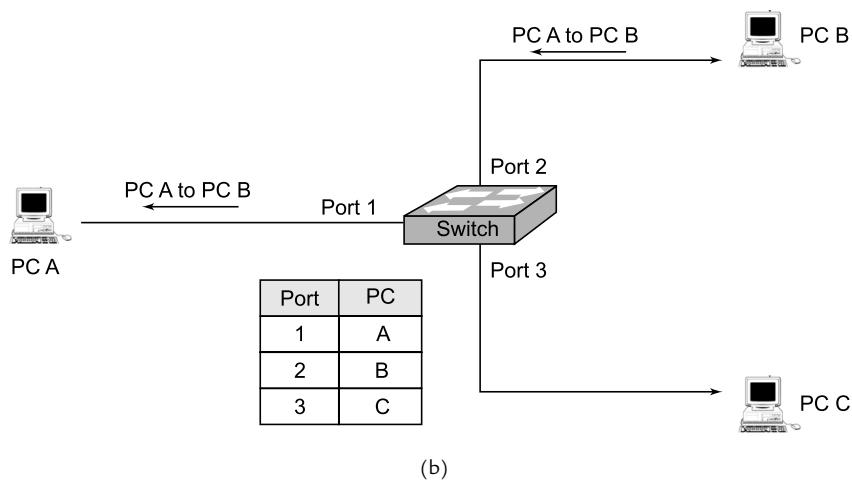
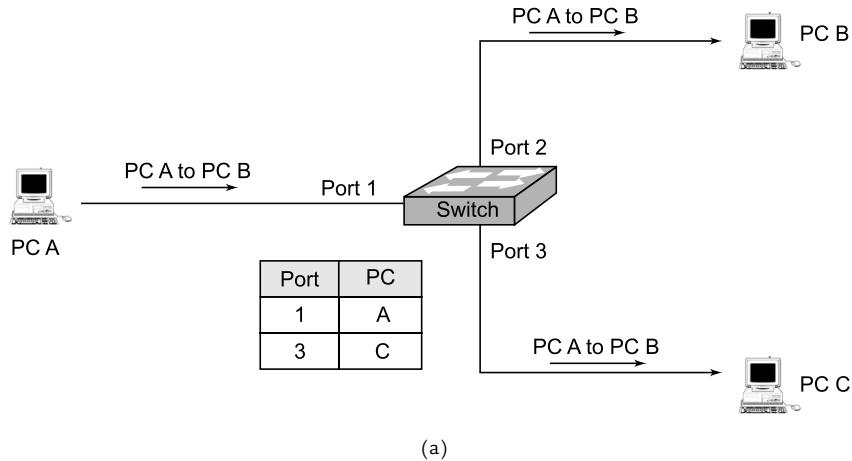
A switch's CAM tables are limited in size and therefore can contain only a limited number of entries at any one time. A network intruder can maliciously flood a switch with a large number of frames from a range of invalid source MAC addresses. If enough new entries are made before old ones expire, new valid entries will not be accepted. Then, when traffic arrives at the switch for a legitimate device located on one of the switch ports that could not create a CAM table entry, the switch must flood frames to that address out all of the ports. This has two adverse effects:

- The switch traffic forwarding is inefficient and voluminous and could potentially slow down the network for all users.
- An intruding device can be connected to any switch port and capture traffic not normally seen on that port.

If the attack is launched, the MAC address table (also referred to as Content Addressable Memory or CAM table) would be full when the majority of devices are powered on. Then frames from those legitimate devices cannot create MAC address table entries when the power is on. If this represents a large number of network devices, the number of MAC addresses flooded with traffic will be high, and any switch port will carry flooded frames from a large number of devices.

If the initial flood of invalid MAC address table entries is a one-time event, the switch eventually ages out older, invalid MAC address table entries, allowing new, legitimate devices to create entries. Traffic flooding ceases and might never be detected, even though the intruder might have captured a significant amount of data from the network.

In the data link layer of the TCP/IP model, a switch delivers an Ethernet frame based on the MAC address. A control address table maintains a list of the switch ports and the destination MAC address by port. This table enables the switch to uniquely deliver information to the intended MAC address. By delivering a frame based on the MAC address, a switch offers considerable security over a hub. A hub simply broadcasts the frame to all ports. An attacker on a hub can listen to the traffic of anyone else connected to the hub.



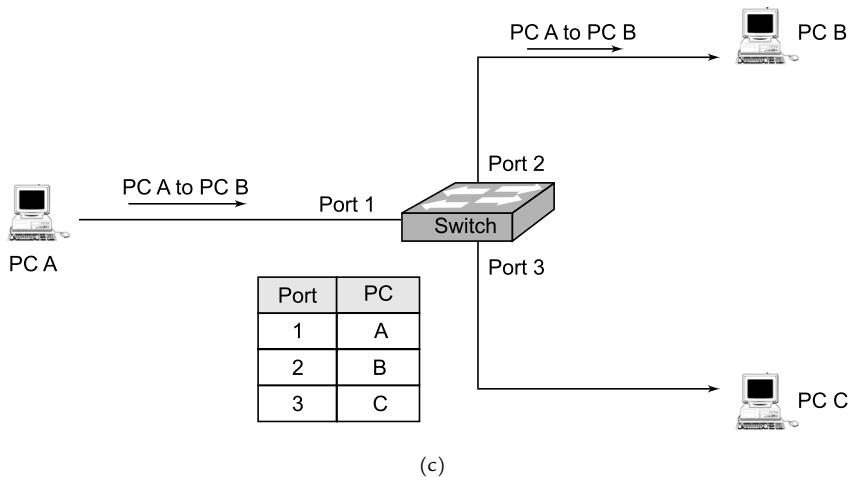
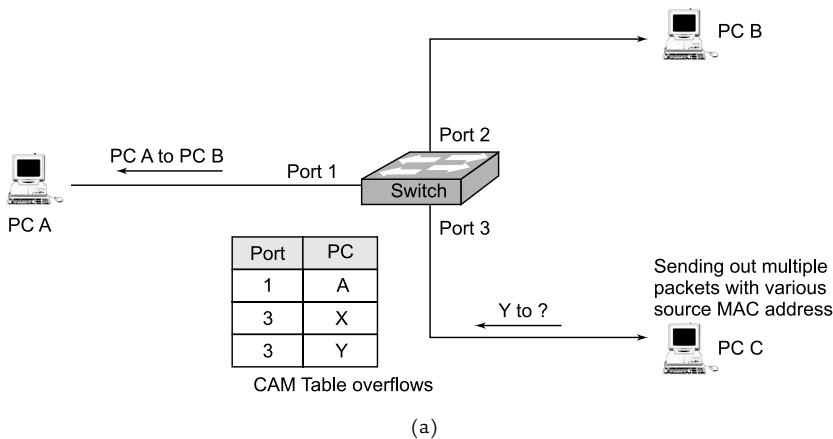


FIGURE 18.7 CAM table overflow attack.

A CAM overflow attack essentially turns a switch into a hub. To succeed, an attacker floods the CAM table with new MAC port mapping. When the CAM table is exhausted beyond the fixed memory, it no longer knows how to deliver based on MAC port binding. Therefore it begins broadcasting Ethernet frames to maintain the flow of traffic. Once the switch begins broadcasting, any connected attacker can hear the traffic that flows through the switch. The CAM table operation and CAM table overflow are as shown in Figure 18.7.

In a CAM overflow attack, an attacker sends thousands of bogus MAC addresses from one port, which looks like valid host's communication to the switch. One of the most popular tools used for this type of attack is Macof.



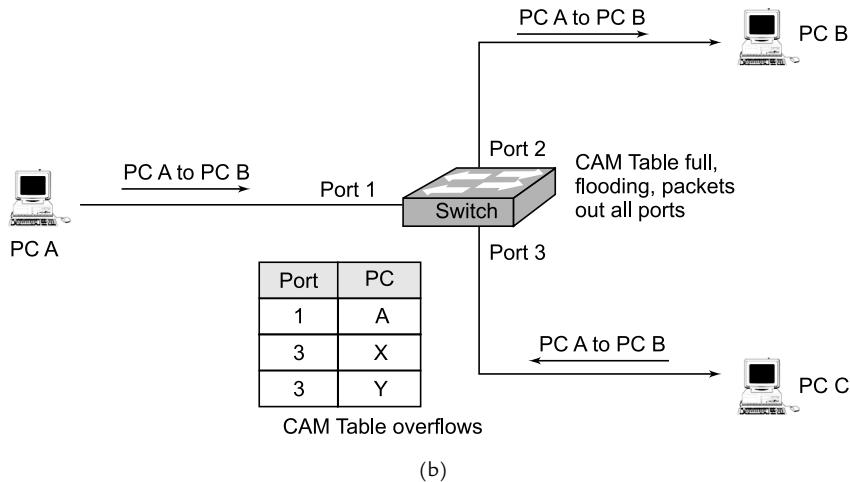


FIGURE 18.8 Flooding packets out all ports.

As Figure 18.8 shows, MAC flooding occurs in this progression; the following steps describe the MAC flooding attack progression.

- Step 1:* Switch forwards traffic based on valid MAC address table entries.
- Step 2:* Attacker (MAC address C) sends out multiple packets with various source MAC addresses.
- Step 3:* Over a short time period, the CAM table in the switch fills up until it cannot accept new entries. As long as the attack is running, the MAC address table on the switch remains full.
- Step 4:* Switch begins to flood all packets that it receives out of every port so that frames sent from Host A to Host B are also flooded out of Port 3 on the switch.

18.6.1 Protection against CAM Overflow Attack

Port security is a feature supported by the switches that restricts a switch port to a specific set or number of MAC addresses. Those addresses can be learned dynamically or configured statically. The port then provides access to frames from only those addresses. If, however, the number of addresses is limited to four but no specific MAC addresses are configured, the port enables any four MAC addresses to be learned dynamically, and port access is limited to those four dynamically learned addresses. The port security feature called sticky learning, available on some switch platforms, combines the features of dynamically learned and statically configured addresses. When

this feature is configured on an interface, the interface converts dynamically learned addresses to sticky secure addresses.

18.7 VLAN HOPPING

In a VLAN hopping attack, an attacker generates traffic with a VLAN ID of an end system it cannot normally reach and sends the traffic. Normally VLANs work by attaching a packet with an identification header. A port can receive only those packets that are of the VLAN. The VLAN information may be carried between switches in the LAN using a trunk port. The trunk ports have access to all VLANs by default. They route traffic for multiple VLANs across the same physical link. Further, the attacker may try to initiate a switch in order to negotiate trunking and send and receive traffic between VLANs.

One of the main limitations of data link layer security is the variety of mechanisms by which packets that are sent from one VLAN may be intercepted or redirected to another VLAN. This is known as VLAN hopping. The VLAN hopping attack allows attackers to bypass a layer 3 device (routers) when communicating from one VLAN to another. A VLAN hopping attack works by taking advantage of an incorrectly configured trunk port. VLAN hopping attacks do not work on a single switch, because in this case the frame will never be forwarded to the destination. In a multi-switch environment, a trunk link would be exploited to transmit the packet. A VLAN hopping attack is classified as switch spoofing and double tagging.

- **Switch spoofing:** In switch spoofing, the network attacker configures a system to spoof itself as a switch by emulating either ISL (inter switch linked) or 802.1q and DTP signaling. This makes the attacker appear to be a switch with a trunk port and therefore a member of all VLANs.
- **Double tagging:** This type of VLAN spoofing attack involves tagging the transmitted frames with two 802.1q headers. Most of the switches in VLAN perform only one level of decapsulation of packets. So when the first switch sees the double tagged frame, it strips the first tag off the frame and then forwards it with the inner 802.1q tag to all switch ports on the attacker's VLAN as well as to all trunk ports. The second switch forwards the packet based on the VLAN ID in the second 802.1q header. This type of attack works even if the trunk ports are set to off. The VLAN hopping with double tagging is shown in Figure 18.9.

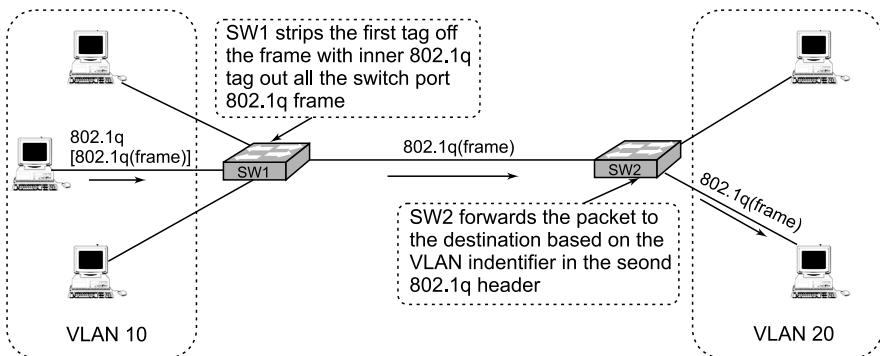


FIGURE 18.9 Double tagging.

18.7.1 Protection against VLAN Hopping Attacks

VLAN hopping attacks can be prevented by adopting the following modifications in the configuration:

1. Strictly use a dedicated VLAN ID for all trunk ports.
2. Group all unused ports, disable them, and place them in an unused VLAN.
3. Using switch port mode access command in the interface configuration mode, set all user ports to non-trunking mode by disabling the dynamic trunk protocol (DTP).
4. In a backbone network, the switch-to-switch connections explicitly configure trunking.
5. Avoid using the native VLAN as the trunk port native VLAN.

18.8 SPANNING TREE PROTOCOL (STP) MANIPULATION ATTACKS

Spanning Tree Protocol (STP) is a layer 2 link management protocol. It prevents forming loops in a redundant switched network environment. Generally the redundant paths are built to provide availability and robustness to the network. These redundant paths may create loops, which can

lead to different links of network attacks. Avoiding loops ensures that broadcast traffic does not become a traffic storm.

18.8.1 Spanning Tree Protocol

Spanning Tree Protocol is a tree like topology with a root switch at the top. A switch is elected as root based on the lowest configured priority of any switch (0 through 65,535). When a switch boots up it begins identifying other switches and determining the root bridge. Once the root bridge is elected, the topology is established from its perspective of the connectivity. The switches determine the path to the root bridge, and all redundant paths are blocked. Any changes in the topology and configuration and acknowledgement are communicated using a bridge protocol data unit (BPDU).

During a breakdown or disconnection of the link, the STP reconfigures the network and redefines the data path by activating an appropriate alternative path. All the switches exchange information for root switch selection and for subsequent configuration of the network. The BPDU carries this information, and all the switches in the network elect a root switch that becomes the focal point in the network and controls the blocked and forwarded link.

18.8.2 STP Attacks

STP protocol is used in switched networks to prevent the creation of bridging loops in an Ethernet network topology. Upon boot up, the switches begin a process of determining a loop-free topology. The switches identify one switch as a root bridge and block all other redundant data paths.

By manipulating the STP root bridge determination calculations, network attackers hope to spoof their system as the root bridge in the topology. To do this, the network attacker broadcasts out STP configuration and topology change bridge protocol data units (BPDUs) in an attempt to force spanning-tree recalculations. The BPDUs sent out by the system of the network attacker announce that the attacking system has a lower bridge priority. If successful, the network attacker becomes the root bridge and can see a variety of frames. By transmitting spoofed STP frames, the network attacker causes the switches to initiate spanning-tree recalculations that then result in having all of the interfaces in the system of the network attacker to be in the forwarding mode. This recalculation causes denial of service conditions on the network by causing an interruption each time the root switch/bridge changes. The generation of a large number of BPDUs per second leads to very high CPU utilization on the switches. Figure 18.10 (a) and (b) show

an attacker using an STP network topology change to force its hosts to be elected by the switch/bridge. As in Figure 18.10 (a), the attacker spoofs a BPDU with a lower priority switch. Now the attacker will be the new root bridge, and the spanning tree topology changes as in Figure 18.10 (b).

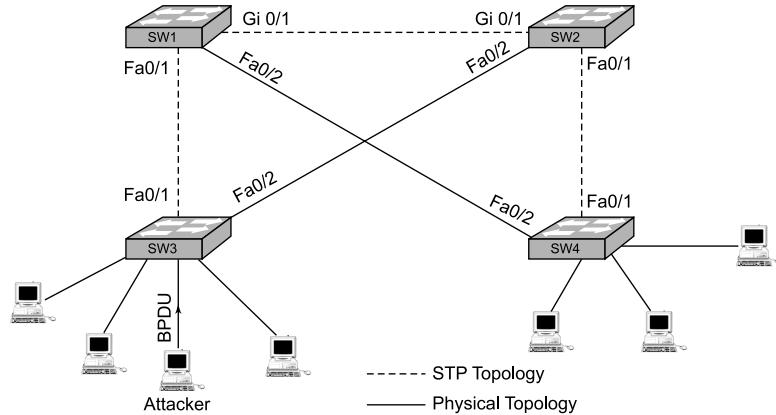


FIGURE 18.10 (a) Attacker spoofs a BPDU with a lower priority switch.

With this new topology, SW3 and SW4 use only SW1 to switch packets, while SW2 is not used by the access switch (SW3 and SW4). Moreover the election of the attacker as root causes the Gigabit Ethernet (SW1 and SW2) to block, causing a suboptimal network.

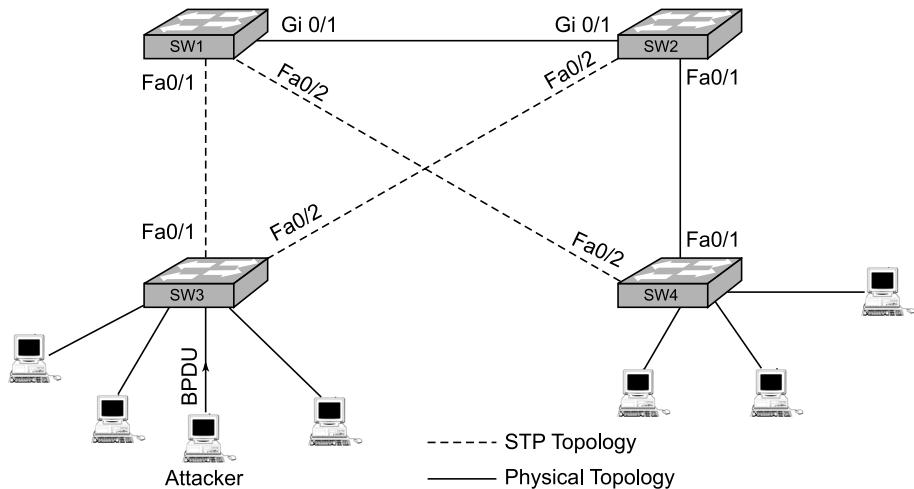


FIGURE 18.10 (b) New root bridge and the spanning tree topology changes.

18.8.3 Protection against STP Manipulation Attacks

To mitigate STP manipulation, use the Root guard and the BPDU guard enhancement commands to enforce the placement of the root bridge in the network and enforce the STP domain borders.

Root Guard

The root guard feature is designed to provide a way to enforce the root bridge placement in the network. It ensures that the port on which the root guard is enabled is the designated port. Normally, root bridge ports are all designated ports, unless two or more ports of the root bridge are connected together. If the bridge receives superior STP Bridge Protocol Data Units (BPDUs) on a root guard-enabled port, the root guard moves this port to a root-inconsistent STP state. This root-inconsistent state is effectively equal to a listening state. No traffic is forwarded across this port. In this way, the root guard enforces the position of the root bridge.

BPDU Guard

The STP BPDU guard is designed to allow network designers to keep the active network topology predictable. While a BPDU guard may seem unnecessary given that the administrator can set the bridge priority to zero, there is still no guarantee that it will be elected as the root bridge. This is because there might be a bridge with priority zero and a lower bridge ID. BPDU guard is best deployed toward user-facing ports to prevent rogue switch network extensions by an attacker.

SUMMARY

- The Data Link layer is used for filtering, access control lists, application control, and authentication to limit access at the highest layers of the protocol design.
- ARP spoofing is a type of attack in which an attacker sends a falsified ARP message over a local area network (LAN).
- MAC flooding is one of the most common network attacks; it is a method of attacking the network switches.
- A port scanning attack exploits the vulnerability of switch devices in a LAN.
- In a DHCP attack, it is possible to attack a DHCP server by causing Denial of Service in the network or by impersonating the DHCP server.

- A CAM overflow attack essentially turns a switch into a hub.
- In a VLAN hopping attack, an attacker generates traffic with a VLAN ID of an end system it cannot normally reach and sends the traffic.
- The Spanning Tree Protocol is used in a switched network to prevent the creation of a bridging loop in an Ethernet network topology.

REVIEW QUESTIONS

1. What is meant by ARP poisoning?
2. List the types of attacks in the data link layer.
3. Explain a MAC flooding attack.
4. What is port stealing? Explain a port stealing attack on a switch.
5. What is a DHCP attack? Explain different types of DHCP attacks.
6. What is a CAM overflow attack? Explain a CAM table.
7. Describe VLAN Hopping. Explain protection against VLAN Hopping.
8. What is the spanning tree protocol (STP)? Explain.

MULTIPLE CHOICE QUESTIONS

1. MAC flooding is the method of attacking the network _____:
(a) router (b) switch (c) hub (d) gateway
2. The _____ allows someone to receive packets that were originally directed to another computer.
(a) port stealing (b) session hijacking
(c) DHCP attack (d) none of the above
3. _____ and _____ are the possible attacks on a DHCP Server:
(a) ARP poisoning; MAC flooding (b) port stealing; MAC flooding
(c) DHCP spoofing; DHCP starvation (d) none of the above

CHAPTER 19

INTRUDERS, VIRUSES, WORMS, AND TROJAN HORSES

19.1 INTRODUCTION

In this chapter, first we discuss the concepts related to securing computer systems, which include certain core IT infrastructures. Within the network there are certain applications that run on top of the operating systems and provide network connectivity essentials on which the entire IT infrastructure depends. If those services go down, if they are corrupted or abused by malicious software and attacks, security can be compromised. All the attacks relate to network security, because the system entry is achieved by means of the network. The internal users maliciously exploit the internal IT resources. A virus or worm may be introduced into the system through malicious software programs or portable storage devices. In this chapter, we cover different malicious software used by the intruders in the networks, their effect on the network and IT infrastructure, and some mitigation techniques.

19.2 VIRUSES

A computer virus is a program written to alter the way a computer operates, without the permission or knowledge of the user. A virus must meet two criteria:

- It must execute itself: it will often place its own code in the path of execution of the other program.

- It must replicate itself: it may replace other executable files with a copy of the virus infected files.

Some virus programs are written to damage the computer by damaging programs, deleting files, or reformatting the hard disk. Others are not designed to do any damage but simply to replicate themselves and make their presence known by presenting text, video, and audio messages. The virus programs typically take up computer memory used by legitimate programs. As a result, they often cause erratic behavior and can result in system crashes.

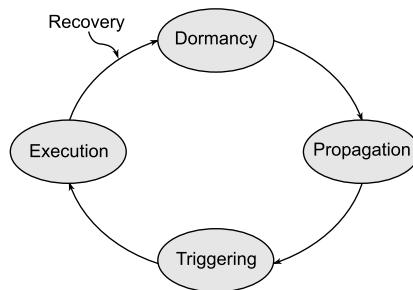


FIGURE 19.1 Virus life cycle.

A virus is a program that can infect other programs by modifying them and inserting a copy of itself into the program. This copy can then go on to infect other programs. Just like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. A virus attaches itself to another program and then executes secretly when the host program is run. During its lifetime a typical virus goes through the following phases, as in Figure 19.1.

Dormant Phase: In this state the virus is idle waiting for some event to happen before it gets activated. Some examples of these events are date/timestamp, presence of another file, or disk usage reaching some capacity.

Propagation Phase: In this stage the virus makes an identical copy of itself and attaches itself to another program. This infected program contains the virus and will in turn enter into a propagation phase to transmit the virus to other programs.

Triggering Phase: In this phase the virus starts performing the function it was intended for. The triggering phase can also be caused by a set of events.

Execution Phase: In this phase the virus performs its function such as damaging programs and data files.

19.2.1 Virus Structure

A virus is a decisively written computer program which consists of two parts, as in Figure 19.2; self-replicating code and the payload, which produces the required action (side effect).

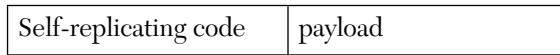


FIGURE 19.2 Virus structure.

In a typical computer virus, the size of the self-replicating code may be between 400 bytes and 2K bytes, while the size of the payload will depend on the side effects (a few hundred bytes). A virus, before infecting an executable, tries to determine whether they have already infected it, by testing for some infection signature. If the signature is there, the executable is already infected and it will not be reinfected. In case of the virus Jerusalem, it does not test correctly for its own signature, which results in reinfection and thus unlimited growth of executable images.

19.2.2 Classification of Computer Viruses

Computer viruses are classified according to their nature of infection and behavior as follows:

Boot Sector Virus

A Boot Sector Virus infects the first sector of the hard drive, where the Master Boot Record (MBR) is stored. The Master Boot Record (MBR) stores the disk's primary partition table and stores bootstrapping instructions which are executed after the computer's BIOS passes execution to the machine code. If a computer is infected with the Boot Sector Virus, the virus launches immediately when the computer is turned on and is loaded into memory, enabling it to control the computer.

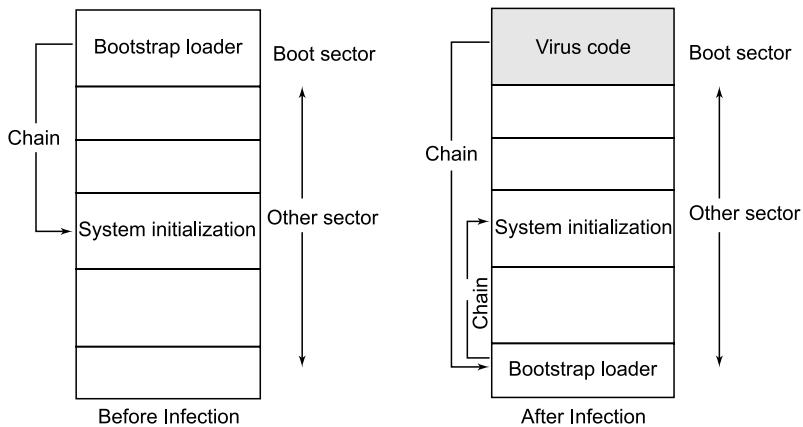


FIGURE 19.3 Boot sector virus infection.

Boot sector viruses take advantages of the boot process of the computers. Because most computers do not contain operating systems in their ROM, this needs to be loaded from the disk or from the network. In early systems, however, the boot order could not be defined, and thus the computer would boot from the diskette, allowing great opportunity for viruses to load before the operating system. On newer systems, each partition is further divided into additional partitions. The disk is always divided into heads, tracks, and sectors. The MBR is located at head 0, track 0, and sectors, which is the first sector on the hard disk. The MBR contains generic process-specific code to locate the active boot partition from the partition table records. The partition table is stored in the data area of the MBR. At the front of the MBR is some tiny code, often called a bootstrap loader. The MBR and boot sector infection technique are explained in the following section.

MBR is a tiny code of size 512 bytes. The MBR gets infected immediately upon booting from an infected diskette in drive A. The classic type of MBR viruses uses the INT 13H BIOS disk routine to access the disk for read and write access. Most MBR infectors replace the bootstrap code with their own copy and do not change the partition table. The stoned virus (Boot Sector Computer Virus) is a typical example of this technique. The virus stores the original MBR on sector 7. After the virus gets control via the replaced MBR, it reads the stored MBR located on sector 7 in memory and gives its control. A couple of empty sectors are typically available after the MBR, and the stoned virus takes advantages of this. This is exactly why some MBR viruses make a system unbootable after infection.

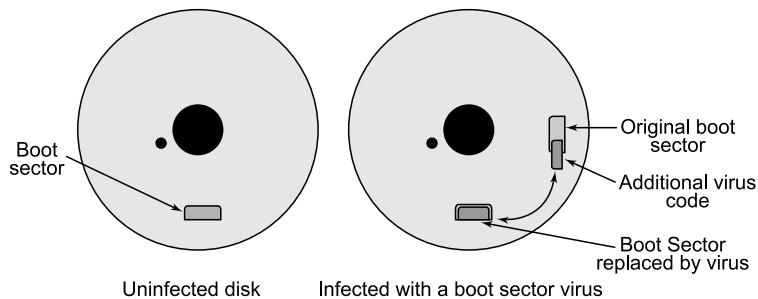


FIGURE 19.4 Disk before and after a stoned infection.

Viruses can also infect the MBR by overwriting the bootstrap code, leaving the partition table entries in place but not saving the original MBR anywhere. Such viruses need to perform the function of the original MBR code. In particular they need to locate the active partition, load it, and give control to it after themselves.

File infecting viruses: File infecting viruses are the viruses that infect files. Sometimes these viruses are memory resident. However, they will commonly infect most, if not all, of the executable files (those with the extensions .COM, .EXE, .OVL, and other overlay files) on a system. Some file infecting viruses will only attack operating system files (such as COMMAND.COM), while others will attack any file that is executable.

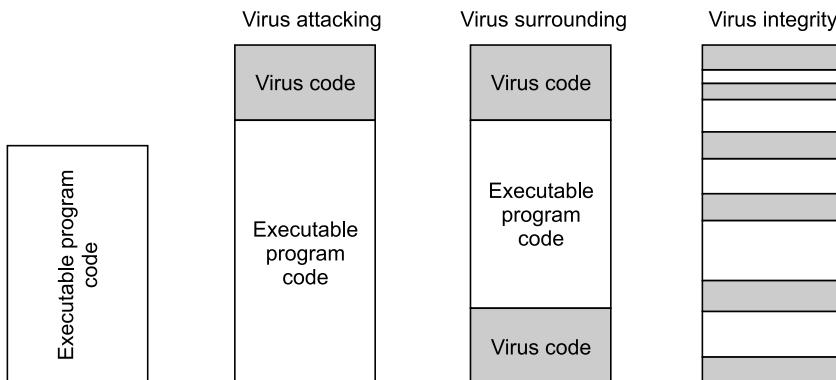


FIGURE 19.5 File infector virus.

Some of these viruses act like boot sector infectors. They replace the program load instructions in an executable file with their own instructions,

and move the original program load instructions to a different part of the file. This usually increases the file's size, making detection a little easier. Other file infecting viruses work by using companion files. They rename all files with .COM extensions to .EXE, then write a file with the same name and a .COM extension. This new file will usually have the "hidden" attribute, making it difficult to detect with ordinary file handling commands. By default, MS-DOS executes the .COM file before the .EXE file so that the .COM file is executed first, loading the virus.

Mass Mailer Viruses: Mass Mailer Viruses search e-mail programs like MS Outlook for e-mail addresses which are stored in the address book and replicate by e-mailing themselves to the addresses stored in the address book of the e-mail program. At a minimum, infections of this class of viruses generate high volumes of traffic that flood networks and overload mail relays. At worst these worms can disclose confidential information from systems they infect, with some mass-mailing worms going so far as to install keystroke loggers and backdoor services for collecting passwords and other sensitive information from the infected system. Although they all use e-mail to propagate, some of these worms can also spread through other means, such as by scanning for unprotected Windows shares on the infected computer's local network. Figure 19.6 shows how NIMDA uses four methods to infect systems.

The Melissa virus first released in March 1999 was the first mass-mailing worm of global consequence. Network administrators were not prepared to deal with this new kind of malware threat, and the virus's rapid infection of millions of PCs caused widespread disruption and economic damage. Later mass-mailing worms such as SirCam and Klez have demonstrated increasingly more sophisticated and polymorphic behavior, and have proved capable of infecting new victims in large numbers, long after their initial appearance on the Internet.

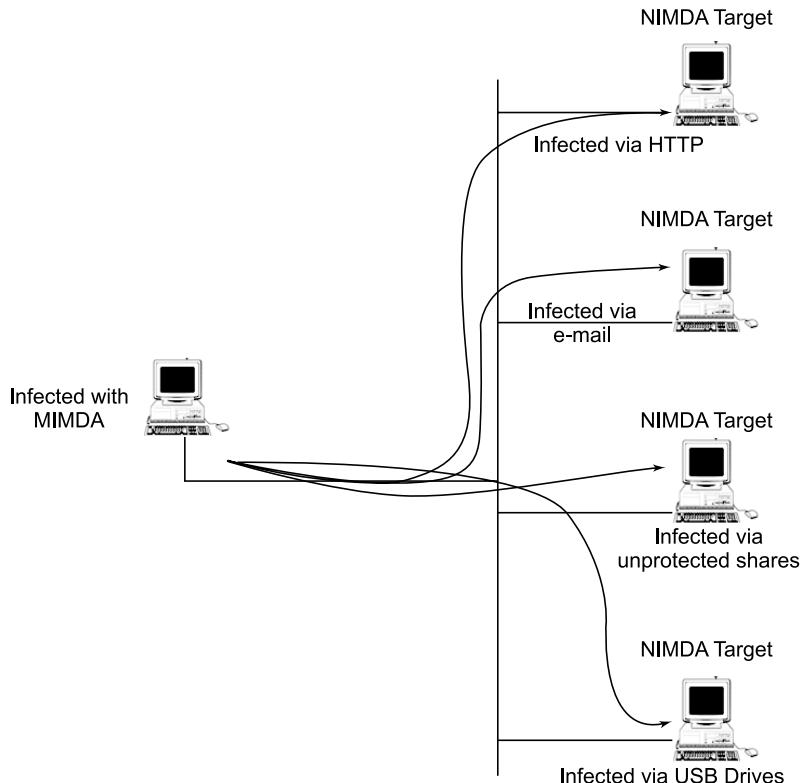


FIGURE 19.6 Targets of mass mailing worms.

Macro Viruses

A macro virus is a virus composed of a sequence of instructions that is interpreted rather than executed directly. Macro viruses can infect either executables or data files (Highland's Lotus 1-2-3 spreadsheet virus). For example, Duff's shell virus can execute on any system that can interpret the instructions. This is a piece of self-replicating code written in an application's macro language. A macro virus requires an auto-execute macro which is executed in response to some event, for example, opening or closing a file or starting an application. Once the macro virus is running, it can copy itself to other documents, delete files, and so on.

Macro viruses are written by using macro programming languages like VBA, which is a feature of the MS Office package. These macros are usually stored as part of the document or spreadsheet and can travel to other systems when these files are transferred to other computers. Macros are short snippets of code written in a language which is typically interpreted by the

application, a language which provides enough functionality to write a virus. Thus, macro viruses are better thought of as data file infectors, but since their predominant form has been macros, they have been named macro viruses.

When a macro-containing document is loaded by the application, the macros can be caused to run automatically, which gives control to the macro virus. Some applications alert/warn the user about the presence of macros in a document, but these warnings may be easily ignored.

The operation of the “Concept” virus operation which targets Microsoft Word documents is shown in Figure 19.7. Word has a persistent, global set of macros which applies to all edited documents, and this is Concept’s target: once installed in the global macros, it can infect all documents edited in the future. A document infected by Concept includes two macros that have special properties in Word.

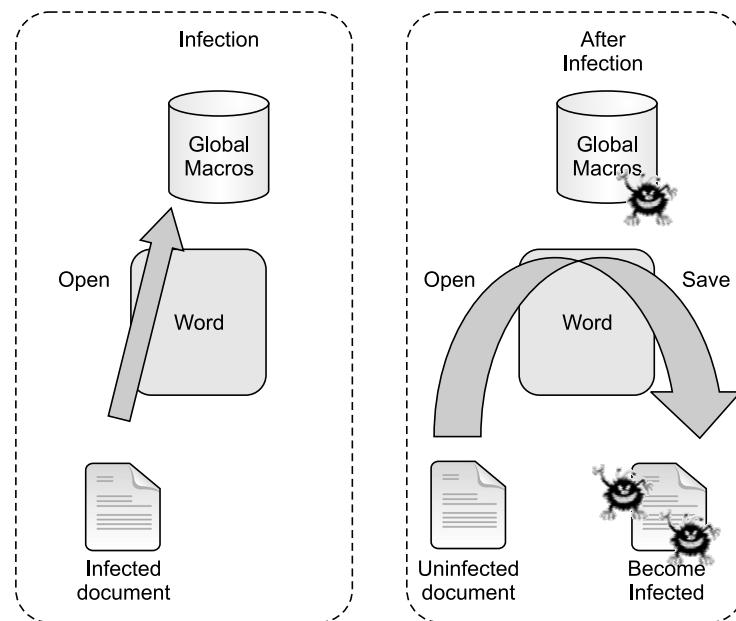


FIGURE 19.7 Concept in action.

Polymorphic Viruses: Polymorphic Viruses have the capability to change their appearance and change their code every time they infect a different system. This helps the Polymorphic Viruses to hide from antivirus

software. A polymorphic virus is characterized by its behaviors like encryption, self-multiplication, and changing of one or more components of itself so that it remains elusive. It is designed to avoid detection as it is capable of creating modified copies of itself.

Thus, a polymorphic virus is a self-encrypted malicious software that has the tendency to change itself in more than one way before multiplying onto the same computer or to computer networks. Since it changes its components properly and is encrypted, the polymorphic virus can be said to be one of the intelligent malware that is hard to detect; by the time the antivirus detects it, the virus has already multiplied after changing one or more of its components (morphing into something else).

Armored Viruses: An armored virus is another type of virus that is designed and written to make itself difficult to detect or analyze. An armored virus may also have the ability to protect itself from antivirus programs, making it more difficult to disinfect.

Armored viruses are a very complex type of virus designed to make its examination much more difficult than in the case of traditional viruses. By using various methods an armored virus can also protect itself from antivirus software by misleading it into believing that the virus location is somewhere other than the real location, which of course makes the detection and removal process more difficult.

Stealth Viruses: Stealth viruses have the capability to hide from operating systems or antivirus software by making changes to file sizes or directory structure. Stealth viruses are anti-heuristic in nature, which helps them to hide from heuristic detection.

A stealth virus will actively take steps to conceal the infection itself, not just the virus body. Further, a stealth virus tries to hide from everything, not just antivirus software. For example, an infected file's original time stamp can be restored after infection, so that the file doesn't look freshly changed.

A virus can store all pre-infection information about a file, including its time stamp, file size, and the file contents. These system's I/O calls can be intercepted and the virus would play back the original information in response to any I/O operations on the infected files, making it appear unified. This technique is applicable to boot I/O too.

Retrovirus: Retrovirus is another type virus which tries to attack and disable the antivirus application running on the computer. A retrovirus can be considered anti-antivirus. Some retroviruses attack the antivirus application and stop it from running, or some others destroy the virus definition database.

- Avgw. Exe
- F-Port. Exe
- Navw32. Exe
- Regedit. Exe
- Scan32. Exe
- Zonealarm. Exe

FIGURE 19.8 Retrovirus.

When it infects a computer, a retrovirus will enumerate the currently running processes and kill off any processes which match one of the names in the list. A partial list is given in Figure 19.8. A more aggressive retrovirus can target the antivirus software on the disk as well as in memory so that antivirus protection is disabled even after the infected system is rebooted. This approach tries to starve antivirus software of CPU time. A retrovirus with appropriate permission would reduce the priority of antivirus software to the minimum value possible, to (ideally) keep it from running. Most operating system schedulers have a mechanism to book the priority of CPU starved processes, however, so attacking antivirus software by reducing process priority is likely to be very effective. For a computer connected in a network, the way to disable antivirus software is to adjust the way a computer looks up host name information on the network, to prevent antivirus software from being able to connect to the antivirus company's server and update its database.

Multiple Characteristic Viruses: Multiple Characteristic viruses have different characteristics of viruses and have different capabilities.

19.3 WORMS

A computer worm is a self-replicating program that penetrates an operating system with the intent of spreading malicious code. A virus typically requires some human intervention (such as opening a file) to propagate itself, whereas a worm typically propagates by itself. A worm also uses network connections to propagate from one machine to another. Some examples of these connections are: electronic mail facility, remote execution facility, and remote login facility. A worm will typically have similar phases as a virus, such as dormant

phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase for a worm uses the following two steps:

- Search the host tables to determine other systems that can be infected.
- Establish a connection with the remote system, copy the worm to the remote system, and cause it to execute just like a virus; network worms are also difficult to detect. However, properly designed system security applications can minimize the threat of worms.

Due to the nature of replication through the network, a worm normally consumes a large amount system resources, including network bandwidth, causing network servers to stop responding.

19.3.1 Worm Attack Procedure

The anatomy of a worm attack can be divided into three steps as follows:

Step 1: The first stage goes by the name of enabling vulnerability and is the first step of the process wherein the work gets installed on a vulnerable system.

Step 2: After getting installed on a vulnerable system, the worm then spreads its wings further by spreading to new targets. It does so by automatic replication of itself and attacking other systems.

Step 3: During the last or the payload stage, the hacker tries to raise the level of his/her access. Normally the hackers get to use the targeted system as a privileged user but, with the passage of time, they normally aim to have access rights equivalent to that of an administrator and can cause maximum damage after that.

- Once the previous cycle is complete, the worm propagates through automatic transmission mechanisms using the vulnerability of the other systems attached to the initially targeted system, and the entire process starts all over again.
- In this manner, the hacker may be able to gain access to multiple systems and could even use the entire array of such infected systems to cause botnet attacks.

Figure 19.9 shows a description of such a propagation wherein the network is supposedly safe and secure with a firewall.

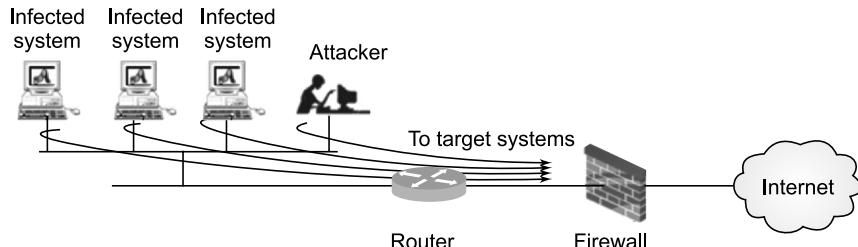


FIGURE 19.9 Worm attack.

19.3.2 How to Mitigate Worm Attacks

Due to the specific nature of worm attacks and their methods of self-propagation, it would be really difficult if not impossible to control and respond to worm attacks. Proper and systematic planning is necessary for worm attack mitigation, and normally the process can be divided into the following four steps:

1. **Containment:** This step involves compartmentalization of the network into infected and not infected parts. This helps to contain the spread of the worm attack.
2. **Inoculation:** This step involves scanning and patching of the vulnerable systems.
3. **Quarantine:** In this step, the infected machine is detected, disconnected, and removed. If removal is not possible, the infected machines are blocked.
4. **Treat:** This is the step where cleaning up and patching is done. Some worms may need reinstalling of the entire system for a thorough cleanup.

19.3.3 Types of Computer Worms

E-mail Worms: E-mail worms spread through infected e-mail messages as an attachment or a link of an infected website. They use the e-mail client to spread itself. It will either send a link within the e-mail that, when clicked, will infect the computer, or it will send an attachment that, when opened, will start the infection. Once the worm is installed, it will search the host computer for any e-mail addresses contained on it. It will then start the process again, sending the worm without any input from the user. A well-known example of this type of worm is the “ILOVEYOU” worm, which infected millions of computers worldwide in 2000.

Instant Messaging Worms: Instant messaging worms spread by sending links to the contact list of instant messaging applications. This works in a similar way to e-mail worms. The infected worm will use the contact list of the user's chat-room profile or instant-message program to send links to infected Websites. These are not as effective as e-mail worms, as the recipient needs to accept the message and click the link. They tend to effect only the users of the particular program.

Internet Worms: Internet worms will scan all available network resources using local operating system services and/or scan the Internet for vulnerable machines. If a computer is found vulnerable, it will attempt to connect and gain access to them.

Internet worms are completely autonomous programs. They use an infected machine to scan the Internet for other vulnerable machines. When a vulnerable machine is located, the worm will infect it and begin the process again. Internet worms are often created to exploit recently discovered security issues on machines that haven't installed the latest operating-system and security updates.

Internet Relay Chat (IRC) Worms: IRC worms spread through IRC chat channels, sending infected files or links to infected Websites.

W32.IRCBot is detection for worms that spread using Internet Relay Chat (IRC). The IRC connection serves as a back door, allowing an attacker to perform a variety of actions on the compromised computer. An attacker usually gathers a large number of details of computers that are infected with W32.IRCBot worms and uses them as a bot network, controlled through IRC. The use of IRC separates threats from their traditional back door and worm counterparts in that the hacker does not issue commands directly to the back door. Rather, they are routed through the IRC server and channel, and then on to the compromised computer as shown in Figure 19.10. Without the IRC server or channel, the attacker is unable to control the compromised computer.

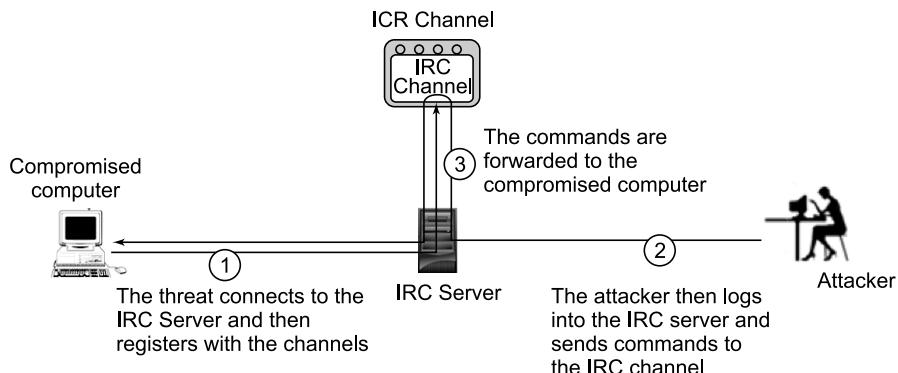


FIGURE 19.10 How IRC bots work.

File-sharing Network Worms: File-sharing network worms place a copy of them in a shared folder and spread via a P2P network.

File-sharing worms take advantage of the fact that file sharers do not know exactly what they are downloading. The worm will copy itself into a shared folder with an unassuming name. When another user on the network downloads files from the shared folder, they will accidentally download the worm, which then copies itself and repeats the process. In 2004, a worm called “Phatbot” infected millions of computers in this way and had the ability to steal personal information, including credit card details, and send spam on an unprecedented scale.

19.4 TROJAN HORSES

A Trojan horse, often shortened to Trojan, is a type of malware designed to provide unauthorized, remote access to a user’s computer. Trojan horses do not have the ability to replicate themselves like viruses; however, they can lead to viruses being installed on a machine, since they allow the computer to be controlled by the Trojan creator. Trojan horses are one of the most common methods a criminal uses to infect the computer and collect personal information from your computer. Trojan horses are also explained in Chapter 14.

19.4.1 Types of Trojan Horses

1. Remote Access Trojans (RATs) or Backdoor Trojans and Trojan Dropper Programs

One of the most common but dangerous Trojans are called Backdoor Trojans. They are also known as Remote Access Trojans or RATs. A PC with a sophisticated backdoor Trojan installed may also be referred to as a zombie or bot. These types of threats are almost invisible to the user, and if they succeed in entering the system, the attacker or intruder has remote access to the system. The attacker, with the help of the Trojan, is now in control of the computer. Some capabilities of backdoor Trojans are to collect information, terminate or run a task or process, download and upload files, report to the attacker’s machine, change critical settings in operating systems, shutdown or restart the PC, and perform Denial of Service (DoS) attacks.

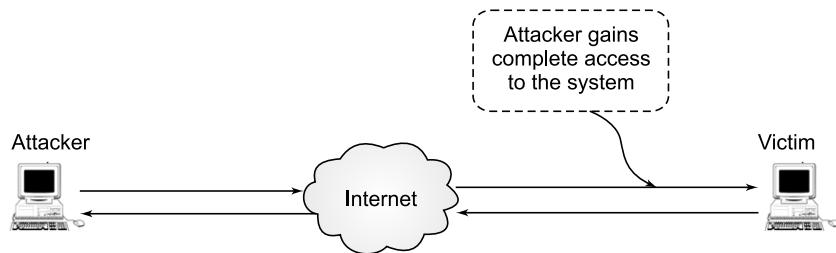


FIGURE 19.11 Remote access Trojans.

This Trojan works like remote desktop access. The attacker gains complete GUI access to the remote system. As in Figure 19.11, the attacker does the following:

- Infects the victim's computer with server.exe and plants a reverse-connecting Trojan.
- The Trojan connects to port 80 to the attacker on the Internet, establishing a reverse connection.
- Now the attacker has complete control over the victim's machine.

Common Infection Methods

- Use specifically crafted e-mail attachments, web links, download packages, or Torrent files as a mechanism for installation of the software.
- Deceive desired targets installing such malicious software via social engineering tactics or temporary physical access to the computer.

2. The File Serving Trojan Horse Virus

Trojan horse viruses from this category are able to create a file server on the infected machine. Usually this server is configured as an FTP server, and with its help the intruder will be able to control network connections and upload and download various files. These Trojan horse viruses are rather small in size, sometimes not more than 10KB, which makes it difficult to detect them. They are often attached to e-mails or hidden in other files that users may download from the Internet. Regularly these Trojan viruses spread with the help of funny forwarded messages that a user receives from friends. Trojan horse viruses may also be hidden in small downloadable games.

3. Distributed Denial of Service Attack Trojan Horse Virus

A lot of computers can be misled by installing the Distributed Denial of Service Trojan so that the hacker can gain control over one, several, or all computers through a client that is connected with a master server. Using the primary computer within one huge zombie network of machines, hackers are able to send attacks at particular targets, including companies and Websites. They simply flood the target server with traffic, thus making it impossible for simple users to access certain Websites or systems. Often these attacks are used to stop the activity of famous brands that could handle different financial demands.

4. Keyloggers and System File Killers

A keylogger Trojan is malicious secret software that monitors the user's keystrokes, logging them to a file and sending them off to remote attackers. The keylogger may record all keystrokes, or they may be sophisticated enough to monitor for a specific activity like opening a Web browser pointing to a user's online banking site. When the desired behavior is observed, the keylogger goes into record mode, capturing user's login username and password. Some sites attempt to thwart keyloggers by having the users respond to visual cues that they must point to with their mouse. However, some keylogger Trojans also capture screenshots, thereby negating the effect of this strategy.

To understand keyloggers, we need to understand the structure and the workings of a keyboard. The keyboard is a matrix of circuits which communicates character codes to a keyboard driver in the computer, as shown in Figure 19.12. When a key is pressed the circuit location in the key matrix is translated to a code in the keyboard processor and placed in the keyboard buffer. From the buffer the keyboard codes are sent to the keyboard controller. The keyboard controller interfacing with the operating system passes keyboard data to waiting applications.

The most common type of keylogger consists of a piece of malware installed and managed by a rootkit. A typical keylogger replaces the operating system's kernel components. As keyboard codes move from the keyboard controller to the operating system, the keylogger captures all keyboard entries before passing them on to the target application. Other types of Trojans are known as keyloggers and system file killers:

- **Keylogging Trojans** - Keyloggers will record every mouse click or pressed key of an infected computer. The recorded action from the computer is sent to a remote location. Spyware distributors and hackers are the common attackers of keyloggers to gain information that they need.

A keylogger Trojan can gain access to valuable data such as online banking credentials, e-mail account usernames and passwords, and credit card details.

- **System file killing Trojans** - These types of Trojans often end or shut down important system or program files. An example of this Trojan is the Trojan KillAV.

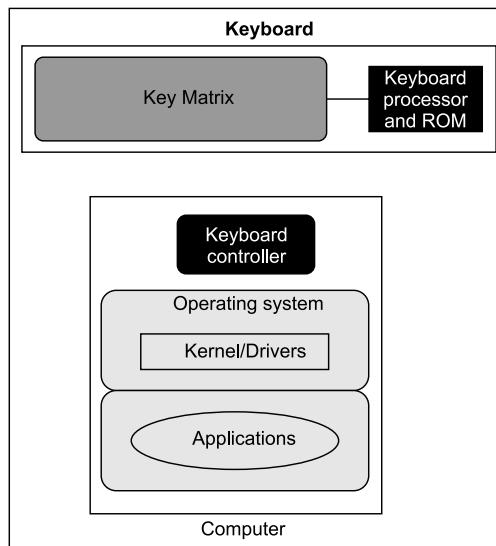


FIGURE 19.12 Structure of a keyboard.

5. The Password Stealing Trojan Horse Virus

The name speaks for itself—Trojans from this category are used to steal passwords. The Trojan transmits information about passwords to the hacker through e-mail. Just like keylogging Trojans, this malware is used mainly for a hacker's financial benefit (a lot of people use passwords to access their bank accounts or credit cards). As in Figure 19.13, the Trojan prompts the user to enter his password in the compromised system. The Trojan then transmits the password to the hacker's system.

One early form of the Trojan horse was a fake login screen. The screen looks just like the login screen used for the system, but when the user attempts to login, the username and password are captured and stored in some secret location accessible to the intruder.

This technique for stealing passwords is designed for a public setting such as a computer lab in which multiple users might use a common set of

terminals or workstations. In recent years, operating systems have become better at preventing or detecting this form of password capture. Microsoft claims that NT and post-NT Windows systems are immune from this form of password-capture attack, because the security subsystem suspends all background processes.

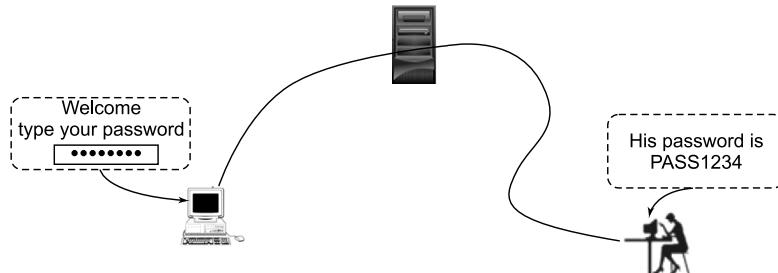


FIGURE 19.13 Stealing passwords with a Trojan horse login program.

6. The System Killing Trojan Horse Virus

These Trojans are meant to destroy everything in the system starting with drive Z and ending with drive A. One of the recent Trojan horse viruses of this type is called Trojan.Killfiles.904. The reasons for creating such Trojans are unknown, but the results could be catastrophic.

19.5 ADWARE

Adware is a software package which automatically plays, displays, or downloads advertisements to a computer. These advertisements can be in the form of a pop-up. The object of the adware is to generate revenue for its author. Adware, by itself, is harmless; however, some adware may come with integrated spyware such as keyloggers and other privacy-invasive software. Advertising functions are integrated into or bundled with the software, which is often designed to note what Internet sites the user visits and to present advertising pertinent to the types of goods or services featured there. Adware is usually seen by the developer as a way to recover development costs, and in some cases it may allow the software to be provided to the user free of charge or at a reduced price. The income derived from presenting advertisements to the user may allow or motivate the developer to continue

to develop, maintain, and upgrade the software product. Conversely, the advertisements may be seen by the user as interruptions or annoyances, or as distractions from the task at hand.

Some adware is also shareware, and so the word may be used as a term of distinction to differentiate between types of shareware software. What differentiates adware from other shareware is that it is primarily advertising-supported, like many free smartphone apps. Users may also be given the option to pay for a “registered” or “licensed” copy to do away with the advertisements. Pandora Radio offers both a free version (with ads) and a paid subscription (without ads).

19.6 MALICIOUS SOFTWARE

Table 19.1 denotes the different malicious software or malware that have caused all kinds of damage across computers from all over the world.

TABLE 19.1 Different Malware.

Malware	Targeted to	Damages
Viruses		
I LOVE YOU	Mailing systems	The I LOVE YOU virus is considered to be one of the most infectious computer viruses ever created. The total damage was estimated up to \$10 billion. The virus was so powerful that governments and large corporations were forced to take their mailing systems off-line to prevent infection. The I LOVE YOU viruses would send itself to everyone in the mailing list and proceed to overwrite all computer files with itself, making the computer unbootable.
Melissa	Word documents	This virus was named after an exotic dancer from Florida. The virus was created by David L. Smith in 1999. The virus started as an infected word document that was posted upon the alt.sex UseNet group, claiming to be a list of passwords for pornographic sites.

(continued)

Malware	Targeted to	Damages
Stuxnet	To disrupt the nuclear efforts of Iranians	Stuxnet is believed to have been created by the Israeli Defense Force together with the American government. It was created with the intention to disrupt the nuclear efforts of Iranians. Stuxnet managed to ruin a fifth of Iran's nuclear centrifuges and nearly 60% of infections were concentrated in Iran. It is also designed to attack industrial programmable logic controllers (PLC), which allow for automation processes in machinery and was specifically aimed at those created by Siemens. The virus was spread through infected USB drives.
Sasser	Buffer overflow	The effects of this virus were incredibly disruptive, with millions of computers being infected as important, critical infrastructure was affected. This took advantage of a buffer overflow vulnerability in the Local Security Authority Subsystem Service (LSASS), which controls the security policy of local accounts, causing crashes to the computer. It uses system resources to propagate itself to other machines through the Internet and infect others automatically.
NIMDA	e-Mail	Its name was derived from the word “admin” when spelled backward. The virus spread through e-mails, server weak points, shared folders, and file transfers.
Jerusalem	MSDOS virus	Jerusalem, being one of the first MSDOS viruses, was one of the most dangerous computer viruses of all time. In 1987 this virus caused a lot of damage in universities, colleges, and corporations all over the world. The name of the virus comes from the fact that it first spread in Jerusalem University.
BlockPOS	POS systems running Microsoft windows	The BlockPOS is a point-of-sale (POS) malware targeting credit and debit card data swiped at POS systems running in MS Windows systems. It uses a RAM scraper to grab card data from the memory of the infected POS devices and transmits the collected data to a compromised server, then uploads it to a file transfer protocol (FTP). Block POS is designed to bypass firewall software.
Worms		
Conficker	Windows computer	Win32 Conficker has been known for its aggressive attacks to disable antivirus and anti-spyware applications and then further infect systems.

Malware	Targeted to	Damages
CodeRed	Microsoft IIS web server	The CodeRed worm is targeted to the computers installed with the Microsoft IIS web server, exploiting a buffer overflow problem in the system. The worm hardly leaves any traces on the hard disk, because it is able to run entirely on memory. Once infected the CodeRed worm will proceed to make a number of copies of itself, but due to a bug in its programming, will duplicate even more and end up consuming a lot of the system's resources.
Mydoom	e-mail	This worm is named by McAfee employee Craig Schmeegar. The name was coined from "mydoom," a line of text in the program code which means my domain, and the word "doom" because of the dangers the worm posed. Mydoom spreads itself by appearing as an e-mail transmission error and contains an attachment of itself. Once executed the worm sends itself to e-mail addresses that are in the user's address book and copies itself to any P2P program's folder to propagate itself through the network.
Trojans		
Zeus	Windows user	It tries to retrieve information from the infected computers. Once installed it tries to download configuration files and updates from the Internet. Zeus has the ability to steal information and attack banking and financial institutions around the world.
Gameover	Banking systems (credit card users)	It is also one of the variants of the Zeus family. This type of malware relies on a peer-to-peer (P2P) botnet infrastructure. The generated peers in the botnet can act as independent command and control servers and are able to download commands or configuration files between them, finally sending the stolen data to the malicious servers. It has infected over 1 million users around the world.
Copyto Locker	Windows	Copyto Locker is a form of Trojan ransomware targeted at computers running Windows. It uses several methods to spread itself, such as e-mail, and once a computer is infected, it will proceed to encrypt certain files on the hard drive and any amount of storage connected to it with RSA public key cryptography.

(continued)

Malware	Targeted to	Damages
Flashback	Mac	It is one of the few Mac malware to have gained notoriety, as it showed that the Mac isn't immune to such attacks. The Trojan propagates itself by using compromised websites containing JavaScript code that will download the payload. Once installed, the Mac becomes part of a botnet of other infected Macs.
Qatbot	Bank accounts	The Qatbot acts as botnet that steals passwords and attaches itself to file shares to spread. This malware makes its rounds to connect to command and control servers waiting for instructions to carry out malicious actions on infected computers. These instructions set provide the Qatbot the tasks of stealing login information, which then could be used by remote attackers to infiltrate online accounts such as banking accounts.
Sykipot	Civil aviation and smart cards	It has the ability to bypass two-factor authentication. Such authentication measures are highly targeted in password related breaches, which led to a massive amount of data being stolen.
Koler	Android mobile devices	Koler is an android Trojan extorting mobile device users for money to unlock their data. The Trojan poses as a valid video player that offers premium access to pornography. The Koler automatically downloads during a browsing session. Once infected it prevents the user from accessing mobile home screens and displays bogus messages purporting to be from the national police service. The message claims the user has been accessing child abuse websites and demands payment to escape prosecution.

19.7 INTRUSION DETECTION SYSTEMS (IDS) AND INTRUSION PREVENTION SYSTEMS (IPS)

The main objective of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) is to continuously monitor traffic flow along a network path, normally at the interface between the trusted and the untrusted sections. An IDS is simply a passive system which only monitors activity in the bypass mode, not obstructing traffic directly, and only raising an alarm

if any anomaly is noted. The IPS instead is an active service which directly intercepts traffic and allows only permitted packets to pass through it, blocking everything else. These systems can be deployed using appropriate hardware, software, or a combination of both.

19.7.1 Intrusion Detection System (IDS)

An IDS monitors network traffic and suspicious activities in the network. Once these anomalies are detected, the IDS will alert the system administrator. In some cases the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address from accessing the network. Intrusion detection systems provide the following:

- Monitoring and analysis of system and user activity.
- Auditing of the system configuration and vulnerability.
- Assessing the integrity of critical systems and data files.
- Statistical analysis of activity patterns based on matching to known attacks.
- Abnormal activity analysis.
- Operating system audit.

Classification of IDS: Intrusion Detection Systems are classified based on the following criteria:

1. **Based on the type of system the IDS protects:** Network IDS (NIDS) and Host IDS (HIDS)
2. **Method of working:** Signature based IDS and Anomaly based detection
3. **Based on functions:** Passive IDS and Active IDS

Network IDS (NIDS): NIDS is strategically positioned in a network to detect any attack on the hosts of the network. It is designed to receive all packets on a particular network segment. To capture all the data passing through the network, NIDS is positioned at the entry and exit points of data from the network to the external network. If an attack is identified the IDS will alert the system administrator. NIDS are placed at important points in the network, as in Figure 19.14, so that it can keep monitoring the traffic flowing in and out of the network. Depending on how they function, NIDS can be divided into two types:

- Statistical anomaly IDS
- Pattern matching IDS

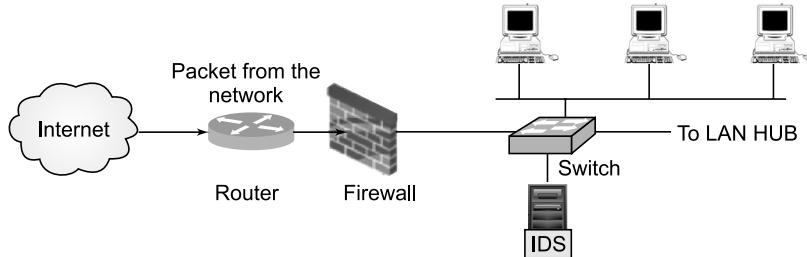


FIGURE 19.14 Network-based IDS.

Statistical anomaly IDS: In a statistical based IDS model, the IDS tries to find out a user's or system's behavior that seems abnormal. The IDS makes a profile of every user and system during normal operation time. When the deviation of this normal behavior is detected, the IDS triggers its alarm for intrusion. One of the main advantages of this type of IDS is that it can detect the type of intrusion that has no records of its previous occurrence. In that sense, a statistical anomaly can detect a new type of attack pattern. A large number of false alarms are the main problem with this system.

Pattern matching IDS: In pattern based system, the IDS maintains a database of known exploits and their attack pattern. During the analysis of network packets, if it finds any pattern match to one of those known attack patterns, then it triggers an alarm. For operation, this type of IDS needs to analyze every packet in the network to look for known attack patterns. Since this type of IDS mainly looks for patterns, it has a very quick deploy and implementation time, unlike statistical based IDS. Another advantage is that it produces fewer false positives. The main problem with pattern based IDS is that it cannot detect anything that is unknown to it or that of which it has no data in its pattern matching database. Some popular Network based IDS are:

- Real Secure
- SecureNet
- Snort

Host IDS (HIDS)

HIDS works on the individual systems connected to the network. It constantly monitors the incoming and outgoing traffic and also audits system

files. The system alerts the system administrator in case of discrepancy. HIDS is installed on workstations which are to be monitored, as in Figure 19.15. The agent monitors the operating system and writes data to log files and/or trigger alarms. A HIDS is additional software installed on a system such as a workstation or a server. It provides protection to the individual host and can detect potential attacks and protect critical operating system files. The primary goal of any HIDS is to monitor traffic. An example of a host based IDS is PortSentry.

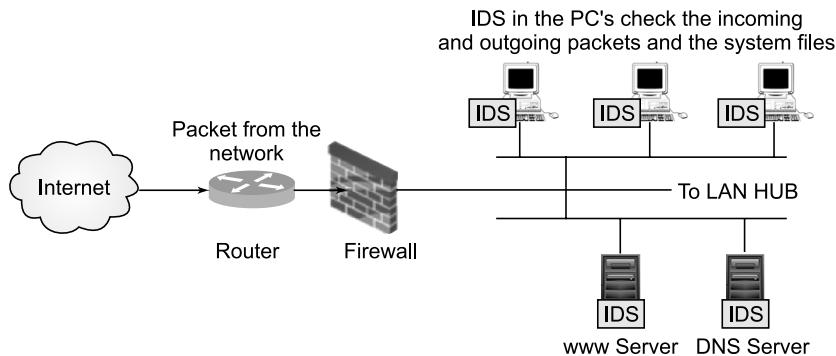


FIGURE 19.15 Host-based IDS.

There are different types of host based Intrusion Detection Systems:

Operating System Level: These types of HIDS function by working on the operating system log files. These HIDS determine unauthorized activities based on the following criteria:

- Application initiated on a system
- Logon and logoff credentials like login location, time, and date
- Addition/deletion or modification of system entities
- Access to system resources like files/folders/memory location or registry

The information obtained is compiled and compared to the signature available in the database using special algorithms.

Application Level: These types of HIDS are very similar to operating system HIDS; the main difference is that these HIDS concentrate more on the application level log files rather than the system level log files. These intrusion detection attempts are saved to a database application residing on the host computer.

Network Level: HIDS also performs some of the functions very similar to NIDS. A network level HIDS works on the network packets that are

addressed to a particular host. If a packet is not addressed to the host, the network level HIDS will not collect and work on the network packets.

Signature based IDS: This type of IDS works based on the principle of matching. The data is analyzed and compared with the signature of the known attacks. If it matches an alert is generated and sent to the system administrator. Signature based IDS uses a database of known vulnerabilities or known attack patterns. The attack tools used here are for the launch of a SYS flooding attack on a server by simply entering the IP address of the system to attack. The attack tool then floods the target system with the SYN packets, but never completes the three-way TCP handshake with the final ACK packet.

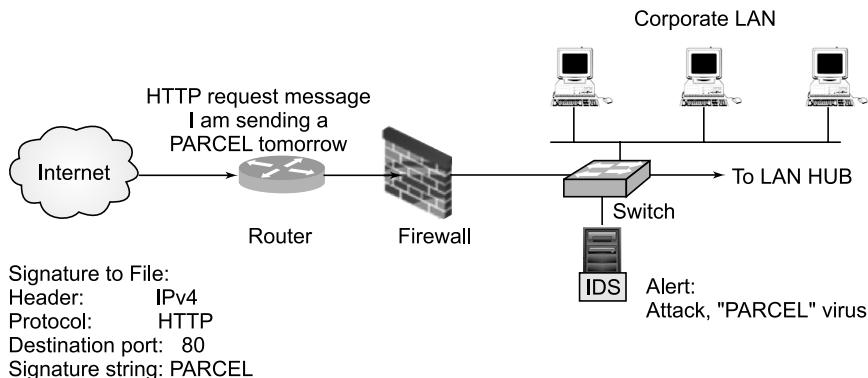


FIGURE 19.16 Signature-based IDS.

In signature-based detection, the intrusion detection is based on comparing the traffic with the known signatures for possible attacks as in Figure 19.16. They can only detect known threats and hence are not efficient in detecting unknown threats. To detect an attack, the signature matching has to be precise; otherwise, even if the attack has a small variation from the known threat signature, then the system will not be able to detect it. For example, in Figure 19.16, if the subject is "love you," instead of "love you," the system may not detect the threat. Hence, it is very easy for the attackers to compromise and breach into the trusted network.

Anomaly based Detection: Anomaly based IDS is also called heuristic based or behavior based detection. It does this by creating a performance baseline under normal operating conditions, as in Figure 19.17. It consists of a statistical mode of normal network traffic which consists of the bandwidth used, the protocols defined for the traffic, and the ports and devices which

are part of the network. It regularly monitors network traffic and compares it with the statistical model. An “anomaly” is anything that is abnormal. If any traffic is found to be abnormal from the baseline, then an alert is triggered by the IDS suspecting an intrusion. In case of any anomaly or discrepancy, the administrator is alerted.

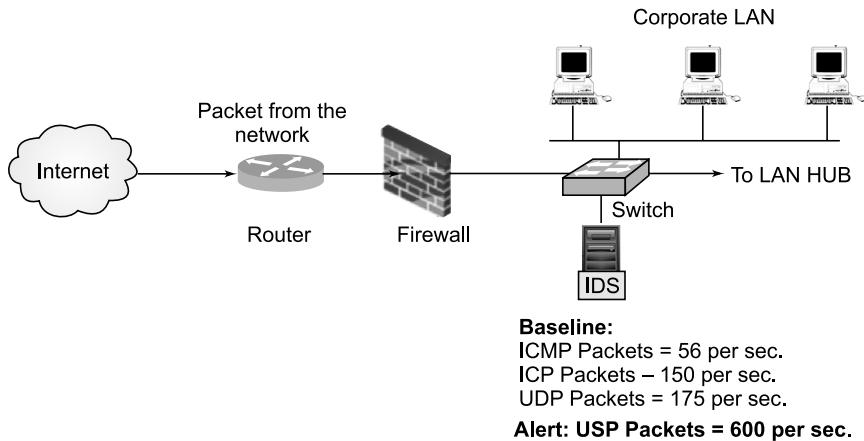


FIGURE 19.17 Anomaly-based IDS.

The IDS first creates a baseline profile that represents the normal behavior of the traffic. The baseline profile is created by allowing the IDS system to learn the traffic over a period of time so that it can study traffic behavior during peak hours, non-peak hours, night hours, early hours of business, and as per the organizational network behavior. After learning, the traffic collected over a period of time is statistically studied and a baseline profile is created. Once the IDS is changed from learning mode to detection/prevention mode, it starts comparing the regular traffic with the profile that was created, and if any abnormality or deviation from the baseline profile is found, then an alert is triggered, cautioning the possible intrusion.

Typical examples of anomalous behavior are:

- Too many Telnet sessions on a single day
- HTTP traffic on a non-standard port
- Heavy SNMP traffic

Passive IDS: A Passive IDS logs the attack and may also raise an alert to notify someone. Most of IDSs are passive by default. It simply detects the kind of malware operating and issues an alert to the system administrator.

Active IDS: Active IDS not only detects the threat but also performs specific actions by removing the suspicious connection or blocking the network traffic from the suspicious source. It is also known as an *Intrusion Prevention System*.

How Does an IDS Work?

An IDS is essentially a network-based solution, typically designed around a UNIX or Linux kernel. Figure 19.18 depicts how an IDS device is incorporated in a network. While other forms of defense such as routers and firewalls are required in a network, IDSs act as a complementary means to further strengthen security.

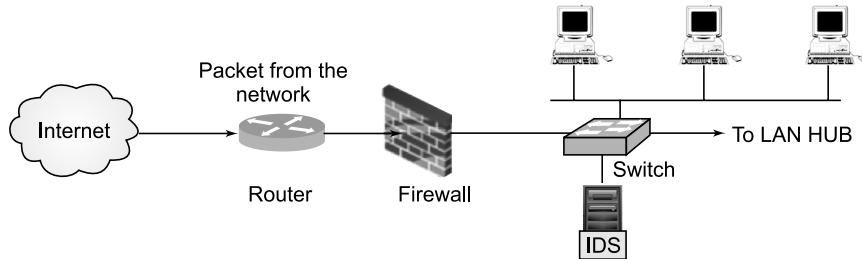


FIGURE 19.18 Installing an IDS device in a typical network.

From the installation point of view, the IDS device is usually situated in a demilitarized zone (DMZ), whereby the basic level of protection is taken care of by routers and firewalls, followed by a further level of intelligent intrusion detection. It comes equipped with network interfaces capable of handling heavy network traffic and is configured to work in promiscuous mode, which enables it to sniff the network traffic without causing disruption or slowdowns.

It monitors all network packets right from OSI Layer 2 (data link) to Layer 7 (applications), and stores this vast amount of information in its database. It also assimilates that information by applying intelligence to it to take security decisions. Intrusion detection mainly focuses on the intention of an attack, rather than just on the methodology. This is made possible by running multiple built-in intelligent algorithms called statistical anomaly-based detection logic.

For example, instead of only looking for a virus signature, an IDS device checks network packets and establishes a relationship between the information in the packets and its potential impact on the network from the security viewpoint. This approach helps the IDS to minimize false alarms.

An IDS can be configured to look for distributed-denial-of-service (DDoS) attacks on a Website. While all HTTP traffic coming to the Web server may be legitimate, it takes extra intelligence to check if the traffic is really legitimate or part of a possible attack. An IDS does this by storing all requests and using its intelligence to check each network packet, Web request, XML, and other forms of Web data, and performing historic analysis before the request reaches the Web server. Due to this difference in the approach to detection, IDSs are “must-have” components in modern network security infrastructures.

Advantages of IDSs

- The network or computer is constantly monitored for any invasion or attack.
- The system can be modified and changed according to needs of a specific client and can help outside as well as inner threats to the system and network.
- It effectively prevents any damage to the network.
- It provides a user friendly interface which allows easy security management systems.
- Any alterations to files and directories on the system can be easily detected and reported.

Disadvantages of IDSs

The only disadvantage of an Intrusion Detection System is it cannot detect the source of the attack, and in case of attack just lock the whole network.

Open Source Intrusion Protection Solutions

Snort: Snort is the most popular open source IDS/IPS system available. It is capable of performing real-time protocol analysis and content searches to detect malware, similar to a commercial IDS system. Snort supports a wide range of operating systems from XP to Linux, AIX, Solaris, and so on, and has its own rule-based language to design intrusion-detection policies and protective actions.

OSSEC: OSSEC is another host-based open source project that addresses intrusion-protection needs. It comes with ample documentation, and supports multiple operating systems.

19.7.2 Intrusion Prevention Systems (IPS)

An Intrusion Prevention System (IPS) is a security solution that provides security against unauthorized access and malicious activities at the network level. It not only monitors the network traffic, it also ensures protection against intrusions that take place on the network. The main function of an Intrusion Prevention System is to analyze all the inbound and outbound network traffic for suspicious activities and perform appropriate actions instantaneously to prevent the intruders from entering the internal network.

An IPS detects the unwanted or malicious data packets that may cause severe damage to the network, hosts, and the resources that the network may have. Using IPS the various network security attacks such as brute force attacks, Denial of Service (DoS) attacks, and vulnerability detection can be handled easily. Moreover, an IPS also ensures prevention against protocol exploits.

The other functions that an Intrusion Prevention System can perform include:

- Blocks network traffic from the offending source IP addresses.
- Resets the TCP connection.
- Corrects un-fragment packet streams.
- Corrects Cyclic Redundancy Check (CRC) errors.
- Checks TCP sequencing issues.
- Sanitizes unsolicited transport and network layer options.

IPS Architecture

The important component of intrusion prevention system deployment is to have one or more sensors. Each sensor is strategically positioned to monitor traffic for particular network segments. Each network segment is monitored by a separate sensor; also a single sensor can monitor several network segments simultaneously. In order to monitor key network segments throughout the network, IPS sensors are often deployed wherever networks with different security policies connect, such as Internet connection points or where internal user networks connect to internal server networks. Figure 19.19 depicts the deployment of an IPS.

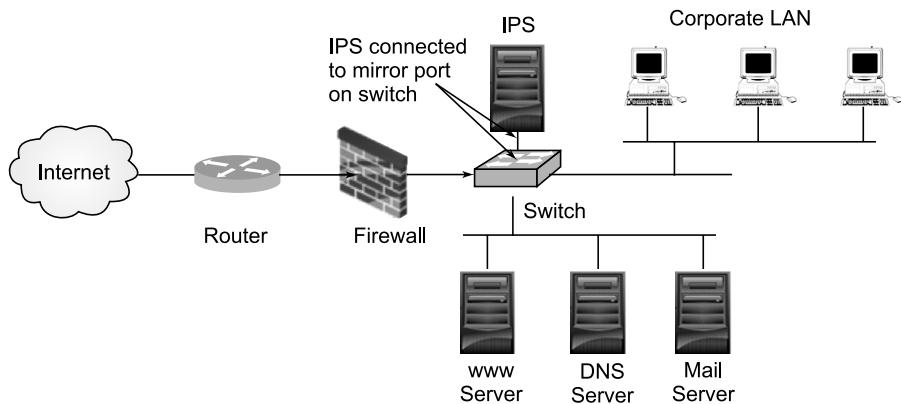


FIGURE 19.19 Intrusion prevention system.

Types of IPS

Intrusion prevention systems (IPs) can be classified into four different types, as shown in Figure 19.20.

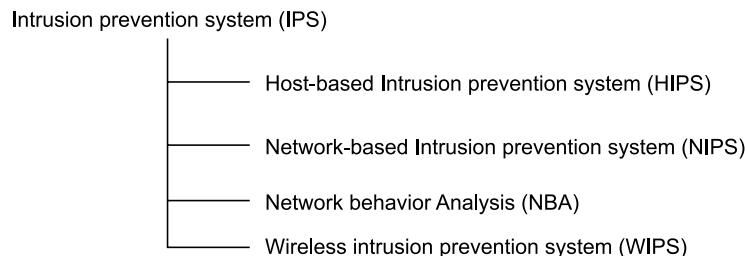


FIGURE 19.20 Classification of IPS.

Host-based Intrusion Prevention System (HIPS): A Host Intrusion Prevention System (HIPS) aims to stop malware by monitoring the behavior of code. This makes it possible to help keep the system secure without depending on a specific threat to be added to a detection update. It is an installed software package which monitors a single host for suspicious activity by analyzing events occurring within that host. The normal method of a HIPS is runtime detection. It intercepts actions when they occur, but some HIPS also offer pre-execution detection. This means that the nature of an executable is analyzed before it runs, to check for suspicious behavior.

Network-based Intrusion Prevention System (NIPS): The NIPS monitors the network for malicious activity or suspicious traffic by analyzing the protocol activity. Once the NIPS is installed in a network, it is used to create physical security zones. This, in turn, makes the network intelligent and quickly distinguishes good traffic from bad traffic. In other words, the NIPS becomes like a prison for hostile traffic such as Trojans, worms, viruses, and polymorphic threats. NIPS are reported to have a high rate of false positives but have blocked thousands of known attacks.

Network Behavior Analysis (NBA): Network behavior analysis (NBA) is a way to enhance the security of a proprietary network by monitoring traffic and noting unusual actions or departures from normal operation. Conventional intrusion prevention system solutions defend a network's perimeter by using packet inspection, signature detection, and real-time blocking. NBA solutions watch what's happening inside the network, aggregating data from many points to support off-line analysis. CAMNEP is a NBA tool which uses several anomaly detection algorithms to classify legitimate and malicious traffic.

Wireless Intrusion Prevention System (WIPS): A wireless intrusion prevention system (WIPS) is a network device that monitors the radio spectrum for the presence of unauthorized access points (intrusion detection), and can automatically take countermeasures (intrusion prevention). The primary purpose of a WIPS is to prevent unauthorized network access to local area networks and other information assets by wireless devices. These systems are typically implemented as an overlay to an existing Wireless LAN infrastructure, although they may be deployed as stand-alones to enforce no-wireless policies within an organization. Some advanced wireless infrastructure has integrated WIPS capabilities.

19.8 HONEYPOTS

Honeypots are closely monitoring decoys or lure systems that are employed in a network to study the behaviors of hackers and to alert network administrators of a possible intrusion. A honeypot is a useful tool for network forensics and intrusion detection. It can play an important role in the real-time adjustment of the security policies of the enterprise domains where it is deployed. Honeypots can simulate a variety of internal and external devices, including Web servers, mail servers, database servers, application servers, and even firewalls.

The two important reasons for setting up honeypots are:

1. To learn how an intruder probes and attempts to gain access to a corporate network. This information will provide an insight into attack methodologies, which also helps the organizations with better security strategies.
2. To gather forensic information required to aid the anxiety of intruders.

19.8.1 Levels or Layers of Tracking

The information provided on an intruder depends on the levels of tracking that are enabled on the honeypot. Typical tracking levels include the firewall, system logs on the honeypot, and sniffer tools.

Firewalls: The firewall provides activity logging capabilities which can be used to identify how an intruder is attempting to get into a honeypot. Also, studying the order, sequence, time stamps, and type of packets used by an intruder to gain access to the honeypot will help to identify the tools and methodology being used by the intruder and their intrusion.

System logs: The system logging capacities built in the operating systems help to identify what changes or attempts have been made.

Sniffer tools: Sniffer tools provide the capabilities of seeing all of the information or packets going between the firewall and honeypot system. Most of the sniffers available are capable of decoding HTTP, Telnet, and SMTP packets. This enables them to determine which methods the intruder is trying to use in much more detail than with a firewall or system logging alone.

19.8.2 Types of Honeypots

Honeypots are generally divided into the following three categories:

1. **High Interaction Honeypot Network (Honeynet):** A high interaction honeypot network is an actual network of honeypots accessible through a controlled gateway simulating all aspects of a network. It provides much more information about the possible attackers. As a honeynet is a dynamic environment, this allows the quick modification of the honeynet infrastructure to adapt it to different environments or industrial sectors. Critical data acquisition obtained from the possible attacks received by the honeynet will allow the development of mechanisms, tools, and procedures to mitigate these attacks in the near future

or recover in a most effective way. The high interaction Honeypot network as a Honeynet is shown in Figure 19.21.

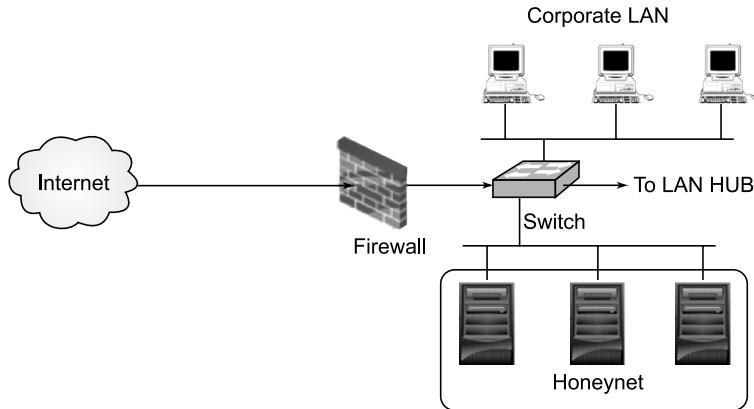


FIGURE 19.21 Honeynet.

2. **Production Honeypot:** Production Honeypots are used to protect the corporate network from attacks, and they are implemented inside the production network to improve overall security. They capture a limited amount of information; mostly low interaction honeypots are used as production honeypots.

Normally production honeypots are low-interaction honeypots which are easier to deploy. They provide limited information about the attacks or attackers. The purpose of production honeypots is to help mitigate risk in an organization.

3. **Research Honeypot:** A research honeypot is used to learn about the tactics and techniques of the Blackhat community. The honeypot operator gains knowledge about the Blackhats and tactics. When a system is compromised, the administrators usually find the tools used by the attackers. The research honeypot provides a real-time insight on how the attack happened.

19.8.3 Locations for the Deployment of Honeypots

If the function of a honeypot is to be effective, it must be deployed in the correct location. A production honeypot used for detection of vulnerabilities may be deployed in one area, while a research honeypot used to learn

about attacks may be deployed in another. For example, if the goal is to detect attackers who have penetrated the corporate network, then the honeypot is placed on the internal network behind the perimeter firewalls. However, if the goal is to research how many attack attempts were made against the organization each day, such a honeypot would have the greatest value deployed outside the perimeter firewall. This way the honeypot could detect all the activity to which the corporate network is vulnerable. In general, most production honeypots are placed behind an organization's security perimeter. To protect the entire organization network, honeypots provide the best value placed on internal networks or networks at high risk, such as DMZs. Access control devices such as firewalls keep the intruders out, while the honeypots work best by interacting with anything that gets through the perimeter security. The three types of production honeypots—prevention, detection, and response—and the best practices for deploying them are discussed as follows.

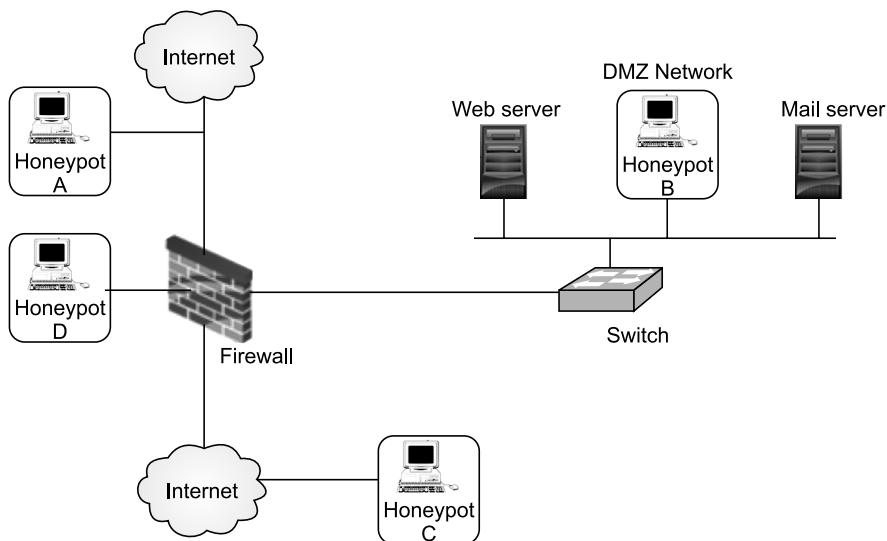


FIGURE 19.22 Different locations to deploy a honeypot.

Placement for Prevention

The purpose of prevention honeypots is to deceive or deter attackers. Such honeypots most likely have the greatest value on the DMZ or internal networks. Figure 19.22 shows a network diagram with four possible locations for

deploying a honeypot, each with its advantages and disadvantages. The location for deployment of a honeypot depends on what is to be achieved. In the case of production honeypots used for deception or deterrence, this would be Honeypot B or Honeypot C. These honeypots are meant to deceive any attackers who successfully penetrate the firewall.

Placing the honeypot on the outside of an organization may only increase risk. In Figure 19.22, Honeypot A is on the outside. Attackers scanning the corporate network would find this honeypot and attempt to interact with it. This successful interaction may raise awareness about the network and bring in more attackers.

Placement for Detection

If the purpose of a honeypot is to detect attacks or unauthorized activity, then there is no question that it should be placed behind the firewall or perimeter security mechanisms. For detection of security threats, the honeypots are to be placed on the internal networks or networks at high risk, such as the DMZ. Once again, this would be Honeypots B and C in Figure 19.22.

Placement for Response

The honeypots for this purpose are similar to both prevention and detection honeypots. They also have the greatest value behind perimeter defenses. The value of a reaction honeypot is providing information on successful attacks. It mirrors the production system in functionality and appearance. When the production system is compromised, the honeypot may be compromised also, since it has the same vulnerabilities. Thus, reaction honeypots should be placed in the same location as the production systems they are mirroring. If the honeypot mirrors a production Web server, it should be placed next to the Web server, such as Honeypot B in Figure 19.22. If the production Web server is attacked, the reaction honeypot also will be attacked; this provides additional information that will assist the system administrator in reacting to the attack.

19.8.4 Advantages and Disadvantages of Honeypots

Advantages: The following are some of the advantages of setting up a honeypot:

1. Using a honeypot in a network, one can learn about incident response; setting up a system that intruders can break into will provide knowledge on detecting hackers' break-ins and cleaning up after them.

2. The knowledge of hacking techniques can protect the real system from similar attacks.
3. The honeypot can be used as an early warning system; setting it up will alert administrators of any hostile intent long before the real system gets compromised.
4. The honeypot detects attacks which are not caught by other security systems.
5. A honeypot collects small amount of data, but almost all of this data is about real attacks of unauthorized activity.
6. Reduce false positive – the honeypot almost always detects or captures attacks or unauthorized activities, which reduces false positives.
7. False negative – the honeypot detects and records any unseen or unnoticed attacks or behavior.

Disadvantages: The disadvantages of the system are as follows:

1. First and foremost is that the honeypot may be used as a stepping stone to further compromise the network, whether the user's own internal network or some network on the Internet.
2. The honeypots add complexity to the network. Increased complexity may lead to increased exposure to exploits.
3. The honeypots must be maintained just like any other networking equipment and services. Maintenance not only requires that the system be shut off but also requires just as much use of resources as a real system.
4. Building a honeypot requires at least a whole system dedicated to it, and this may be an expensive resource for some corporations.
5. Risk of takeover – Since there are many security holes in honeypots, a malicious attacker can take over the honeypot and use it to gain access and hack other networks.

Some of the honeypots available are:

- Trip Wire – Trip Wire
- Network Associate – CyberCop Sting
- Fred Cohen and Associates - Deception Toolkit
- Resource Technologies – Man Trap

SUMMARY

- The computer virus is a program written to alter the way a computer operates without the permission or knowledge of the user.
- The virus goes through the four phases: Dormant phase, Propagation phase, Triggering phase, and Execution phase.
- Computer viruses are classified as boot sector viruses, file infecting viruses, mass mailer viruses, macro viruses, polymorphic viruses, armored viruses, stealth viruses, retro viruses, and multiple characteristic viruses.
- A computer worm is a self-replicating program that penetrates an operating system with the intent of spreading malicious code.
- The different types of computer worms are: e-mail worms, instant messaging worms, Internet worms, Internet Relay Chat (IRC) worms, and file sharing network worms.
- Adware is a software package which automatically plays, displays, or downloads advertisements to a computer.
- An IDS continuously monitors traffic flow along a network path, normally at the interface between the trusted and the untrusted sections. The IPS instead is an active service which directly intercepts traffic and allows only permitted packets to pass through it, blocking everything else.
- Honeypots are closely monitoring decoys or lure systems that are employed in a network to study the behavior of hackers and to alert the network administrator of a possible intrusion.

REVIEW QUESTIONS

1. Explain the four phases of a virus.
2. Differentiate between adware and a virus.
3. Explain the boot sector virus.
4. Describe the worm attack procedure.
5. Explain different types of worms.
6. What are the detection methods used by IDS?
7. Explain the different types of IDS.

8. Explain in detail about network-based IDS and host-based IDS.
9. Describe the IDS and their approaches in protecting networks and host information assets.
10. What is the purpose of signature based IDS?
11. Mention the limitations of IDS.
12. Discuss the different types of intrusion detection and prevention systems with suitable examples.
13. Why is IDS needed?
14. Mention the limitations of IDS.
15. Explain the different types of intrusion detection systems.
16. Explain statistical anomaly detection and rule based intrusion detection.
17. What are honeypots? Explain the functions of honeypots.
18. Define padded cells in honeypots.
19. Explain the classification of honeypots.
20. Give the advantages and disadvantages of honeypots.
21. Write a short note on packet sniffers, honeypots, and honeynets.
22. What are the measures that may be used for intrusion detection?
23. How does a signature-based IDPS differ from a behavior-based IDPS?
24. Describe IDS and their approaches in protecting network and host information assets.
25. Write short notes on:
 - (a) honeypots, honeynets
 - (b) Padded cell systems
 - (c) Trap and Trace systems
 - (d) Active intrusion prevention
26. Write briefly about the signature-based Intrusion Detection Systems.
27. Explain about host-based Intrusion Detection Systems in brief.

28. What is an audit record? What is the use of an audit record in intrusion detection?
29. What is IDS? Explain profile-based IDS.
30. How is the behavior of an intruder found?

MULTIPLE CHOICE QUESTIONS

1. _____ infect the first sector of the hard drive, where the master boot record is stored.
(a) Boot sector viruses (b) File infection viruses
(c) Mass mailer viruses (d) None of the above
2. Mass mailer viruses search _____ programs.
(a) txt (b) jpeg
(c) e-mail (d) .doc
3. A(n) _____ virus is a virus composed of a sequence of instructions that is interpreted rather than executed directly.
(a) macro (b) polymorphic
(c) armored (d) stealth
4. _____ have the capabilities to change their appearance and change their code every time they infect a different system.
(a) Macro viruses (b) Mass mailing viruses
(c) File infecting viruses (d) Polymorphic viruses
5. Stealth viruses have the capabilities to hide from _____ or _____ software by making changes.
(a) operating systems; antivirus
(b) file systems; antivirus
(c) operating systems; application program
(d) none of the above

CHAPTER 20

FIREWALLS AND VIRTUAL PRIVATE NETWORKS (VPN)

20.1 INTRODUCTION

A firewall is an access control device that looks at the IP packets, compares with policy rules, and decides whether to allow, deny, or take some other action on the packet. A firewall is one of the very important security mechanisms used in a network. The basic function of a firewall is to secure the network traffic against unauthorized access of the network. It imposes restrictions on network services. The audit and controlled access performed by a firewall device can generate an alarm for abnormal behavior of the users. It provides a perimeter defense. A firewall is usually placed between two networks to act as a gateway.

Firewalls are the first line of defense between the corporate network and the Internet. The early forms of firewalls were dedicated hardware devices, which were commercially available. The router-based access control lists only provide basic protection and isolation of networks. This enables providing complete security requirements for the network. Due to the rapid development in hacking techniques, firewalls evolved over time and also their functionality covered all the seven layers of the OSI model. A virtual private network (VPN) provides virtual network links based on encrypting and isolating traffic at the packet level during data transmission. Typically VPNs are a default approach to secure communication between any two parties. In this chapter we provide an in-depth overview of firewalls and VPNs, their relevance in a network, and their role in protecting the network.

The need for firewalls: The Internet connects peoples of all ages and of many professions. It is considered as a backbone for the growth of e-commerce. In many organizations the access to the Internet is not restricted. Depending on the organization's local policy, authorization may be restricted to computers located in offices, in which there is an individual who is responsible for use of these machines. Such a policy may be enforced in order to provide some means of security against hacking remote services. The method of implementing such security tends to be dependent on the workstations and servers. Strong security features on these computers secure the network from intrusion.

When a security breach is discovered, each potentially affected system must be upgraded to fix the breach. This requires scalable configuration management and aggressive patching to function effectively. While difficult, this is possible and is necessary if only host-based security is used. A widely accepted alternative or at least complement to host-based security services is the firewall. A firewall may be a single computer system or a set of two or more systems that cooperate to perform the firewall function. It enforces an access control policy between two networks. The firewall security mechanism is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter. The firewall can be thought of as a pair of mechanisms; one that exists to block traffic and another that exists to permit traffic. The main function of a firewall is to protect a corporate network from Internet-based attacks while allowing the users to access the Internet from within the company safely.

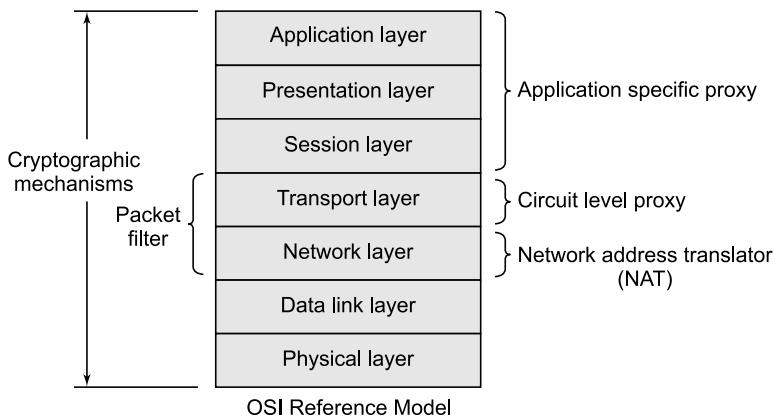


FIGURE 20.1 Firewall functions with reference to the OSI model.

A firewall functions as:

- Packet filtering routers
- Stateful inspection firewall
- Bastion hosts
- Network address translators
- Circuit-level gateways
- Application-level proxies
- Proxy firewalls

In general, the various firewall security mechanisms address themselves to specific layers in the open systems interconnection (OSI) reference model as in Figure 20.1. Several mechanisms can be combined into a comprehensive firewall system, but the mechanisms should be chosen and coordinated so that they do not work against each other.

Basic functions: Firewalls are systems which protect networks or network devices, such as PCs, servers, network devices, control systems, cameras, and so on from unauthorized access by preventing network traffic to or from these systems. The first broad distinction here is the difference between host firewalls and network firewalls. A host-based firewall is installed on a computer (host) or already provided by the operating system as a software feature, such as the Microsoft Windows system firewall or the IP tables firewall provided with most Linux systems.

Network firewalls are devices which have been developed especially for use as a firewall and are placed in the network, rather than on a PC. These host, or hardware, firewalls are important elements in corporate network facilities, especially when they are connected to the external networks or when wired transmissions are combined with less secure network technologies (e.g., wireless networks). As in Figure 20.2, a network firewall serves to set up the network boundary as the first line of defense against attacks and only allows desired traffic into and out of the network.

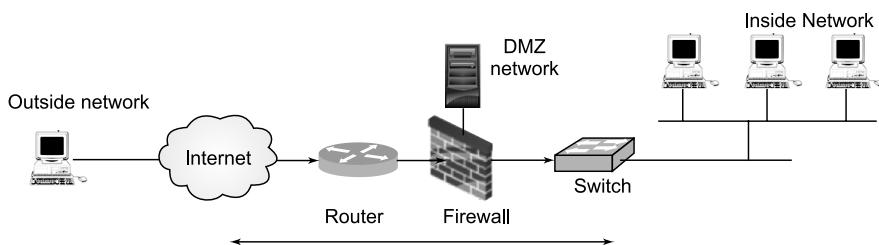


FIGURE 20.2 Network firewall.

Packet filtering: The Packet Filtering Firewall is one of the most basic firewalls. The first step in protecting internal users from external network threats is to implement this type of firewall. Most of the routers have packet filtering built-in, but the problem with the routers is that they are difficult to configure and don't provide extensive logs of the incidents.

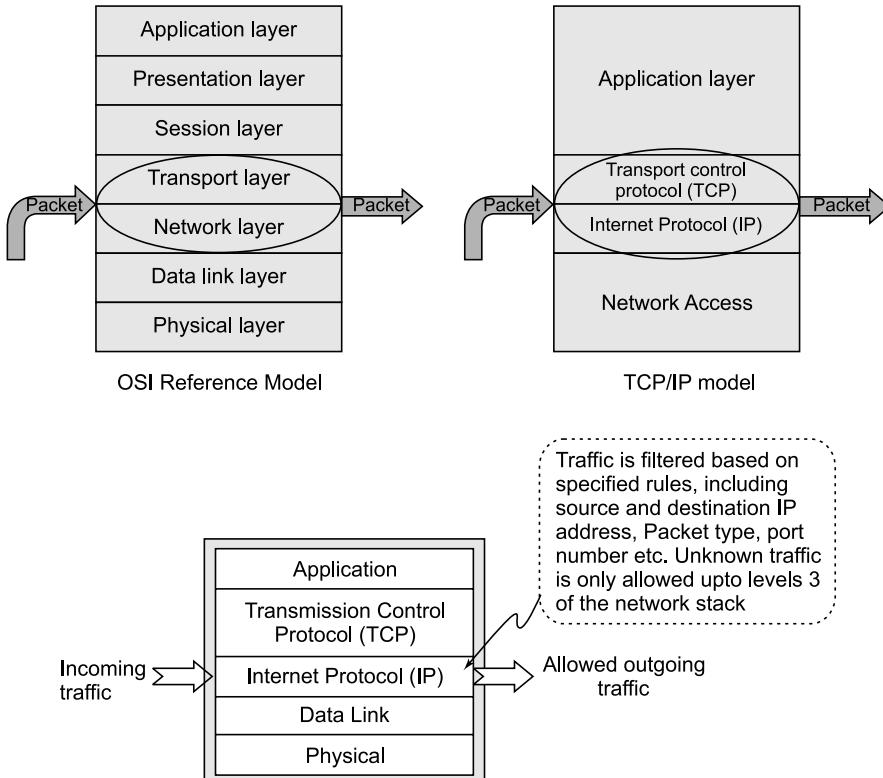


FIGURE 20.3 Packet filtering related layers.

Referring to the definition and the purpose of the firewall, the firewall is the first destination for the traffic coming to the internal network. So, anything which comes to the internal network passes through the firewall. Also any outgoing traffic will also pass through the firewall before leaving the network completely. Due to these functions this type of firewall filter is also called a screening router. Each packet has a header which provides the information about the packet, its source and destination, and so on. The

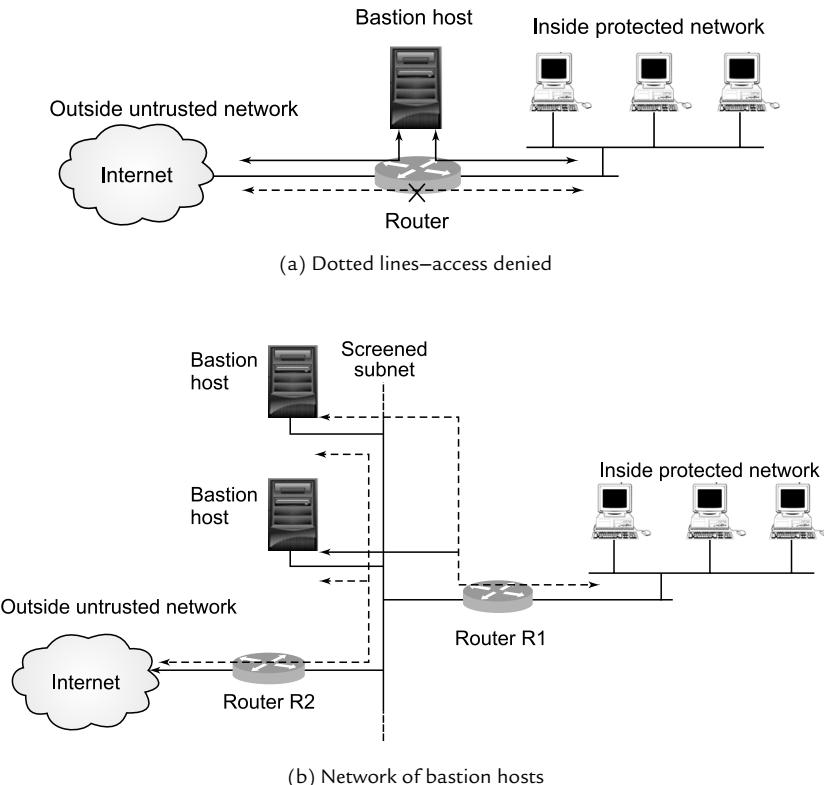
packet filtering firewall inspects these packets to allow or deny them. The information may or may not be remembered by the firewall. Each packet is compared with a set of filter rules and based on any match, the packet is either allowed, denied, or dropped. Packet filtering works on the network layer and transport layer of the OSI reference or TCP and IP layer of TCP/IP, as Figure 20.3 illustrates. It does not remember the state, and hence it is called a stateless firewall.

The types of information in layers 3 and 4 that are used by packet filters include:

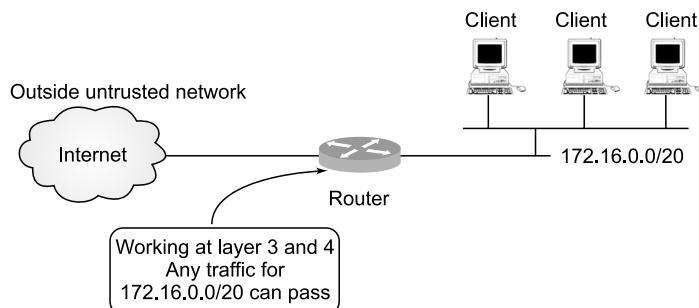
1. *Source IP address of the packet.* This is necessary because IP spoofing might have changed the source IP address to reflect the origin of a packet from somewhere else, rather than replacing the original source.
2. Destination IP Address. The firewall rules should check for IP address rather than DNS names. This prevents abuse of DNS servers.
3. IP Protocol ID.
4. TCP/UDP port number.
5. ICMP message type.
6. Fragmentation flags.
7. IP Options settings.

20.2 PACKET FILTERING ROUTERS AND BASTION HOSTS

The packet-filter firewalls deny access to traffic that does not meet a set of rules and pass traffic that does. In Figure 20.4, the dotted line indicates the access denied and the solid line with arrowheads indicates the traffic allowed access. This is indicated by a dotted line with X and a solid line with arrowheads. In a screened-host firewall, a router at network level controls access to and from a single host—called a bastion host—through which all traffic to and from the protected network must travel. Direct access to the protected network is denied and the bastion host does not forward packets. The bastion host is a highly defended, secured strongpoint that—one hopes—can resist attack. In a screened-subnet firewall, a pair of routers controls access to a small network of *bastion hosts*. The screened subnet is also called a “demilitarized zone” (DMZ).

**FIGURE 20.4** Packet filtering routers and bastion hosts.

Packet filters can be implemented using access control lists (ACLs), which are commonly found in some of the routers. Figure 20.5 shows a router between the private network and the Internet.

**FIGURE 20.5** Basic packet filter.

The packet filter examines every packet against the ACLs for matches. If a match is found, the packet is either permitted or denied passage through the interface. If a match is not found, the packet is implicitly denied passage. Packet filters process information only up to layer 4, making them very fast and efficient. However, packet filters don't track the TCP session information generated when two computers are communicating with one another. When computers first start to communicate using TCP, they perform a three-way handshake, which is used to establish the TCP session. Because these sessions aren't monitored by packet filters, the computers become vulnerable to spoofing.

20.2.1 Important Features of Packet Filters

The firewalls normally follow a few specific rules upon which features are incorporated during firewall designing. A few are listed as follows:

1. TCP ACK bit filtering: When TCP ACK bit filtering is enabled, only response packets (packets with the TCP ACK bit set in three-way handshake) are allowed through. This effectively blocks all connection attempts from being initiated through this filter rule. TCP ACK bit filtering is often applied to all TCP inbound filters to prevent external hosts from initiating TCP connections to internal hosts; however, it should not be used in the following circumstances:

- An internal service is provided to an external host (because the first TCP connection packet does not have the ACK bit set).
- TCP applications, such as FTP, require incoming connections.
- Packets, such as UDP packets, do not have an ACK bit.

2. Dynamic packet filtering: Dynamic packet filtering tracks the outgoing packets it has allowed to pass and allows only the corresponding response packets to return. When the first packet is transmitted to the Internet, a reverse filter is dynamically created to allow the response packet to return. To be counted as a response, the incoming packet must be from the host and port to which the outbound packet was sent.

Stateful packet filtering supports both connection oriented and connectionless protocols (TCP, UDP, ICMP, etc.). Dynamic packet filtering monitors each connection and creates a temporary (time-limited) inbound filter exception for the connection. This allows blocking the incoming traffic originating from a particular port number and address while still allowing return traffic from that same port number and address. The reverse filter is created by extracting the following packet information:

- Source IP address
- Source interface
- Source port
- Destination IP address
- Destination interface
- Destination port
- Protocol type

3. Fragmented packet filtering: When an IP datagram is fragmented, only the first packet has the complete header and transport information. All subsequent packets do not have the transport information, such as the port number. Formerly, it was safe to allow these fragments to pass through the firewall unchecked, as long as the first packet was dropped. The target host would eventually drop all subsequent packets, and reassembly could not be completed without the first packet.

This is no longer true today. Fragmented packets can be used to launch denial-of-service attacks by continuously flooding the network with fragment packets to consume network bandwidth and resources on the target host. Tiny fragments can also be used to bypass the firewall with overlapped fragments or by manipulating the fragment flags.

To protect against fragment attacks with manipulation of the fragment flags, an automatic fragmented packet filtering mechanism is added. Fragmented packet filtering discards the first packet, which carries the complete header and transport information, and all subsequent fragmented packets if they have the same source and destination IP addresses and interfaces.

20.2.2 Types of Packet Filtering

A packet filtering firewall allows only those packets to pass which are allowed as per your firewall policy. Each packet passing through is inspected, and then the firewall decides to pass it or not. The packet filtering can be divided into two parts:

- Stateless packet filtering
- Stateful packet filtering

Stateless Packet Filtering: If the information about the passing packets is not remembered by the firewall, then this type of filtering is called stateless packet filtering. These types of firewalls are not smart enough and can be deceived very easily by the hackers. These are especially dangerous for UDP types of data packets. The reason is that the allow/deny decisions are taken on packet-by-packet basis, and these are not related to the previous allowed/denied packets.

Stateful Packet Filtering: If the firewall remembers the information about the previously passed packets, then that type of filtering is stateful packet filtering. These can be called smart firewalls. This packet filtering is also known as dynamic packet filtering. In the stateful packet filtering method, the packet filtering process goes beyond basic packet filtering. It keeps track of the state of connection flows for all the packets, in both directions. It also keeps track of all the IP addresses currently connected at any point of time.

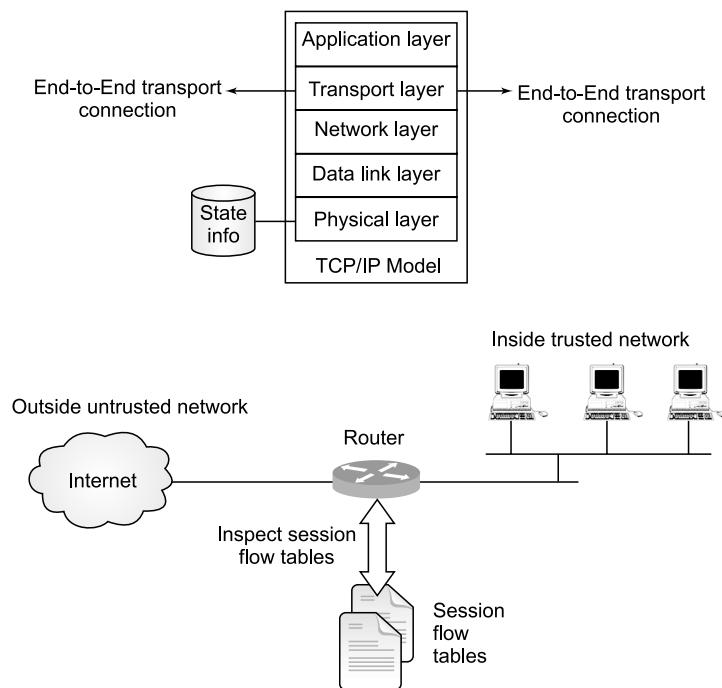


FIGURE 20.6 Stateful inspection.

This type of firewall combines the speed of packet filters with the enhanced security of stored session information characterized by proxies. While traffic is being forwarded through the firewall, stateful inspections of the packets create slots in session flow tables. These tables contain source and destination IP addresses, port numbers, and TCP protocol information. Before traffic can travel back through the firewall, stateful inspections of the packets are cross-referenced to the session flow tables for an existing connection slot. If a match is found in the tables, the packets are forwarded; otherwise, the packets are dropped or rejected. Figure 20.6 shows the client and session flow tables being used.

20.2.3 Circuit-Level Gateways

Many firewalls now include built-in support for Socks (the name derives from Unix Sockets), software that allows applications to access a variety of communication protocols. Thus Socks can handle many different types of traffic, routing packets between compatible clients and servers in the untrusted network and the protected one. In effect, it forms a circuit between a client and server as in Figure 20.7; but it acts as a proxy, too, forwarding only those packets deemed acceptable.

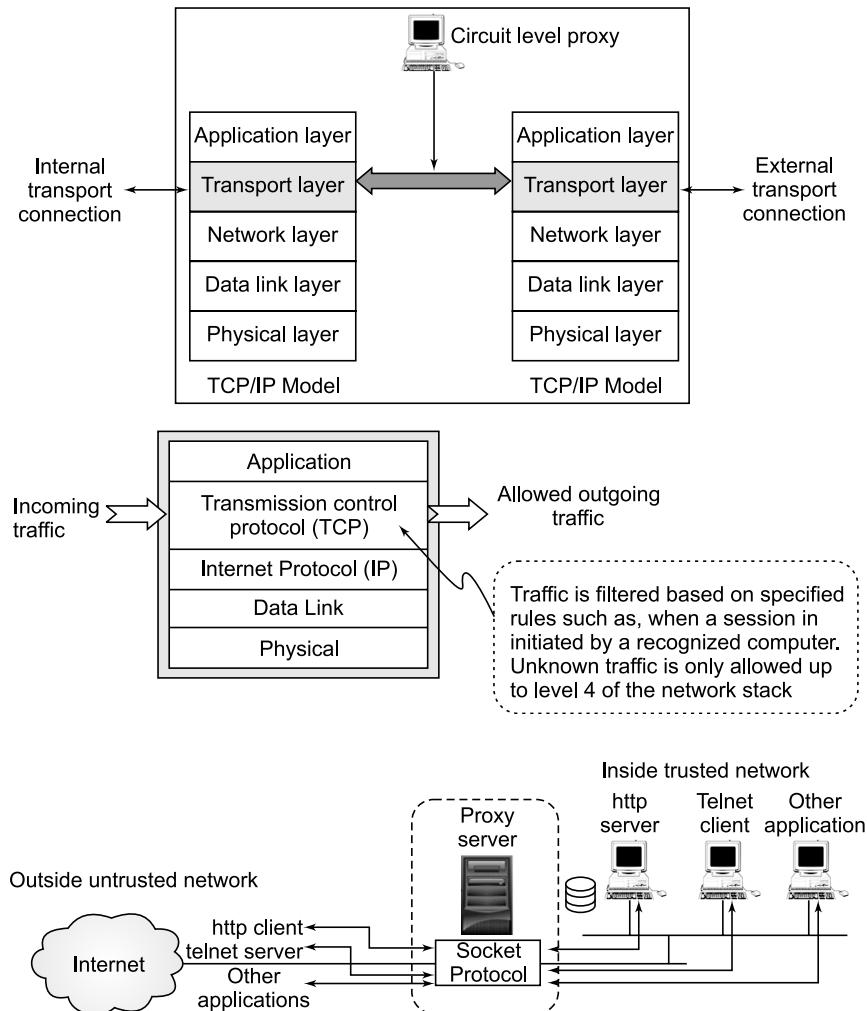


FIGURE 20.7 Circuit-level gateway.

Application Level Gateways (Proxy): A firewall is also capable of inspecting application level protocols. This requires the firewall to understand certain specific application protocols. An application-level firewall uses application-specific proxies. This can interact with the source and destination of a message to verify whether it meets security standards, and then allows or denies access on the basis of its evaluation. Separate proxies are needed for each application. Further, a so-called dual-homed application-level firewall can be built by installing two interfaces, one on each network. So a popular location for such a firewall is a bastion host, either a screened-host or screened-subnet firewall. A bastion host provides an external facing point of entry into a network containing network instances. This host can provide a single point of fortification or audit and can be started and stopped to enable or disable inbound SSH communication from the Internet.

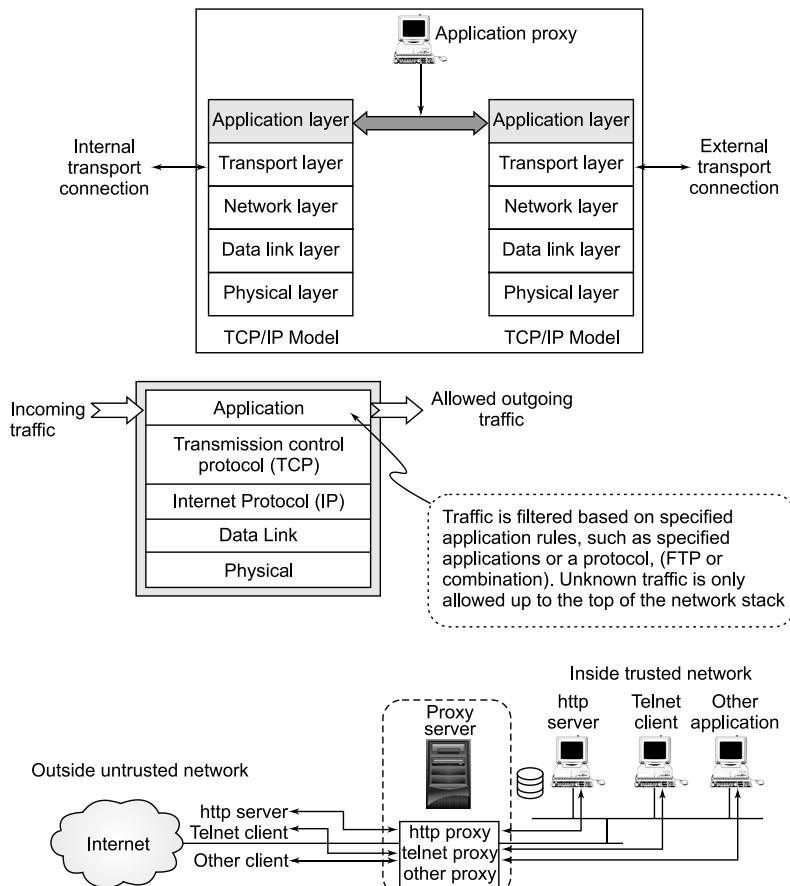


FIGURE 20.8 Application-level gateway.

The application level gateway (ALG) shown in Figure 20.8 manages specific application protocols such as SIP (Session Initiation Protocol) and FTP (File Transfer Protocol). An ALG acts as an intermediary between the Internet and an application server that can understand the application protocol. The ALG appears as the end point server and controls whether to allow or deny traffic to the application server. It does this by intercepting and analyzing the specified traffic, allocating resources, and defining dynamic policies to allow traffic to pass through the gateway.

ALG has the Following Functions

- Application level gateways work on the Application layer of the OSI model and provide protection for a specific Application Layer Protocol. A proxy server is the best example of application level gateway firewalls.
- Application level gateways work only for the protocol which is configured. For example, if we install a web proxy based firewall, than it will only allow HTTP Protocol Data. They are supposed to understand application specific commands such as HTTP:GET and HTTP:POST, as they are deployed on the Application Layer for a Specific Protocol.
- Application level firewalls can also be configured as Caching Servers, which in turn increase the network performance and make it easier to log traffic.
- It allows client applications to use dynamic TCP/UDP ports to communicate with known ports used by server applications, even if the firewall configuration allows traffic through only a limited number of ports. Without an ALG, the ports would either get blocked, or the network administrator would need to open up a large number of ports in the firewall, weakening the network and allowing potential attacks on those ports.
- It recognizes application specific commands and offers security controls over them.
- It can convert the network layer address information that is found in an application payload.
- Synchronizes multiple streams or sessions between hosts.

The step-by-step process of an application proxy firewall is:

Step 1: The client attempts to connect to the Web server located on the outside. The client also requests a Web page from the proxy.

Step 2: The proxy server receives the request and forwards that to the appropriate Web server.

Step 3: The Web server receives the requests and responds back to the proxy server with the requested information.

Step 4: The proxy server receives the information and forwards it to the original client.

20.3 PROXY FILTER (SERVER)

Proxy filters, also known as application proxy servers, extend beyond the reach of packet filters by examining information from layers 4-7. A proxy server sits between the client and the destination working as a middleman between the two communicating parties, as in Figure 20.9. It requires the client to establish a session with the proxy itself, which in turn creates a second session between itself and the destination. Consider, for instance, a client computer that requests information from a remote website. The client creates a session with the proxy server, which can then authenticate the user for valid access to the Internet before creating a second session between the website and itself. As the information comes back from the website, the proxy server examines layers 4-7 for a valid connection to the inside network.

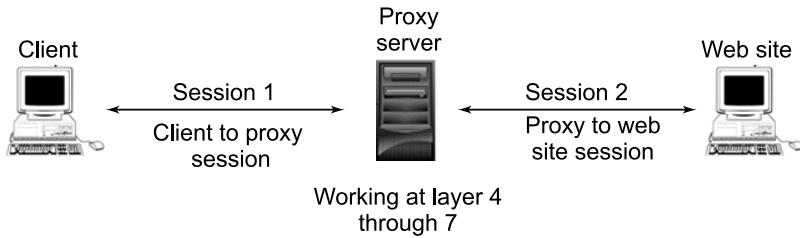


FIGURE 20.9 Proxy server sessions.

Although proxies can provide some of the most effective measures of protection, they can introduce speed and performance issues, particularly when a large number of sessions are being simultaneously negotiated. They are also built on general-purpose operating systems such as Unix, Linux, or Microsoft Windows, which can make them vulnerable to OS-related attacks.

Reverse Proxy Firewall

A reverse proxy firewall functions in the same way as proxy firewalls, with the exceptions that they are used to protect the servers and not the clients.

Clients connecting to the Web server may unknowingly be sent to the proxy server, where it services the requests on behalf of the client. The proxy server may also be able to load balance the requests to multiple servers, consequently spreading the workload.

The step-by-step process of a reverse proxy firewall is:

Step 1: The client opens a Web browser and enters the URL that directs them to the associated proxy Web server requesting information.

Steps 2 and 3: The proxy server can have multiple locations from which to gather information (different servers as Application server 1 and Application server 2).

Steps 4 and 5: The proxy server prepares the content received from Application servers 1 and 2 for distribution to the requesting client.

Step 6: The proxy server responds to the client with the requested information.

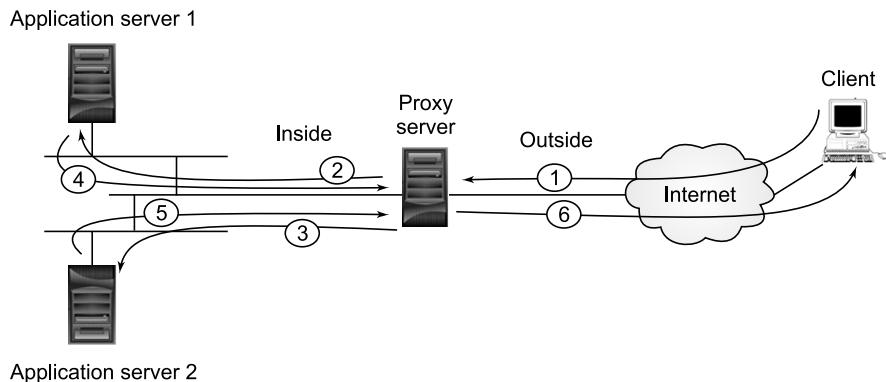


FIGURE 20.10 Reverse proxy firewall.

Core Functions of Firewalls

Firewalls also perform certain functions in addition to controlling application communication. These include network address translation (NAT), an important function performed by a firewall—the process of converting an IP address to another and logging of traffic.

20.4 NETWORK ADDRESS TRANSLATION (NAT)

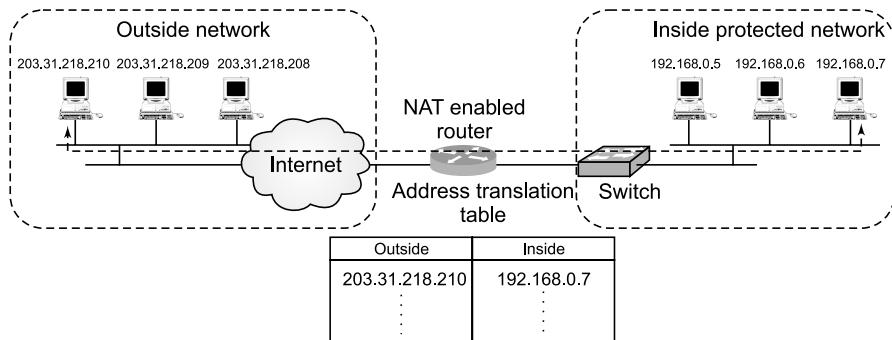


FIGURE 20.11 Address translation table.

A network address translator (NAT) hides internal addresses from the outside world. Network address translation routers contain a table of outside and inside addresses as in Figure 20.11. They translate the outside address of an incoming message into the hidden inside address, and do the reverse for an outgoing message. The primary version of IP addresses used on the Internet is IPv4 or IPv6. When a network is connected to the Internet, it must translate its private IP address into the public IP address in order to be routable. By doing this a large number of hosts behind a firewall can take turns or share a few public addresses when accessing the Internet. Table 20.1 represents some of the private addresses and their range.

TABLE 20.1 Private Addresses.

Address	Mask	Range of Private Addresses
10.0.0.0	255.0.0.0	10.0.0.0 – 10.255.255.255
172.16.0.0	255.240.0.0	172.16.0.0 – 172.31.255.255
192.168.0.0	255.255.0.0	192.168.0.0 – 192.168.255.255

NAT is usually implemented in a firewall separately from the policy or rule set. The NAT has been defined to translate addresses between one host and another, but it does not mean those hosts will be able to communicate. It is controlled by the policy defined in the firewall rule set.

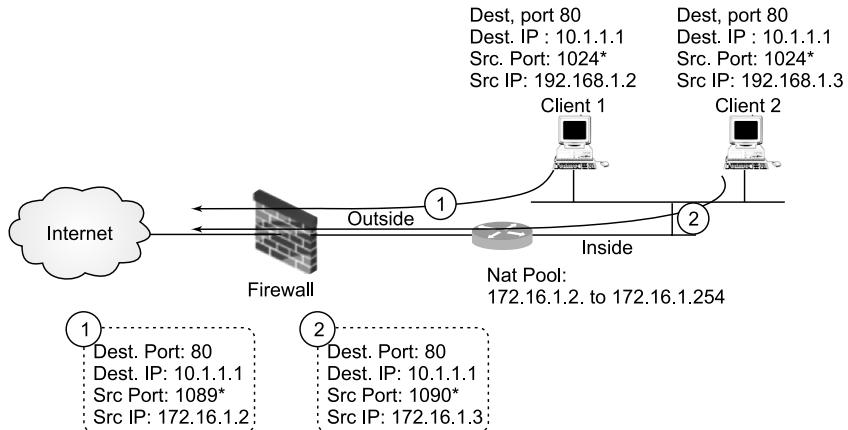


FIGURE 20.12 Network address translation.

The hosts have both public and private IP addresses. This IP information present within a packet header will change depending on where the packet is viewed. To understand this concept, the address when viewed on the trusted side of the firewall is referred as a local address. The firewall translates this address; the address will be the host's global address.

Figure 20.12 shows two clients located on the inside of the firewall. Client 1 has an IP address 192.168.1.2 and Client 2 has an IP address of 192.168.1.3. A NAT pool of addresses has been assigned to the firewall using IP addresses 172.16.1.2 through 172.16.1.254. When Client 1 attempts to connect to the Internet, the firewall has been configured to take an IP address from the pool and change the client's source address to the address from the pool. The connection passes through the firewall, and the source address is changed from 192.168.1.2 to 172.16.1.2. When Client 2 establishes a connection through the firewall, it will get a second address from the pool. The size of the pool is directly proportional to the number of clients allowed through. When the 255th client attempts to make a connection through the firewall, the pool of addresses will have been completely allocated and the connection will be maintained.

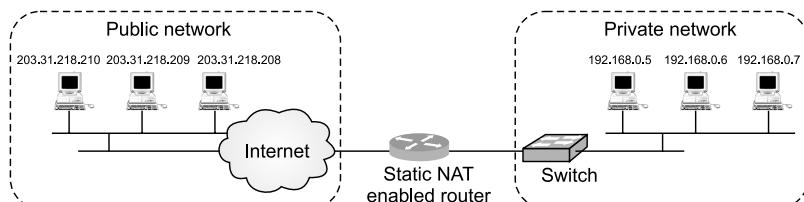


FIGURE 20.13 Network configured for static NAT mode.

Static NAT: Static NAT is mainly created to allow hosts on a private network to be directly accessible via the Internet using real public IPs. It allows the mapping of public IP addresses to hosts inside the internal network. This means that with appropriate firewall settings, the public is able to access hosts on the private network. In Figure 20.13, the private network is connected to the Internet via the router, which has been configured for static NAT mode. In this mode each private host has a single public IP address mapped to it. The private host 192.168.0.5 has the public IP address 203.31.218.208 mapped to it. Therefore, any packets generated by 192.168.0.5 that need to be routed to the Internet will have their source IP field replaced with IP address 203.31.218.208.

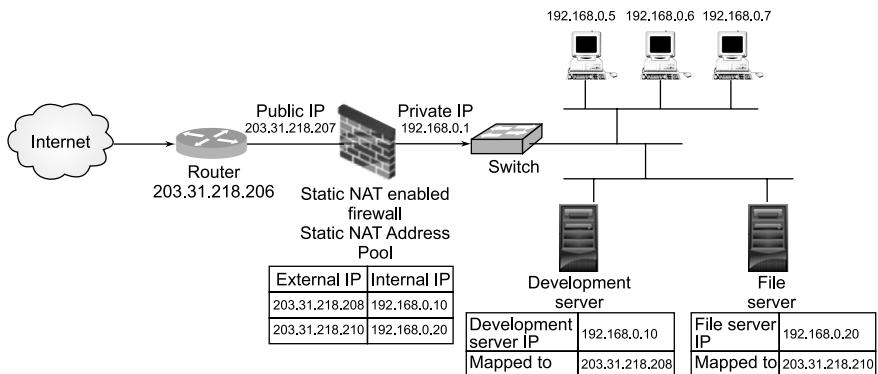


FIGURE 20.14 Static network address translation.

For example, in Figure 20.14 the development server (192.168.0.10) connected in the network needs to be secured and also allowed to access certain users to gain access to various services it offers for development purposes. At the same time the users are also given access to the file server (192.168.0.20). In this case the Static NAT is used, with a set of complex filters to make sure only authorized IP addresses get server access. The firewall device has been set up for Static NAT mode. Packets arrive at the public interface and are checked and then translated within the firewall's memory, where the source IP is replaced with that of the intended internal host. The router simply routes packets between networks.

One of the common uses of static NAT is to provide Internet access to a trusted host inside the firewall perimeter or inbound access to a specific host, such as a web server that needs to be accessible via a public IP address.

Dynamic NAT: Dynamic NAT has some similarities and differences compared to Static NAT. Like in Static NAT, the NAT router creates a one-to-one mapping between an inside local and inside global address and

changes the IP addresses in packets as they exit and enter the inside network. The Dynamic NAT does the same but without making the mapping to the public IP static and usually uses a group of available public IPs. This mapping of an inside local address to an inside global address happens dynamically. Dynamic NAT sets up a pool of possible inside global addresses and defines matching criteria to determine which inside local IP addresses should be translated with NAT. The NAT is then maintained in the firewall table, until some event causes it to be terminated. This event is often a timer that expires after a predefined period of inactivity from the inside host. It removes the NAT entry, and then the address can be reused by a different host. Dynamic NAT allows the dynamic mapping of public IP addresses to each internal host. The mapping is random and will remain for as long as the session is lost.

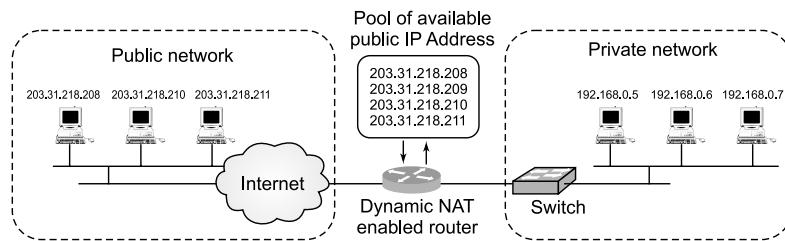


FIGURE 20.15 Dynamic NAT.

In Figure 20.15, is a router which is configured to perform Dynamic NAT for the network. The mapping of a pool of public IP addresses is requested from the ISP (203.31.218.208 – 203.31.218.211), which will be dynamically mapped by the router to the internal hosts. For example, the workstation in a private network with IP address 192.168.0.5 sends a request to the Internet and is assigned the public IP address 203.31.218.208. This mapping between the workstation's private and public IP address will remain until the session finishes. The router is configured with a special NAT timeout, and after this timeout is reached, the router will expire the particular mapping and reuse it for a different internal host.

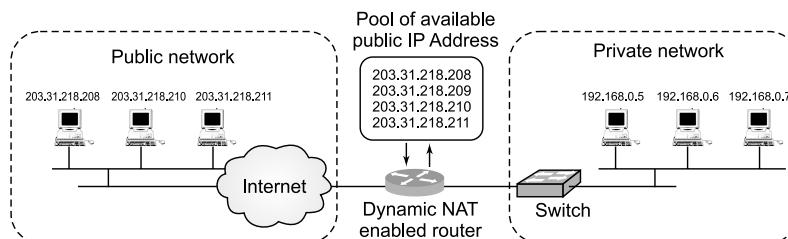


FIGURE 20.16 Dynamic NAT after reallocating IP addresses.

If the browser is restarted after a timeout period, the old mapping once created by the router is deleted and created anew by reallotting the IP address from the pool as in Figure 20.16.

In the example shown in Figure 20.17, the workstations from different workgroups are permitted to access the development servers in the restricted development network. The development network is a separate network designed only for development activities, and it has very limited access to the rest of the networks. Each user in the workgroups is registered with the firewall connected to the development network and therefore can gain access to the development network.

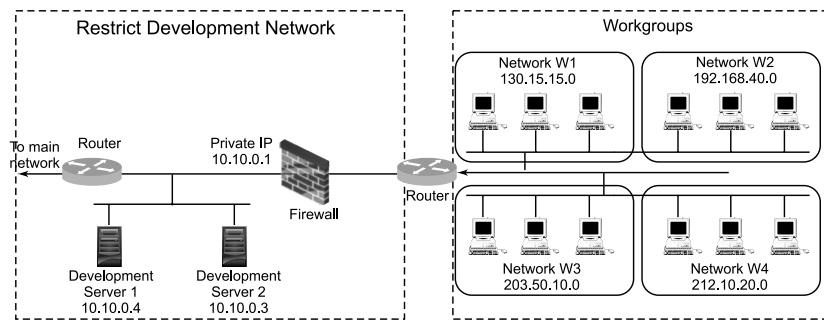


FIGURE 20.17 Restricted access for the workgroup to access the development network.

The development network has configured its firewall only to access particular members of the workgroup W2 network (192.168.40.0/24) to access the development server. This is the main workgroup network.

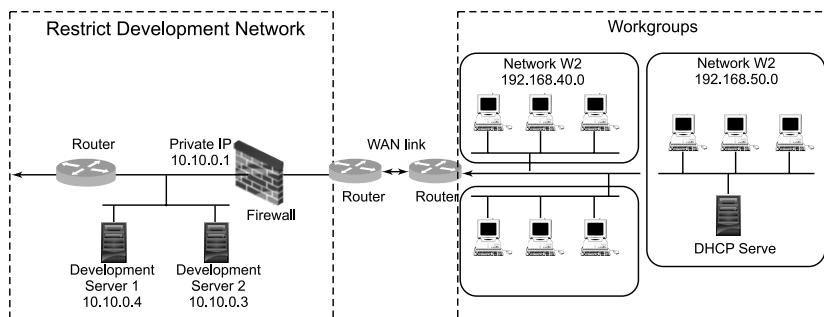


FIGURE 20.18 New network W2 with DHCP server.

If the workgroup network W2 is expanding its development network with many more terminals, a separate network (192.168.50.0/24) is created that also needs access to the development network. All hosts on this new

network will be using the new DHCP server, which means that they will have a dynamic IP address. But the firewall device in the development network strictly prohibits any other workgroup network to gain access except 192.168.40.0. In order for the new network to access the development server, the firewall in the development network has to think that any workstation from the new network is actually part of the 192.168.40.0 network, as in Figure 20.18, and then it won't deny the access of the development server.

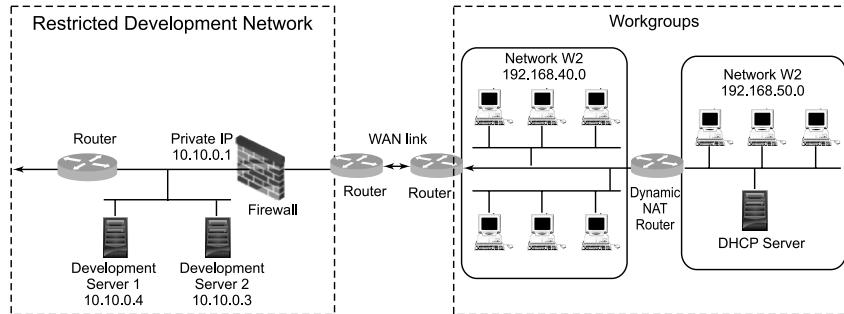


FIGURE 20.19 Dynamic NAT router to map hosts from a new network to an existing network.

To provide maximum security for this new network, dynamic IPs make it more difficult for someone to track a particular host from it by using its IP address. To meet this requirement it implements a dynamic NAT router as shown in Figure 20.19. The router is placed between the existing 192.168.40.0 and the new network (192.168.50.0). Some IP addresses need to be reserved from the 192.168.40.0 network in order to allow the dynamic NAT router to use them for mapping hosts on the new network to the existing network. Using this method the firewall used in the development network can allow the IP address in the new network to access the development server.

20.5 PORT ADDRESS TRANSLATION (PAT)

Port Address Translation (PAT) is a modified form of dynamic NAT that maps multiple private IP addresses to a single public IP address (many-to-one) by using different ports. Static NAT and Dynamic NAT both require a one-to-one mapping from the inside local address to the inside global address. With Port Address Translation, the entire inside local address space can be mapped to a single global address. This is done by modifying the

communication port addresses in addition to the source and destination IP addresses. Thus, the firewall can use a single IP address for multiple communications by tracking which ports are associated with which sessions. PAT allows overloading or the mapping of more than one inside local address to the same inside global address. But this also means that return packets would all have the same destination address as they reach the NAT router. To deliver the packet that belongs to an inside local, the NAT entries in the transaction table are provided with extended entries. This not only tracks the relevant IP address, but also the protocol types and ports. By translating both IP and the port number of a packet (up to 65535), inside local addresses could theoretically be mapped to a single inside global address.

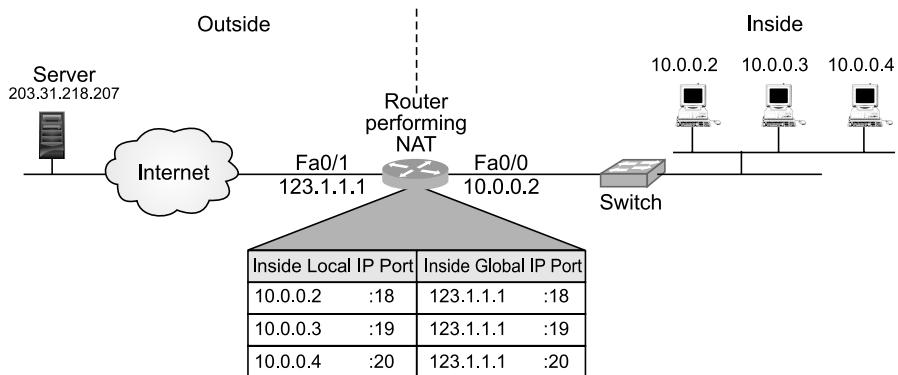


FIGURE 20.20 Port address translation.

When using PAT the router maintains a unique source port number on the inside global IP address to distinguish between translations. In Figure 20.20, each host is assigned to the same public IP address 123.1.1.1 but with different port numbers.

PAT provides an increased level of security because it cannot be used for incoming connections. However, a downside to PAT is that it limits connection-oriented protocols, such as TCP. Some firewalls will try to map UDP and ICMP connections, allowing DNS, Network Time Protocol (NTP), and ICMP echo replies to return to the proper host on the inside network. However, even those firewalls that do use PAT on UDP cannot handle all cases. With no defined end of session, they will usually time out the PAT entry after some predetermined time. This timeout period must be set to be relatively short (from seconds to a few minutes) to avoid filling the PAT table

(although, on modern firewalls, the tables used for these sessions, commonly called *translation tables*, can frequently handle tens of thousands or even millions of sessions). Connection-oriented protocols have a defined end of session built into them that can be picked up by the firewall. The timeout period associated with these protocols can be set to a relatively long period (hours or even days).

Auditing and Logging

Firewalls are excellent auditors. Given plenty of disk space or remote logging capabilities, they can record any traffic that passes through them. Attack attempts will leave evidence in logs, and if administrators are watching systems diligently, attacks can be detected before they are successful. Therefore, it is important that system activity be logged and monitored. Firewalls should record system events that are both successful and unsuccessful. Verbose logging and timely reviews of those logs can alert administrators to suspicious activity before a serious security breach occurs. Verbose logging tracks all changes and settings applied using group policy and its extension to the local computer and to users who log on to the computer. The log file is located at *SystemDrive/Debug*. This folder is a hidden folder. Enabling verbose logging involves adding the registry key for verbose logging. Since this can generate a huge volume of log traffic, the logs are best sent to a Security Information and Event Management (SIEM) system that can filter, analyze, and perform heuristic behavior detection to help the network and security administrators.

Advantages and Disadvantages of Firewalls

Advantages

1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits vulnerability, and provides protection from spoofing and routing attacks.
2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.
3. A firewall is a convenient platform for several Internet functions that are not security related, which include network address translator and network management function.
4. A firewall can serve as the platform for IPsec. Using the tunnel mode capability, the firewall can be used to implement virtual private networks.

Disadvantages

1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

20.6 VIRTUAL PRIVATE NETWORK (VPN)

A VPN is a shared network where private data is segmented from other traffic so that only the intended recipient has access. The term VPN was originally used to describe a secure connection over the Internet. Today however VPN is also used to describe private networks, such as Frame Relay, Asynchronous Transfer Mode (ATM), and Multiprotocol Label Switch (MPLS). Normally data passing over a public network is not secure. The VPN uses a combination of hardware and software to build an encrypted data tunnel through the public network. This tunnel creates a virtual private network within the public network, providing secure data transmission.

20.6.1 VPN Types and Working

VPN are generally divided into two basic categories:

1. Remote Access VPN
2. Site-to-Site VPN

Remote Access VPN: Remote access VPNs are usually used to link to a private network from various remote locations. It enables mobile users to establish a connection to an organization server by using this infrastructure provided by the ISP. A remote access VPN allows users to connect to their corporate intranet or extranet whenever or wherever needed. Users have access to all the resources on the organization's network, as if they are physically located in the organization. The VPN device at the ISP accepts the user's login, then establishes the tunnel to the VPN

device at the organization, and finally begins forwarding packets over the Internet. A remote access VPN connection over the Internet enables a remote access client to initiate a dial-up connection to a local ISP instead of connecting to a corporate network access server (NAS). Using this established physical connection to the local ISP, the remote access client initiates a VPN connection across the Internet to the organization VPN server. After establishing a VPN connection, the remote access client can access the resources of the private intranet. Figure 20.21 shows remote access over the Internet.

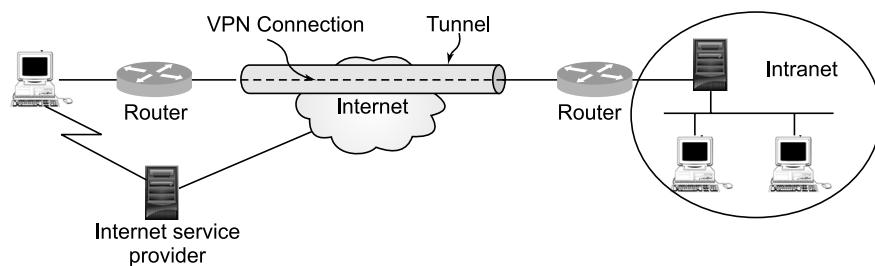


FIGURE 20.21 Remote client accessing resources using VPN.

Site-to-Site VPN: Site-to-Site VPN is used to connect a branch office network to another organization over the Internet. A routed VPN connection across the Internet logically operates as a dedicated wide area network link. In a site-to-site VPN connection, the packets sent from either router across the VPN connection typically do not originate at the routers. As in Figure 20.22 the two routers mutually authenticate to establish the connection.

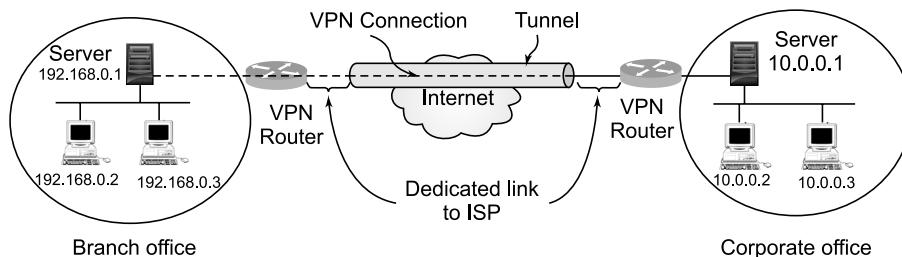


FIGURE 20.22 Site-to-site VPN.

Intranet based VPN Connection: VPN connections help to provide the required security to enable a network segment such as the development network to be physically connected to the intranet. A VPN server separates the development network from the rest of the network as in Figure 20.23.

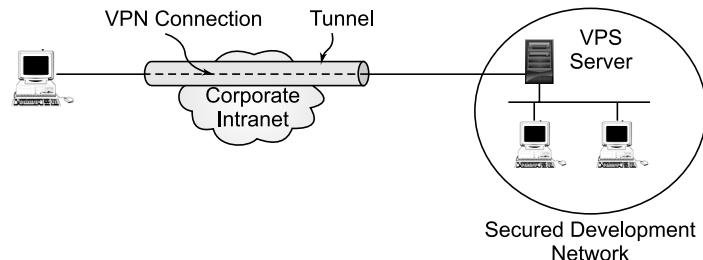


FIGURE 20.23 Intranet-based VPN connection.

The VPN server does not provide a direct routed connection between the corporate intranet and the separate development network segment. Users on the corporate intranet with appropriate permissions can establish a remote access VPN connections with the VPN server and gain access to the protected resources. Once the connection is established, all communication across the VPN connection is authorized to access the development network that is hidden from view.

Site-to-Site VPN Connections Over an Intranet

Two secured networks can be connected over an intranet using a site-to-site VPN connection, as shown in Figure 20.24. To interconnect the two departments from separate locations, this type of VPN connection is used. Their data are highly sensitive and must be communicated in a secured intranet channel. When a VPN connection is established, users on computers on either network can exchange sensitive data across the corporate intranet.

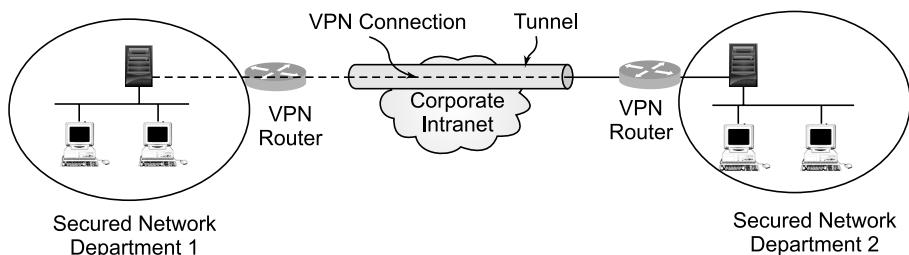


FIGURE 20.24 Site-to-site VPN connection over an intranet.

20.6.2 Components of VPN

A VPN uses numerous methods for keeping the connection and data safe and secure. Some of the essential components that are used in VPN are Authentication, Encryption, Internet Security Protocol (IPsec), and tunneling.

Authentication: Authentication of connection is implemented by using authentication mechanisms like passwords and biometric and cryptographic methods. User name and password are commonly used techniques, because they are easy to deploy and use. The ultimate goal of authentication in a VPN connection is:

- To identify the machine: The machine certificate identifies the system as a valid system for establishing the IPsec security.
- To identify the user: The user proves who they are based on user name, OTP, certificate, or some other mechanism, but the basic function is determining whether the user has permission to establish a connection.

Some of the common Authentication Protocols: The following authentication protocols are used to identify VPN users and grant or deny user access to network resources based on the user's credentials.

Password Authentication Protocol (PAP)

It is a cleartext authentication scheme. The network access server (NAS) requests the user name and password, and PAP returns them in cleartext (unencrypted). This authentication process is not secure, because a malicious user could capture the user's name and password and use it to get subsequent access to the NAS and all of the resources provided by the NAS. PAP provides no protection against replay attacks or remote client impersonation once the user's password is compromised.

Shiva Password Authentication Protocol (SPAP)

The SPAP is a reversible encryption mechanism employed by Shiva Corporation. Generally a computer running Windows XP Professional uses SPAP when connecting to a Shiva LAN Rover. A Shiva client that connects to a server running Routing and Remote Access also uses SPAP. Currently, this form of authentication is more secure than plaintext but less secure than CHAP or MS-CHAP.

Challenge Handshake Authentication Protocol (CHAP)

This mechanism uses an encrypted password authentication technique. The Challenge Handshake Authentication Protocol (CHAP) mechanism

prevents transmission of the actual password on the connection. The NAS sends a challenge, which consists of a session ID and an arbitrary challenge string, to the remote client. The remote client must use the MD5 one-way hashing algorithm to return the user name and a hash of the challenge, session ID, and the client's password. The user name is sent as plaintext.

CHAP protects against replay attacks by using an arbitrary challenge string for each authentication attempt. It also protects against remote-client impersonation by unpredictably sending repeated challenges to the remote client throughout the duration of the connection.

MS-CHAP

Microsoft Challenge Handshake Authentication Protocol (MS-CHAP) is an encrypted authentication mechanism very similar to CHAP. In this protocol also the NAS sends a challenge, which consists of a session ID and an arbitrary challenge string, to the remote client. The remote client must return the user name and an encrypted form of the challenge string, the session ID, and the MD4-hashed password. This design, which uses the MD4 hash of the password, helps to provides an additional level of security because it allows the server to store hashed passwords instead of cleartext passwords or passwords that are stored using reversible encryption. MS-CHAP also provides additional error codes, including a password-expired code, and additional encrypted client-server messages that permit users to change their passwords during the authentication process. In MS-CHAP, both the client and the NAS independently generate a common initial encryption key for subsequent data encryption.

MS-CHAP v2

MS-CHAP version 2 (MS-CHAP v2) is an updated encrypted authentication mechanism. This provides stronger security for the exchange of user name and password identifications and determination of encryption keys. With MS-CHAP v2, the NAS sends a challenge to the client that consists of a session identifier and an arbitrary challenge string. The remote access client sends a response that contains the user name, an arbitrary peer challenge string, and an encrypted form of the received challenge string, the peer challenge string, the session identifier, and the user's password. The NAS checks the response from the client and sends back a response containing an indication of the success or failure of the connection attempt. It provides an authenticated response based on the sent challenge string, the peer challenge string, the encrypted response of the client, and the user's

password. The remote access client verifies the authentication response and, if correct, uses the connection. If the authentication response is not correct, the remote access client terminates the connection.

Extensible Authentication Protocol (EAP)

Extensible Authentication Protocol (EAP) is a Point-to-Point Protocol (PPP) authentication protocol that allows for an arbitrary authentication method. There is a difference in the EAP authentication process compared to other authentication process in that during the authentication phase, EAP does not actually perform authentication. In phase 2, EAP only negotiates the user of a common EAP authentication method. Phase 2 performs the actual authentication in EAP. In this phase, the NAS collects the authentication data and then validates the data against its own user database or a central authentication data server.

EAP is an IETF standard extension to PPP that allows for an arbitrary authentication mechanism for the validation of the PPP connection. EAP provides the highest flexibility to allow for a more secure authentication method. Architecturally, an EAP infrastructure consists of the following three segments:

- **EAP peer computer:** that is attempting to access a network, also known as an access client.
- **EAP authenticator:** An access point or network access server (NAS) that is requiring EAP authentication prior to granting access to a network.
- **Authentication server -** A server computer that negotiates the use of a specific EAP method with an EAP peer, validates the EAP peer's credentials, and authorizes access to the network. Typically, the authentication server is a Remote Authentication Dial-In User Service (RADIUS) server.

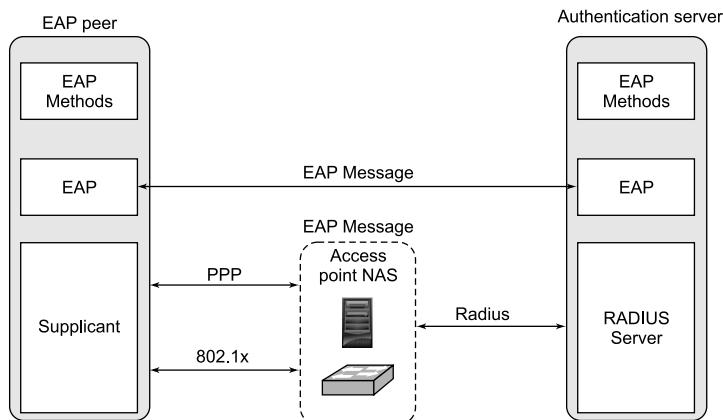


FIGURE 20.25 EAP message exchange.

The RADIUS protocol is used to provide centralized administration of authentication, authorization, and accounting (AAA) and an industry-standard security infrastructure. RADIUS is defined in RFCs 2138 and 2139 in the IETF RFC Database. RADIUS enables administrators to manage a set of authorization policies, accumulate accounting information, and access an account database from a central location. This enables the VPN server to send the authentication credentials to a central authenticating device, and the same user account can be used for both dial-up remote access and VPN-based remote access.

The EAP peer and the EAP authenticator send EAP messages using a supplicant—a software component that uses EAP to authenticate network access—and a data link layer transport protocol such as PPP or IEEE 802.1X. The EAP authenticator and the authentication server send EAP messages using RADIUS. The end result is that EAP messages are exchanged between the EAP components on the EAP peer and the authentication server as in Figure 20.25.

An EAP-MD5 challenge is a required EAP type that includes a challenge and response hand-shake protocol, where the challenges and responses are sent as EAP messages. A typical use of the EAP-MD5 challenge is to authenticate the credentials of remote access clients by using user name and password security systems.

EAP-TLS uses a TLA handshake to authenticate client and server mutually with a certificate. It is a required authentication technique for smart cards and digital certificate-based authentication. With EAP-TLS, a client presents a user certificate to the server, and the server presents a server certificate to the client. The first provides strong user authentication to the server, the second provides assurance that the VPN client has reached a trusted VPN server. Both client and server systems rely on a chain of trusted certificate authorities (CAs) to verify the validity of the offered certificate.

20.6.3 VPN Encryption

To ensure confidentiality of the data in a communication network, it is encrypted by the sender and decrypted by the receiver. The data encryption and decryption are performed by the VPN client or VPN server. Once the Internet connection is established between the client and server, the user creates a VPN connection with the corporate VPN server. If the VPN connection is encrypted, there is no need to use encryption on the dial-up networking connection between the client and the ISP.

The remote access VPN does not provide end-to-end data encryption. End-to-end data encryption can be achieved by using IPSec, which creates a secure connection after the remote access connection has been made. In VPN protocols such as PPP and PPTP, data encryption is available only if MS-CHAP, MS-CHAP-v2, or EAP-TLS are used as the authentication protocols. The encryption and decryption process depends on both the sender and the receiver having knowledge of a common encryption key. The length of the encryption key is negotiated at the time of establishing connection.

VPN uses protocols and some encryption algorithms for the ultimate privacy protections. The three commonly used VPN encryption algorithms are Advanced Encryption Standard (AES), RSA, and Secure Hash Algorithm (SHA).

VPN Tunneling

Tunneling is a process of placing an entire packet within another packet and sending it over a network. Tunneling is a VPN process that enables the encryption of one type of protocol within the datagram of a different protocol. The protocol of the outer packet is understood by the network and the remote ends called tunnel interfaces, where the packet enters and exits the network. After the tunnel is established data can be sent. The tunnel client and server use the tunnel data transfer protocol header for the payload. The client then sends the resulting encapsulated payload across the network, which is to the tunnel server. The tunnel server accepts the packets, removes the tunnel data transfer protocol header, and forwards the payload to the target network.

Types of Tunneling

There are two types of tunneling

- Voluntary tunneling
- Compulsory tunneling

Voluntary tunneling: In the tunneling process a user or a client computer can issue a VPN request to configure and create a voluntary tunnel. The user computer is a tunnel end point and acts as the tunnel client. The voluntary tunneling occurs when a client computer or routing server creates a virtual connection to the target tunnel server. This is accomplished by using tunneling client software and appropriate tunneling protocols installed in the client computer.

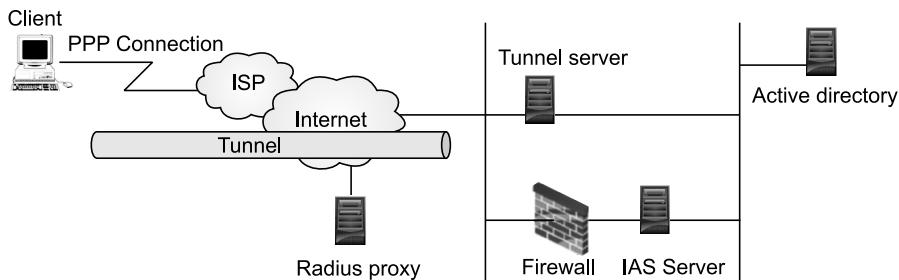


FIGURE 20.26 Voluntary tunnel created by dial-up.

In Figure 20.26, IAS is used for voluntary tunneling. A dial-up client establishes a dial-up connection to an ISP. The dial-up client calls an ISP that is providing Internet access for all the users of an organization. Based on the dial-up connection parameters, the NAS dialed by the dial-up client sends an access request packet to a configured RADIUS proxy computer. The RADIUS proxy server, based on the legitimacy of the user, forwards the access-request packet to the IAS server that is reachable on the Internet through a firewall. The organization IAS server authenticates and authorizes the connection attempt of the dial-up client and sends an access-accept packet to the RADIUS proxy. The RADIUS proxy forwards the access-accept packet to the ISP NAS and the ISP NAS connects the dial-up client to the Internet.

Now the dial-up client initiates a tunnel connection with the organization tunnel server on the Internet. Based on the tunnel connection parameters, the tunnel server sends an access-request packet to the organization IAS server. The organization IAS server authenticates and authorizes the connection attempt of the tunnel client and sends an access-accept packet back to the tunnel server. The tunnel server completes the tunnel creation and the tunnel client can now send packets to the organization intranet through the tunnel.

Compulsory tunneling: In compulsory tunneling, instead of the user a VPN remote access server configures and creates a tunnel. Hence the end point is the remote server not the user. The main difference between the voluntary and compulsory tunneling is that with compulsory tunneling the VPN connection setup is managed by the carrier network provider. Initially the client makes their usual connection through the Internet via their carrier (ISP). This carrier then immediately forges a secure VPN connection between the VPN server and the client.

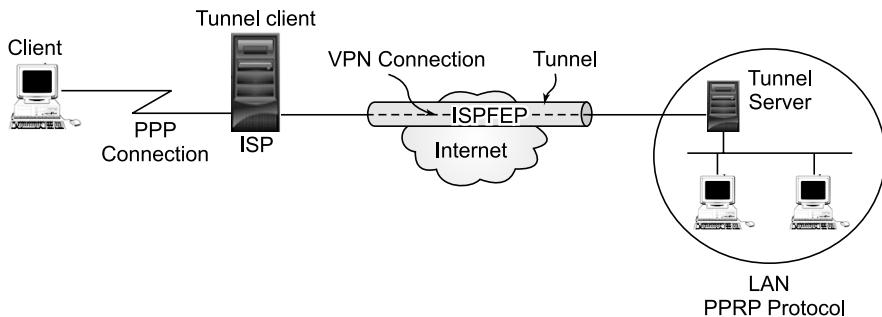


FIGURE 20.27 Compulsory tunneling created by tunnel-enabled NAS.

The computers in the network devices providing the tunnel for the client computer are generally known as a front end processor (FEP) for PPTP and L2TP access concentrator. An organization can contract with an ISP to deploy a set of FEPs. These FEPs can establish tunnels across the Internet to a tunnel server connected to the corporation's private network (LAN), thereby consolidating calls from geographically diverse locations into a single Internet connection at the corporate network. Figure 20.27 shows the client computer placing a dial-up call to a tunneling-enabled NAS at the ISP, in order to authenticate against an IAS server on the other side of the tunnel.

A dial-up client establishes a dial-up connection to an ISP. The dial-up client calls an ISP that is providing tunneled access across the Internet for all the users of an organization. Based on the dial-up connection parameters, the NAS dialed by the dial-up client sends an access-request packet to a configured IAS server. The ISP IAS server authorizes the tunnel connection and sends back an access-accept packet with a series of tunnel attributes. If needed, the IAS NAS creates a tunnel to the organization tunnel server on the Internet.

Typically IAS provides both authentication and authorization. In this case, however, it is common for the ISP IAS server to provide authorization only. Because the dial-in client is performing authentication against the organization tunnel server, authentication against the ISP NAS is not necessary. The ISP NAS then sends a PPP message to the dial-up client to restart the authentication process so that the dial-up user can be authenticated against the organization tunnel server. The dial-up client sends its authentication information to the IAS NAS, which encapsulates it and sends it through the tunnel to the tunnel server. After the authentication credentials are received by the tunnel server, the tunnel server sends an access-request packet to

the organization IAS server. The organization IAS server authenticates and authorizes the connection of the dial-up client to the tunnel server and sends an access-accept packet to the tunnel server. The tunnel server then completes the connection to the dial-up client.

20.7 TUNNELING PROTOCOLS

Point-to-Point Tunneling Protocol (PPTP)

PPTP encapsulates point-to-point protocol frames into IP datagrams for transmission over an IP based network, such as an Internet or intranet. PPTP creates a VPN tunnel via the following two steps:

1. The PPTP client connects to their Internet Service Provider (ISP) using PPP dial-up.
2. PPTP creates a TCP control connection between the VPN clients and VPN server to establish a tunnel. These connections are made using TCP port 1723.

Once the VPN tunnel is established, it supports the flow of control messages for managing and eventually tearing down the VPN connection data packets that pass through the tunnel, to or from the VPN client and layer 2 tunneling protocol (L2TP). PPTP assumes the availability of an IP network between a PPTP client and a PPTP server. The PPTP client might already be attracted to an IP network that can reach the PPTP server, or the PPTP client might have to use a dial-up connection to a NAS to establish IP connectivity as in the case of dial-up Internet users. Authentication during the creation of PPTP based VPN uses authentication mechanisms such as EAP, MS-CHAP, MS-CHAPv2, CHAP, SPAP (Shiva Password Authentication Protocol), and PAP (Password Authentication Protocol).

A typical deployment of PPTP starts with a remote or mobile PPTP client that needs access to a private enterprise LAN by using a local ISP. Clients using computers running Windows NT Server version 4.0 or Windows NT Workstation version 4.0 use Dial-up Networking and the remote access protocol PPP to connect to an ISP. PPTP is a simpler, single tunnel only protocol, and it only works with ISP. Both use PPP as the transport mechanism between the user and the ISP.

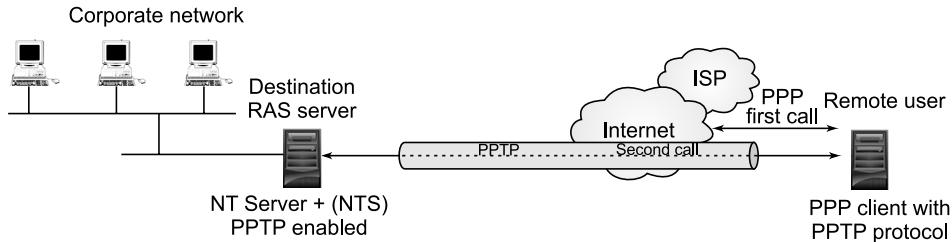


FIGURE 20.28 A PPTP virtual private network.

As in Figure 20.28, two calls are placed. The first call is the call to the ISP which establishes the PPP session. After the client has made the initial PPP connection to the ISP, a second dial-up networking call is made over the existing PPP connection. Data sent using this second connection is in the form of IP datagrams that contain PPP packets, referred to as encapsulated PPP packets. The second call creates the VPN connection to a PPTP server on the private enterprise's LAN; this is referred to as a tunnel. The PPTP tunnel is shown in Figure 20.29.

Tunneling is the process of sending packets to a computer on a private network by routing them over some other network, such as the Internet. The other network routers cannot access the computer that is on the private network. However, tunneling enables the routing network to transmit the packet to an intermediary computer, such as a PPTP server, that is connected to both the routing network and the private network. Both the PPTP client and the PPTP server use tunneling to securely route packets to a computer on the private network by using routers that only know the address of the private network intermediary server.

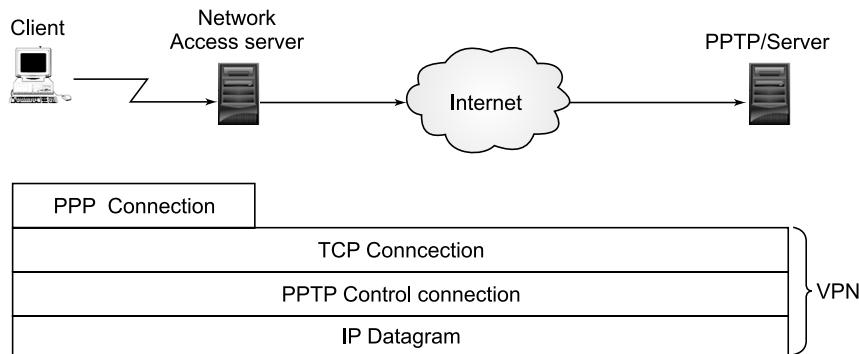


FIGURE 20.29 The PPTP tunnel.

When the PPTP server receives the packet from the routing network, it sends it across the private network to the destination computer. The PPTP server does this by processing the PPTP packet to obtain the private network computer name or address information in the encapsulated PPP packet. Note that the encapsulated PPP packet can contain multi-protocol data such as TCP/IP, IPX, or Net BEUI protocols. Because the PPTP server is configured to communicate across the private network by using private network protocols, it is able to read multi-protocol packets. The PPTP packet has a huge amount of overhead.

Figure 20.30 illustrates the multi-protocol support built into PPTP. A packet sent from the PPTP client to the PPTP server passes through the PPTP tunnel to a destination computer on the corporate network.

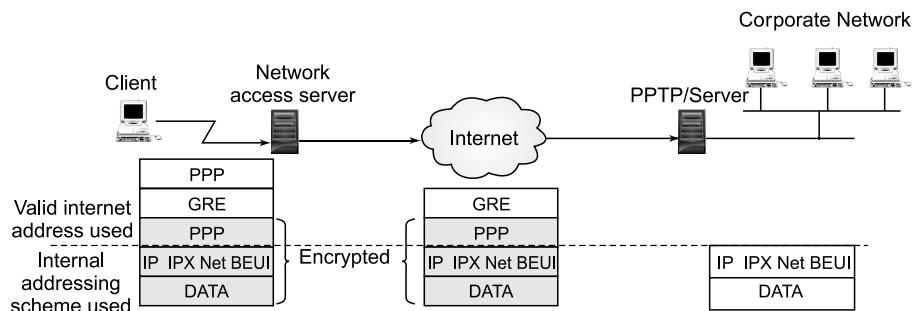


FIGURE 20.30 Connecting a dial-up networking PPTP client to the corporate network.

PPTP encapsulates the encrypted and compressed PPP packets into IP datagrams for transmission over the Internet. These IP datagrams are routed over the Internet until they reach the PPTP server that is connected to the Internet and the private network. The PPTP server disassembles the IP datagram into a PPP packet and then decrypts the PPP packet using the network protocol of the private network. These network protocols on the corporate network that are supported by PPTP are IPX, NetBEUI, or TCP/IP.

The GRE provides services similar to an IP connection: ACK, flow control, and call ID. And like IP, there is no error correction (it is connectionless). Packets are discarded if errors are encountered.

PAP vs. CHAP - with PAP the password is transmitted as-is across the network. With CHAP, the password is encrypted (40- or 128-bit key) before being transmitted.

Problem with PPTP – encryption is only possible in a Microsoft environment – not Apple or UNIX.

20.7.1 PPTP Architecture

PPTP is designed to provide a secure method for reaching private networks over the Internet. Examining the PPTP reveals the secure design features of the PPTP protocol. The architecture of PPTP is applicable to the Windows NT Server version 4.0 or Windows NT Workstation version 4.0 as discussed here.

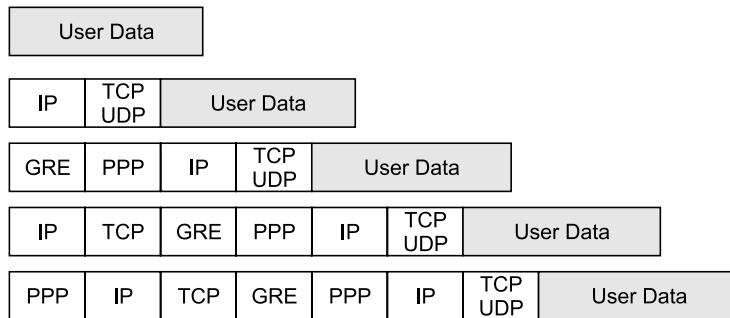


FIGURE 20.31 PPTP packet contraction.

PPTP Architecture Overview

The secure communication created using the PPTP protocol typically involves three processes. Each of these processes requires the successful completion of the previous process.

- (i) PPP protocol - PPP connection and communication
- (ii) PPTP control connection
- (iii) PPTP data tunneling

PPP protocol – PPP Connection and Communication

A PPTP client uses PPP to connect to an ISP by using a standard telephone line. This connection uses the PPP protocol to establish the connection and encrypt data packets. PPP is a remote access protocol used by PPTP to send multi-protocol data across TCP/IP-based networks. PPP encapsulates IP, IPX, and NetBEUI packets between PPP frames and sends the encapsulated packets by creating a point-to-point link between the sending and receiving computers. Most PPTP sessions are started by a client dialing up an ISP network access server (NAS). The PPP protocol is used to create the

dial-up connection between the client and network access server and performs the following three functions:

- Establishes and ends the physical connection—the PPP protocol uses a sequence defined in RFC 1661 to establish and maintain connections between remote computers.
- Authenticates users—PPTP clients are authenticated by using the PPP protocol. Cleartext, encrypted, or Microsoft encrypted authentication can be used by the PPP protocol.
- Creates PPP datagrams that contain encrypted IPX, NetBEUI, or TCP/IP packets—PPP creates datagrams that contain one or more encrypted TCP/IP, IPX, or NetBEUI data packets. Because the network packets are encrypted, all traffic between a PPP client and a network access server is secure.

This entire process is illustrated in Figure 20.32.

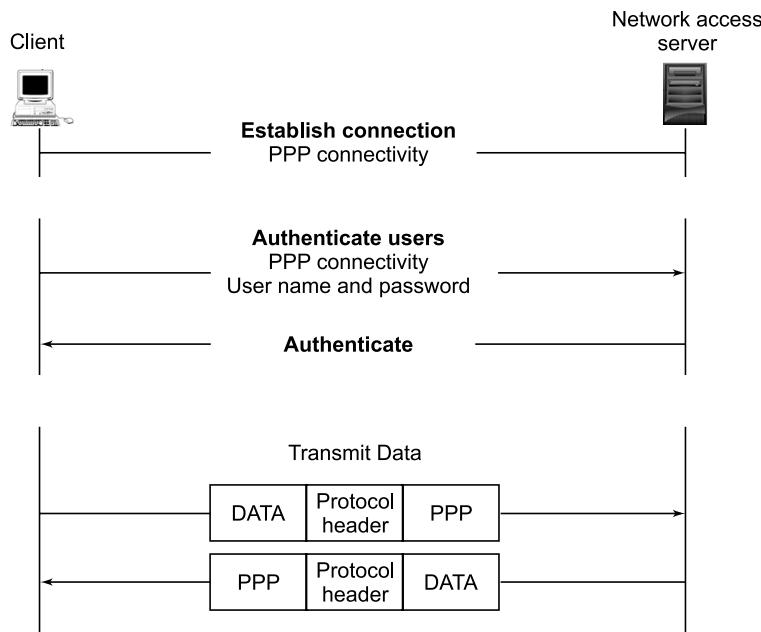


FIGURE 20.32 Dial-up networking PPP connection to ISP.

PPTP Control Connection: Control messages are transmitted in control packets in a TCP datagram. The TCP connection is created between the PPTP client and the PPTP server. This connection is used to exchange

control messages. The control messages are sent in TCP datagrams containing the control messages. A datagram contains a PPP header, a TCP header, a PPTP control message, and appropriate trailers, as in Figure 20.33. PTP control connection packets consist of a data link layer header and trailer, an IP header, a TCP header, and a PPTP control message.

Using the connection to the Internet established by the PPP protocol, the PPTP protocol creates a control connection from the PPTP client to a PPTP server on the Internet. This connection uses TCP to establish the connection and is called a PPTP tunnel. The PPTP protocol specifies a series of control messages sent between the PPTP enabled client and the PPTP server. The control messages establish, maintain, and end the PPTP tunnel. The following list as in Table 20.2 presents the primary control messages used to establish and maintain the PPTP tunnel.

Date link Header	IP	TCP	PPTP Control message	Date Link Trailer
------------------	----	-----	----------------------	-------------------

FIGURE 20.33 PPTP control connection packet.

The PPTP control connection carries the PPTP call control and management messages that are used to maintain the PPTP tunnel. This includes the transmission of periodic PPTP Echo_request and PPTP Echo_reply messages to detect a connectivity failure between the PPTP client and PPTP server.

TABLE 20.2 PPTP Control Message Types.

Message Types	Purpose
PPTP_START_SESSION_REQUEST	Starts Session
PPTP_START_SESSION_REPLY	Replies to start session request
PPTP_ECHO_REQUEST	Maintains session
PPTP_ECHO_REPLY	Replies to maintain session request
PPTP_WAN_ERROR_NOTIFY	Reports an error on the PPP connection
PPTP_SET_LINK_INFO	Configures the connection between client and PPTP Server
PPTP_STOP_SESSION_REQUEST	Ends session
PPTP_STOP_SESSION_REPLY	Replies to end session request

Control messages are transmitted in control packets in a TCP datagram. One TCP connection is created between the PPTP client and the PPTP server. This connection is used to exchange control messages. The control messages are sent in TCP datagrams containing the control messages. A datagram contains a PPP header, a TCP header, a PPTP control message, and appropriate trailers as in Figure 20.34.

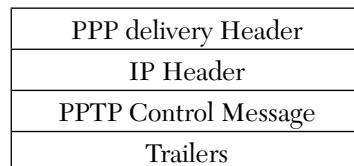


FIGURE 20.34 PPTP TCP datagram with control messages.

The exchange of messages between the PPTP client and the PPTP server over the TCP connection is used to create and maintain a PPTP tunnel. This entire process is illustrated in Figure 20.35.

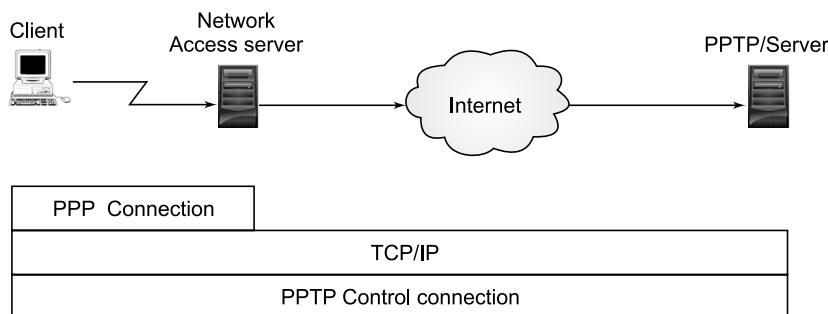


FIGURE 20.35 PPTP control connection to PPTP server over PPP connection to ISP.

Note that in this illustration, the control connection is for the scenario in which the remote access client is the PPTP client. In this scenario the remote access client is not PPTP enabled and uses a PPTP enabled ISP network access server; the PPTP control connection begins at the ISP server. Finally, the PPTP protocol creates IP datagrams containing encrypted PPP packets that are then sent through the PPTP tunnel to the PPTP server. The PPTP server disassembles the IP datagrams, decrypts the PPP packets, and then routes the decrypted packets to the private network.

PPTP Data Tunneling

PPTP data tunneling is performed through multiple levels of encapsulation. The structure of the tunneled PPTP data is shown in Figure 20.36. The initial PPP payload is encrypted and encapsulated with a PPP header to create a PPP frame. The PPP frame is then encapsulated with a modified GRE header, which is designed to provide a simple, general purpose mechanism for encapsulating data sent over IP networks.

Date link Header	IP Header	GRE Header	PPP header	Encryption Payload (IP diagram)	Date Link Trailer
------------------	-----------	------------	------------	------------------------------------	-------------------

FIGURE 20.36 Tunneled PPTP data.

20.7.2 Encapsulation of Data Link Layer

To transmit the data over LAN or WAN networks, the IP datagram is encapsulated with a header and trailer for the data link layer of the outgoing physical interface. For example, if the IP datagrams are sent over an Ethernet interface, the IP datagrams are encapsulated with the Ethernet header and trailer. If the IP datagrams are sent over a WAN link, the IP datagram is encapsulated with a PPP header and trailer. After the PPTP tunnel is established, user data is transmitted between the client and PPTP server. Data is transmitted in IP datagrams containing PPP packets. The IP datagrams are created using a modified version of the Internet Generic Routing Encapsulation (GRE) protocol. (GRE is defined in RFCs 1701 and 1702.) The IP datagram created by PPTP is shown in Figure 20.37.

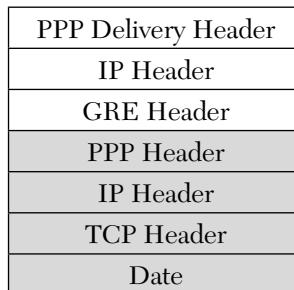


FIGURE 20.37 IP datagram containing encrypted PPP packet as created by PPTP.

The IP delivery header provides the information necessary for the datagram to traverse the Internet. The GRE header is used to encapsulate the PPP packet within the IP datagram. The PPP packet was created by RAS. Note that the PPP packet is just one unintelligible block because it is encrypted. Even if the IP datagram were intercepted, it would be nearly impossible to decrypt the data.

Understanding PPTP Security

PPTP extends the strict authentication and encryption security available to computers running RAS under the Windows NT Server version 4.0 and Windows NT Workstation version 4.0 to PPTP clients on the Internet. PPTP also can protect the PPTP server and private network by ignoring all but PPTP traffic. Despite the strict security, it is very simple to use PPTP with existing firewalls. This section will help you understand security features of PPTP.

- **Authentication and access control:** Initial dial-in authentication may be required by an ISP network access server. If this authentication is required, it is strictly to log on to the ISP network access server; it is not related to Windows NT-based authentication. Authentication of remote PPTP clients is done by using the same PPP authentication methods used for any RAS client dialing directly to a RAS server. Microsoft's implementation of the Remote Access Service (RAS) supports the Challenge Handshake Authentication Protocol (CHAP), the Microsoft Challenge Handshake Authentication Protocol (MS-CHAP), and the Password Authentication Protocol (PAP) authentication schemes.

After authentication, all access to a private LAN continues to use the Windows NT-based security model. Access to resources on NTFS drives, or to other network resources, requires the proper permissions. It is recommended that the NTFS file system be used for file resources that are accessed by PPTP clients.

- **Data encryption:** For data encryption, PPTP uses the RAS “shared-secret” encryption process. It is referred to as a shared-secret because both ends of the connection share the encryption key. Under the Microsoft implementation of RAS, the shared secret is the user password. (Other encryption methods are based on public key encryption).

The user name and password of the PPTP client are available to the PPTP server and supplied by the PPTP client. An encryption key is derived from the hashed password stored on both the client and

server. The RSA RC4 standard is used to create this 40-bit session key based on the client password. This key is used to encrypt all data that is passed over the Internet, keeping the remote connection private and secure.

- **PPTP packet filtering:** Network security from malicious activity can be enhanced by enabling PPTP filtering on the PPTP server. When PPTP filtering is enabled, the PPTP server on the private network accepts and routes only PPTP packets from authenticated users. This prevents all other packets from entering the PPTP server and private network. In conjunction with PPP encryption, this ensures that only authorized encrypted data enters or leaves the private LAN.
- **Using third-party firewalls:** PPTP traffic uses TCP port 1723, and IP protocol uses ID 47, as assigned by the Internet Assigned Numbers Authority (IANA). PPTP can be used with most firewalls and routers by enabling traffic destined for port 1723 to be routed through the firewall or router.

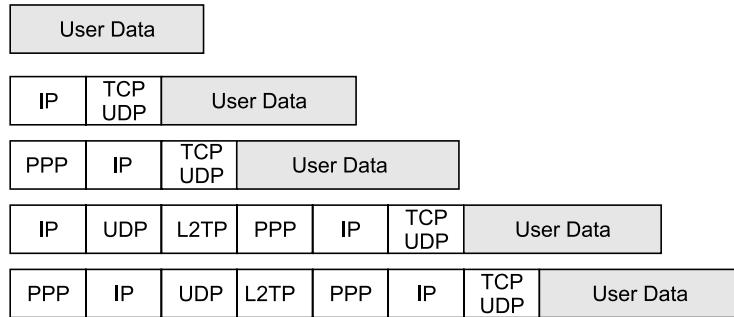
Firewalls ensure corporate network security by strictly regulating data that comes into the private network from the Internet. An organization can deploy a PPTP server running Windows NT Server version 4.0 behind its firewall. The PPTP server accepts PPTP packets passed to the private network from the firewall and extracts the PPP packet from the IP datagram, decrypts the packet, and forwards the packet to the computer on the private network.

Layer 2 Tunneling Protocol (L2TP)

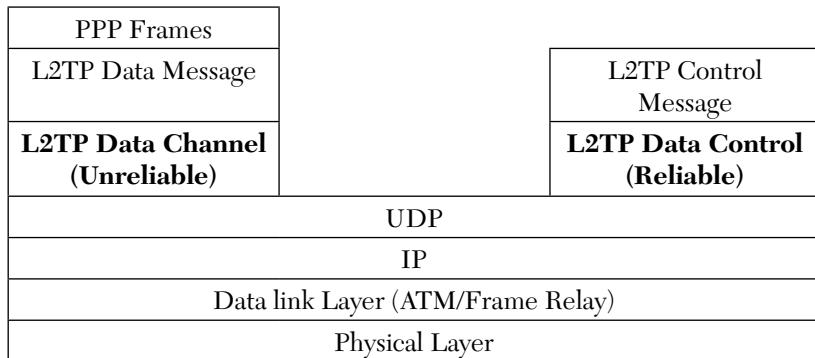
L2TP is a combination of PPTP and Layer 2 Forwarding (L2F) mechanisms. It is described in RFC 2661 in the IETF RFC Database. L2TP encapsulates PPP frames that are sent over the IP network, X.25, frame relay, or ATM network. When sent over an IP network, L2TP frames are encapsulated as user datagram protocol (UDP) messages. L2TP can be used as a tunneling protocol over the Internet or over a private intranet. The L2TP frames include the following:

- L2TP connection maintenance messages that include the L2TP header.
- L2TP tunneled data that include a PPP header and a PPP payload.

Here encryption is provided through the use of the Internet Protocol security (IPSec) Encapsulating Security Payload (ESP) header and trailer.

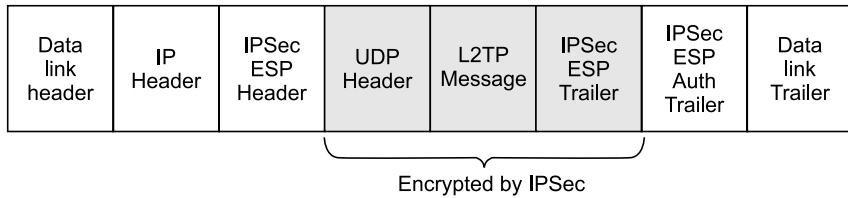
**FIGURE 20.38** L2TP packet construction.

L2TP Protocols use 2 stacks—both are sending normally unreliable UDP packets, but the control messages are reliable due to extra overhead placed there by L2TP, and the data messages are left as standard UDP and are unreliable.

**FIGURE 20.39** L2TP protocol stack.

L2TP Connection Maintenance Messages

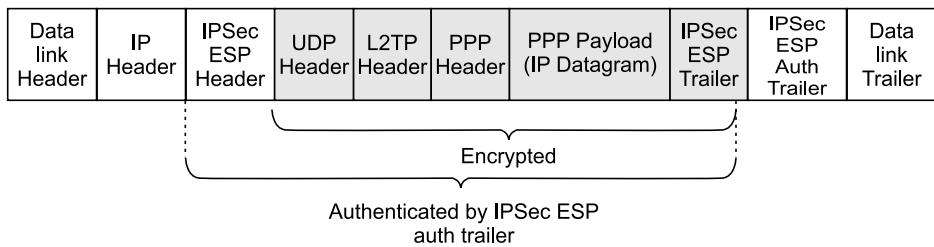
The L2TP tunnel maintenance is not performed over a separate TCP connection. L2TP call control and message management traffic is sent as a UDP message between the L2TP client and the L2TP server. This control message is sent as the encrypted payload of the IPSec ESP transport mode as shown in Figure 20.40.

**FIGURE 20.40** L2TP control message encrypted by IPsec.

The L2TP control message also uses sequencing to ensure delivery of the message. Within the L2TP control message, the Next-Received field (similar to the TCP Acknowledgment field) and the Next-Sent field (similar to the TCP Sequence Number field) are used to maintain the sequence of control messages. Out-of-sequence packets are dropped. The Next-Sent and Next-Received fields can also be used for sequenced delivery and flow control for tunneled data. L2TP supports multiple calls for each tunnel. In the L2TP control message the L2TP header for tunneled data has a Tunnel ID that identifies the tunnel and a Call ID that identifies a call within the tunnel.

20.7.3 L2TP Data Tunneling

L2TP data tunneling is performed using multiple levels of encapsulation. Figure 20.41 shows the resulting structure of tunneled L2TP over IPsec data.

**FIGURE 20.41** L2TP packet encapsulation.

L2TP encapsulation: The initial PPP payload is encapsulated with a PPP header and an L2TP header.

UDP encapsulation: The encapsulated L2TP packet is then encapsulated with a UDP header with the source and destination ports set to 1701.

IP encapsulation: The UDP message is encrypted and encapsulated with an IPSec ESP header and trailer and an ESP Authentication (Auth) trailer.

Encapsulation of the Data-Link Layer

To forward the message on a LAN or WAN link, the IP datagram is finally encapsulated with a header and trailer for the data-link layer technology of the outgoing physical interface. If it is an Ethernet interface, then the IP datagram is encapsulated with an Ethernet header and trailer. In a point-to-point WAN link, the IP datagram is encapsulated with a PPP header and trailer.

L2TP with Internet Protocol Security (L2TP/IPSec)

Tunneling protocols such as PPTP and L2TP are implemented at the data-link layer of the ISO-OSI reference model and provide data security by helping to create secure tunnels. In contrast, the IPSec protocol is implemented at the network layer and helps secure data at the packet level. IPSec provides the two security protocols Authentication Header (AH) and ESP.

SUMMARY

- The basic function of a firewall is to secure the network traffic for the purpose of unauthorized access.
- A virtual private network provides virtual network links based on encryption and isolating traffic at the packet level during data transmission.
- The functions of firewalls are as packet filtering routers, stateful inspection firewalls, Bastion hosts, NAT circuit level gateways, application level proxies, and as proxy firewalls.
- Packet filter firewalls deny access to traffic that does not meet a set of rules and pass traffic that does.
- Packet filtering can be divided into two parts: stateless packet filtering and stateful packet filtering.
- If the information about the passing packets is not remembered by the firewall, then this type of filtering is called stateless packet filtering.
- If the firewall remembers the information about the previously passed packets, then that type of filtering is stateful packet filtering.
- A firewall understands certain specific application protocols. An application-level firewall uses application-specific proxies.

- A proxy filter (server) sits between the client and the destination working as a middleman between the two communication parties.
- A reverse proxy firewall functions in the same way as a proxy firewall, with the exception that they are used to protect the server and not the clients.
- A network address translator (NAT) hides internal addresses from the outside world. A network address translation router contains a table of outside and inside addresses.
- Port address translation (PAT) is a modified form of dynamic NAT that maps multiple private IP addresses to single public IP address (many-to-one) by using different ports.
- VPNs are generally divided into two basic categories: remote access VPNs and site-to-site VPNs.
- Remote access VPNs are usually used to link private networks from various remote locations.
- Site-to-site VPN is used to connect a branch office network to another organization over the Internet Security Protocol (IPSec) and tunneling.
- VPN tunneling is a process of placing an entire packet within another packet and sending it over a network.
- There are two types of tunneling: voluntary tunneling and compulsory tunneling.
- PPTP encapsulates point-to-point protocol frames into IP datagrams for transmission over an IP based network such as an Internet and intranet.
- Layer 2 Tunneling Protocol (L2TP) is a combination of a PPTP and Layer 2 Forwarding (L2F) mechanism.

REVIEW QUESTIONS

1. How does a screened host architecture for a firewall differ from a screened subnet firewall architecture? Which offers more security for the information assets that remain on the trusted network? Explain with a neat sketch.
2. What are the approaches of implementing a firewall?
3. How can a firewall be categorized based on its processing mode?
4. With neat diagrams highlight the differences between a screened host firewall single homed bastion and a screened host firewall dual homed bastion.

5. Discuss the different types of firewall systems.
6. Describe how various types of firewalls interact with the network traffic at various levels of the OSI model.
7. Describe firewall technology and the various approaches to firewall implementation.
8. List the five generations of firewall technology. Which generation is still in common use?
9. Describe how the various types of firewalls interact with the network traffic at various levels of the OSI model.
10. Define firewall. What are its different types? Explain the working of each in detail.

MULTIPLE CHOICE QUESTIONS

1. A network layer firewall has two sub-categories which are:
 - (a) stateful firewall and stateless firewall
 - (b) bit oriented firewall and byte oriented firewall
 - (c) frame firewall and packet firewall
 - (d) none of the mentioned
2. A _____ hides internal addresses from the outside world.
 - (a) NAT
 - (b) PPTP
 - (c) CHAP
 - (d) none of the above
3. Which one is not a VPN authentication protocol?
 - (a) PAP
 - (b) CHAP
 - (c) SPAP
 - (d) ALG
4. The computer network devices providing the tunnel for the client computer is generally known as a _____ for the PPTP and L2TP access concentrator.
 - (a) firewall
 - (b) network address translation
 - (c) front end processor
 - (d) none of the above

APPENDIX



ANSWERS FOR MULTIPLE CHOICE QUESTIONS

Chapter 1

- 1.** (c) **2.** (d) **3.** (a) **4.** (b) **5.** (d) **6.** (a) **7.** (d) **8.** (c)
9. (c) **10.** (b) **11.** (b) **12.** (a) **13.** (c) **14.** (d) **15.** (d) **16.** (b)

Chapter 2

- 1.** (a) **2.** (b) **3.** (a) **4.** (b) **5.** (a) **6.** (b) **7.** (b) **8.** (a)
9. (b) **10.** (c)

Chapter 3

- 1.** (a) **2.** (a) **3.** (c) **4.** (d) **5.** (a) **6.** (b) **7.** (a) **8.** (c)
9. (c) **10.** (b) **11.** (b)

Chapter 4

- 1.** (a) **2.** (a) **3.** (b) **4.** (a) **5.** (d) **6.** (b) **7.** (c) **8.** (a)
9. (a) **10.** (c)

Chapter 5

- 1.** (a) **2.** (b) **3.** (a) **4.** (d) **5.** (a) **6.** (a) **7.** (c) **8.** (d)
9. (c) **10.** (b)

Chapter 6

- 1.** (c) **2.** (a) **3.** (b) **4.** (b) **5.** (d) **6.** (d) **7.** (c) **8.** (b)
9. (b) **10.** (c) **11.** (c)

Chapter 7

- 1.** (b) **2.** (a) **3.** (c) **4.** (b) **5.** (a) **6.** (a) **7.** (b) **8.** (b)
9. (d) **10.** (a)

Chapter 8

- 1.** (b) **2.** (b) **3.** (b) **4.** (c) **5.** (d) **6.** (d) **7.** (a) **8.** (a)
9. (b) **10.** (d) **11.** (a) **12.** (a)

Chapter 9

- 1.** (c) **2.** (c) **3.** (a) **4.** (c) **5.** (a) **6.** (c) **7.** (c) **8.** (a)
9. (b) **10.** (b)

Chapter 10

- 1.** (a) **2.** (b) **3.** (a) **4.** (b) **5.** (b) **6.** (a) **7.** (c) **8.** (a)
9. (a) **10.** (a)

Chapter 11

- 1.** (a) **2.** (b) **3.** (c) **4.** (c) **5.** (a) **6.** (c) **7.** (a) **8.** (c)
9. (a) **10.** (a) **11.** (c)

Chapter 12

- 1.** (b) **2.** (b) **3.** (c) **4.** (a) **5.** (c) **6.** (b) **7.** (a) **8.** (c)
9. (a) **10.** (c)

Chapter 13

- 1.** (d) **2.** (a) **3.** (b) **4.** (d) **5.** (a) **6.** (d) **7.** (a) **8.** (b)
9. (a) **10.** (c) **11.** (a) **12.** (c) **13.** (a) **14.** (b) **15.** (c)

Chapter 14

- 1.** (d) **2.** (c) **3.** (a) **4.** (d) **5.** (a) **6.** (b) **7.** (c) **8.** (d)
9. (b) **10.** (b) **11.** (d) **12.** (d) **13.** (c) **14.** (a)

Chapter 15

- 1.** (b) **2.** (a) **3.** (b) **4.** (b) **5.** (d) **6.** (c) **7.** (b) **8.** (a)
9. (a) **10.** (a)

Chapter 16

- 1.** (c) **2.** (a) **3.** (a) **4.** (a) **5.** (c) **6.** (a) **7.** (a) **8.** (a)
9. (c) **10.** (a) **11.** (d) **12.** (a) **13.** (b) **14.** (a) **15.** (a) **16.** (b)
17. (a)

Chapter 17

- 1.** (b) **2.** (a) **3.** (b) **4.** (c) **5.** (a) **6.** (c) **7.** (a) **8.** (a)
9. (b) **10.** (a) **11.** (c)

Chapter 18

- 1.** (b) **2.** (a) **3.** (c) **4.** (b) **5.** (a) **6.** (b) **7.** (d) **8.** (d)
9. (c) **10.** (b) **11.** (b)

Chapter 19

- 1.** (a) **2.** (c) **3.** (a) **4.** (d) **5.** (a) **6.** (c) **7.** (b) **8.** (d)
9. (a) **10.** (a) **11.** (c)

Chapter 20

- 1.** (a) **2.** (a) **3.** (d) **4.** (c) **5.** (c) **6.** (b) **7.** (d) **8.** (a)
9. (b) **10.** (a)

REFERENCES

- Anderson, J. *Computer Security Threat Monitoring and Surveillance*, Fort Washington, PA: James P. Anderson Co., April 1980.
- Anderson, Rose. *Security Engineering*, Wiley, 2001.
- Black, Uyless. *Internet Security Protocols*, Pearson Education Asia, 2000.
- Bellare, M., & Kohno, T. *Hash Function Balance and Its Impact on Birthday Attacks*, Eurocrypt 2004, 401–418.
- Canetti, R., Goldreich, O., & Halevi, S. *The Random Oracle Methodology, Revisited*, ACM Sym. on Theory of Computation (STOC), 1998, <http://eprint.iacr.org/1998/011.pdf>.
- Chapman D. Brent & Zwicky, E. *Building Internet Firewall*. O'Reilly, Sebastopol, CA, 1995.
- Cheng, P.-C., Garay, J. A., Herzberg, A., & Krawczyk, H. *A Security Architecture for Internet Protocols*, IBM System Journal, November 1998.
- Comer, D. *Computer Networks Internets*, Prentice Hall, 2000.
- Comer, D. *Internetworking with TCP/IP*, Volume I, Prentice Hall, India, 1999.
- Comer, D. *The Internet Book*, Prentice Hall, India, 1999.
- Davis, C. *IPSec Securing VPNs*, Tata McGraw-Hill, 2001.
- Denning, P. *Computer Under Attack: Intruders, Worms and Viruses*, Addison – Wesley, Reading, MA, 1990.
- Easttom, C. *Computer Security Fundamentals*, Pearson, 2012.
- Florian, M., Pramstaller, N., Rechberger, C., & Rijmen, V. *On the Collision Resistance of RIPEMD-160*, Springer-Verlag, Berlin /Heidelberg, 2006.

- Forouzan, Behrouz A. *Cryptography and Network Security*, Tata McGraw Hill Education Private Limited, India, 2012.
- Forouzan, B. *Data Communication and Networking*, Tata McGraw-Hill, 2002.
- Forouzan, B. *TCP/IP*, Tata McGraw-Hill, 2002.
- Goldwasser, S., & Bellare, M. *Lecture Notes on Cryptography*, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2008.
- Hoffman, L. (ed.). *Rogue Programs: Viruses, Worms and Trojan Horses*, Van Nostrand Reinhold, New York, 1990.
- Information Security Handbook for Network Beginners*. National Center of Incident Readiness and Strategy for Cybersecurity (NISC), Government of Japan.
- Kaufman, C., Perlman, R., & Speciner, M. *Network Security*, Pearson Education Asia, 2002.
- Kaufman, C., Perlman R., & Speciner, M. *Network Security*, Prentice Hall, Upper Saddle River, NJ, 2001.
- Kohel, David R. *Cryptography*, <http://creativecommons.org>, 2015.
- Krawetz, N. *Introduction to Network Security*, Charles River Media, an imprint of Thomson Learning, 2007.
- Mohamed, B., Eder, C., & Hanke, T. *An Introduction to Cryptography*, 2018.
- Nanavati, S., Thieme, M., & Nanavati, R. *Biometrics*, Pearson Education Asia, 2002.
- Nash, A., Duane, B., Brink, D., & Joseph, C. *PKI – Implementing and Managing E-Security*, McGraw-Hill, 2000.
- Parmar, S. K. *Information Resource Guide: Computer, Internet, and Network Systems Security: An Introduction to Security*.
- Peng, S. *The Pigeonhole Principle*, Duke University, September 1, 2009.
- Pieprzyk, J., Hardjono, T., & Seberry J. *Fundamentals of Computer Security*, Springer, Berlin, 2003.
- Pistoia, M., Reller, D. F., Gupta, D., Nagnur, M., & Ramani, A. K. *Java 2 Network Security*, Pearson Education, Asia, 2001.
- Rhee, M. *Internet Security*, Wiley & Sons, Hoboken, NJ, 2003.

- Robling Denning, D. E. *Cryptography and Data Security*, Addison-Wesley, 1982.
- Robshaw, M. J. B. *On Recent Results for MD2, MD4 and MD5*, RSA Laboratories, 1996.
- Rivest, R. L. *The MD6 Hash Function*, <http://www.csrc.nist.gov/pki/HashWorkshop/index.html>
- Schneier, B. *Applied Cryptography*, Wiley and Sons, 2001.
- Scott, C., Wolfe, P., & Erwin, M. *Virtual Private Network*, O'Reilly, 2000.
- Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, 2000.
- Smith, R. *Internet Cryptography*, Pearson Education Asia, 1999.
- Stalling, W. *Network Security Essentials*, Pearson Education Asia, 2002.
- Stalling, W. *Cryptography and Network Security*, Pearson Education Asia, 2000.
- Tanenbaum, A. *Computer Networks*, Prentice Hall, India, 1995.
- van Tilborg, Henk C. A. *Fundamentals of Cryptography*, Kluwer Academic.
- Vaudenay, S. *A Classical Introduction to Cryptography*, Springer, New York, 2006.
- Whitman, M. E., & Mattord, H. J. *Principles of Information Security*, Cengage Learning.
- Zwickly, E. D., Cooper, S., & Chapman, D. B. *Building Internet Firewalls*, O'Reilly, 2000.

INDEX

A

Abelian group, 210
Access control, 13, 14
Access Control List (ACL), 414
Active attack, 8
Adding salt, 303
Additive cipher, 36
Advanced Encryption Standard (AES), 151
Adware, 550
AES decryption, 176
AES encryption, 156
AES key expansion, 164
AES structure, 155
Affine cipher, 39
Algebraic attack, 104
Anomaly based IDS, 558
Application layer attack, 411
Application layer security, 421
Application level gateways (proxy), 585
ARP spoofing attack, 514
Asymmetric key encryption, 21, 189
Attacks on DNS, 370
Attacks on RSA, 198
Attacks using ICMP messages, 377
Authentication, 12
Authentication exchange, 14
Authentication Header (AH), 484
Authentication Header (AH) for IPv4 and IPv6, 488
Authentication Server (AS), 344

Autokey cipher, 42
Availability, 3, 5

B

Bigram frequency, 23
Biometric security systems, 316
Biometric solutions, 319
Birthday attack, 236
Blind attack, 395
Block cipher, 21, 68, 83, 85

C

Cryptography, 4, 21

D

Data encryption standard (DES), 109
Data integrity and system security, 4
Data link layer security, 513
Data representation, 152
D-Boxes: (Diffusion Box), 88
DDoS attack trojan horse virus, 548
Decryption algorithm, 20
Denial of service (DoS), 5, 8
DES cryptanalysis, 125
DES function, 111
DHCP attacks, 519
DHCP spoofing, 519
DHCP starvation, 520
Dictionary attack, 11, 297

- Differential cryptanalysis, 103
 Diffie-Hellman key exchange, 203
 Diffie-Hellman problem (DHP), 205
 Diffusion, 96
 Digital certificate, 334
 Digital envelop, 281
 Digital fingerprint, 278
 Digital signature, 14, 277, 278
 Digital Signature Algorithm (DSA), 284
 Digital Signature Standard (DSS), 284
 Discretionary Access Control (DAC), 14
 Distributed attack, 9
 DNS cache poisoning, 370
 DNS caching, 368
 DNS resolution, 368
 DNS spoofing, 373
 Domain Name System (DNS), 368
 DoS and DDoS, 397–399
 Dynamic NAT, 591
- E**
- Eavesdropping, 388
 Electronic codebook mode (ECB), 69
 Elementary attack, 199
 ElGamal encryption using elliptic curve cryptography, 206, 209, 219
 ElGamal public key cryptosystem, 206
 ElGamal signature scheme, 288
 Elliptical Curve Cryptosystem (ECC), 209
 Elliptical Curve Digital Signature Algorithm (ECDSA), 290
 Elliptic curve, 209
 Elliptic Curve Discreet Logarithm Problem (ECDLP), 218
 E-mail infrastructure, 422
 E-mail protocols, 423
 E-mail security, 422
 E-mail security mechanisms, 431
 E-mail security service, 422, 426
 E-mail worms, 544
 Encapsulating Security Payload (ESP) for IPv4 and IPv6, 492
- Encapsulation security payload, 484
 Encryption, 14, 19
 Encryption algorithm, 20
 End-to-end encryption, 27
 Entity authentication, 295
 ESP header, 492
 Expansion D-boxes, 90
 Exploitation of weak key, 104
 Exploit attack, 11
 Extensible Authentication Protocol (EAP), 602
- F**
- Feistel cipher, 99, 100, 102
 Feistel structure, 111
 File infection virus, 537
 File serving trojan horse virus, 547
 Firewalls, 565, 575
 Fractorization attack, 199
 FTP bounce, 412
- G**
- Galois field multiplication, 163
- H**
- Hand-held passcode generator, 313
 Hash collision, 305
 Hashed MAC (HMAC), 245
 Hash function, 230, 231, 232, 251
 Hastad's broadcast attack, 199
 Hijack attack, 10
 Hill cipher, 48
 Honeypots, 564
 Host based IPS, 563
 Host IDS, 556
 HTTPS, 465
 HTTPS certificate, 466
 Hybrid attack, 11
- I**
- IDEA, 129
 IMAP, 425

Improved Davie's attack, 126
 Insider attack 9
 Instant message worms, 545
 Integrity, 4, 12
 Internet, 359
 Internet control message protocol (ICMP), 374
 Internet Key Exchange (IKE), 501, 505
 Internet Relay Chat (IRC) worms, 545
 Internet worms, 545
 Intranet based VPN, 599
 Intrusion Detection System (IDS), 554
 Intrusion Prevention System (IPS), 562
 IPSec, 483
 IPSec core protocol, 483
 IPSec Encapsulating Security Payload (ESP) Format, 492
 ISAKMP, 501, 506
 ISO-OSI Model, 362

K

Kerberos, 342
 Kerckhoff's principle, 33
 Key distribution, 499
 Key Distribution Center (KDC), 343
 Keyed transposition cipher, 61
 Keyless cipher, 88
 Keyless transposition cipher, 58
 Key logger attack, 297
 Keyloggers and system file killers, 548
 Key management, 499
 Key management for IPSec, 499, 501
 Known plain test attack (KPA), 24

L

Layer 2 tunneling protocol (L2TP), 616
 Linear cryptanalysis, 104, 126
 Linear feedback shift register, 65
 Link encryption, 26
 Location of encryption devices, 26
 Lookup table, 302

M

MAC flooding, 515
 Macro virus, 539
 Malicious softwares, 551
 Mandatory access control, 14
 Man-in-the-middle attack (MITM), 393, 394
 Masquerading, 6
 Mass mailer virus, 538
 MD5 algorithm, 258
 Merkle-Damgård strengthening, 252
 Message authentication, 238
 Message Authentication Code (MAC), 241
 Message digest, 231, 254
 Message integrity, 230
 MIME, 440, 441
 Modification, 6
 Modification Detection Code (MDC), 241
 Modified Needham-Schroeder public key identification protocol, 314
 Monoalphabetic cipher, 35
 Monogram frequency, 23
 MS-CHAP, 601
 Multiplicative cipher, 37

N

Need to know, 4
 Nested MAC (NMAC), 244
 Network Address Translation (NAT), 589
 Network attacks and security threats, 383
 Network based IDS, 556
 Network based IPS, 564
 Network communication architecture and protocols, 361
 Network layer security, 479
 Network security model (NSM), 412
 Network security protocols, 359
 Non-repudiation, 12
 Notarization, 14

O

Oakley, 506
 One time password, 307
 Originality integrity, 4
 OSI-layer functions, 363
 Output Feedback (OFB), 75

P

Packet filtering, 578
 Parallel MAC (PMAC), 244
 Passive attack, 8
 Password Attack, 11, 297
 Password authentication, 296
 Password Authentication Protocol (PAP), 600
 Password based attacks, 411
 Password hashing, 300
 Password sniffing, 297
 Password stealing Torjan horse virus, 549
 Pattern attack, 24
 PGP key management, 435
 PGP message format, 437
 PGP message transmission, 438
 Phishing attack, 10
 Pigeonhole principle, 235
 Ping, 375
 Plaintext, 20
 Playfair cipher, 43
 Point-to-point tunneling protocol, 607
 Poly-alphabetic cipher, 41
 Polymorphic virus, 540
 POP3, 425
 Port Address Translation (PAT), 594
 Port stealing, 518
 Pretty Good Privacy (PGP), 431, 432
 Privacy Enhanced Mail (PEM), 431
 Private key, 190
 Product cipher, 96
 Proxy filter (server), 587
 Public key, 190
 Public key cryptography, 190
 Public key distribution, 331

Public key encryption, 190
 Public key infrastructure (PKI), 340

Q

Quadgram Frequency, 23

R

Rabin cryptosystem, 200
 Rail fence cipher, 58
 Rainbow table, 302
 Random oracle model (ROM), 234
 Remote access Trojans, (RATs), 546
 Remote access VPN, 597
 Replaying, 7
 Repudiation, 7
 Retrovirus, 541
 Role based access control (RBAC), 15
 Round key generation, 113
 Rounds, 97
 Routing control, 14
 RSA algorithm, 192
 RSA digital signature, 286

S

S-Boxes (Substitution box), 92, 93
 Secret key, 20
 Secure hash algorithm (SHA), 261
 Secure Shell (SSH), 466
 Secure Socket Layer (SSL), 452
 Security Association (SA), 497
 Security attacks, 5
 Security authentication data, 492
 Security handshake pitfalls, 349
 Security Parameter Index (SPI), 493
 Security policies, 499
 Security services, 12
 Session hijacking, 389
 SHA1, 262
 SHA 256, 266
 SHA 512, 269
 Shift cipher, 37

Shift register based stream cipher, 65
 Shiva Password Authentication Protocol (SPAP), 600
 Signature based IDS, 558
 Site-to-site VPN, 598
 Small private key attack, 199
 Small public key, 199
 S-MIME, 440, 442
 SMTP, 423
 Smurf attack, 406
 Sniffer tools, 565
 Sniffing, 387
 Snooping, 5
 Social engineering, 298, 410
 Spanning Tree Protocol (STP), 526
 Spoof attack, 11
 Spoofing, 385
 SSH connection protocol, 471
 SSH services, 473
 SSH user authentication protocol, 470
 SSL record protocol, 458
 Stateful packet filtering, 583
 Stateless packet filtering, 582
 Static NAT, 591
 Statistical attack, 24
 Stealth virus, 541
 STP attacks, 527
 Stream cipher, 21, 63
 Substitution, 21
 Substitution cipher, 34, 85
 Swap operation, 95
 Symmetric key encryption, 21
 SYN flood, 408
 System killing Trojan horse virus, 550

T

TCP/IP layer architecture model, 364
 TCP/IP protocol suite, 366
 TCP Session hijacking, 392
 Threat to Integrity, 6
 Ticket granting server (TGS), 343
 Ticket granting ticket (TGT), 343

Timestamp, 309
 TLS Handshake protocol, 463
 TLS record protocol, 463
 Traffic analysis, 5
 Traffic padding, 14
 Transport layer protocol, 467
 Transport Layer Security (TLS), 451, 461
 Transport mode, 484, 486
 Transposition cipher, 21, 34, 57, 85
 Trigram frequency, 23
 Triple DES, 109, 126
 Trojan Horse, 396, 546
 Trusted centers, 332
 Tunneling protocol, 607
 Tunnel mode, 485, 486

V

Vigenere cipher, 46
 Virtual Local Area Network (VLAN), 413
 Virtual Private Network (VPN), 575, 597
 Virus structure, 535, 536
 VLAN hopping, 525
 VPN encryption, 603
 VPN tunneling, 604
 Vulnerability and attacks in networks, 360

W

Wireless IPS, 564
 Worms, 542

X

X.509 Certificate, 336
 XOR, 94

