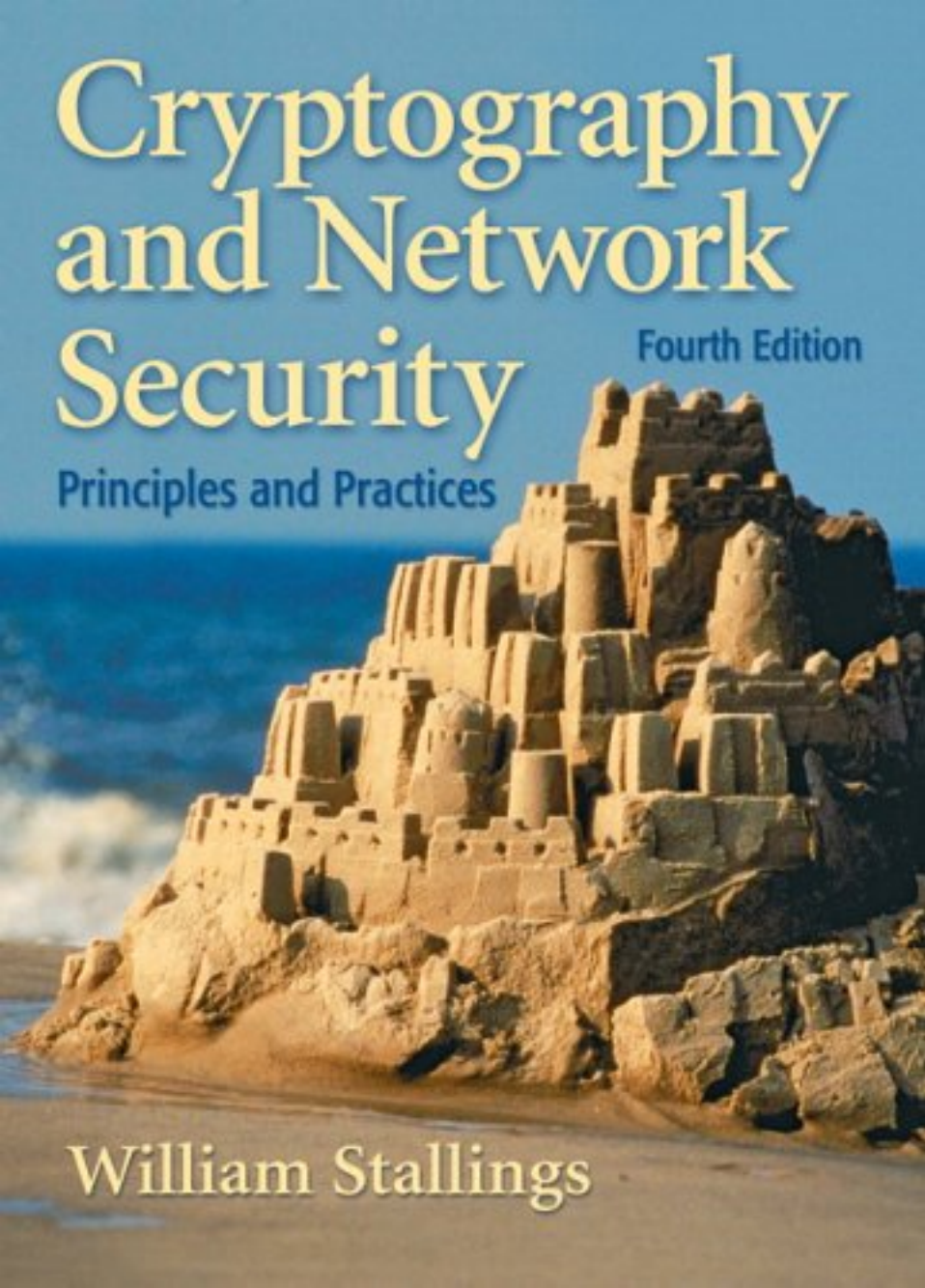
Cryptography and Network Security

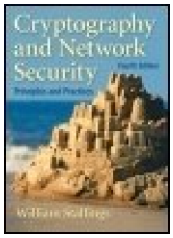
Fourth Edition

Principles and Practices



William Stallings

NEXT



Cryptography and Network Security Principles and Practices, Fourth Edition

By William Stallings

Publisher: **Prentice Hall**

Pub Date: **November 16, 2005**

Print ISBN-10: **0-13-187316-4**

Print ISBN-13: **978-0-13-187316-2**

eText ISBN-10: **0-13-187319-9**

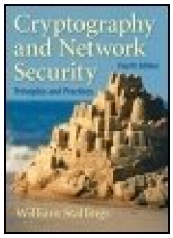
eText ISBN-13: **978-0-13-187319-3**

Pages: **592**

In this age of viruses and hackers, of electronic eavesdropping and electronic fraud, security is paramount.

As the disciplines of cryptography and network security have matured, more practical, readily available applications to enforce network security have developed. This text provides a practical survey of both the principles and practice of cryptography and network security. First, the basic issues to be addressed by a network security capability are explored through a tutorial and survey of cryptography and network security technology. Then, the practice of network security is explored via practical applications that have been implemented and are in use today.

NEXT



Cryptography and Network Security Principles and Practices, Fourth Edition

By William Stallings

Publisher: Prentice Hall

Pub Date: November 16, 2005

Print ISBN-10: 0-13-187316-4

Print ISBN-13: 978-0-13-187316-2

eText ISBN-10: 0-13-187319-9

eText ISBN-13: 978-0-13-187319-3

Pages: 592

- Table of Contents
- Index

Copyright

Library of Congress Cataloging-in-Publication Data on File

Vice President and Editorial Director, ECS: *Marcia J. Horton*

Executive Editor: *Tracy Dunkelberger*

Editorial Assistant: *Christianna Lee*

Executive Managing Editor: *Vince O'Brien*

Managing Editor: *Camille Trentacoste*

Production Editor: *Rose Kernan*

Director of Creative Services: *Paul Belfanti*

Cover Designer: *Bruce Kenselaar*

Managing Editor, AV Management and Production: *Patricia Burns*

Art Editor: *Gregory Dulles*

Manufacturing Manager: *Alexis Heydt-Long*

Manufacturing Buyer: *Lisa McDowell*

Marketing Manager: *Robin O'Brien*

Marketing Assistant: *Barrie Reinhold*

Pearson Prentice Hall

Pearson Education, Inc.

Upper Saddle River, NJ 07458

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Inc., *Toronto*
Pearson Educacíon de Mexico, S.A. de C.V.
Pearson EducationJapan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education Inc., *Upper Saddle River, New Jersey*

Dedication

To Antigone never dull never boring always a Sage

PREV NEXT

Notation

Even the natives have difficulty mastering this peculiar vocabulary.

The Golden Bough, Sir James George Frazer

Symbol	Expression	Meaning
D, K	D(K, Y)	Symmetric decryption of ciphertext Y using secret key K.
D, PR_a	D(PR_a, Y)	Asymmetric decryption of ciphertext Y using A's private key PR_a
D, PU_a	D(PU_a, Y)	Asymmetric decryption of ciphertext Y using A's public key PU_a
E, K	E(K, X)	Symmetric encryption of plaintext X using secret key K.
E, PR_a	E(PR_a, X)	Asymmetric encryption of plaintext X using A's private key PR_a
E, PU_a	E(PU_a, X)	Asymmetric encryption of plaintext X using A's public key PU_a
K		Secret key
PR_a		Private key of user A
PU_a		Public key of user A
C, K	C(K, X)	Message authentication code of message X using secret key K.
GF(p)		The finite field of order p, where p is prime. The field is defined as the set Z_p together with the arithmetic operations modulo p.
GF(2^n)		The finite field of order 2^n.
Z_n		Set of nonnegative integers less than n
gcd	gcd(i, j)	Greatest common divisor; the largest positive integer that divides both i and j with no remainder on division.
mod	a mod m	Remainder after division of a by m.
mod, $\equiv$	$a \equiv b \pmod m$	$a \bmod m = b \bmod m$
mod, $\not\equiv$	$a \not\equiv b \pmod m$	$a \bmod m \not\equiv b \bmod m$
dlog	$\text{dlog}_{a,p}(b)$	Discrete logarithm of the number b for the base a (mod p)
f	f(n)	The number of positive integers less than n and relatively prime to n. This is Euler's totient function.
S	$\sum_{i=1}^{n} a_i$	$a_1 + a_2 + ... + a_n$

Symbol	Expression	Meaning
$\prod$	$\prod_{i=1}^{n} a_i$	$a_1 \times a_2 \times \ldots \times a_n$
|	$i \mid j$	i divides j, which means that there is no remainder when j is divided by i
|,|	$\lvert a \rvert$	Absolute value of a
||	$x \mid\mid y$	x concatenated with y
$\approx$	$x \approx y$	x is approximately equal to y
$\oplus$	$x \oplus y$	Exclusive-OR of x and y for single-bit variables; Bitwise exclusive-OR of x and y for multiple-bit variables
$\lfloor\,\rfloor$	$\lfloor x \rfloor$	The largest integer less than or equal to x
$\in$	$x \in S$	The element x is contained in the set S.
$\leftrightarrow$	$A \leftrightarrow (a_1, a_2, \ldots, a_k)$	The integer A corresponds to the sequence of integers $(a_1, a_2, \ldots, a_k)$

PREV NEXT

Preface

"The tie, if I might suggest it, sir, a shade more tightly knotted. One aims at the perfect butterfly effect. If you will permit me"

"What does it matter, Jeeves, at a time like this? Do you realize that Mr. Little's domestic happiness is hanging in the scale?"

"There is no time, sir, at which ties do not matter."

Very Good, Jeeves! P. G. Wodehouse

In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security does not matter. Two trends have come together to make the topic of this book of vital interest. First, the explosive growth in computer systems and their interconnections via networks has increased the dependence of both organizations and individuals on the information stored and communicated using these systems. This, in turn, has led to a heightened awareness of the need to protect data and resources from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. Second, the disciplines of cryptography and network security have matured, leading to the development of practical, readily available applications to enforce network security.

Objectives

It is the purpose of this book to provide a practical survey of both the principles and practice of cryptography and network security. In the first two parts of the book, the basic issues to be addressed by a network security capability are explored by providing a tutorial and survey of cryptography and network security technology. The latter part of the book deals with the practice of network security: practical applications that have been implemented and are in use to provide network security.

The subject, and therefore this book, draws on a variety of disciplines. In particular, it is impossible to appreciate the significance of some of the techniques discussed in this book without a basic understanding of number theory and some results from probability theory. Nevertheless, an attempt has been made to make the book self-contained. The book presents not only the basic mathematical results that are needed but provides the reader with an intuitive understanding of those results. Such background material is introduced as needed. This approach helps to motivate the material that is introduced, and the author considers this preferable to simply presenting all of the mathematical material in a lump at the beginning of the book.

Intended Audience

The book is intended for both an academic and a professional audience. As a textbook, it is intended as a one-semester undergraduate course in cryptography and network security for computer science, computer engineering, and electrical engineering majors. It covers the material in IAS2 Security Mechanisms, a core area in the Information Technology body of knowledge; NET4 Security, another core area in the Information Technology body of knowledge; and IT311, Cryptography, an advanced course; these subject areas are part of the Draft ACM/IEEE Computer Society Computing Curricula 2005.

The book also serves as a basic reference volume and is suitable for self-study.

Plan of the Book

The book is organized in four parts:

Part One. Conventional Encryption: A detailed examination of conventional encryption algorithms and design principles, including a discussion of the use of conventional encryption for confidentiality.

Part Two. Public-Key Encryption and Hash Functions: A detailed examination of public-key encryption algorithms and design principles. This part also examines the use of message authentication codes and hash functions, as well as digital signatures and public-key certificates.

Part Three. Network Security Practice: Covers important network security tools and applications, including Kerberos, X.509v3 certificates, PGP, S/MIME, IP Security, SSL/TLS, and SET.

Part Four. System Security: Looks at system-level security issues, including the threat of and countermeasures for intruders and viruses, and the use of firewalls and trusted systems.

In addition, the book includes an extensive glossary, a list of frequently used acronyms, and a bibliography. Each chapter includes homework problems, review questions, a list of key words, suggestions for further reading, and recommended Web sites.

A more detailed, chapter-by-chapter summary of each part appears at the beginning of that part.

Internet Services for Instructors and Students

There is a Web site for this book that provides support for students and instructors. The site includes links to other relevant sites, transparency masters of figures and tables in the book in PDF (Adobe Acrobat) format, and PowerPoint slides. The Web page is at WilliamStallings.com/Crypto/Crypto4e.html. As soon as typos or other errors are discovered, an errata list for this book will be available at WilliamStallings.com. In addition, the Computer Science Student Resource site, at WilliamStallings.com/StudentSupport.html, provides documents, information, and useful links for computer science students and professionals.

Projects for Teaching Cryptography and Network Security

For many instructors, an important component of a cryptography or security course is a project or set of projects by which the student gets hands-on experience to reinforce concepts from the text. This book provides an unparalleled degree of support for including a projects component in the course. The instructor's manual not only includes guidance on how to assign and structure the projects, but also includes a set of suggested projects that covers a broad range of topics from the text:

- **Research projects:** A series of research assignments that instruct the student to research a particular topic on the Internet and write a report
- **Programming projects:** A series of programming projects that cover a broad range of topics and that can be implemented in any suitable language on any platform
- **Lab exercises:** A series of projects that involve programming and experimenting with concepts from the book
- **Writing assignments:** A set of suggested writing assignments, by chapter
- **Reading/report assignments:** A list of papers in the literature, one for each chapter, that can be assigned for the student to read and then write a short report

See Appendix B for details.

What's New in the Fourth Edition

In the three years since the third edition of this book was published, the field has seen continued innovations and improvements. In this new edition, I try to capture these changes while maintaining a broad and comprehensive coverage of the entire field. To begin this process of revision, the third edition was extensively reviewed by a number of professors who teach the subject. In addition, a number of professionals working in the field reviewed individual chapters. The result is that, in many places, the narrative has been clarified and tightened, and illustrations have been improved. Also, a large number of new "field-tested" problems have been added.

Beyond these refinements to improve pedagogy and user friendliness, there have been major substantive changes throughout the book. Highlights include the following:

- **Simplified AES:** This is an educational, simplified version of AES (Advanced Encryption Standard), which enables students to grasp the essentials of AES more easily.
- **Whirlpool:** This is an important new secure hash algorithm based on the use of a symmetric block cipher.
- **CMAC:** This is a new block cipher mode of operation. CMAC (cipher-based message authentication code) provides message authentication based on the use of a symmetric block cipher.
- **Public-key infrastructure (PKI):** This important topic is treated in this new edition.
- **Distributed denial of service (DDoS) attacks:**DDoS attacks have assumed increasing significance in recent years.
- **Common Criteria for Information Technology Security Evaluation:** The Common Criteria have become the international framework for expressing security requirements and evaluating products and implementations.
- **Online appendices:** Six appendices available at this book's Web site supplement the material in the text.

In addition, much of the other material in the book has been updated and revised.

Acknowledgments

This new edition has benefited from review by a number of people, who gave generously of their time and expertise. The following people reviewed all or a large part of the manuscript: Danny Krizanc (Wesleyan University), Breno de Medeiros (Florida State University), Roger H. Brown (Rensselaer at Hartford), Cristina Nita-Rotarul (Purdue University), and Jimmy McGibney (Waterford Institute of Technology).

Thanks also to the many people who provided detailed technical reviews of a single chapter: Richard Outerbridge, Jorge Nakahara, Jeroen van de Graaf, Philip Moseley, Andre Correa, Brian Bowling, James Muir, Andrew Holt, Décio Luiz Gazzoni Filho, Lucas Ferreira, Dr. Kemal Bicakci, Routo Terada, Anton Stiglic, Valery Pryamikov, and Yongge Wang.

Joan Daemen kindly reviewed the chapter on AES. Vincent Rijmen reviewed the material on Whirlpool. And Edward F. Schaefer reviewed the material on simplified AES.

The following people contributed homework problems for the new edition: Joshua Brandon Holden (Rose-Hulman Institute if Technology), Kris Gaj (George Mason University), and James Muir (University of Waterloo).

Sanjay Rao and Ruben Torres of Purdue developed the laboratory exercises that appear in the instructor's supplement. The following people contributed project assignments that appear in the instructor's supplement: Henning Schulzrinne (Columbia University); Cetin Kaya Koc (Oregon State University); and David Balenson (Trusted Information Systems and George Washington University).

Finally, I would like to thank the many people responsible for the publication of the book, all of whom did their usual excellent job. This includes the staff at Prentice Hall, particularly production manager Rose Kernan; my supplements manager Sarah Parker; and my new editor Tracy Dunkelberger. Also, Patricia M. Daly did the copy editing.

With all this assistance, little remains for which I can take full credit. However, I am proud to say that, with no help whatsoever, I selected all of the quotations.

Chapter 0. Reader's Guide

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

The Art of War, Sun Tzu

This book, with its accompanying Web site, covers a lot of material. Here we give the reader an overview.

0.1. Outline of this Book

Following an introductory chapter, Chapter 1, the book is organized into four parts:

Part One: Symmetric Ciphers: Provides a survey of symmetric encryption, including classical and modern algorithms. The emphasis is on the two most important algorithms, the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES). This part also addresses message authentication and key management.

Part Two: Public-Key Encryption and Hash Functions: Provides a survey of public-key algorithms, including RSA (Rivest-Shamir-Adelman) and elliptic curve. It also covers public-key applications, including digital signatures and key exchange.

Part Three: Network Security Practice: Examines the use of cryptographic algorithms and security protocols to provide security over networks and the Internet. Topics covered include user authentication, e-mail, IP security, and Web security.

Part Four: System Security: Deals with security facilities designed to protect a computer system from security threats, including intruders, viruses, and worms. This part also looks at firewall technology.

Many of the cryptographic algorithms and network security protocols and applications described in this book have been specified as standards. The most important of these are Internet Standards, defined in Internet RFCs (Request for Comments), and Federal Information Processing Standards (FIPS), issued by the National Institute of Standards and Technology (NIST). Appendix A discusses the standards-making process and lists the standards cited in this book.

0.2. Roadmap

Subject Matter

The material in this book is organized into three broad categories:

> **Cryptology:** This is the study of techniques for ensuring the secrecy and/or authenticity of information. The two main branches of cryptology are **cryptography**, which is the study of the design of such techniques; and **cryptanalysis**, which deals with the defeating such techniques, to recover information, or forging information that will be accepted as authentic.

> **Network security:** This area covers the use of cryptographic algorithms in network protocols and network applications.
>
> **Computer security:** In this book, we use this term to refer to the security of computers against intruders (e.g., hackers) and malicious software (e.g., viruses). Typically, the computer to be secured is attached to a network and the bulk of the threats arise from the network.

The first two parts of the book deal with two distinct cryptographic approaches: symmetric cryptographic algorithms and public-key, or asymmetric, cryptographic algorithms. Symmetric algorithms make use of a single shared key shared by two parties. Public-key algorithms make use of two keys: a private key known only to one party, and a public key, available to other parties.

Topic Ordering

This book covers a lot of material. For the instructor or reader who wishes a shorter treatment, there are a number of opportunities.

To thoroughly cover the material in the first two parts, the chapters should be read in sequence. With the exception of the Advanced Encryption Standard (AES), none of the material in **Part One** requires any special mathematical background. To understand AES, it is necessary to have some understanding of finite fields. In turn, an understanding of finite fields requires a basic background in prime numbers and modular arithmetic. Accordingly, Chapter 4 covers all of these mathematical preliminaries just prior to their use in Chapter 5 on AES. Thus, if Chapter 5 is skipped, it is safe to skip Chapter 4 as well.

Chapter 2 introduces some concepts that are useful in later chapters of Part One. However, for the reader whose sole interest is contemporary cryptography, this chapter can be quickly skimmed. The two most important symmetric cryptographic algorithms are DES and AES, which are covered in Chapters 3 and 5, respectively. Chapter 6 covers two other interesting algorithms, both of which enjoy commercial use. This chapter can be safely skipped if these algorithms are not of interest.

For **Part Two**, the only additional mathematical background that is needed is in the area of number theory, which is covered in Chapter 8. The reader who has skipped Chapters 4 and 5 should first review the material on Sections 4.1 through 4.3.

The two most widely used general-purpose public-key algorithms are RSA and elliptic curve, with RSA enjoying much wider acceptance. The reader may wish to skip the material on elliptic curve cryptography in Chapter 10, at least on a first reading. In Chapter 12, Whirlpool and CMAC are of lesser importance.

Part Three and **Part Four** are relatively independent of each other and can be read in either order. Both parts assume a basic understanding of the material in Parts One and Two.

PREV NEXT

0.3. Internet and Web Resources

There are a number of resources available on the Internet and the Web to support this book and to help one keep up with developments in this field.

Web Sites for This Book

A special Web page has been set up for this book at **WilliamStallings.com/Crypto/Crypto4e.html.** The site includes the following:

- **Useful Web sites:** There are links to other relevant Web sites, organized by chapter, including the sites listed in this section and throughout this book.
- **Errata sheet:** An errata list for this book will be maintained and updated as needed. Please e-mail any errors that you spot to me. Errata sheets for my other books are at **WilliamStallings.com.**
- **Figures:** All of the figures in this book in PDF (Adobe Acrobat) format.
- **Tables:** All of the tables in this book in PDF format.
- **Slides:** A set of PowerPoint slides, organized by chapter.
- **Cryptography and network security courses:** There are links to home pages for courses based on this book; these pages may be useful to other instructors in providing ideas about how to structure their course.

I also maintain the Computer Science Student Resource Site, at **WilliamStallings.com/StudentSupport.html.** The purpose of this site is to provide documents, information, and links for computer science students and professionals. Links and documents are organized into four categories:

- **Math:** Includes a basic math refresher, a queuing analysis primer, a number system primer, and links to numerous math sites
- **How-to:** Advice and guidance for solving homework problems, writing technical reports, and preparing technical presentations
- **Research resources:** Links to important collections of papers, technical reports, and bibliographies
- **Miscellaneous:** A variety of other useful documents and links

Other Web Sites

There are numerous Web sites that provide information related to the topics of this book. In subsequent chapters, pointers to specific Web sites can be found in the *Recommended Reading and Web Sites* section. Because the addresses for Web sites tend to change frequently, I have not included URLs in the book. For all of the Web sites listed in the book, the appropriate link can be found at this book's Web site. Other links not mentioned in this book will be added to the Web site over time.

USENET Newsgroups

A number of USENET newsgroups are devoted to some aspect of cryptography or network security. As with virtually all USENET groups, there is a high noise-to-signal ratio, but it is worth experimenting to see if any meet your needs. The most relevant are

- **sci.crypt.research:** The best group to follow. This is a moderated newsgroup that deals with research topics; postings must have some relationship to the technical aspects of cryptology.
- **sci.crypt:** A general discussion of cryptology and related topics.
- **sci.crypt.random-numbers:** A discussion of cryptographic-strength random number generators.
- **alt.security:** A general discussion of security topics.
- **comp.security.misc:** A general discussion of computer security topics.
- **comp.security.firewalls:** A discussion of firewall products and technology.
- **comp.security.announce:** News, announcements from CERT.
- **comp.risks:** A discussion of risks to the public from computers and users.
- **comp.virus:** A moderated discussion of computer viruses.

Chapter 1. Introduction

The combination of space, time, and strength that must be considered as the basic elements of this theory of defense makes this a fairly complicated matter. Consequently, it is not easy to find a fixed point of departure.

On War, Carl Von Clausewitz

Key Points

- The OSI (open systems interconnection) security architecture provides a systematic framework for defining security attacks, mechanisms, and services.
- **Security attacks** are classified as either passive attacks, which include unauthorized reading of a message of file and traffic analysis; and active attacks, such as modification of messages or files, and denial of service.
- A **security mechanism** is any process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack. Examples of mechanisms are encryption algorithms, digital signatures, and authentication protocols.
- **Security services** include authentication, access control, data confidentiality, data integrity, nonrepudiation, and availability.

The requirements of **information security** within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents. An example of the latter is personnel screening procedures used during the hiring process.

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the Internet. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**.

The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. Network security measures are needed to protect data during their transmission. In fact, the term **network security** is somewhat misleading, because virtually all business, government, and academic organizations interconnect their data processing equipment with a collection of interconnected networks. Such a collection is often referred to as an internet,[1] and the term **internet security** is used.

[1] We use the term *internet*, with a lowercase "i," to refer to any interconnected collection of networks. A corporate intranet is an example of an internet. The Internet with a capital "I" may be one of the facilities used by an organization to construct its internet.

There are no clear boundaries between these two forms of security. For example, one of the most publicized types of attack on information systems is the computer virus. A virus may be introduced into a system physically when it arrives on a diskette or optical disk and is subsequently loaded onto a computer. Viruses may also arrive over an internet. In either case, once the virus is resident on a

computer system, internal computer security tools are needed to detect and recover from the virus.

This book focuses on internet security, which consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information. That is a broad statement that covers a host of possibilities. To give you a feel for the areas covered in this book, consider the following examples of security violations:

1. User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.

2. A network manager, D, transmits a message to a computer, E, under its management. The message instructs computer E to update an authorization file to include the identities of a number of new users who are to be given access to that computer. User F intercepts the message, alters its contents to add or delete entries, and then forwards the message to E, which accepts the message as coming from manager D and updates its authorization file accordingly.

3. Rather than intercept a message, user F constructs its own message with the desired entries and transmits that message to E as if it had come from manager D. Computer E accepts the message as coming from manager D and updates its authorization file accordingly.

4. An employee is fired without warning. The personnel manager sends a message to a server system to invalidate the employee's account. When the invalidation is accomplished, the server is to post a notice to the employee's file as confirmation of the action. The employee is able to intercept the message and delay it long enough to make a final access to the server to retrieve sensitive information. The message is then forwarded, the action taken, and the confirmation posted. The employee's action may go unnoticed for some considerable time.

5. A message is sent from a customer to a stockbroker with instructions for various transactions. Subsequently, the investments lose value and the customer denies sending the message.

Although this list by no means exhausts the possible types of security violations, it illustrates the range of concerns of network security.

Internetwork security is both fascinating and complex. Some of the reasons follow:

1. Security involving communications and networks is not as simple as it might first appear to the novice. The requirements seem to be straightforward; indeed, most of the major requirements for security services can be given self-explanatory one-word labels: confidentiality, authentication, nonrepudiation, integrity. But the mechanisms used to meet those requirements can be quite complex, and understanding them may involve rather subtle reasoning.

2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features. In many cases, successful attacks are designed by looking at the problem in a completely different way, therefore exploiting an unexpected weakness in the mechanism.

3. Because of point 2, the procedures used to provide particular services are often counterintuitive: It is not obvious from the statement of a particular requirement that such elaborate measures are needed. It is only when the various countermeasures are considered that the measures used make sense.

4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement (e.g., at what points in a network are certain security mechanisms needed) and in a logical sense [e.g., at what layer or layers of an architecture such as TCP/IP (Transmission Control Protocol/Internet Protocol) should mechanisms be placed].

5. Security mechanisms usually involve more than a particular algorithm or protocol. They usually also require that participants be in possession of some secret information (e.g., an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There is also a reliance on communications protocols whose behavior may complicate the task of developing the security mechanism. For example, if the proper functioning of the security mechanism requires setting time limits on the transit time of a message from sender to receiver, then any protocol or network that introduces variable, unpredictable delays may render such time limits meaningless.

Thus, there is much to consider. This chapter provides a general overview of the subject matter that structures the material in the remainder of the book. We begin with a general discussion of network security services and mechanisms and of the types of attacks they

are designed for. Then we develop a general overall model within which the security services and mechanisms can be viewed.

PREV NEXT

1.1. Security Trends

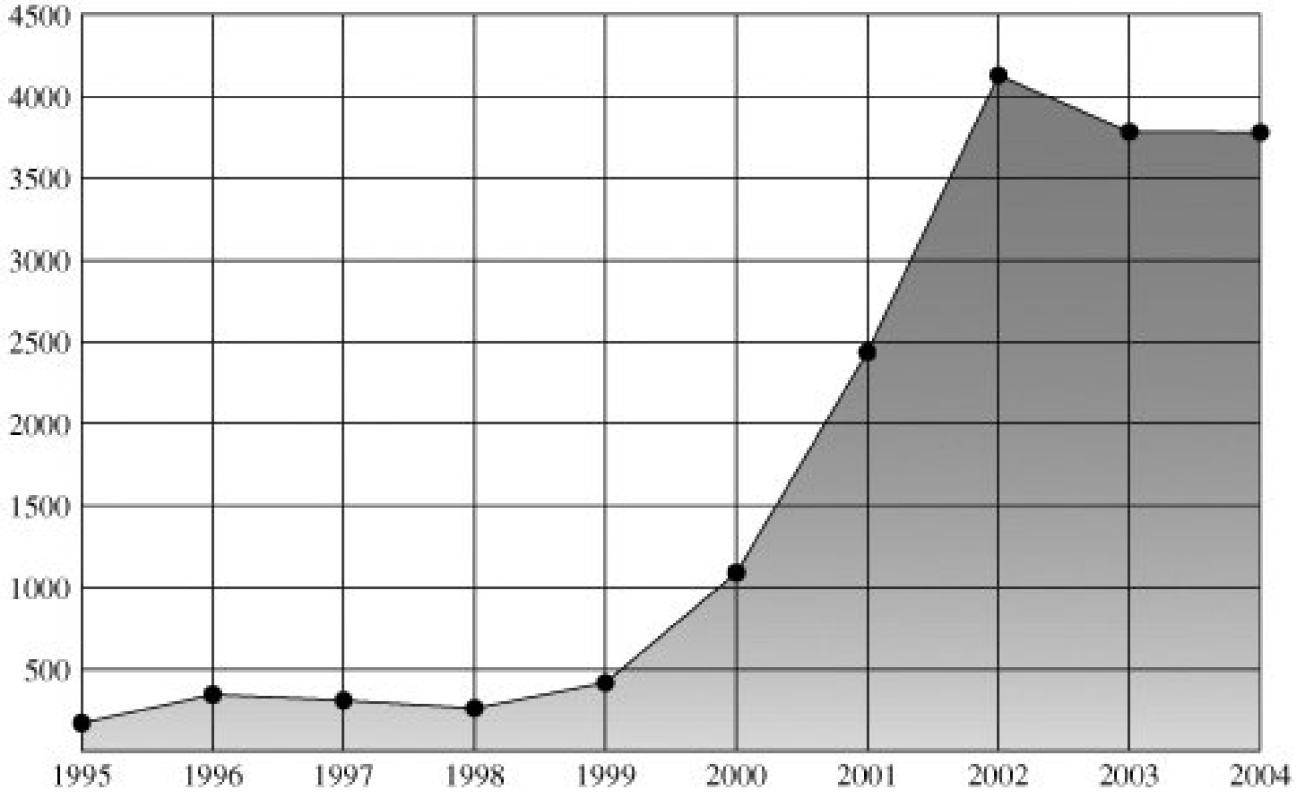
In 1994, the Internet Architecture Board (IAB) issued a report entitled "Security in the Internet Architecture" (RFC 1636). The report stated the general consensus that the Internet needs more and better security, and it identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

These concerns are fully justified. As confirmation, consider the trends reported by the Computer Emergency Response Team (CERT) Coordination Center (CERT/CC). Figure 1.1a shows the trend in Internet-related vulnerabilities reported to CERT over a 10-year period. These include security weaknesses in the operating systems of attached computers (e.g., Windows, Linux) as well as vulnerabilities in Internet routers and other network devices. Figure 1.1b shows the number of security-related incidents reported to CERT. These include denial of service attacks; IP spoofing, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP; and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents.

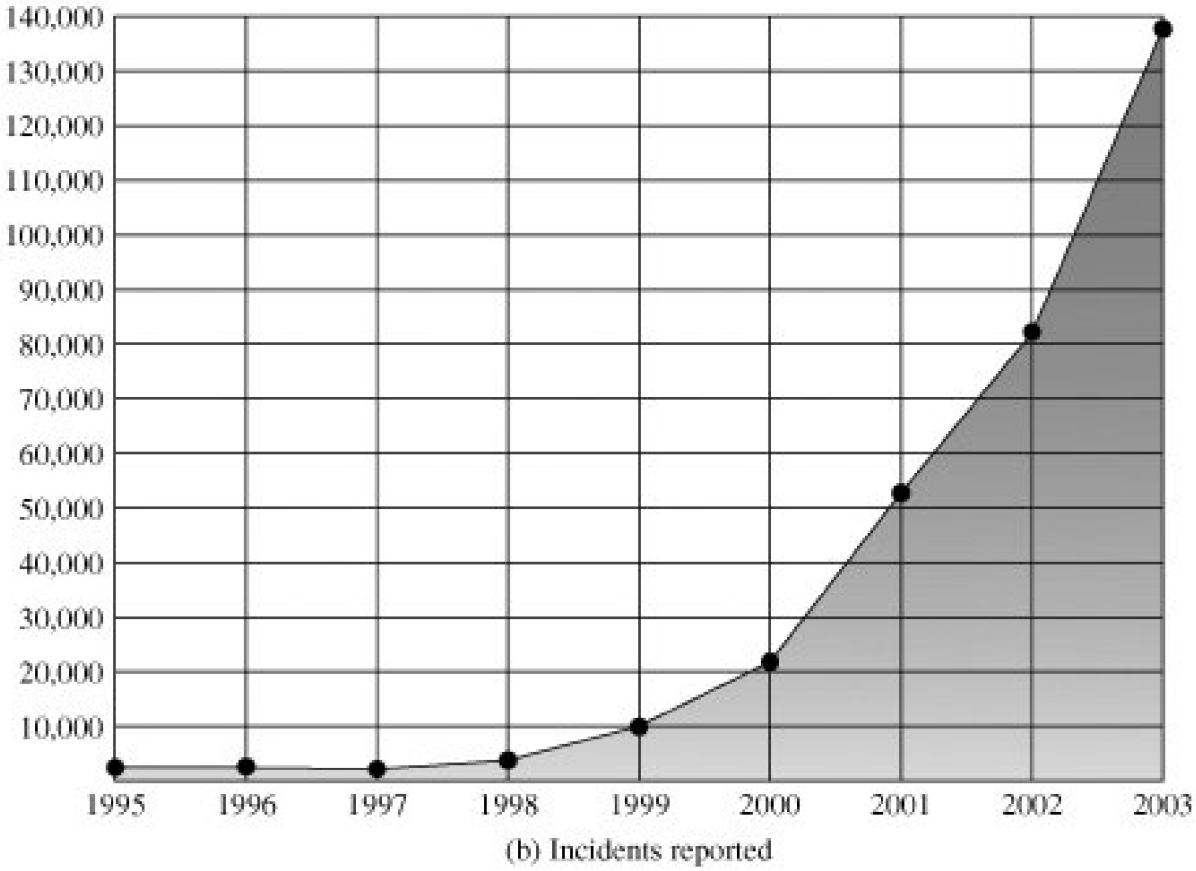
Figure 1.1. CERT Statistics

(This item is displayed on page 10 in the print version)

[View full size image]



(a) Vulnerabilities reported

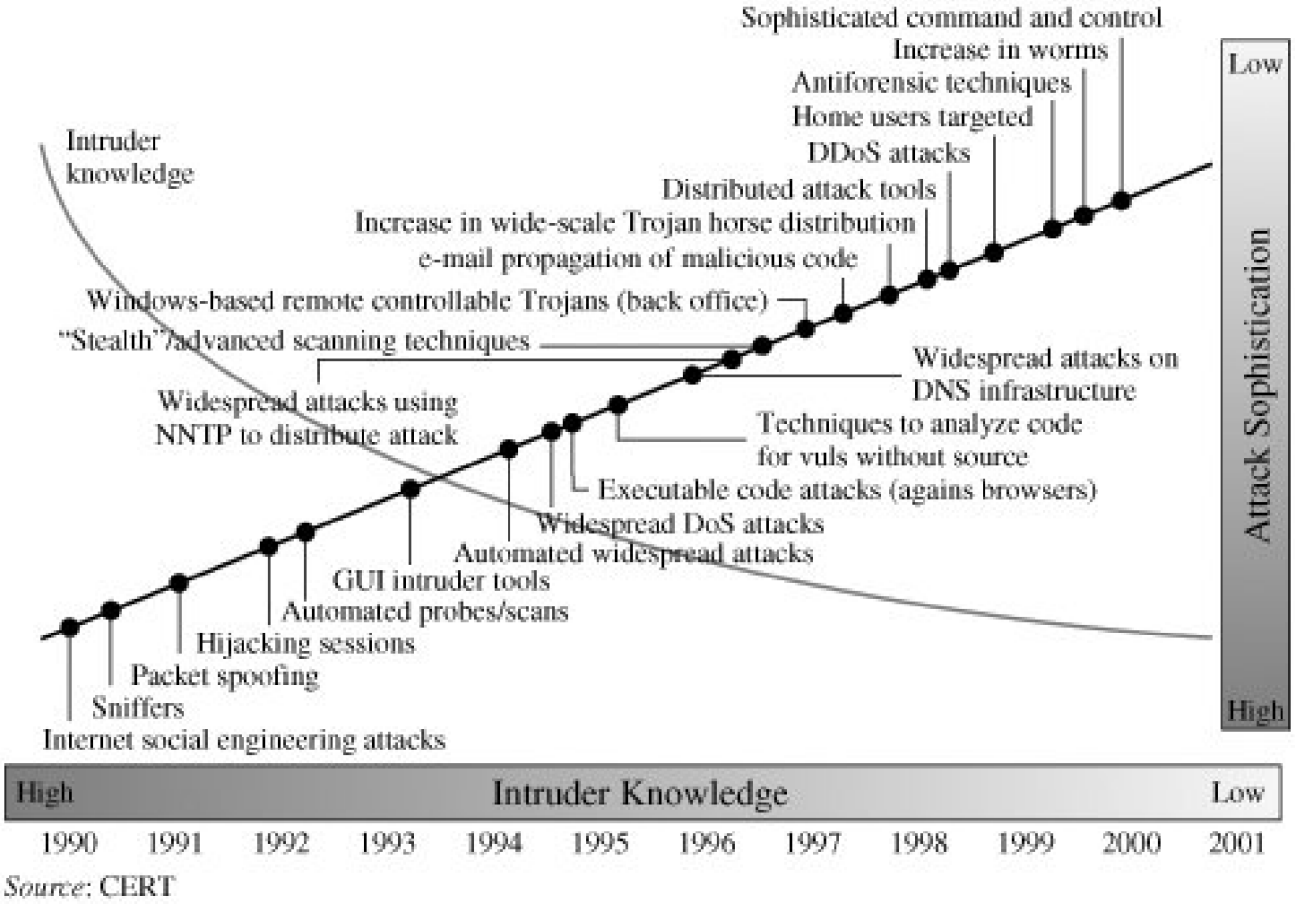


(b) Incidents reported

Over time, the attacks on the Internet and Internet-attached systems have grown more sophisticated while the amount of skill and knowledge required to mount an attack has declined (Figure 1.2). Attacks have become more automated and can cause greater amounts of damage.

Figure 1.2. Trends in Attack Sophistication and Intruder Knowledge

[View full size image]



Source: CERT

This increase in attacks coincides with an increased use of the Internet and with increases in the complexity of protocols, applications, and the Internet itself. Critical infrastructures increasingly rely on the Internet for operations. Individual users rely on the security of the Internet, email, the Web, and Web-based applications to a greater extent than ever. Thus, a wide range of technologies and tools are needed to counter the growing threat. At a basic level, cryptographic algorithms for confidentiality and authentication assume greater importance. As well, designers need to focus on Internet-based protocols and the vulnerabilities of attached operating systems and applications. This book surveys all of these technical areas.

PREV NEXT

1.2. The OSI Security Architecture

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. This is difficult enough in a centralized data processing environment; with the use of local and wide area networks, the problems are compounded.

ITU-T[2] Recommendation X.800, *Security Architecture for OSI*, defines such a systematic approach.[3] The OSI security architecture is useful to managers as a way of organizing the task of providing security. Furthermore, because this architecture was developed as an international standard, computer and communications vendors have developed security features for their products and services that relate to this structured definition of services and mechanisms.

[2] The International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T) is a United Nationssponsored agency that develops standards, called Recommendations, relating to telecommunications and to open systems interconnection (OSI).

[3] The OSI security architecture was developed in the context of the OSI protocol architecture, which is described in Appendix H. However, for our purposes in this chapter, an understanding of the OSI protocol architecture is not required.

For our purposes, the OSI security architecture provides a useful, if abstract, overview of many of the concepts that this book deals with. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

In the literature, the terms *threat* and *attack* are commonly used to mean more or less the same thingTable 1.1 provides definitions taken from RFC 2828, *Internet Security Glossary*.

Table 1.1. Threats and Attacks (RFC 2828)

Threat	A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.
Attack	An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

1.3. Security Attacks

A useful means of classifying security attacks, used both in X.800 and RFC 2828, is in terms of *passive attacks* and *active attacks*. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

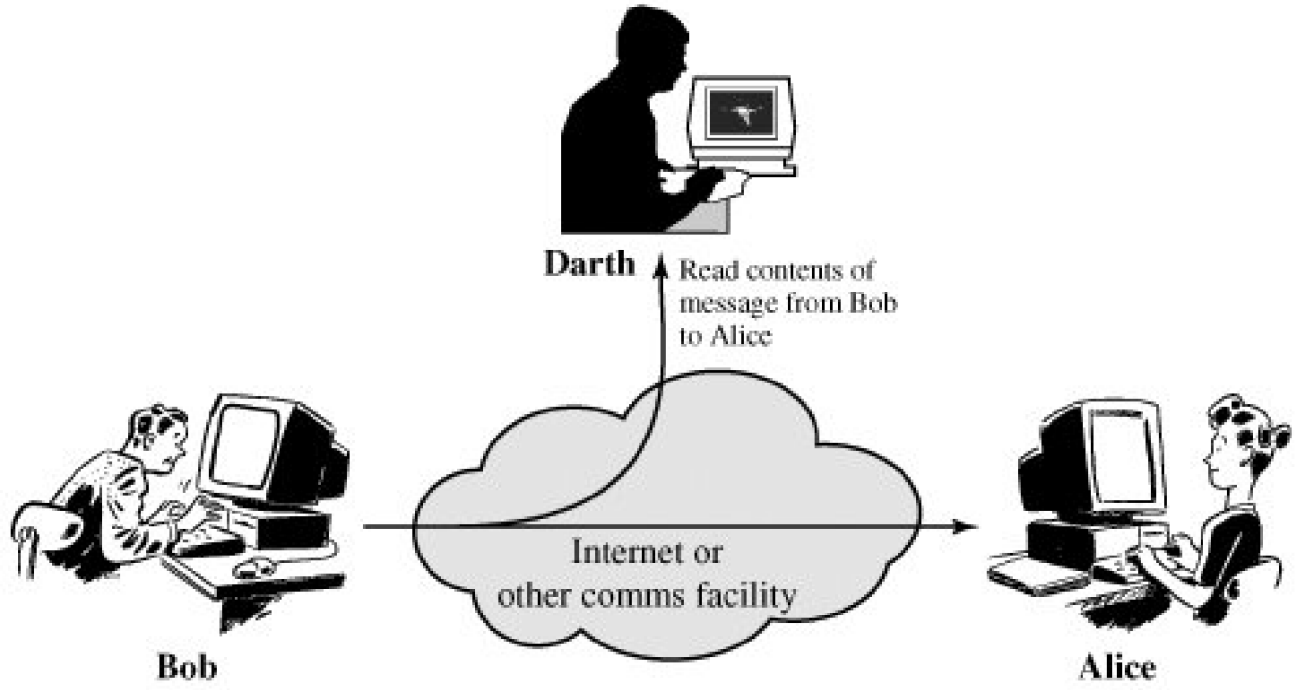
Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

The **release of message contents** is easily understood (Figure 1.3a). A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

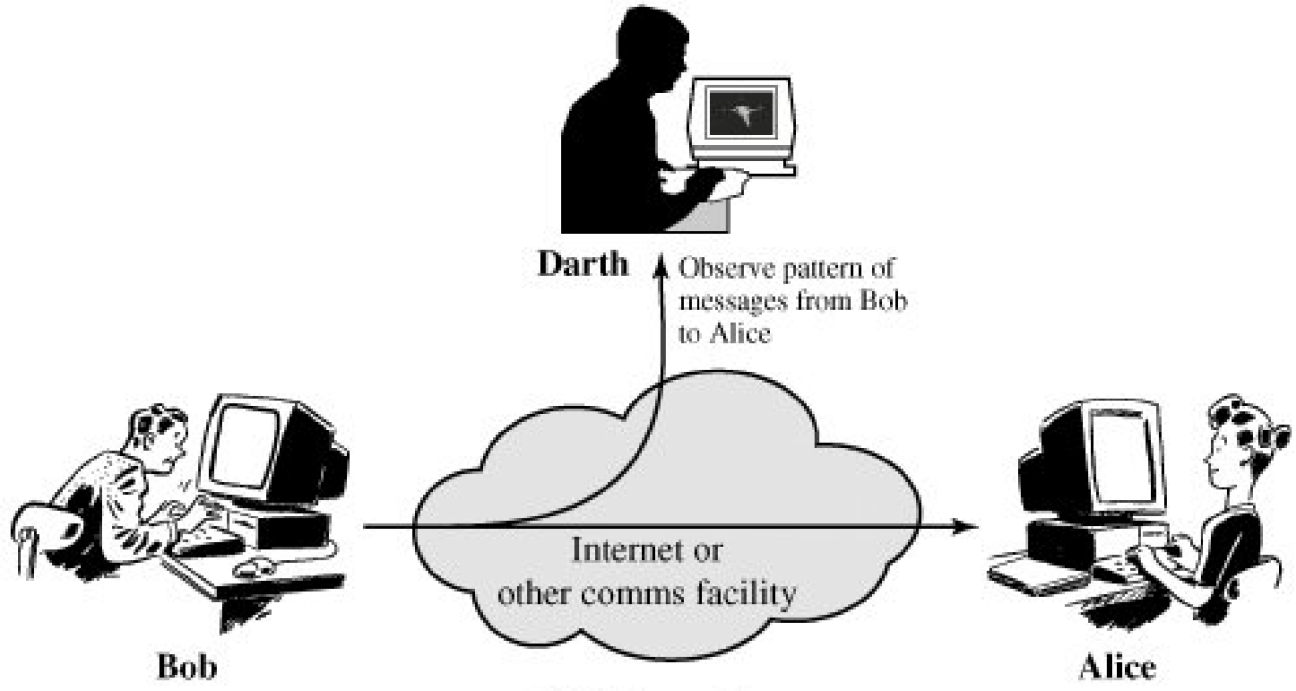
Figure 1.3. Passive Attacks

(This item is displayed on page 14 in the print version)

[View full size image]



(a) Release of message contents



(b) Traffic analysis

A second type of passive attack, **traffic analysis**, is subtler (Figure 1.3b). Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

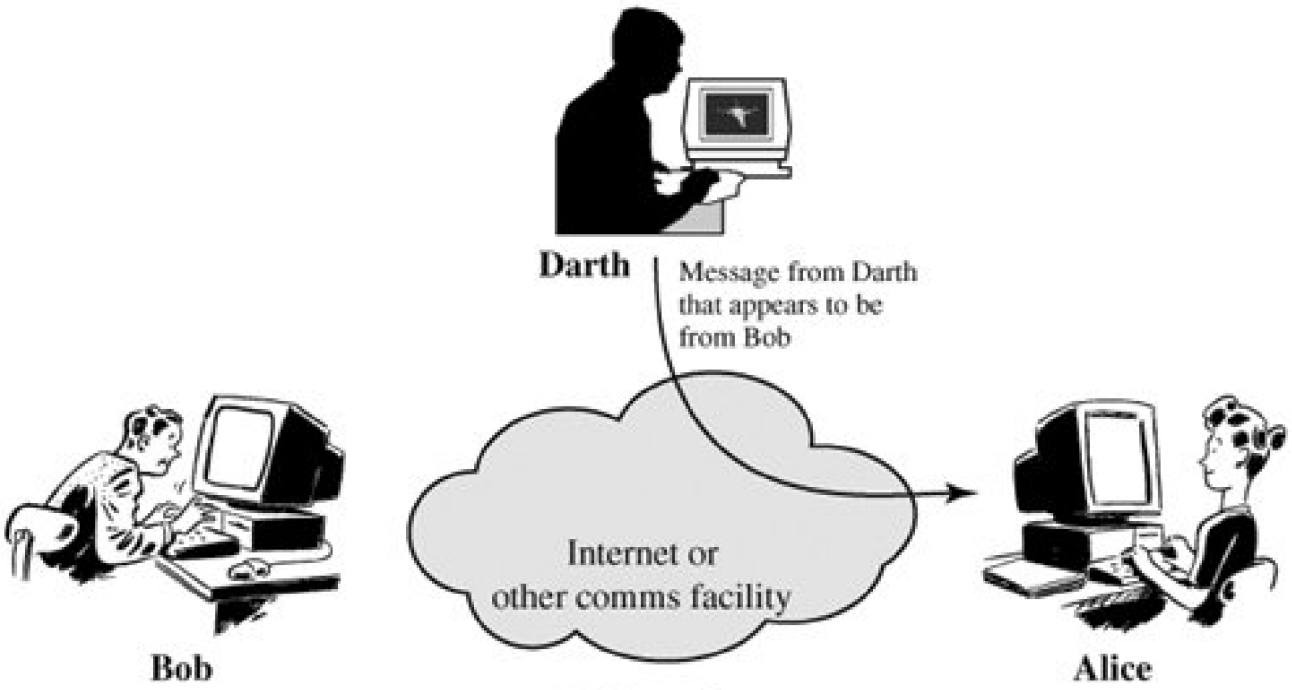
Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

A **masquerade** takes place when one entity pretends to be a different entity (Figure 1.4a). A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

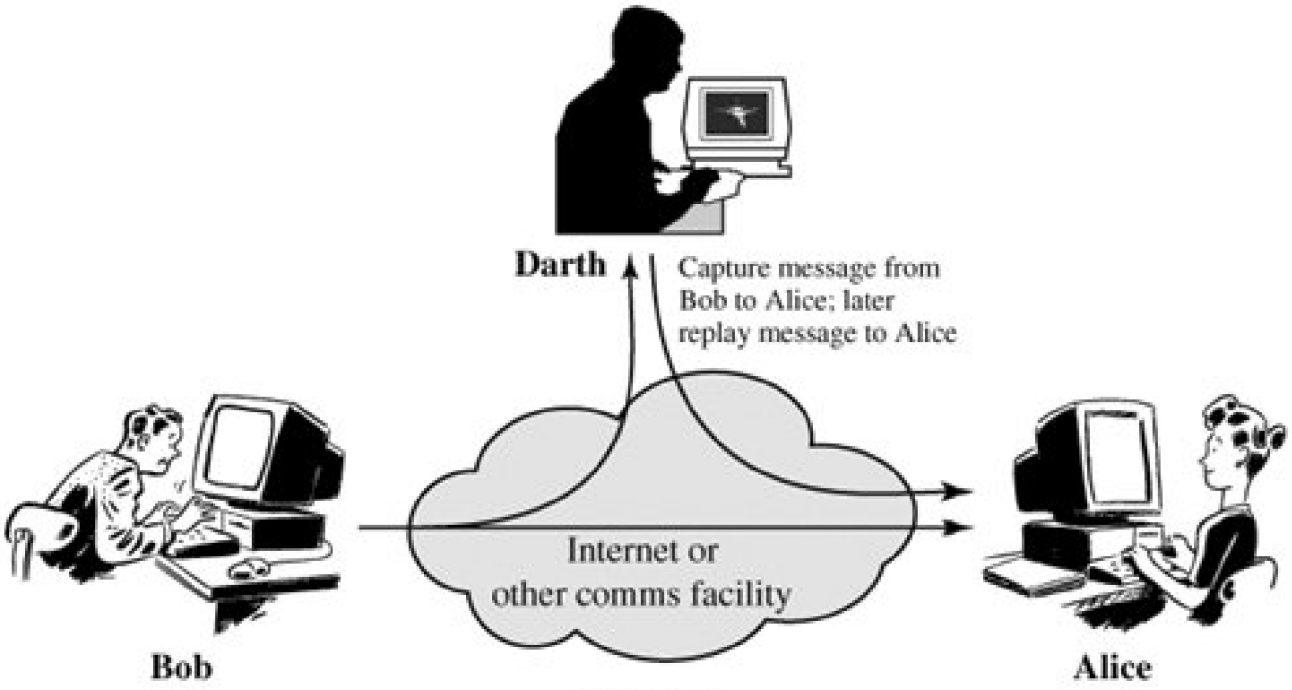
Figure 1.4. Active Attacks

(This item is displayed on pages 15 - 16 in the print version)

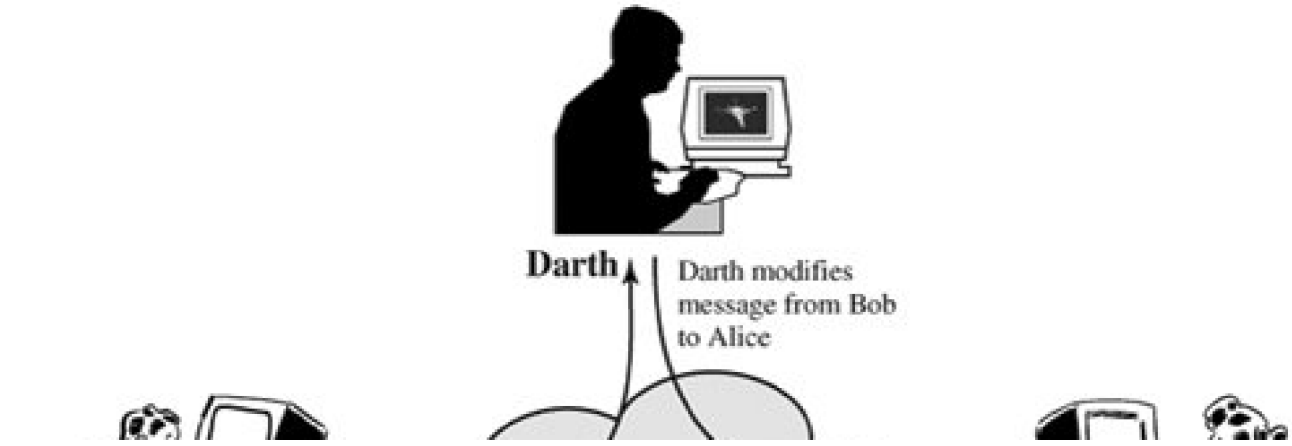
[View full size image]

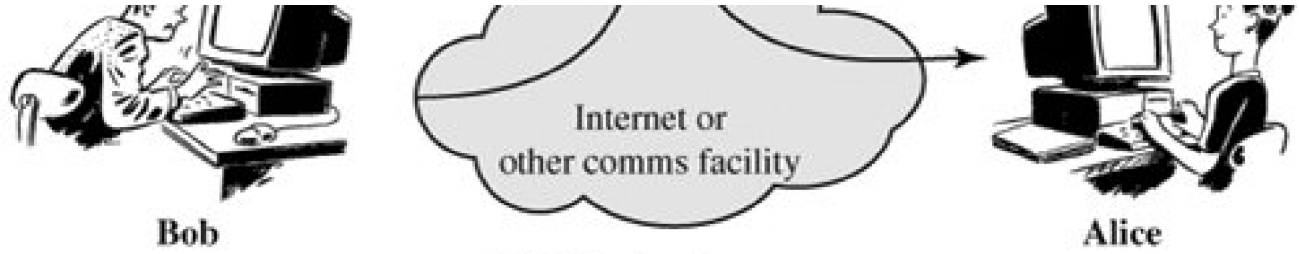


(a) Masquerade

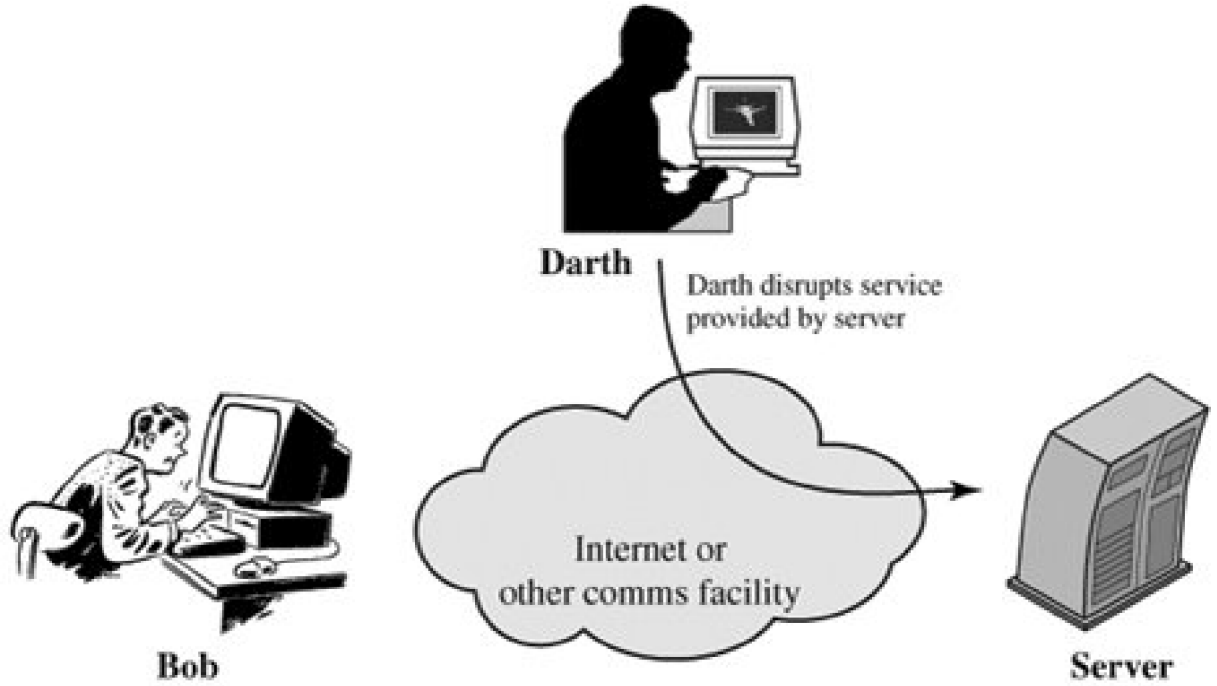


(b) Replay





(c) Modification of messages



(d) Denial of service

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (Figure 1.4b).

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (Figure 1.4c). For example, a message meaning "Allow John Smith to read confidential file *accounts*" is modified to mean "Allow Fred Brown to read confidential file *accounts*."

The **denial of service** prevents or inhibits the normal use or management of communications facilities (Figure 1.4d). This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

1.4. Security Services

X.800 defines a security service as a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers. Perhaps a clearer definition is found in RFC 2828, which provides the following definition: a processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms.

X.800 divides these services into five categories and fourteen specific services (Table 1.2). We look at each category in turn.[4]

[4] There is no universal agreement about many of the terms used in the security literature. For example, the term *integrity* is sometimes used to refer to all aspects of information security. The term *authentication* is sometimes used to refer both to verification of identity and to the various functions listed under integrity in this chapter. Our usage here agrees with both X.800 and RFC 2828.

Table 1.2. Security Services (X.800)

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data Origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality

The protection of all user data on a connection.

Connectionless Confidentiality

The protection of all user data in a single data block

Selective-Field Confidentiality

The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic Flow Confidentiality

The protection of the information that might be derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery

As above, but provides only detection without recovery.

Selective-Field Connection Integrity

Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity

Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity

Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

NONREPUDIATION

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin

Proof that the message was sent by the specified party.

Nonrepudiation, Destination

Proof that the message was received by the specified party.

Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of, or at times during the data transfer phase of, a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
- **Data origin authentication:** Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

Access Control

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights

can be tailored to the individual.

Data Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

Data Integrity

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection.

A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

Nonrepudiation

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

Availability Service

Both X.800 and RFC 2828 define availability to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system (i.e., a system is available if it provides services according to the system design whenever users request them). A variety of attacks can result in the loss of or reduction in availability. Some of these attacks are amenable to automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

X.800 treats availability as a property to be associated with various security services. However, it makes sense to call out specifically an availability service. An availability service is one that protects a system to ensure its availability. This service addresses the security concerns raised by denial-of-service attacks. It depends on proper management and control of system resources and thus depends on access control service and other security services.

PREV NEXT

1.5. Security Mechanisms

Table 1.3 lists the security mechanisms defined in X.800. As can be seen the mechanisms are divided into those that are implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. These mechanisms will be covered in the appropriate places in the book and so we do not elaborate now, except to comment on the definition of encipherment. X.800 distinguishes between reversible encipherment mechanisms and irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted. Irreversible encipherment mechanisms include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications.

Table 1.3. Security Mechanisms (X.800)

SPECIFIC SECURITY MECHANISMS

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment

The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature

Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control

A variety of mechanisms that enforce access rights to resources.

Data Integrity

A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange

A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding

The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control

Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization

The use of a trusted third party to assure certain properties of a data exchange.

PERVASIVE SECURITY MECHANISMS

Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality

That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label

The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection

Detection of security-relevant events.

Security Audit Trail

Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery

Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Table 1.4, based on one in X.800, indicates the relationship between security services and security mechanisms.

Table 1.4. Relationship between Security Services and Mechanisms

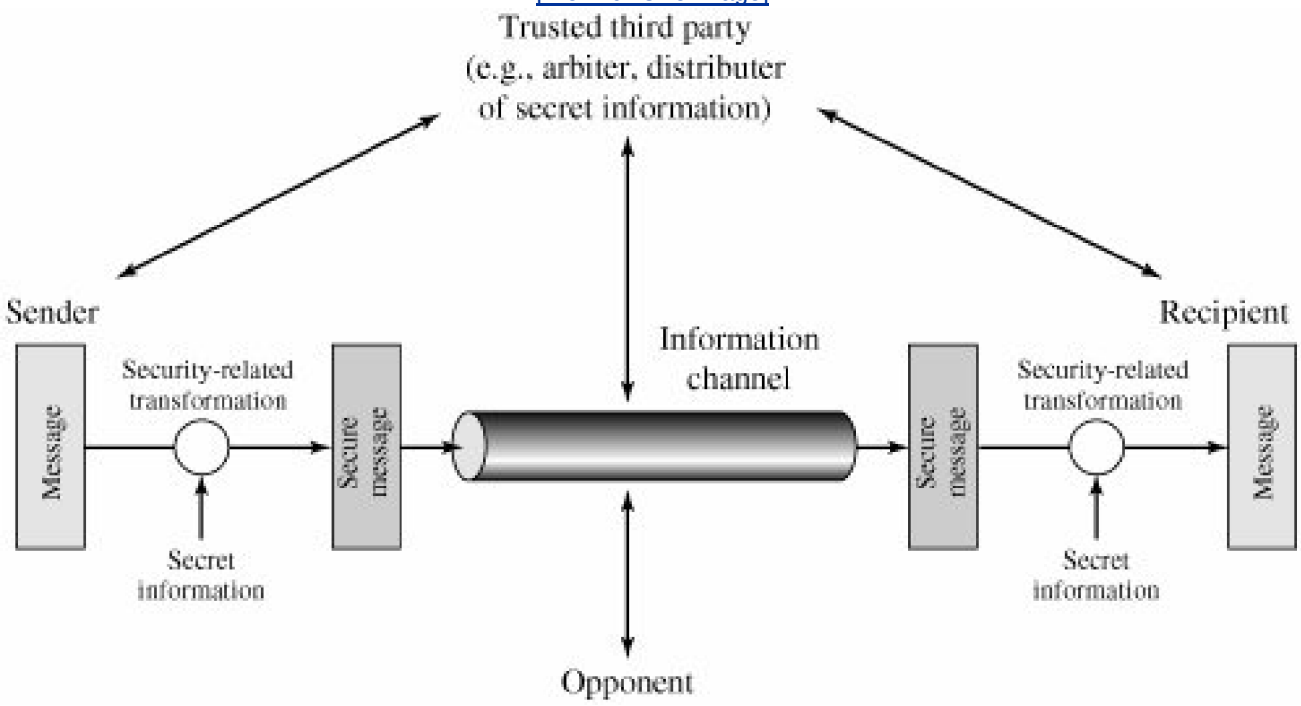
	Mechanism							
Service	**Encipherment**	**Digital Signature**	**Access Control**	**Data Integrity**	**Authentication Exchange**	**Traffic Padding**	**Routing Control**	**Notarization**
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

1.6. A Model for Network Security

A model for much of what we will be discussing is captured, in very general terms, in Figure 1.5. A message is to be transferred from one party to another across some sort of internet. The two parties, who are the *principals* in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

Figure 1.5. Model for Network Security

[View full size image]



Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.[5]

[5] Part Two discusses a form of encryption, known as public-key encryption, in which only one of the two principals needs to have the secret information.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

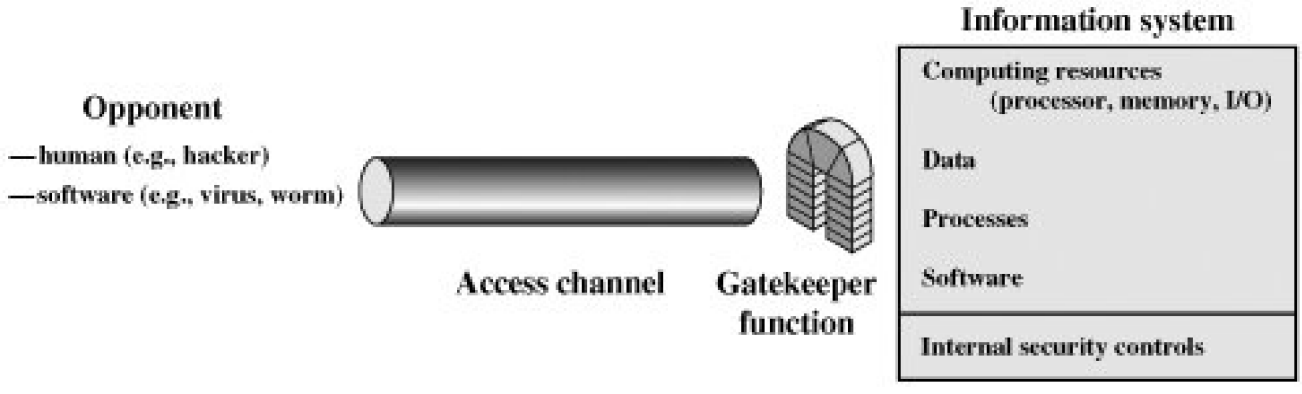
This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of the secret information.
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Parts One through Three of this book concentrates on the types of security mechanisms and services that fit into the model shown in Figure 1.5. However, there are other security-related situations of interest that do not neatly fit this model but that are considered in this book. A general model of these other situations is illustrated by Figure 1.6, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. Or, the intruder can be a disgruntled employee who wishes to do damage, or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

Figure 1.6. Network Access Security Model

[View full size image]



Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers. Programs can present two kinds of threats:

- **Information access threats** intercept or modify data on behalf of users who should not have access to that data.
- **Service threats** exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

The security mechanisms needed to cope with unwanted access fall into two broad categories (see Figure 1.6). The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access, the second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders. These issues are explored in Part Four.

PREV NEXT

1.7. Recommended Reading and Web Sites

[PFLE02] provides a good introduction to both computer and network security. Two other excellent surveys are [PIEP03] and [BISH05]. [BISH03] covers much the same ground as [BISH05] but with more mathematical detail and rigor. [SCHN00] is valuable reading for any practitioner in the field of computer or network security: it discusses the limitations of technology, and cryptography in particular, in providing security, and the need to consider the hardware, the software implementation, the networks, and the people involved in providing and attacking security.

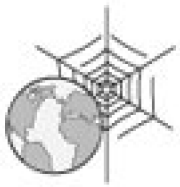
BISH03 Bishop, M. *Computer Security: Art and Science* Boston: Addison-Wesley, 2003.

BISH05 Bishop, M. *Introduction to Computer Security*. Boston: Addison-Wesley, 2005.

PFLE02 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2002.

PIEP03 Pieprzyk, J.; Hardjono, T.; and Seberry, J. *Fundamentals of Computer Security*. New York: Springer-Verlag, 2003.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World* New York: Wiley 2000.



Recommended Web Sites

The following Web sites[6] are of general interest related to cryptography and network security:

[6] Because URLs sometimes change, they are not included. For all of the Web sites listed in this and subsequent chapters, the appropriate link is at this book's Web site at **williamstallings.com/Crypto/Crypto4e.html**.

- **COAST:** Comprehensive set of links related to cryptography and network security.
- **IETF Security Area:** Material related to Internet security standardization efforts.
- **Computer and Network Security Reference Index:** A good index to vendor and commercial products, FAQs, newsgroup archives, papers, and other Web sites.

- **The Cryptography FAQ:** Lengthy and worthwhile FAQ covering all aspects of cryptography.
- **Tom Dunigan's Security Page:** An excellent list of pointers to cryptography and network security Web sites.
- **Helgar Lipma's Cryptology Pointers:** Another excellent list of pointers to cryptography and network security Web sites.
- **IEEE Technical Committee on Security and Privacy:**Copies of their newsletter, information on IEEE-related activities.
- **Computer Security Resource Center:** Maintained by the National Institute of Standards and Technology (NIST); contains a broad range of information on security threats, technology, and standards.
- **Security Focus:** A wide variety of security information, with an emphasis on vendor products and end-user concerns.
- **SANS Institute:** Similar to Security Focus. Extensive collection of white papers.

PREV NEXT

1.8. Key Terms, Review Questions, and Problems

Key Terms

access control

active threat

authentication

authenticity

availability

data confidentiality

data integrity

denial of service

encryption

integrity

intruder

masquerade

nonrepudiation

OSI security architecture

passive threat

replay

security attacks

security mechanisms

security services

traffic analysis

Review Questions

1.1 What is the OSI security architecture?

1.2 What is the difference between passive and active security threats?

1.3 List and briefly define categories of passive and active security attacks.

1.4 List and briefly define categories of security services.

1.5 List and briefly define categories of security mechanisms.

Problems

1.1 Draw a matrix similar to Table 1.4 that shows the relationship between security services and attacks.

1.2 Draw a matrix similar to Table 1.4 that shows the relationship between security mechanisms and attacks.

Part One: Symmetric Ciphers

> *Cryptography is probably the most important aspect of communications security and is becoming increasingly important as a basic building block for computer security.*
>
> *Computers at Risk: Safe Computing in the Information Age*, National Research Council, 1991

> *The increased use of computer and communications systems by industry has increased the risk of theft of proprietary information. Although these threats may require a variety of countermeasures, encryption is a primary method of protecting valuable electronic information.*
>
> *Communications Privacy: Federal Policy and Actions*, General Accounting Office Report GAO/OSI-94-2, November 1993

By far the most important automated tool for network and communications security is encryption. Two forms of encryption are in common use: conventional, or symmetric, encryption and public-key, or asymmetric, encryption. Part One provides a survey of the basic principles of symmetric encryption, looks at widely used algorithms, and discusses applications of symmetric cryptography.

Road Map for Part One

Chapter 2: Classical Encryption Techniques

Chapter 2 describes classical symmetric encryption techniques. It provides a gentle and interesting introduction to cryptography and cryptanalysis and highlights important concepts.

Chapter 3: Block Ciphers and the Data Encryption Standard

Chapter 3 introduces the principles of modern symmetric cryptography, with an emphasis on the most widely used encryption technique, the Data Encryption Standard (DES). The chapter includes a discussion of design considerations and cryptanalysis and introduces the Feistel cipher, which is the basic structure of most modern symmetric encryption schemes.

Chapter 4: Finite Fields

Finite fields have become increasingly important in cryptography. A number of cryptographic algorithms rely heavily on properties of finite fields, notably the Advanced Encryption Standard (AES) and elliptic curve cryptography. This chapter is positioned here so that concepts relevant to AES can be introduced prior to the discussion of AES. Chapter 4 provides the necessary background to the understanding of arithmetic over finite fields of the form GF(2^n).

Chapter 5: Advanced Encryption Standard

The most important development in cryptography in recent years is the adoption of a new symmetric cipher standard, AES. Chapter 5 provides a thorough discussion of this cipher.

Chapter 6: More on Symmetric Ciphers

Chapter 6 explores additional topics related to symmetric ciphers. The chapter begins by examining multiple encryption and, in particular, triple DES. Next, we look at the concept of block cipher modes of operation, which deal with ways of handling plaintext longer than a single block. Finally, the chapter discusses stream ciphers and describes RC4.

Chapter 7: Confidentiality Using Symmetric Encryption

Beyond questions dealing with the actual construction of a symmetric encryption algorithm, a number of design issues relate to the use of symmetric encryption to provide confidentiality. Chapter 7 surveys the most important of these issues. The chapter includes a discussion of end-to-end versus link encryption, techniques for achieving traffic confidentiality, and key distribution techniques. An important related topic, random number generation, is also addressed.

Chapter 2. Classical Encryption Techniques

Many savages at the present day regard their names as vital parts of themselves, and therefore take great pains to conceal their real names, lest these should give to evil-disposed persons a handle by which to injure their owners.

The Golden Bough, Sir James George Frazer

Key Points

- Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption.
- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The two types of attack on an encryption algorithm are cryptanalysis, based on properties of the encryption algorithm, and brute-force, which involves trying all possible keys.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.
- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption in the 1970s. It remains by far the most widely used of the two types of encryption. Part One examines a number of symmetric ciphers. In this chapter, we begin with a look at a general model for the symmetric encryption process; this will enable us to understand the context within which the algorithms are used. Next, we examine a variety of algorithms in use before the computer era. Finally, we look briefly at a different approach known as steganography. Chapter 3 examines the most widely used symmetric cipher: DES.

Before beginning, we define some terms. An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. The process of converting from plaintext to ciphertext is known as **enciphering** or **encryption**; restoring the plaintext from the ciphertext is **deciphering** or **decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system** or a **cipher**. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code." The areas of cryptography and cryptanalysis together are called **cryptology**.

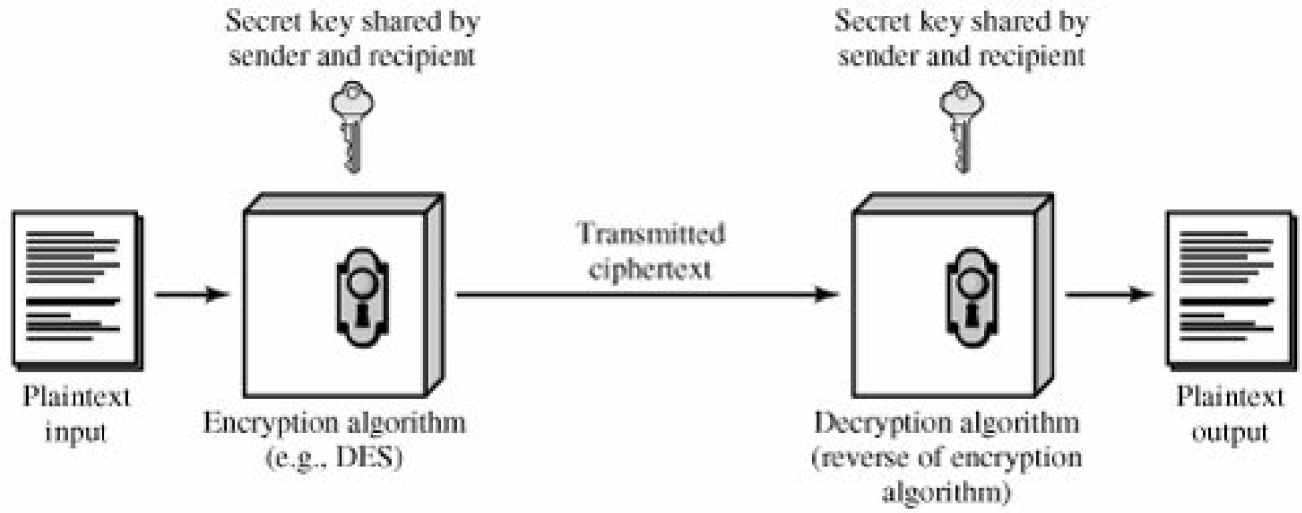
2.1. Symmetric Cipher Model

A symmetric encryption scheme has five ingredients (Figure 2.1):

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

Figure 2.1. Simplified Model of Conventional Encryption

[View full size image]



There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

---

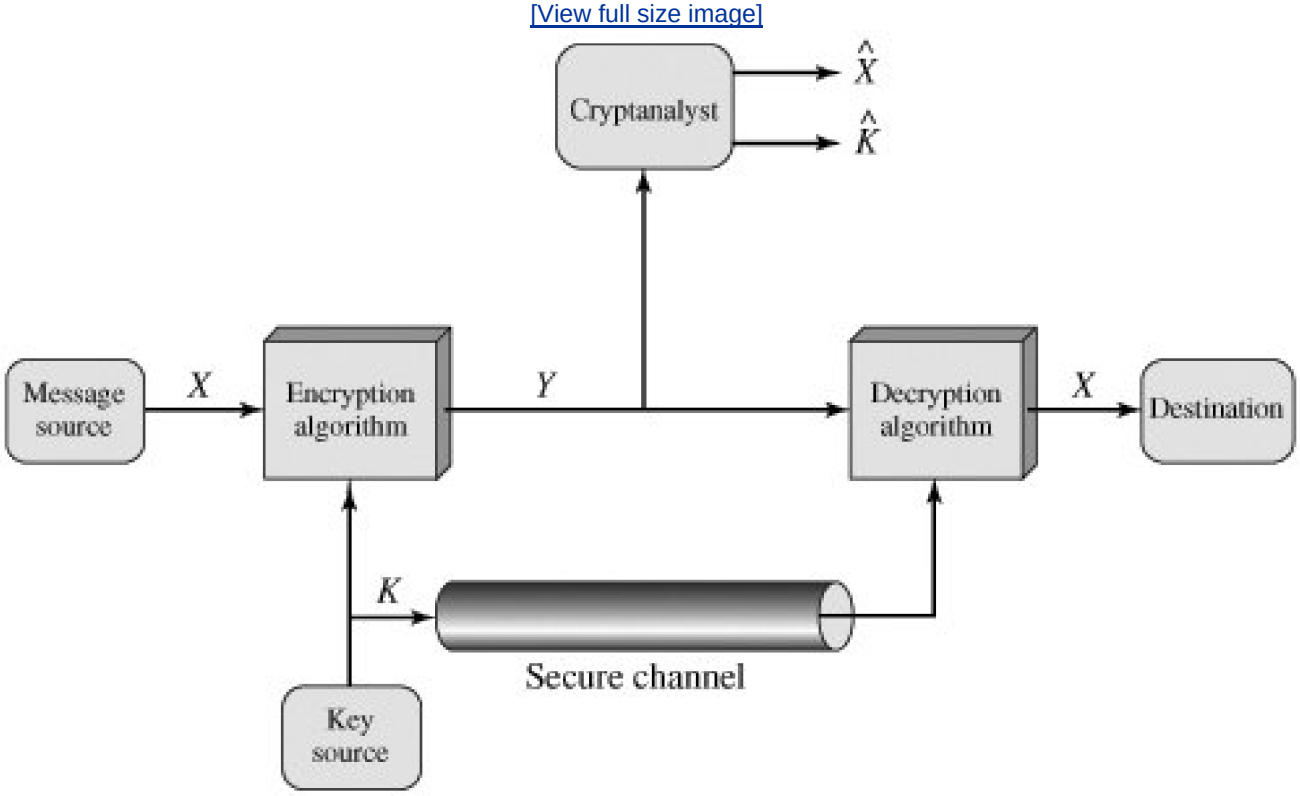
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

We assume that it is impractical to decrypt a message on the basis of the ciphertext *plus* knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 2.2. A source produces a message in plaintext, $X = [X_1, X_2, ..., X_M]$. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet {0, 1} is typically used. For encryption, a key of the form $K = [K_1, K_2, ..., K_J]$ is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

Figure 2.2. Model of Conventional Cryptosystem

[View full size image]



With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext $Y = [Y_1, Y_2, ..., Y_N]$. We can write this

as

$$Y = E(K, X)$$

This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

The intended receiver, in possession of the key, is able to invert the transformation:

$$X = D(K, Y)$$

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate $\hat{X}$. Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate $\hat{K}$.

Cryptography

Cryptographic systems are characterized along three independent dimensions:

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as *product systems*, involve multiple stages of substitutions and transpositions.

2. **The number of keys used.** If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

3. **The way in which the plaintext is processed.** A *block cipher* processes the input one block of elements at a time, producing an output block for each input block. A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along.

Cryptanalysis

Typically, the objective of attacking an encryption system is to recover the key in use rather then simply to recover the plaintext of a single ciphertext. There are two general approaches to attacking a conventional encryption scheme:

- **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

- **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

If either type of attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

We first consider cryptanalysis and then discuss brute-force attacks.

Table 2.1 summarizes the various types of **cryptanalytic attacks**, based on the amount of information known to the cryptanalyst. The most difficult problem is presented when all that is available is the *ciphertext only*. In some cases, not even the encryption algorithm is known, but in general we can assume that the opponent does know the algorithm used for encryption. One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical. Thus, the opponent must rely on an analysis of the ciphertext itself, generally applying various statistical tests to it. To use this approach, the opponent must have some general idea of the type of plaintext that is concealed, such as English or French text, an EXE file, a Java source listing, an accounting file, and so on.

Table 2.1. Types of Attacks on Encrypted Messages

Type of Attack	Known to Cryptanalyst
Ciphertext only	• Encryption algorithm • Ciphertext
Known plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	• Encryption algorithm • Ciphertext • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

The ciphertext-only attack is the easiest to defend against because the opponent has the least amount of information to work with. In many cases, however, the analyst has more information. The analyst may be able to capture one or more plaintext messages as well as their encryptions. Or the analyst may know that certain plaintext patterns will appear in a message. For example, a file that is encoded in the Postscript format always begins with the same pattern, or there may be a standardized header or banner to an electronic funds transfer message, and so on. All these are examples of *known plaintext*. With this knowledge, the analyst may be able to deduce the key on the basis of the way in which the known plaintext is transformed.

Closely related to the known-plaintext attack is what might be referred to as a probable-word attack. If the opponent is working with the

encryption of some general prose message, he or she may have little knowledge of what is in the message. However, if the opponent is after some very specific information, then parts of the message may be known. For example, if an entire accounting file is being transmitted, the opponent may know the placement of certain key words in the header of the file. As another example, the source code for a program developed by Corporation X might include a copyright statement in some standardized position.

If the analyst is able somehow to get the source system to insert into the system a message chosen by the analyst, then a *chosen-plaintext* attack is possible. An example of this strategy is differential cryptanalysis, explored in Chapter 3. In general, if the analyst is able to choose the messages to encrypt, the analyst may deliberately pick patterns that can be expected to reveal the structure of the key.

Table 2.1 lists two other types of attack: chosen ciphertext and chosen text. These are less commonly employed as cryptanalytic techniques but are nevertheless possible avenues of attack.

Only relatively weak algorithms fail to withstand a ciphertext-only attack. Generally, an encryption algorithm is designed to withstand a known-plaintext attack.

Two more definitions are worthy of note. An encryption scheme is **unconditionally secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available. That is, no matter how much time an opponent has, it is impossible for him or her to decrypt the ciphertext, simply because the required information is not there. With the exception of a scheme known as the one-time pad (described later in this chapter), there is no encryption algorithm that is unconditionally secure. Therefore, all that the users of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria:

- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the information.

An encryption scheme is said to be **computationally secure** if either of the foregoing two criteria are met. The rub is that it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully.

All forms of cryptanalysis for symmetric encryption schemes are designed to exploit the fact that traces of structure or pattern in the plaintext may survive encryption and be discernible in the ciphertext. This will become clear as we examine various symmetric encryption schemes in this chapter. We will see in Part Two that cryptanalysis for public-key schemes proceeds from a fundamentally different premise, namely, that the mathematical properties of the pair of keys may make it possible for one of the two keys to be deduced from the other.

A **brute-force attack** involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. Table 2.2 shows how much time is involved for various key spaces. Results are shown for four binary key sizes. The 56-bit key size is used with the DES (Data Encryption Standard) algorithm, and the 168-bit key size is used for triple DES. The minimum key size specified for AES (Advanced Encryption Standard) is 128 bits. Results are also shown for what are called substitution codes that use a 26-character key (discussed later), in which all possible permutations of the 26 characters serve as keys. For each key size, the results are shown assuming that it takes 1 *m*s to perform a single decryption, which is a reasonable order of magnitude for today's machines. With the use of massively parallel organizations of microprocessors, it may be possible to achieve processing rates many orders of magnitude greater. The final column of Table 2.2 considers the results for a system that can process 1 million keys per microsecond. As you can see, at this performance level, DES can no longer be considered computationally secure.

Table 2.2. Average Time Required for Exhaustive Key Search

Key size (bits)	Number of alternative keys	Time required at 1 decryption/*m*s	Time required at 10^6 decryption/*m*s
32	$2^{32} = 4.3 \times 10^9$	2^{31} *m*s = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} *m*s = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} *m*s = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} *m*s = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} *m*s = 6.4×10^{12} years	6.4×10^6 years

PREV NEXT

2.2. Substitution Techniques

In this section and the next, we examine a sampling of what might be called classical encryption techniques. A study of these techniques enables us to illustrate the basic approaches to symmetric encryption used today and the types of cryptanalytic attacks that must be anticipated.

The two basic building blocks of all encryption techniques are substitution and transposition. We examine these in the next two sections. Finally, we discuss a system that combines both substitution and transposition.

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.[1] If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

[1] When letters are involved, the following conventions are used in this book. Plaintext is always in lowercase; ciphertext is in uppercase; key values are in italicized lowercase.

Caesar Cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

plain: meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

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

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter *p*, substitute the ciphertext letter *C*:[2]

[2] We define *a* mod *n* to be the remainder when *a* is divided by *n*. For example, 11 mod 7 = 4. See Chapter 4 for a further discussion of modular arithmetic.

$C = E(3, p) = (p + 3) \bmod 26$

A shift may be of any amount, so that the general Caesar algorithm is

$C = E(k, p) = (p + k) \bmod 26$

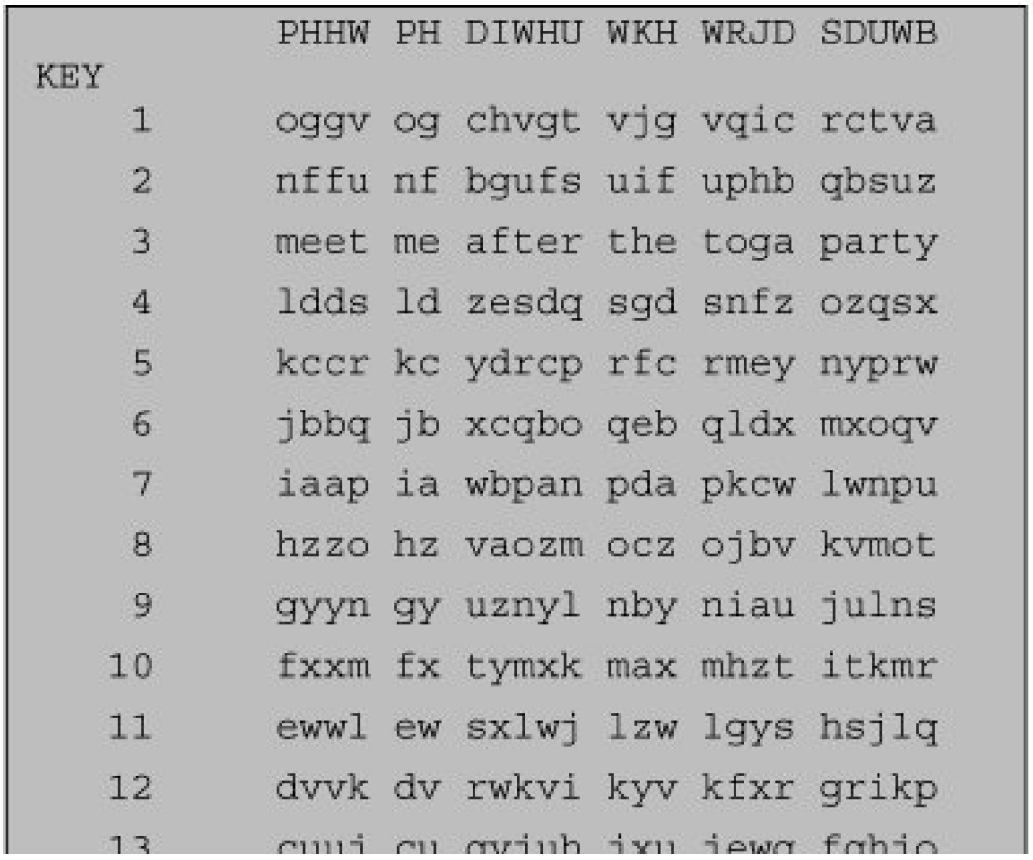
where *k* takes on a value in the range 1 to 25. The decryption algorithm is simply

$p = D(k, C) = (C\ k) \bmod 26$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys. Figure 2.3 shows the results of applying this strategy to the example ciphertext. In this case, the plaintext leaps out as occupying the third line.

Figure 2.3. Brute-Force Cryptanalysis of Caesar Cipher

(This item is displayed on page 37 in the print version)



KEY	
	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhio

14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsgre	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjjy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

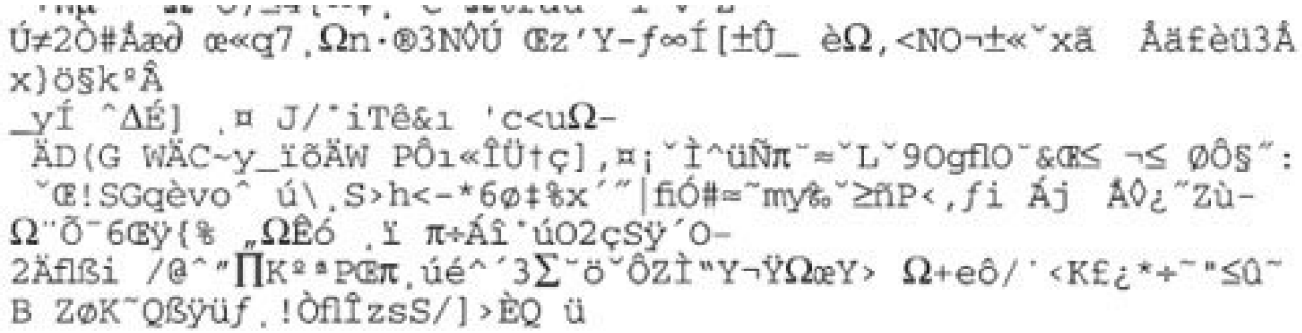
Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

In most networking situations, we can assume that the algorithms are known. What generally makes brute-force cryptanalysis impractical is the use of an algorithm that employs a large number of keys. For example, the triple DES algorithm, examined in Chapter 6, makes use of a 168-bit key, giving a key space of 2^{168} or greater than 3.7×10^{50} possible keys.

The third characteristic is also significant. If the language of the plaintext is unknown, then plaintext output may not be recognizable. Furthermore, the input may be abbreviated or compressed in some fashion, again making recognition difficult. For example, Figure 2.4 shows a portion of a text file compressed using an algorithm called ZIP. If this file is then encrypted with a simple substitution cipher (expanded to include more than just 26 alphabetic characters), then the plaintext may not be recognized when it is uncovered in the brute-force cryptanalysis.

Figure 2.4. Sample of Compressed Text



Monoalphabetic Ciphers

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Recall the assignment for the Caesar cipher:

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

If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4×10^{26} possible keys. This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a **monoalphabetic substitution cipher**, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

There is, however, another line of attack. If the cryptanalyst knows the nature of the plaintext (e.g., noncompressed English text), then the analyst can exploit the regularities of the language. To see how such a cryptanalysis might proceed, we give a partial example here that is adapted from one in [SINK66]. The ciphertext to be solved is

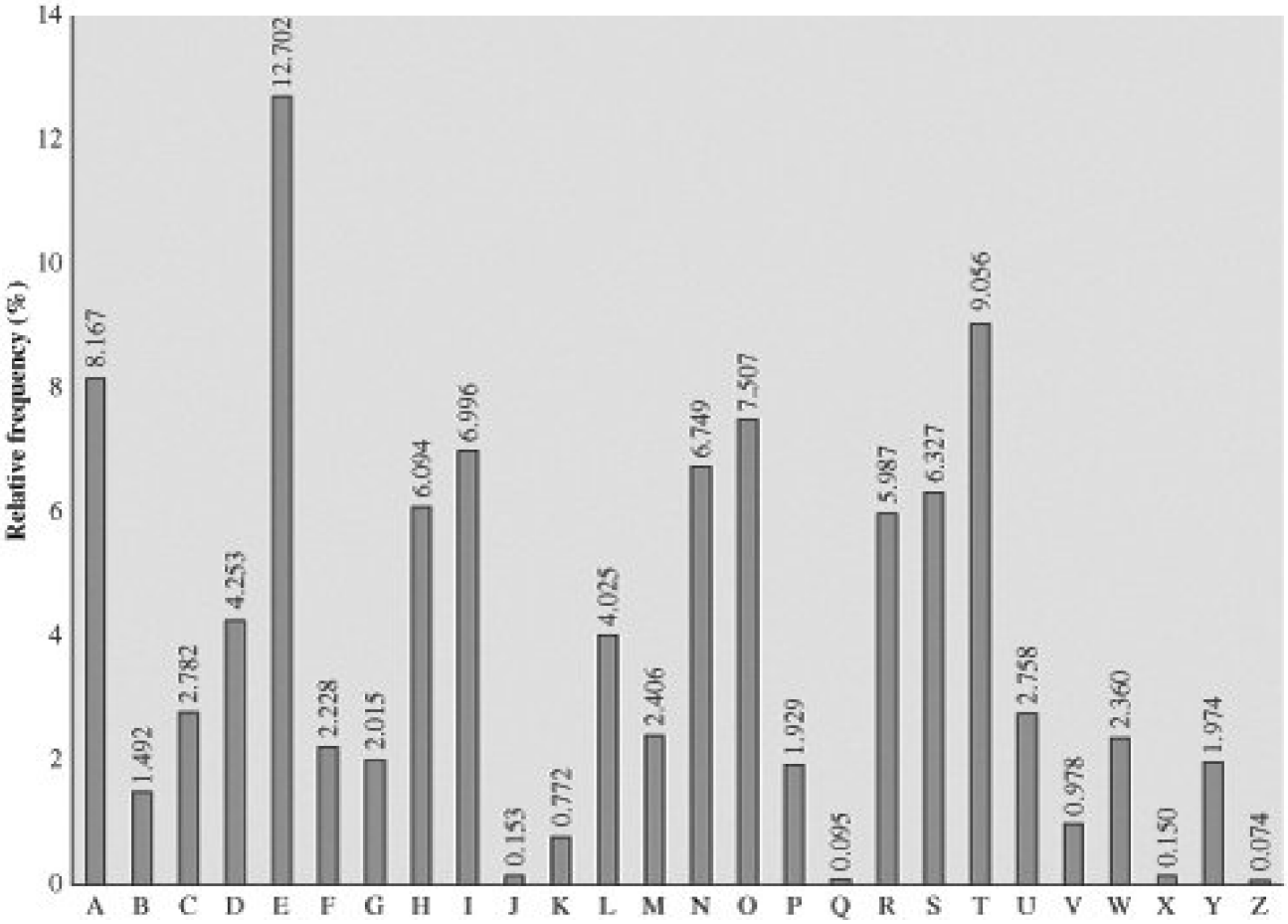
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure 2.5 (based on [LEWA00]). If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. In any case, the relative frequencies of the letters in the ciphertext (in percentages) are as follows:

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

Figure 2.5. Relative Frequency of Letters in English Text

[View full size image]



Comparing this breakdown with Figure 2.5, it seems likely that cipher letters P and Z are the equivalents of plain letters e and t, but it is not certain which is which. The letters S, U, O, M, and H are all of relatively high frequency and probably correspond to plain letters from the set {a, h, i, n, o, r, s}.The letters with the lowest frequencies (namely, A, B, G, Y, I, J) are likely included in the set {b, j, k, q, v, x, z}.

There are a number of ways to proceed at this point. We could make some tentative assignments and start to fill in the plaintext to see if it looks like a reasonable "skeleton" of a message. A more systematic approach is to look for other regularities. For example, certain words may be known to be in the text. Or we could look for repeating sequences of cipher letters and try to deduce their plaintext equivalents.

A powerful tool is to look at the frequency of two-letter combinations, known as digrams. A table similar to Figure 2.5 could be drawn up showing the relative frequency of digrams. The most common such digram is th. In our ciphertext, the most common digram is ZW, which appears three times. So we make the correspondence of Z with t and W with h. Then, by our earlier hypothesis, we can equate P with e. Now notice that the sequence ZWP appears in the ciphertext, and we can translate that sequence as "the." This is the most frequent trigram (three-letter combination) in English, which seems to indicate that we are on the right track.

Next, notice the sequence ZWSZ in the first line. We do not know that these four letters form a complete word, but if they do, it is of the

form th_t. If so, S equates with a.

[Page 40]

So far, then, we have

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
t a e e te a that e e a a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
e t ta t ha e ee a e th t a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
e e e tat e the t

Only four letters have been identified, but already we have quite a bit of the message. Continued analysis of frequencies plus trial and error should easily yield a solution from this point. The complete plaintext, with spaces added between words, follows:

it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow

Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet. A countermeasure is to provide multiple substitutes, known as homophones, for a single letter. For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone used in rotation, or randomly. If the number of symbols assigned to each letter is proportional to the relative frequency of that letter, then single-letter frequency information is completely obliterated. The great mathematician Carl Friedrich Gauss believed that he had devised an unbreakable cipher using homophones. However, even with homophones, each element of plaintext affects only one element of ciphertext, and multiple-letter patterns (e.g., digram frequencies) still survive in the ciphertext, making cryptanalysis relatively straightforward.

Two principal methods are used in substitution ciphers to lessen the extent to which the structure of the plaintext survives in the ciphertext: One approach is to encrypt multiple letters of plaintext, and the other is to use multiple cipher alphabets. We briefly examine each.

Playfair Cipher

The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams.[3]

[3] This cipher was actually invented by British scientist Sir Charles Wheatstone in 1854, but it bears the name of his friend Baron Playfair of St. Andrews, who championed the cipher at the British foreign office.

[Page 41]

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's *Have His Carcase*:[4]

[4] The book provides an absorbing account of a probable-word attack.

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

In this case, the keyword is *monarchy*. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.

4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 x 26 = 676 digrams, so that identification of individual digrams is more difficult. Furthermore, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult. For these reasons, the Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and still enjoyed considerable use by the U.S. Army and other Allied forces during World War II.

Despite this level of confidence in its security, the Playfair cipher is relatively easy to break because it still leaves much of the structure of the plaintext language intact. A few hundred letters of ciphertext are generally sufficient.

One way of revealing the effectiveness of the Playfair and other ciphers is shown in Figure 2.6, based on [SIMM93]. The line labeled *plaintext* plots the frequency distribution of the more than 70,000 alphabetic characters in the *Encyclopaedia Brittanica* article on cryptology.[5] This is also the frequency distribution of any monoalphabetic substitution cipher. The plot was developed in the following way: The number of occurrences of each letter in the text was counted and divided by the number of occurrences of the letter e (the most frequently used letter). As a result, e has a relative frequency of 1, t of about 0.76, and so on. The points on the horizontal axis correspond to the letters in order of decreasing frequency.

[5] I am indebted to Gustavus Simmons for providing the plots and explaining their method of construction.

Figure 2.6. Relative Frequency of Occurrence of Letters

[View full size image]

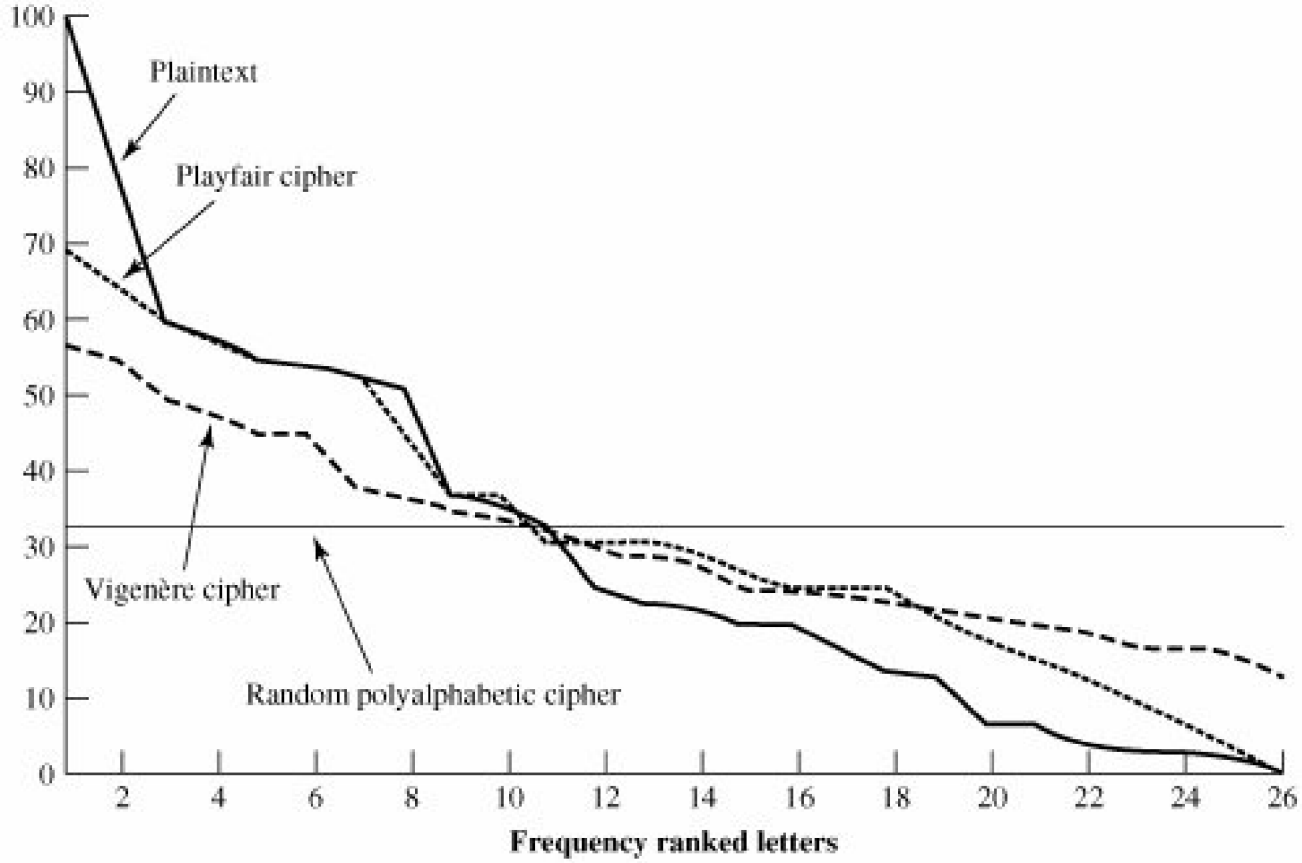


Figure 2.6 also shows the frequency distribution that results when the text is encrypted using the Playfair cipher. To normalize the plot, the number of occurrences of each letter in the ciphertext was again divided by the number of occurrences of e in the plaintext. The resulting plot therefore shows the extent to which the frequency distribution of letters, which makes it trivial to solve substitution ciphers, is masked by encryption. If the frequency distribution information were totally concealed in the encryption process, the ciphertext plot of frequencies would be flat, and cryptanalysis using ciphertext only would be effectively impossible. As the figure shows, the Playfair cipher has a flatter distribution than does plaintext, but nevertheless it reveals plenty of structure for a cryptanalyst to work with.

Hill Cipher[6]

[6] This cipher is somewhat more difficult to understand than the others in this chapter, but it illustrates an important point about cryptanalysis that will be useful later on. This subsection can be skipped on a first reading.

Another interesting multiletter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25). For $m = 3$, the system can be described as follows:

$$c_1 = (k_{11}P_1 + k_{12}P_2 + k_{13}P_3) \bmod 26$$

$$c_2 = (k_{21}P_1 + k_{22}P_2 + k_{23}P_3) \bmod 26$$

$c_3 = (k_{31}P1 + k_{32}P2 + k_{33}P3) \bmod 26$

This can be expressed in term of column vectors and matrices:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

or

$$\mathbf{C} = \mathbf{KP} \bmod 26$$

where **C** and **P** are column vectors of length 3, representing the plaintext and ciphertext, and **K** is a 3 x 3 matrix, representing the encryption key. Operations are performed mod 26.

For example, consider the plaintext "paymoremoney" and use the encryption key

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the plaintext are represented by the vector

$$\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}. \text{Then } \mathbf{K}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} = \text{LNS. Continuing in this fashion,}$$

the ciphertext for the entire plaintext is LNSHDLEWMTRW.

Decryption requires using the inverse of the matrix **K**. The inverse $\mathbf{K}^{-1}$ of a matrix **K** is defined by the equation $\mathbf{K}\mathbf{K}^{-1} = \mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$, where **I** is the matrix that is all zeros except for ones along the main diagonal from upper left to lower right. The inverse of a matrix does not always exist, but when it does, it satisfies the preceding equation. In this case, the inverse is:

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as follows:

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \end{pmatrix}\begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 24 & 0 & 17 \end{pmatrix} \quad \begin{pmatrix} 494 & 52 & 365 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

It is easily seen that if the matrix $\mathbf{K}^{-1}$ is applied to the ciphertext, then the plaintext is recovered. To explain how the inverse of a matrix is determined, we make an exceedingly brief excursion into linear algebra.[7] For any square matrix (m x m) the **determinant** equals the sum of all the products that can be formed by taking exactly one element from each row and exactly one element from each column, with certain of the product terms preceded by a minus sign. For a 2 x 2 matrix

[7] The basic concepts of linear algebra are summarized in the Math Refresher document at the Computer Science Student Resource site at WilliamStallings.com/StudentSupport.html. The interested reader may consult any text on linear algebra for greater detail.

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

the determinant is $k_{11}k_{22}$ $k_{12}k_{21}$. For a 3 x 3 matrix, the value of the determinant is $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23}$ $k_{31}k_{22}k_{13}$ $k_{21}k_{12}k_{33}$ $k_{11}k_{32}k_{23}$. If a square matrix **A** has a nonzero determinant, then the inverse of the matrix is computed as $[\mathbf{A}^{-1}]_{ij} = (1)^{i+j}(D_{ij})/\text{ded}(\mathbf{A})$, where (D_{ij}) is the subdeterminant formed by deleting the ith row and the jth column of **A** and det(**A**) is the determinant of **A**. For our purposes, all arithmetic is done mod 26.

In general terms, the Hill system can be expressed as follows:

$\mathbf{C} = E(\mathbf{K}, \mathbf{P}) = \mathbf{KP} \bmod 26$

$\mathbf{P} = D(\mathbf{K}, \mathbf{P}) = \mathbf{K}^{-1}\mathbf{C} \bmod 26 = \mathbf{K}^{-1}\mathbf{KP} = \mathbf{P}$

As with Playfair, the strength of the Hill cipher is that it completely hides single-letter frequencies. Indeed, with Hill, the use of a larger matrix hides more frequency information. Thus a 3 x 3 Hill cipher hides not only single-letter but also two-letter frequency information.

Although the Hill cipher is strong against a ciphertext-only attack, it is easily broken with a known plaintext attack. For an m x m Hill cipher, suppose we have m plaintext-ciphertext pairs, each of length m. We label the pairs

$$\mathbf{P}_j = \begin{pmatrix} p_{1j} \\ p_{2j} \\ \vdots \\ p_{mj} \end{pmatrix} \text{ and } \mathbf{C}_j = \begin{pmatrix} c_{1j} \\ c_{2j} \\ \vdots \\ c_{mj} \end{pmatrix} \text{ such that } \mathbf{C}_j = \mathbf{KP}_j \text{ for } 1 \le j \le m \text{ and for some}$$

unknown key matrix **K**. Now define two m x m matrices $\mathbf{X} = (P_{ij})$ and $\mathbf{Y} = (C_{ij})$. Then we can form the matrix equation $\mathbf{Y} = \mathbf{KX}$. If **X** has an inverse, then we can determine $\mathbf{K} = \mathbf{YX}^{-1}$. If **X** is not invertible, then a new version of **X** can be formed with additional plaintext-ciphertext pairs until an invertible **X** is obtained.

We use an example based on one in [STIN02]. Suppose that the plaintext "friday" is encrypted using a 2 x 2 Hill cipher to yield the ciphertext PQCFKU. Thus, we know that

$$\mathbf{K}\begin{pmatrix}5\\17\end{pmatrix} \bmod 26 = \begin{pmatrix}15\\16\end{pmatrix}; \mathbf{K}\begin{pmatrix}8\\3\end{pmatrix} \bmod 26 = \begin{pmatrix}2\\5\end{pmatrix}; \quad \text{and} \quad \mathbf{K}\begin{pmatrix}0\\24\end{pmatrix} \bmod 26 = \begin{pmatrix}10\\20\end{pmatrix}$$

Using the first two plaintext-ciphertext pairs, we have

$$\begin{pmatrix}15 & 2\\16 & 5\end{pmatrix} = \mathbf{K}\begin{pmatrix}5 & 8\\17 & 3\end{pmatrix} \bmod 26$$

The inverse of **X** can be computed:

$$\begin{pmatrix}5 & 8\\17 & 3\end{pmatrix}^{-1} = \begin{pmatrix}9 & 2\\1 & 15\end{pmatrix}$$

so

$$\mathbf{K} = \begin{pmatrix}15 & 2\\16 & 5\end{pmatrix}\begin{pmatrix}9 & 2\\1 & 15\end{pmatrix} = \begin{pmatrix}137 & 60\\149 & 107\end{pmatrix} \bmod 26 = \begin{pmatrix}7 & 8\\19 & 3\end{pmatrix}$$

This result is verified by testing the remaining plaintext-ciphertext pair.

Polyalphabetic Ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**. All these techniques have the following features in common:

1. A set of related monoalphabetic substitution rules is used.
2. A key determines which particular rule is chosen for a given transformation.

The best known, and one of the simplest, such algorithm is referred to as the Vigenère cipher. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter a. Thus, a Caesar cipher with a shift of 3 is denoted by the key value *d*.

To aid in understanding the scheme and to aid in its use, a matrix known as the Vigenère tableau is constructed (Table 2.3). Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple: Given a key letter *x* and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled *x* and the column labeled y; in this case the ciphertext is V.

Table 2.3. The Modern Vigenère Tableau

(This item is displayed on page 46 in the print version)

[View full size image]

Key	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
a	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
b	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	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	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

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is *deceptive*, the message "we are discovered save yourself" is encrypted as follows:

key: *deceptivedeceptivedeceptive*
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column.

The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured. However, not all knowledge of the plaintext structure is lost. For example, Figure 2.6 shows the frequency distribution for a Vigenère cipher with a keyword of length 9. An improvement is achieved over the Playfair cipher, but considerable frequency information remains.

It is instructive to sketch a method of breaking this cipher, because the method reveals some of the mathematical principles that apply in cryptanalysis.

First, suppose that the opponent believes that the ciphertext was encrypted using either monoalphabetic substitution or a Vigenère cipher. A simple test can be made to make a determination. If a monoalphabetic substitution is used, then the statistical properties of the ciphertext should be the same as that of the language of the plaintext. Thus, referring to Figure 2.5, there should be one cipher letter with a relative frequency of occurrence of about 12.7%, one with about 9.06%, and so on. If only a single message is available for analysis, we would not expect an exact match of this small sample with the statistical profile of the plaintext language. Nevertheless, if the correspondence is close, we can assume a monoalphabetic substitution.

If, on the other hand, a Vigenère cipher is suspected, then progress depends on determining the length of the keyword, as will be seen in a moment. For now, let us concentrate on how the keyword length can be determined. The important insight that leads to a solution is the following: If two identical sequences of plaintext letters occur at a distance that is an integer multiple of the keyword length, they will generate identical ciphertext sequences. In the foregoing example, two instances of the sequence "red" are separated by nine character positions. Consequently, in both cases, r is encrypted using key letter *e*, e is encrypted using key letter *p*, and d is encrypted using key letter *t*. Thus, in both cases the ciphertext sequence is VTW.

An analyst looking at only the ciphertext would detect the repeated sequences VTW at a displacement of 9 and make the assumption that the keyword is either three or nine letters in length. The appearance of VTW twice could be by chance and not reflect identical plaintext letters encrypted with identical key letters. However, if the message is long enough, there will be a number of such repeated ciphertext sequences. By looking for common factors in the displacements of the various sequences, the analyst should be able to make a good guess of the keyword length.

Solution of the cipher now depends on an important insight. If the keyword length is *N*, then the cipher, in effect, consists of *N* monoalphabetic substitution ciphers. For example, with the keyword DECEPTIVE, the letters in positions 1, 10, 19, and so on are all encrypted with the same monoalphabetic cipher. Thus, we can use the known frequency characteristics of the plaintext language to attack each of the monoalphabetic ciphers separately.

The periodic nature of the keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself. Vigenère proposed what is referred to as an **autokey system**, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

key: *deceptivewearediscoveredsav*
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

Even this scheme is vulnerable to cryptanalysis. Because the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied. For example, e enciphered by *e*, by Figure 2.5, can be expected to occur with a frequency of $(0.127)^2 \approx 0.016$, whereas t enciphered by *t* would occur only about half as often. These regularities can be exploited to achieve successful cryptanalysis.[8]

[8] Although the techniques for breaking a Vigenère cipher are by no means complex, a 1917 issue of *Scientific American* characterized this system as "impossible of translation." This is a point worth remembering when similar claims are made for modern algorithms.

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918. His system works on binary data rather than letters. The system can be expressed succinctly as follows:

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext

k_i = ith binary digit of key

c_i = ith binary digit of ciphertext

$\oplus$ = exclusive-or (XOR) operation

Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key. Because of the properties of the XOR, decryption simply involves the same bitwise operation:

$$p_i = c_i \oplus k_i$$

The essence of this technique is the means of construction of the key. Vernam proposed the use of a running loop of tape that eventually repeated the key, so that in fact the system worked with a very long but repeating keyword. Although such a scheme, with a long key, presents formidable cryptanalytic difficulties, it can be broken with sufficient ciphertext, the use of known or probable plaintext sequences, or both.

One-Time Pad

An Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is unbreakable. It produces random output that bears no statistical relationship to the plaintext. Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.

An example should illustrate our point. Suppose that we are using a Vigenère scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message. Thus, the tableau of Table 2.3 must be expanded to 27 x 27. Consider the ciphertext

ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

We now show two different decryptions using two different keys:

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key: *pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih*
plaintext: mr mustard with the candlestick in the hall

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key: *mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt*
plaintext: miss scarlet with the knife in the library

Suppose that a cryptanalyst had managed to find these two keys. Two plausible plaintexts are produced. How is the cryptanalyst to decide which is the correct decryption (i.e., which is the correct key)? If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and therefore which plaintext is correct.

In fact, given any plaintext of equal length to the ciphertext, there is a key that produces that plaintext. Therefore, if you did an exhaustive search of all possible keys, you would end up with many legible plaintexts, with no way of knowing which was the intended plaintext. Therefore, the code is unbreakable.

The security of the one-time pad is entirely due to the randomness of the key. If the stream of characters that constitute the key is truly random, then the stream of characters that constitute the ciphertext will be truly random. Thus, there are no patterns or regularities that a cryptanalyst can use to attack the ciphertext.

In theory, we need look no further for a cipher. The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.

2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

Because of these difficulties, the one-time pad is of limited utility, and is useful primarily for low-bandwidth channels requiring very high security.

PREV NEXT

[Page 49 (continued)]

2.3. Transposition Techniques

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following:

[Page 50]

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

```
Key:        4 3 1 2 5 6 7
Plaintext:   a t t a c k p
             o s t p o n e
             d u n t i l t
             w o a m x y z
Ciphertext:  TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the ciphertext in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm,

```
Key:        4 3 1 2 5 6 7
Input:      t t n a a p t
            m t s u o a o
            d w c o i x k
            n l y p e t z
Output:     NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

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 26 27 28

After the first transposition we have

03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28

which has a somewhat regular structure. But after the second transposition, we have

17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28

This is a much less structured permutation and is much more difficult to cryptanalyze.

PREV NEXT

2.4. Rotor Machines

The example just given suggests that multiple stages of encryption can produce an algorithm that is significantly more difficult to cryptanalyze. This is as true of substitution ciphers as it is of transposition ciphers. Before the introduction of DES, the most important application of the principle of multiple stages of encryption was a class of systems known as rotor machines.[9]

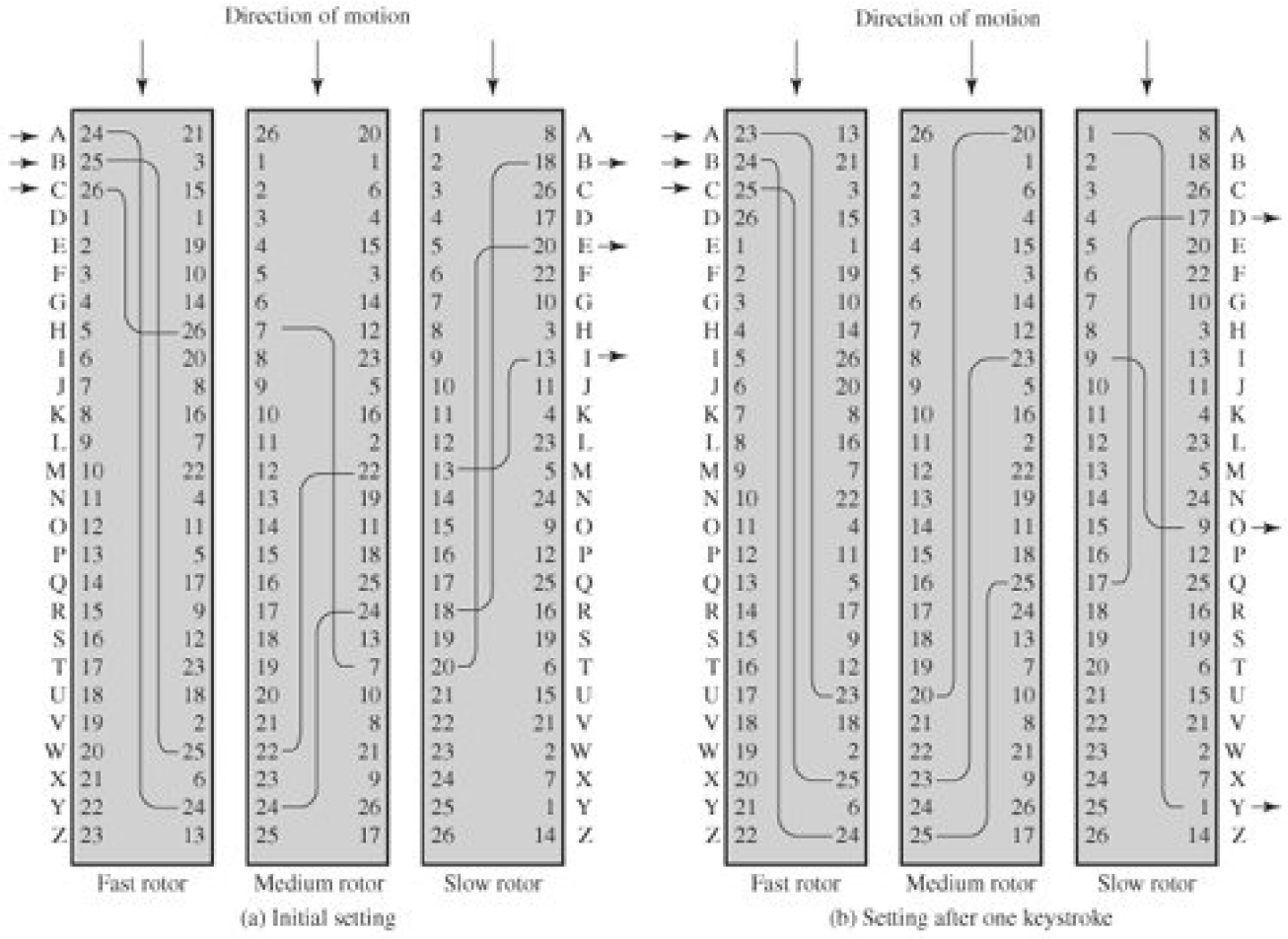
[9] Machines based on the rotor principle were used by both Germany (Enigma) and Japan (Purple) in World War II. The breaking of both codes by the Allies was a significant factor in the war's outcome.

The basic principle of the rotor machine is illustrated in Figure 2.7. The machine consists of a set of independently rotating cylinders through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin. For simplicity, only three of the internal connections in each cylinder are shown.

Figure 2.7. Three-Rotor Machine with Wiring Represented by Numbered Contacts

(This item is displayed on page 52 in the print version)

[View full size image]



If we associate each input and output pin with a letter of the alphabet, then a single cylinder defines a monoalphabetic substitution. For example, in Figure 2.7, if an operator depresses the key for the letter A, an electric signal is applied to the first pin of the first cylinder and flows through the internal connection to the twenty-fifth output pin.

Consider a machine with a single cylinder. After each input key is depressed, the cylinder rotates one position, so that the internal connections are shifted accordingly. Thus, a different monoalphabetic substitution cipher is defined. After 26 letters of plaintext, the cylinder would be back to the initial position. Thus, we have a polyalphabetic substitution algorithm with a period of 26.

A single-cylinder system is trivial and does not present a formidable cryptanalytic task. The power of the rotor machine is in the use of multiple cylinders, in which the output pins of one cylinder are connected to the input pins of the next. Figure 2.7 shows a three-cylinder system. The left half of the figure shows a position in which the input from the operator to the first pin (plaintext letter a) is routed through the three cylinders to appear at the output of the second pin (ciphertext letter B).

With multiple cylinders, the one closest to the operator input rotates one pin position with each keystroke. The right half of Figure 2.7 shows the system's configuration after a single keystroke. For every complete rotation of the inner cylinder, the middle cylinder rotates one pin position. Finally, for every complete rotation of the middle cylinder, the outer cylinder rotates one pin position. This is the same type of operation seen with an odometer. The result is that there are 26 x 26 x 26 = 17,576 different substitution alphabets used before the system repeats. The addition of fourth and fifth rotors results in periods of 456,976 and 11,881,376 letters, respectively. As David Kahn eloquently put it, referring to a five-rotor machine [KAHN96, page 413]:

> A period of that length thwarts any practical possibility of a straightforward solution on the basis of letter frequency. This general solution would need about 50 letters per cipher alphabet, meaning that all five rotors would have to go through their combined cycle 50 times. The ciphertext would have to be as long as all the speeches made on the floor of the Senate and the House of Representatives in three successive sessions of Congress. No cryptanalyst is likely to bag that kind of trophy in his lifetime; even diplomats, who can be as verbose as politicians, rarely scale those heights of loquacity.

The significance of the rotor machine today is that it points the way to the most widely used cipher ever: the Data Encryption Standard (DES). This we examine in Chapter 3.

2.5. Steganography

We conclude with a discussion of a technique that is, strictly speaking, not encryption, namely, steganography.

A plaintext message may be hidden in one of two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.[10]

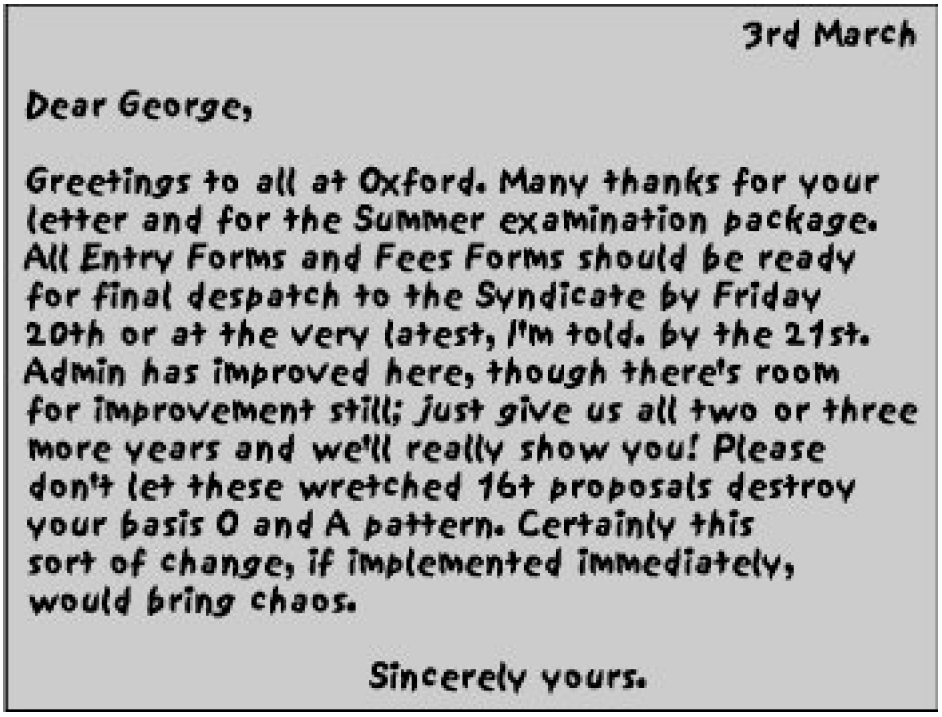
[10] *Steganography* was an obsolete word that was revived by David Kahn and given the meaning it has today [KAHN96].

A simple form of steganography, but one that is time-consuming to construct, is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. For example, the sequence of first letters of each word of the overall message spells out the hidden message. Figure 2.8 shows an example in which a subset of the words of the overall message is used to convey the hidden message.

Figure 2.8. A Puzzle for Inspector Morse

(This item is displayed on page 54 in the print version)

(From* The Silent World of Nicholas Quinn,*by Colin Dexter)



Various other techniques have been used historically; some examples are the following [MYER91]:

- **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.
- **Invisible ink:** A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

- **Pin punctures:** Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.
- **Typewriter correction ribbon:** Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Although these techniques may seem archaic, they have contemporary equivalents. [WAYN93] proposes hiding a message by using the least significant bits of frames on a CD. For example, the Kodak Photo CD format's maximum resolution is 2048 by 3072 pixels, with each pixel containing 24 bits of RGB color information. The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image. The result is that you can hide a 2.3-megabyte message in a single digital snapshot. There are now a number of software packages available that take this type of approach to steganography.

Steganography has a number of drawbacks when compared to encryption. It requires a lot of overhead to hide a relatively few bits of information, although using some scheme like that proposed in the preceding paragraph may make it more effective. Also, once the system is discovered, it becomes virtually worthless. This problem, too, can be overcome if the insertion method depends on some sort of key (e.g., see Problem 2.11). Alternatively, a message can be first encrypted and then hidden using steganography.

The advantage of steganography is that it can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered. Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide.

2.6. Recommended Reading and Web Sites

For anyone interested in the history of code making and code breaking, the book to read [KAHN96]. Although it is concerned more with the impact of cryptology than its technical development, it is an excellent introduction and makes for exciting reading. Another excellent historical account is [SING99].

A short treatment covering the techniques of this chapter, and more, is [GARD72]. There are many books that cover classical cryptography in a more technical vein; one of the best is [SINK66]. [KORN96] is a delightful book to read and contains a lengthy section on classical techniques. Two cryptography books that contain a fair amount of technical material on classical techniques are [GARR01] and [NICH99]. For the truly interested reader, the two-volume [NICH96] covers numerous classical ciphers in detail and provides many ciphertexts to be cryptanalyzed, together with the solutions.

An excellent treatment of rotor machines, including a discussion of their cryptanalysis is found in [KUMA97].

[KATZ00] provides a thorough treatment of steganography. Another good source is [WAYN96].

GARD72 Gardner, M. *Codes, Ciphers, and Secret Writing.* New York: Dover, 1972.

GARR01 Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.

KAHN96 Kahn, D. *The Codebreakers: The Story of Secret Writing.* New York: Scribner, 1996.

KATZ00 Katzenbeisser, S., ed. *Information Hiding Techniques for Steganography and Digital Watermarking.* Boston: Artech House, 2000.

KORN96 Korner, T. *The Pleasures of Counting.* Cambridge, England: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.

NICH96 Nichols, R. *Classical Cryptography Course.* Laguna Hills, CA: Aegean Park Press, 1996.

NICH99 Nichols, R. ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.

SING99 Singh, S. *:The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* New York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary Cryptanalysis: A Mathematical Approach*. Washington, DC: The Mathematical Association of America, 1966.

WAYN96 Wayner, P. *Disappearing Cryptography.* Boston: AP Professional Books, 1996.

Recommended Web Sites

- **American Cryptogram Association:** An association of amateur cryptographers. The Web site includes information and links to sites concerned with classical cryptography.

- **Crypto Corner:** Simon Singh's Web site. Lots of good information, plus interactive tools for learning about cryptography.
- **Steganography:** Good collection of links and documents.

2.7. Key Terms, Review Questions, and Problems

Key Terms

block cipher

brute-force attack

Caesar cipher

cipher

ciphertext

computationally secure

conventional encryption

cryptanalysis

cryptographic system

cryptography

cryptology

deciphering

decryption

enciphering

encryption

Hill cipher

monoalphabetic cipher

one-time pad

plaintext

Playfair cipher

polyalphabetic cipher

rail fence cipher

single-key encryption

steganography

stream cipher

symmetric encryption

transposition cipher

unconditionally secure

Vigenère cipher

Review Questions

2.1 What are the essential ingredients of a symmetric cipher?

2.2 What are the two basic functions used in encryption algorithms?

2.3 How many keys are required for two people to communicate via a cipher?

2.4 What is the difference between a block cipher and a stream cipher?

2.5 What are the two general approaches to attacking a cipher?

2.6 List and briefly define types of cryptanalytic attacks based on what is known to the attacker.

2.7 What is the difference between an unconditionally secure cipher and a computationally secure cipher?

2.8 Briefly define the Caesar cipher.

2.9 Briefly define the monoalphabetic cipher.

2.10 Briefly define the Playfair cipher.

2.11 What is the difference between a monoalphabetic cipher and a polyalphabetic cipher?

2.12 What are two problems with the one-time pad?

2.13 What is a transposition cipher?

2.14 What is steganography?

Problems

2.1 A generalization of the Caesar cipher, knows as the affine Caesar cipher, has the following form: For each plaintext letter *p*, substitute the ciphertext letter *C*:

$C = E([a, b], p) = (ap + b) \bmod 26$

A basic requirement of any encryption algorithm is that it be one-to-one. That is, if p $\neq$ q, then E(*k*, *p*) $\neq$ E(*k*, *q*). Otherwise, decryption is impossible, because more than one plaintext character maps into the same ciphertext character. The affine Caesar cipher is not one-to-one for all values of *a*. For example, for *a* = 2 and *b* = 3, then E([*a*, *b*], 0) = E([*a*, *b*], 13) = 3.

a. Are there any limitations on the value of *b*? Explain why or why not.

b. Determine which values of *a* are not allowed.

c. Provide a general statement of which values of *a* are and are not allowed. Justify your statement.

2.2 How many one-to-one affine Caesar ciphers are there?

2.3 A ciphertext has been generated with an affine cipher. The most frequent letter of the ciphertext is 'B', and the second most frequent letter of the ciphertext is 'U'. Break this code.

2.4 The following ciphertext was generated using a simple substitution algorithm:

53‡‡† 305))6*;4826)4‡.)4‡);806*;48† 8¶60))85;;]8*;:‡*8† 83

(88)5*† ;46(;88*96*?;8)*‡(;485);5*† 2:*‡(;4956*2(5*-4)88*

;4069285);)6† 8)4‡‡[ddagger];1(‡9;48081;8:8‡1;48† 85;4)485† 528806*81

(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;

Decrypt this message. *Hints:*

1. As you know, the most frequently occurring letter in English is e. Therefore, the first or second (or perhaps third?) most common character in the message is likely to stand for e. Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a character in the ciphertext that decodes to e.

2. The most common word in English is "the." Use this fact to guess the characters that stand for t

and h.

3. Decipher the rest of the message by deducing additional words.

Warning: The resulting message is in English but may not make much sense on a first reading.

2.5 One way to solve the key distribution problem is to use a line from a book that both the sender and the receiver possess. Typically, at least in spy novels, the first sentence of a book serves as the key. The particular scheme discussed in this problem is from one of the best suspense novels involving secret codes, *Talking to Strange Men*, by Ruth Rendell. Work this problem without consulting that book!

Consider the following message:

SIDKHKDM AF HCRKIABIE SHIMC KD LFEAILA

This ciphertext was produced using the first sentence of *The Other Side of Silence* (a book about the spy Kim Philby):

The snow lay thick on the steps and the snowflakes driven by the wind looked black in the headlights of the cars.

A simple substitution cipher was used.

a. What is the encryption algorithm?

b. How secure is it?

c. To make the key distribution problem simple, both parties can agree to use the first or last sentence of a book as the key. To change the key, they simply need to agree on a new book. The use of the first sentence would be preferable to the use of the last. Why?

2.6 In one of his cases, Sherlock Holmes was confronted with the following message.

534 C2 13 127 36 31 4 17 21 41

DOUGLAS 109 293 5 37 BIRLSTONE

26 BIRLSTONE 9 127 171

Although Watson was puzzled, Holmes was able immediately to deduce the type of cipher. Can you?

2.7 This problem uses a real-world example, from an old U.S. Special Forces manual (public domain). A copy is available at **ftp://shell.shore.net/members/w/s/ws/Support/Crypto/FM-31-4.pdf**

a. Using the two keys (memory words) *cryptographic* and *network security*, encrypt the following message:

Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful

bring two friends.

Make reasonable assumptions about how to treat redundant letters and excess letters in the memory words and how to treat spaces and punctuation. Indicate what your assumptions are. *Note:* The message is from the Sherlock Holmes novel,*The Sign of Four*.

b. Decrypt the ciphertext. Show your work.

c. Comment on when it would be appropriate to use this technique and what its advantages are.

2.8 A disadvantage of the general monoalphabetic cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A common technique for avoiding this is to use a keyword from which the cipher sequence can be generated. For example, using the keyword *CIPHER*, write out the keyword followed by unused letters in normal order and match this against the plaintext letters:

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

If it is felt that this process does not produce sufficient mixing, write the remaining letters on successive lines and then generate the sequence by reading down the columns:

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

This yields the sequence

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

Such a system is used in the example in Section 2.2 (the one that begins "it was disclosed yesterday"). Determine the keyword.

2.9 When the PT-109 American patrol boat, under the command of Lieutenant John F. Kennedy, was sunk by a Japanese destroyer, a message was received at an Australian wireless station in Playfair code:

KXJEY UREBE ZWEHE WRYTU HEYFS
KREHE GOYFI WTTTU OLKSY CAJPO
BOTEI ZONTX BYBNT GONEY CUZWR
GDSON SXBOU YWRHE BAAHY USEDQ

The key used was*royal new zealand navy*. Decrypt the message. Translate TT into tt.

2.10

a. Construct a Playfair matrix with the key*largest*.

b. Construct a Playfair matrix with the key *occurrence*. Make a reasonable assumption about how to treat redundant letters in the key.

2.11

a. Using this Playfair matrix

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

---

encrypt this message:

Must see you over Cadogan West. Coming at once.

Note: The message is from the Sherlock Holmes story,*The Adventure of the Bruce-Partington Plans*.

b. Repeat part (a) using the Playfair matrix from Problem 2.10a.

c. How do you account for the results of this problem? Can you generalize your conclusion?

2.12

a. How many possible keys does the Playfair cipher have? Ignore the fact that some keys might produce identical encryption results. Express your answer as an approximate power of 2.

b. Now take into account the fact that some Playfair keys produce the same encryption results. How many effectively unique keys does the Playfair cipher have?

2.13 What substitution system results when we use a 25 x 1 Playfair matrix?

2.14

a. Decipher the message YITJP GWJOW FAQTQ XCSMA ETSQU SQAPU SQGKC PQTYJ using the Hill cipher with the inverse key $\begin{pmatrix} 5 & 1 \\ 2 & 7 \end{pmatrix}$. Show your calculations and the result.

b. Decipher the message MWALO LIAIW WTGBH JNTAK QZJKA ADAWS SKQKU AYARN CSODN IIAES OQKJY B using the Hill cipher with the inverse key $\begin{pmatrix} 2 & 23 \\ 21 & 7 \end{pmatrix}$. Show your calculations and the result.

2.15

a. Encrypt the message "meet me at the usual place at ten rather than eight oclock" using the Hill cipher with the key $\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$. Show your calculations and the result.

b. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext.

2.16 We have shown that the Hill cipher succumbs to a known plaintext attack if sufficient plaintext-ciphertext pairs are provided. It is even easier to solve the Hill cipher if a chosen plaintext attack can be mounted. Describe such an attack.

2.17 It can be shown that the Hill cipher with the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ requires that (ad bc) is relatively prime to 26; that is the only common positive factor of (ad bc) and 26 is 1. Thus, if (ad bc) = 13 or is even, the matrix is not allowed. Determine the number of different (good) keys there are for a 2 x 2 Hill cipher without counting them one by one, using the following steps:

a. Find the number of matrices whose determinant is even because one or both rows are even. (A row is "even" if both entries in the row are even.)

b. Find the number of matrices whose determinant is even because one or both columns are even. (A column is "even" if both entries in the column are even.)

c. Find the number of matrices whose determinant is even because all of the entries are odd.

d. Taking into account overlaps, find the total number of matrices whose determinant is even.

e. Find the number of matrices whose determinant is a multiple of 13 because the first column is a multiple of 13.

f. Find the number of matrices whose determinant is a multiple of 13 where the first column is not a multiple of 13 but the second column is a multiple of the first modulo 13.

g. Find the total number of matrices whose determinant is a multiple of 13.

h. Find the number of matrices whose determinant is a multiple of 26 because they fit case (a) and (e). (b) and (e). (c) and (e). (a) and (f). And so on ...

i. Find the total number of matrices whose determinant is neither a multiple of 2 nor a multiple of 13.

2.18 Using the Vigenère cipher, encrypt the word "explanation" using the key *leg*.

2.19 This problem explores the use of a one-time pad version of the Vigenère cipher. In this scheme, the key is a stream of random numbers between 0 and 26. For example, if the key is 3 19 5 ..., then the first letter of plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, the third with a shift of 5 letters, and so on.

a. Encrypt the plaintext sendmoremoney with the key stream 9 0 1 7 23 15 21 14 11 11 2 8 9.

b. Using the ciphertext produced in part a, find a key so that the cipher text decrypts to the plaintext

cashnotneeded.

2.20 What is the message embedded in Figure 2.8?

2.21 In one of Dorothy Sayers's mysteries, Lord Peter is confronted with the message shown in Figure 2.9. He also discovers the key to the message, which is a sequence of integers:

787656543432112343456567878878765654

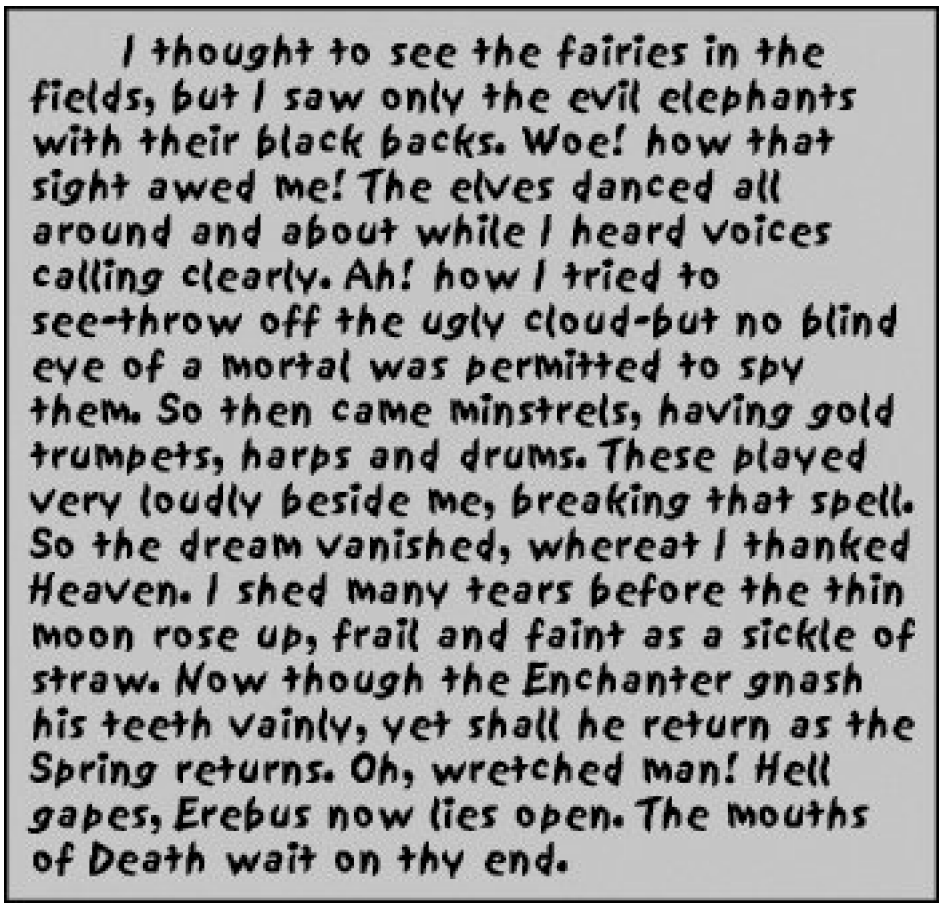
343211234345656787887876565443321 1234

a. Decrypt the message. *Hint:* What is the largest integer value?

b. If the algorithm is known but not the key, how secure is the scheme?

c. If the key is known but not the algorithm, how secure is the scheme?

Figure 2.9. A Puzzle for Lord Peter



Programming Problems

2.22 Write a program that can encrypt and decrypt using the general Caesar cipher, also known as an additive cipher.

2.23 Write a program that can encrypt and decrypt using the affine cipher described in Problem 2.1.

2.24 Write a program that can perform a letter frequency attack on an additive cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify "give me the top 10 possible plaintexts".

2.25 Write a program that can perform a letter frequency attack on any monoalphabetic substitution cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify "give me the top 10 possible plaintexts".

2.26 Create software that can encrypt and decrypt using a 2 x 2 Hill cipher.

2.27 Create software that can perform a fast known plaintext attack on a Hill cipher, given the dimension m. How fast are your algorithms, as a function of m?

Chapter 3. Block Ciphers and the Data Encryption Standard

All the afternoon Mungo had been working on Stern's code, principally with the aid of the latest messages which he had copied down at the Nevin Square drop. Stern was very confident. He must be well aware London Central knew about that drop. It was obvious that they didn't care how often Mungo read their messages, so confident were they in the impenetrability of the code.

Talking to Strange Men, Ruth Rendell

Key Points

- A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Many block ciphers have a Feistel structure. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves. The original key is expanded so that a different key is used for each round.
- The Data Encryption Standard (DES) has been the most widely used encryption algorithm until recently. It exhibits the classic Feistel structure. DES uses a 64-bit block and a 56-bit key.

- Two important methods of cryptanalysis are differential cryptanalysis and linear cryptanalysis. DES has been shown to be highly resistant to these two types of attack.

The objective of this chapter is to illustrate the principles of modern symmetric ciphers. For this purpose, we focus on the most widely used symmetric cipher: the Data Encryption Standard (DES). Although numerous symmetric ciphers have been developed since the introduction of DES, and although it is destined to be replaced by the Advanced Encryption Standard (AES), DES remains the most important such algorithm. Further, a detailed study of DES provides an understanding of the principles used in other symmetric ciphers. We examine other important symmetric ciphers, including AES, in Chapters 5 and 6.

This chapter begins with a discussion of the general principles of symmetric block ciphers, which are the type of symmetric ciphers studied in this book (with the exception of the stream cipher RC4 in Chapter 6). Next, we cover full DES. Following this look at a specific algorithm, we return to a more general discussion of block cipher design.

Compared to public-key ciphers such as RSA, the structure of DES, and most symmetric ciphers, is very complex and cannot be explained as easily as RSA and similar algorithms. Accordingly, the reader may with to begin with a simplified version of DES, which is described in Appendix C. This version allows the reader to perform encryption and decryption by hand and gain a good understanding of the working of the algorithm details. Classroom experience indicates that a study of this simplified version enhances understanding of DES.[1]

[1] However, you may safely skip Appendix C, at least on a first reading. If you get lost or bogged down in the details of DES, then you can go back and start with simplified DES.

3.1. Block Cipher Principles

Most symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher [FEIS73]. For that reason, it is important to examine the design principles of the Feistel cipher. We begin with a comparison of stream ciphers and block ciphers. Then we discuss the motivation for the Feistel block cipher structure. Finally, we discuss some of its implications.

Stream Ciphers and Block Ciphers

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher. A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. Using some of the modes of operation explained in Chapter 6, a block cipher can be used to achieve the same effect as a stream cipher.

Far more effort has gone into analyzing block ciphers. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers. Accordingly, the concern in this chapter, and in our discussions throughout the book of symmetric encryption, will focus on block ciphers.

Motivation for the Feistel Cipher Structure

A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular. The following examples illustrate nonsingular and singular transformation for $n = 2$.

Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

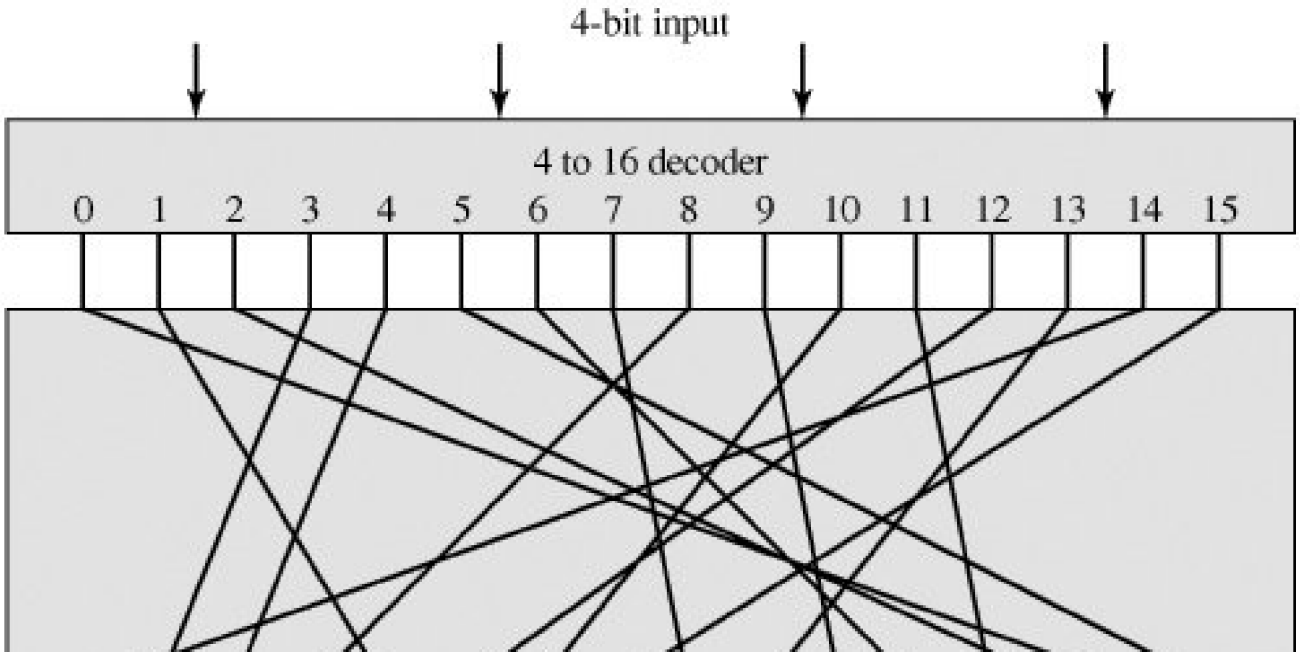
Plaintext	Ciphertext
00	11
01	10
10	01
11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.

Figure 3.1 illustrates the logic of a general substitution cipher for $n = 4$. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits. The encryption and decryption mappings can be defined by a tabulation, as shown in Table 3.1. This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the *ideal block cipher*, because it allows for the maximum number of possible encryption mappings from the plaintext block [FEIS75].

Figure 3.1. General *n*-bit-*n*-bit Block Substitution (shown with *n* = 4)

(This item is displayed on page 65 in the print version)



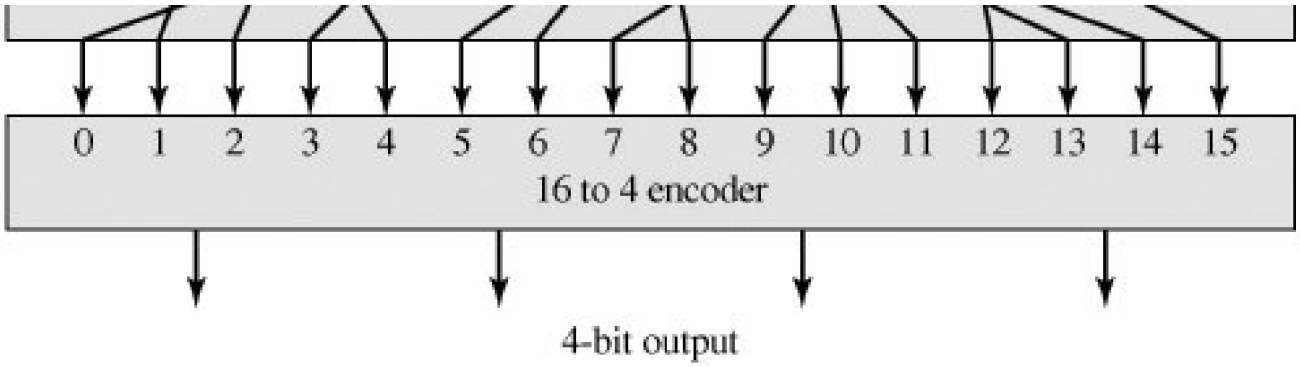

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 to 4 encoder
4-bit output

Table 3.1. Encryption and Decryption Tables for Substitution Cipher of Figure 3.4

(This item is displayed on page 65 in the print version)

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000

But there is a practical problem with the ideal block cipher. If a small block size, such as $n = 4$, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, are vulnerable to a statistical analysis of the plaintext. This weakness is not inherent in the use of a substitution cipher but rather results from the use of a small block size. If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

An arbitrary reversible substitution cipher (the ideal block cipher) for a large block size is not practical, however, from an implementation and performance point of view. For such a transformation, the mapping itself constitutes the key. Consider again Table 3.1, which defines one particular reversible mapping from plaintext to ciphertext for $n = 4$. The mapping can be defined by the entries in the second column, which show the value of the ciphertext for each plaintext block. This, in essence, is the key that determines the specific mapping from among all possible mappings. In this case, using this straightforward method of defining the key, the required key length is (4 bits) x (16 rows) = 64 bits. In general, for an n-bit ideal block cipher, the length of the key defined in this fashion is $n \times 2^n$ bits. For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is $64 \times 2^{64} = 2^{70} \approx 10^{21}$ bits.

In considering these difficulties, Feistel points out that what is needed is an approximation to the ideal block cipher system for large n, built up out of components that are easily realizable [FEIS75]. But before turning to Feistel's approach, let us make one other observation. We could use the general block substitution cipher but, to make its implementation tractable, confine ourselves to a subset of the possible reversible mappings. For example, suppose we define the mapping in terms of a set of linear equations. In the case of $n = 4$, we have

$$y_1 = k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4$$

$$y_2 = k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4$$

$$y_3 = k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4$$

$$y_4 = k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4$$

where the x_i are the four binary digits of the plaintext block, the y_i are the four binary digits of the ciphertext block, the k_{ij} are the binary coefficients, and arithmetic is mod 2. The key size is just n^2, in this case 16 bits. The danger with this kind of formulation is that it may be vulnerable to cryptanalysis by an attacker that is aware of the structure of the algorithm. In this example, what we have is essentially the Hill cipher discussed in Chapter 2, applied to binary data rather than characters. As we saw in Chapter 2, a simple linear system such as this is quite vulnerable.

The Feistel Cipher

Feistel proposed [FEIS73] that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the $2^n!$ transformations available with the ideal block cipher.

In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations. In fact, this is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions [SHAN49]. We look next at these concepts of diffusion and confusion and then present the Feistel cipher. But first, it is worth commenting on this remarkable fact: The Feistel cipher structure, which dates back over a quarter century and which, in turn, is based on Shannon's proposal of 1945, is the structure used by many significant symmetric block ciphers currently in use.

Diffusion and Confusion

The terms *diffusion* and *confusion* were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system [SHAN49].[2] Shannon's concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows. Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words). If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key. In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used. The arbitrary substitution cipher that we discussed previously (Figure 3.1) is such a cipher, but as we have seen, is impractical.

[2] Shannon's 1949 paper appeared originally as a classified report in 1945. Shannon enjoys an amazing and unique position in the history of computer and information science. He not only developed the seminal ideas of modern cryptography but is also responsible for inventing the discipline of information theory. In addition, he founded another discipline, the application of Boolean algebra to the study of digital circuits; this last he managed to toss off as a master's thesis.

Other than recourse to ideal systems, Shannon suggests two methods for frustrating statistical cryptanalysis: diffusion and confusion. In **diffusion**, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally this is equivalent to having each ciphertext digit be affected by many plaintext digits. An example of diffusion is to encrypt a message $M = m_1, m_2, m_3, \ldots$ of characters with an averaging operation:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

adding k successive letters to get a ciphertext letter y_n. One can show that the statistical structure of the plaintext has been dissipated. Thus, the letter frequencies in the ciphertext will be more nearly equal than in the plaintext; the digram frequencies will also be more nearly equal, and so on. In a binary block cipher, diffusion can be achieved by repeatedly performing some permutation on the data followed by applying a function to that permutation; the effect is that bits from different positions in the original plaintext contribute to a single bit of ciphertext.[3]

[3] Some books on cryptography equate permutation with diffusion. This is incorrect. Permutation *by itself*, does not change the statistics of the plaintext at the level of individual letters or permuted blocks. For example, in DES, the permutation swaps two 32-bit blocks, so statistics of strings of 32 bits or less are preserved.

Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key. The mechanism of diffusion seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key. On the other hand, **confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm. In contrast, a simple linear substitution function would add little confusion.

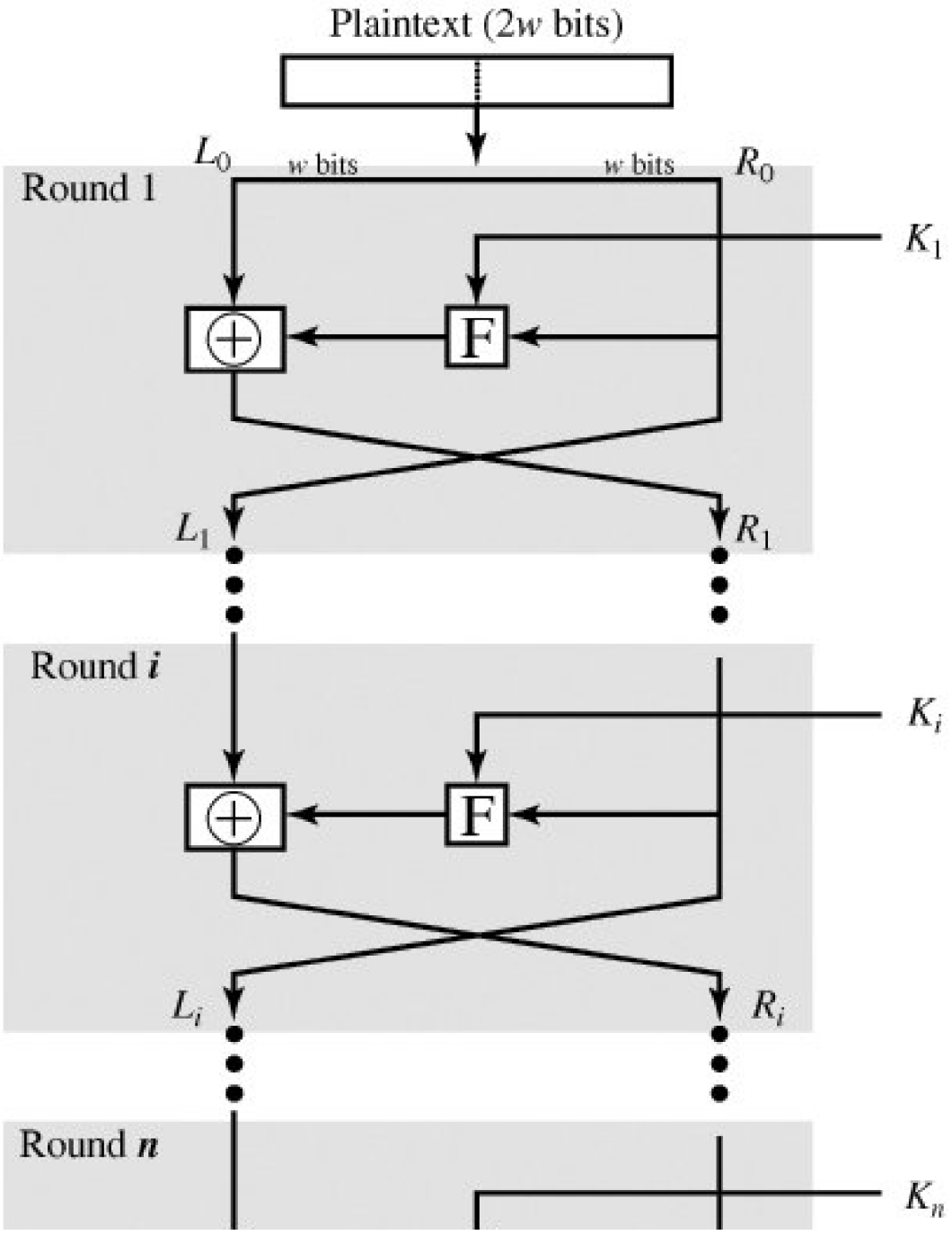
As [ROBS95b] points out, so successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design.

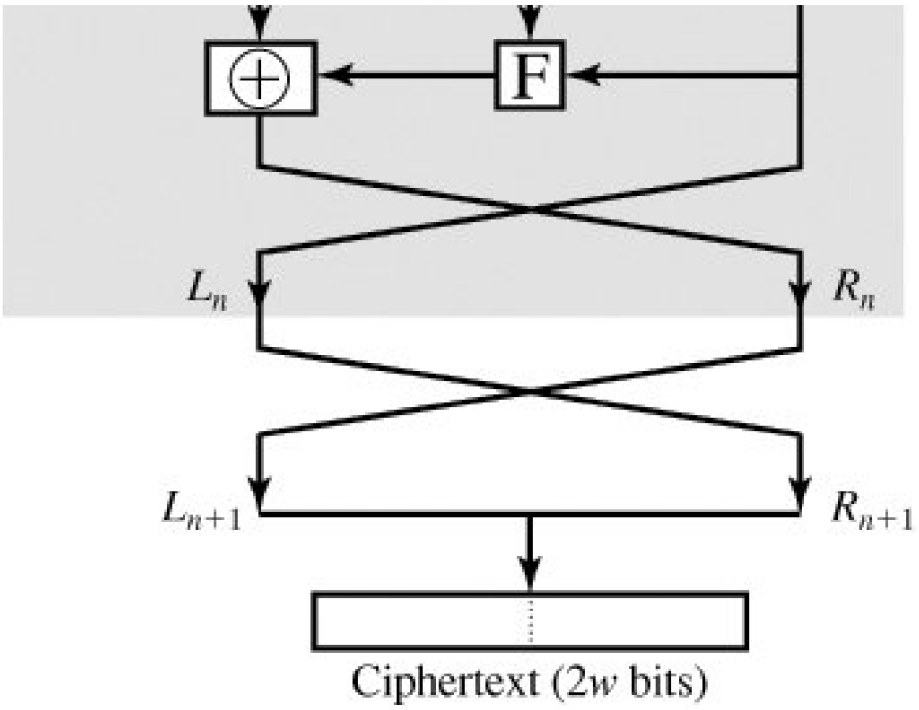
Feistel Cipher Structure

Figure 3.2 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length $2w$ bits and a key K. The plaintext block is divided into two halves, L_0 and R_0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as a subkey K_i, derived from the overall K. In general, the subkeys K_i are different from K and from each other.

Figure 3.2. Classical Feistel Network

(This item is displayed on page 69 in the print version)





All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey K_i. Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data.[4] This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

[4] The final round is followed by an interchange that undoes the interchange that is part of the final round. One could simply leave both interchanges out of the diagram, at the sacrifice of some consistency of presentation. In any case, the effective lack of a swap in the final round is done to simplify the implementation of the decryption process, as we shall see.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

- **Key size:** Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

- **Round function:** Again, greater complexity generally means greater resistance to cryptanalysis.

There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.

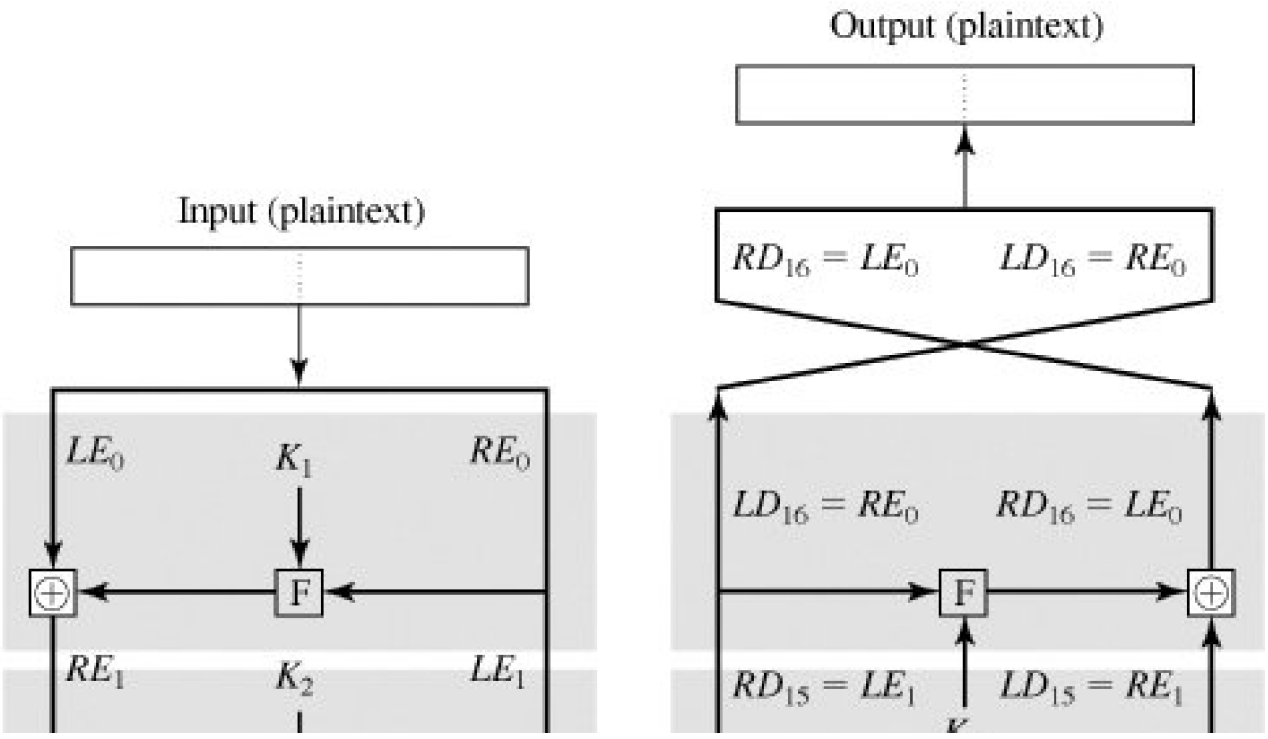
Feistel Decryption Algorithm

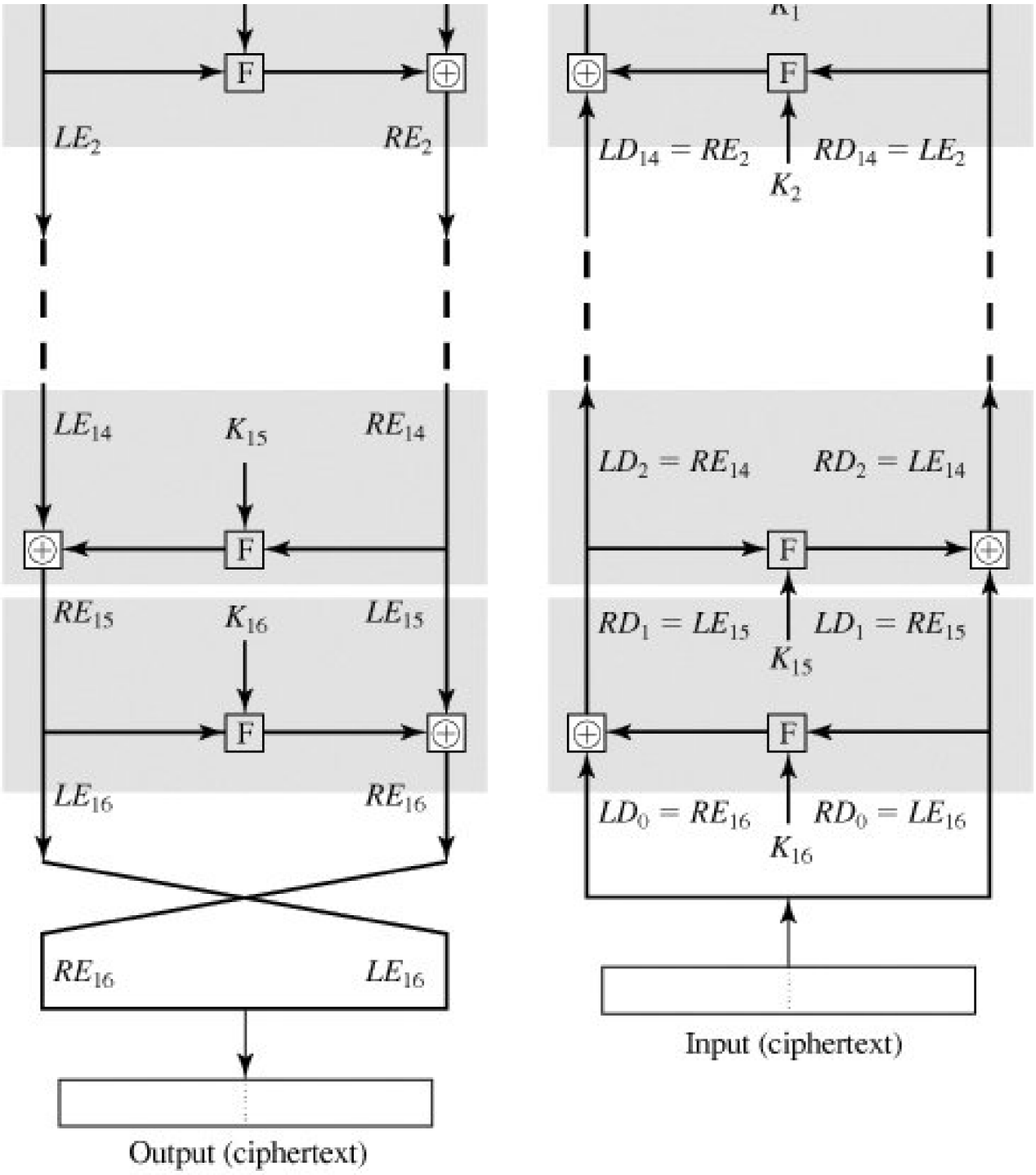
The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys K_i in reverse order. That is, use K_n in the first round, K_{n-1} in the second round, and so on until K_1 is used in the last round. This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.

To see that the same algorithm with a reversed key order produces the correct result, consider Figure 3.3, which shows the encryption process going down the left-hand side and the decryption process going up the right-hand side for a 16-round algorithm (the result would be the same for any number of rounds). For clarity, we use the notation LE_i and RE_i for data traveling through the encryption algorithm and LD_i and RD_i for data traveling through the decryption algorithm. The diagram indicates that, at every round, the intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped. To put this another way, let the output of the ith encryption round be $LE_i||RE_i$ (L_i concatenated with R_i). Then the corresponding input to the $(16-i)$th decryption round is $RE_i||LE_i$ or, equivalently, $RD_{16-i}||LD_{16-i}$.

Figure 3.3. Feistel Encryption and Decryption

(This item is displayed on page 71 in the print version)





Let us walk through Figure 3.3 to demonstrate the validity of the preceding assertions.[5] After the last iteration of the encryption process, the two halves of the output are swapped, so that the ciphertext is $RE_{16}||LE_{16}$. The output of that round is the ciphertext. Now take that ciphertext and use it as input to the same algorithm. The input to the first round is $RE_{16}||LE_{16}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.

[5] To simplify the diagram, it is untwisted, not showing the swap that occurs at the end of each iteration. But please note that the intermediate result at the end of the ith stage of the encryption process is the $2w$-bit quantity formed by concatenating LE_i and RE_i, and that the intermediate result at the end of the ith stage of the decryption process is the $2w$-bit quantity formed by concatenating LD_i and RD_i.

[Page 71]

Now we would like to show that the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First, consider the encryption process. We see that

$LE_{16} = RE_{15}$

$RE_{16} = LE_{15} \times F(RE_{15}, K_{16})$

[Page 72]

On the decryption side,

$LD_1 = RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0 \times F(RD_0, K_{16})$

$= RE_{16} \times F(RE_{15}, K_{16})$

$= [LE_{15} \times F(RE_{15}, K_{16})] \times F(RE_{15}, K_{16})$

The XOR has the following properties:

$[A \times B] \times C = A \times [B \times C]$

$D \times D = 0$

$E \times 0 = E$

Thus, we have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$. Therefore, the output of the first round of the decryption process is $LE_{15}||RE_{15}$, which is the 32-bit swap of the input to the sixteenth round of the encryption. This correspondence holds all the way through the 16 iterations, as is easily shown. We can cast this process in general terms. For the ith iteration of the encryption algorithm,

$LE_i = RE_{i-1}$

$RE_i = LE_{i-1} \times F(RE_{i-1}, K_i)$

Rearranging terms,

$RE_{i-1} = LE_i$

$LE_{i-1} = RE_i \times F(RE_{i-1}, K_i 2 = RE_i \times F(LE_i, K_i)$

Thus, we have described the inputs to the ith iteration as a function of the outputs, and these equations confirm the assignments shown in the right-hand side of Figure 3.3.

Finally, we see that the output of the last round of the decryption process is $RE_0||LE_0$. A 32-bit swap recovers the original plaintext, demonstrating the validity of the Feistel decryption process.

Note that the derivation does not require that F be a reversible function. To see this, take a limiting case in which F produces a constant output (e.g., all ones) regardless of the values of its two arguments. The equations still hold.

3.2. The Data Encryption Standard

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA).[6] For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

[6] The terminology is a bit confusing. Until recently, the terms *DES* and *DEA* could be used interchangeably. However, the most recent edition of the DES document includes a specification of the DEA described here plus the triple DEA (TDEA) described in Chapter 6. Both DEA and TDEA are part of the Data Encryption Standard. Further, until the recent adoption of the official term *TDEA*, the triple DEA algorithm was typically referred to as *triple DES* and written as 3DES. For the sake of convenience, we use the term 3DES.

The DES enjoys widespread use. It has also been the subject of much controversy concerning how secure the DES is. To appreciate the nature of the controversy, let us quickly review the history of the DES.

In the late 1960s, IBM set up a research project in computer cryptography led by Horst Feistel. The project concluded in 1971 with the development of an algorithm with the designation LUCIFER [FEIS73], which was sold to Lloyd's of London for use in a cash-dispensing system, also developed by IBM. LUCIFER is a Feistel block cipher that operates on blocks of 64 bits, using a key size of 128 bits. Because of the promising results produced by the LUCIFER project, IBM embarked on an effort to develop a marketable commercial encryption product that ideally could be implemented on a single chip. The effort was headed by Walter Tuchman and Carl Meyer, and it involved not only IBM researchers but also outside consultants and technical advice from NSA. The outcome of this effort was a refined version of LUCIFER that was more resistant to cryptanalysis but that had a reduced key size of 56 bits, to fit on a single chip.

In 1973, the National Bureau of Standards (NBS) issued a request for proposals for a national cipher standard. IBM submitted the results of its Tuchman-Meyer project. This was by far the best algorithm proposed and was adopted in 1977 as the Data Encryption Standard.

Before its adoption as a standard, the proposed DES was subjected to intense criticism, which has not subsided to this day. Two areas drew the critics' fire. First, the key length in IBM's original LUCIFER algorithm was 128 bits, but that of the proposed system was only 56 bits, an enormous reduction in key size of 72 bits. Critics feared that this key length was too short to withstand brute-force attacks. The second area of concern was that the design criteria for the internal structure of DES, the S-boxes, were classified. Thus, users could not be sure that the internal structure of DES was free of any hidden weak points that would enable NSA to decipher messages without benefit of the key. Subsequent events, particularly the recent work on differential cryptanalysis, seem to indicate that DES has a very strong internal structure. Furthermore, according to IBM participants, the only changes that were made to the proposal were changes to the S-boxes, suggested by NSA, that removed vulnerabilities identified in the course of the evaluation process.

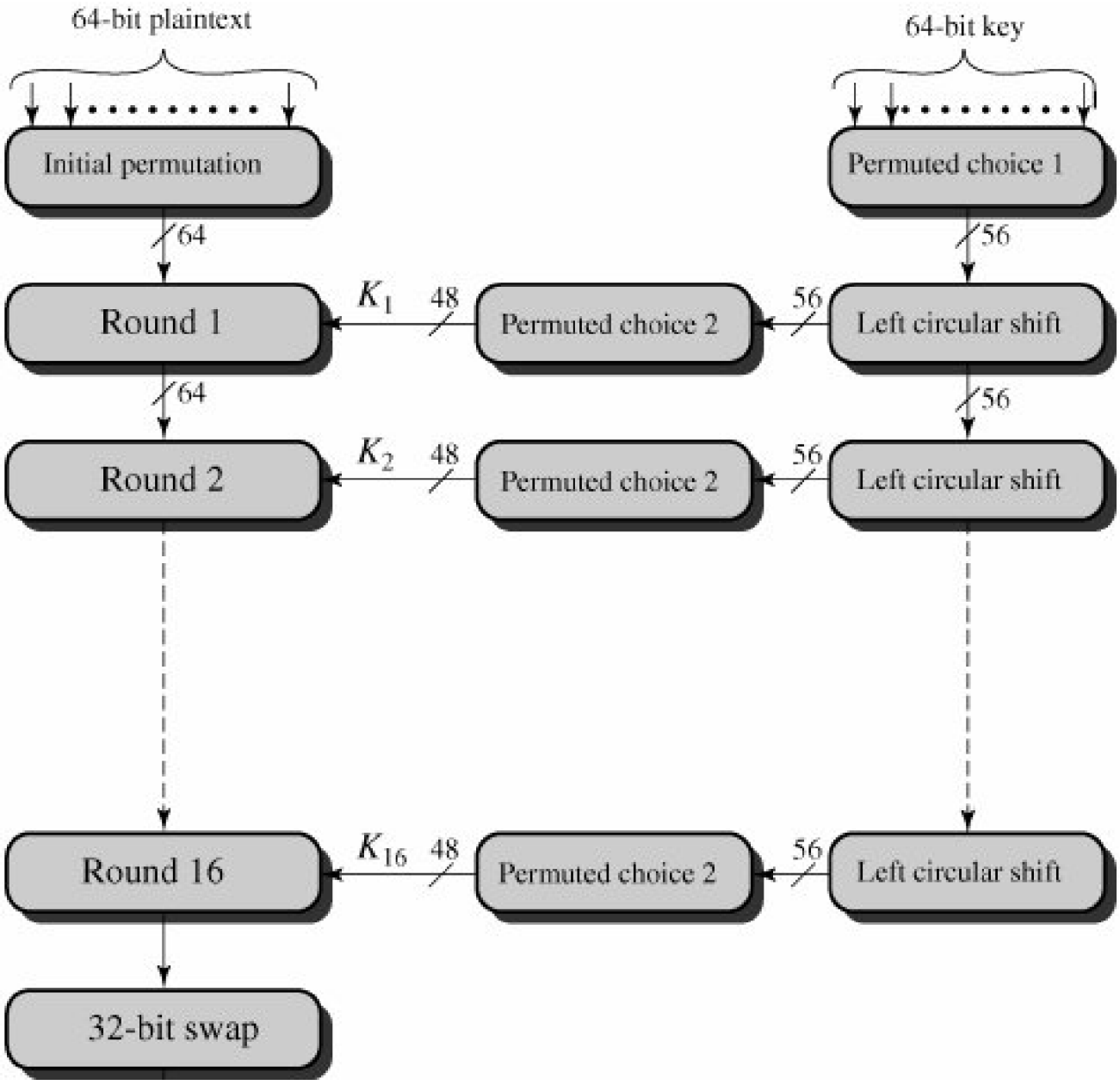
Whatever the merits of the case, DES has flourished and is widely used, especially in financial applications. In 1994, NIST reaffirmed DES for federal use for another five years; NIST recommended the use of DES for applications other than the protection of classified information. In 1999, NIST issued a new version of its standard (FIPS PUB 46-3) that indicated that DES should only be used for legacy systems and that triple DES (which in essence involves repeating the DES algorithm three times on the plaintext using two or three different keys to produce the ciphertext) be used. We study triple DES in Chapter 6. Because the underlying encryption and decryption algorithms are the same for DES and triple DES, it remains important to understand the DES cipher.

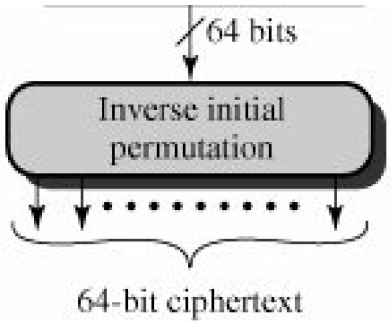
DES Encryption

The overall scheme for DES encryption is illustrated in Figure 3.4. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.[7]

[7] Actually, the function expects a 64-bit key as input. However, only 56 of these bits are ever used; the other 8 bits can be used as parity bits or simply set arbitrarily.

Figure 3.4. General Depiction of DES Encryption Algorithm





Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**. Finally, the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher, as shown in Figure 3.2.

The right-hand portion of Figure 3.4 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the 16 rounds, a *subkey* (K_i) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation

The initial permutation and its inverse are defined by tables, as shown in Tables 3.2a and 3.2b, respectively. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

Table 3.2. Permutation Tables for DES

(This item is displayed on page 76 in the print version)

(a) Initial Permutation (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

(b) Inverse Initial Permutation (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

(c) Expansion Permutation (E)

	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	

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10

(a) Initial Permutation (IP)							
19	13	30	6	22	11	4	25

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

where M_i is a binary digit. Then the permutation$X = \text{IP}(M)$ is as follows:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

If we then take the inverse permutation $Y = \text{IP}^{-1}(X) = \text{IP}^{-1}(\text{IP}(M))$, it can be seen that the original ordering of the bits is restored.

Details of Single Round

Figure 3.5 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel

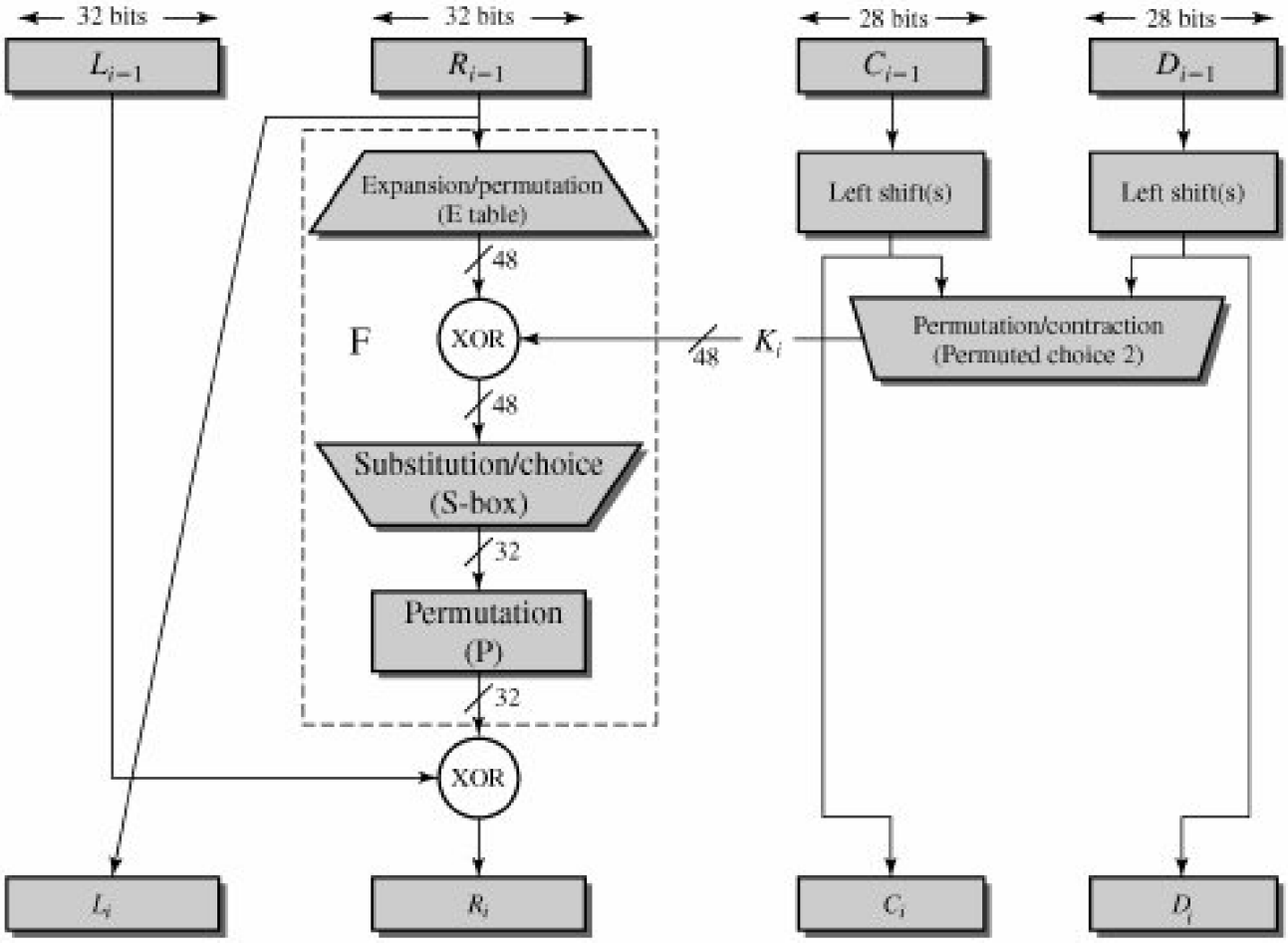
cipher, the overall processing at each round can be summarized in the following formulas:

$L_i = R_{i-1}$

$R_i = L_{i-1} \times F(R_{i-1}, K_i)$

Figure 3.5. Single Round of DES Algorithm

[View full size image]



The round key K_i is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits (Table 3.2c). The resulting 48 bits are XORed with K_i. This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 3.2d.

The role of the S-boxes in the function F is illustrated in Figure 3.6. The substitution consists of a set of eight S-boxes, each of which

accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S_1 for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

Figure 3.6. Calculation of F(R, K)

(This item is displayed on page 78 in the print version)

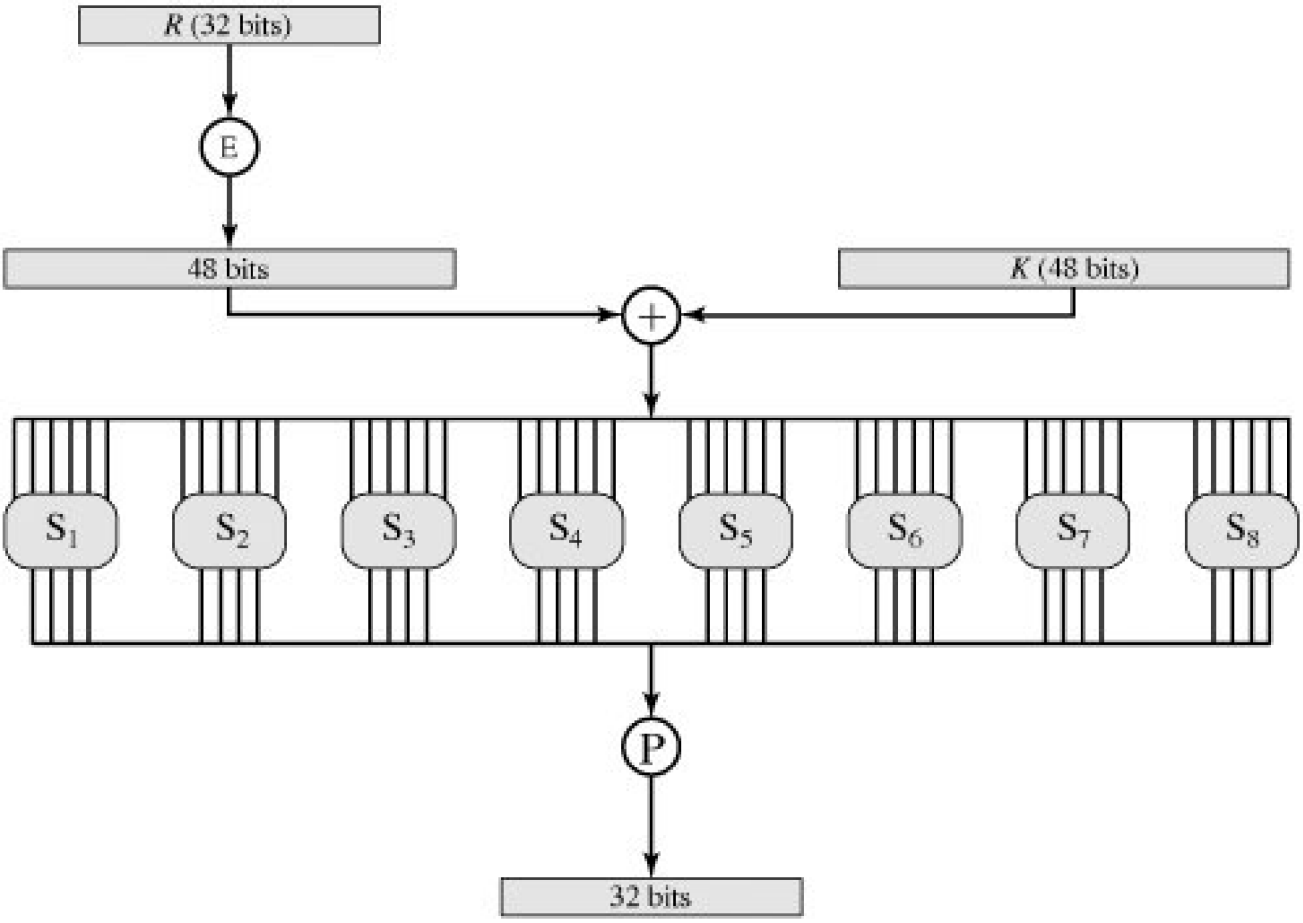


Table 3.3. Definition of DES S-Boxes

(This item is displayed on page 79 in the print version)

[View full size image]

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

S_7

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

S_8

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

Each row of an S-box defines a general reversible substitution. Figure 3.1 may be useful in understanding the mapping. The figure shows the substitution for row 0 of box S_1.

The operation of the S-boxes is worth further comment. Ignore for the moment the contribution of the key (K_i). If you examine the expansion table, you see that the 32 bits of input are split into groups of 4 bits, and then become groups of 6 bits by taking the outer bits from the two adjacent groups. For example, if part of the input word is

---

... efgh ijkl mnop ...

this becomes

... defghi hijklm lmnopq ...

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round the output from each S-box immediately affects as many others as possible.

Key Generation

Returning to Figures 3.4 and 3.5, we see that a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a. The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b). The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0. At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift, or rotation, of 1 or 2 bits, as governed by Table 3.4d. These shifted values serve as input to the next round. They also serve as input to Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$.

Table 3.4. DES Key Schedule Calculation

(a) Input Key

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
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (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

(c) Permuted Choice Two (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

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

The Avalanche Effect

A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext. In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

DES exhibits a strong avalanche effect Table 3.5 shows some results taken from [KONH81]. In Table 3.5a, two plaintexts that differ by one bit were used:

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

with the key

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

Table 3.5. Avalanche Effect in DES

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

The Table 3.5a shows that after just three rounds, 21 bits differ between the two blocks. On completion, the two ciphertexts differ in 34 bit positions.

Table 3.5b shows a similar test in which a single plaintext is input:

01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

with two keys that differ in only one bit position:

1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Again, the results show that about half of the bits in the ciphertext differ and that the avalanche effect is pronounced after just a few rounds.

3.3. The Strength of Des

Since its adoption as a federal standard, there have been lingering concerns about the level of security provided by DES. These concerns, by and large, fall into two areas: key size and the nature of the algorithm.

The Use of 56-Bit Keys

With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16}. Thus, on the face of it, a brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years (see Table 2.2) to break the cipher.

However, the assumption of one encryption per microsecond is overly conservative. As far back as 1977, Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond [DIFF77]. This would bring the average search time down to about 10 hours. The authors estimated that the cost would be about $20 million in 1977 dollars.

DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98]. And, of course, hardware prices will continue to drop as speeds increase, making DES virtually worthless.

It is important to note that there is more to a key-search attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. If the message is just plain text in English, then the result pops out easily, although the task of recognizing English would have to be automated. If the text message has been compressed before encryption, then recognition is more difficult. And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate. Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed. The EFF approach addresses this issue as well and introduces some automated techniques that would be effective in many contexts.

Fortunately, there are a number of alternatives to DES, the most important of which are AES and triple DES, discussed in Chapters 5 and 6, respectively.

The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes. This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered. Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.[8]

[8] At least, no one has publicly acknowledged such a discovery.

Timing Attacks

We discuss timing attacks in more detail in Part Two, as they relate to public-key algorithms. However, the issue may also be relevant for symmetric ciphers. In essence, a timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts. A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs. [HEVI99] reports on an approach that yields the Hamming weight (number of bits equal to one) of the secret key. This is a long way from knowing the actual key, but it is an intriguing first step. The authors conclude that DES appears to be fairly resistant to a successful timing attack but suggest some avenues to explore. Although this is an interesting line of attack, it so far appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES.

3.4. Differential and Linear Cryptanalysis

For most of its life, the prime concern with DES has been its vulnerability to brute-force attack because of its relatively short (56 bits) key length. However, there has also been interest in finding cryptanalytic attacks on DES. With the increasing popularity of block ciphers with longer key lengths, including triple DES, brute-force attacks have become increasingly impractical. Thus, there has been increased emphasis on cryptanalytic attacks on DES and other symmetric block ciphers. In this section, we provide a brief overview of the two most powerful and promising approaches: differential cryptanalysis and linear cryptanalysis.

Differential Cryptanalysis

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis. In this section, we discuss the technique and its applicability to DES.

History

Differential cryptanalysis was not reported in the open literature until 1990. The first published effort appears to have been the cryptanalysis of a block cipher called FEAL by Murphy [MURP90]. This was followed by a number of papers by Biham and Shamir, who demonstrated this form of attack on a variety of encryption algorithms and hash functions; their results are summarized in [BIHA93].

The most publicized results for this approach have been those that have application to DES. Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 2^{55} complexity. The scheme, as reported in BIHA93], can successfully cryptanalyze DES with an effort on the order of 2^{47} encryptions, requiring 2^{47} chosen plaintexts. Although 2^{47} is certainly significantly less than 2^{55} the need for the adversary to find 2^{47} chosen plaintexts makes this attack of only theoretical interest.

Although differential cryptanalysis is a powerful tool, it does not do very well against DES. The reason, according to a member of the IBM team that designed DES [COPP94], is that differential cryptanalysis was known to the team as early as 1974. The need to strengthen DES against attacks using differential cryptanalysis played a large part in the design of the S-boxes and the permutation P. As evidence of the impact of these changes, consider these comparable results reported in [BIHA93]. Differential cryptanalysis of an eight-round LUCIFER algorithm requires only 256 chosen plaintexts, whereas an attack on an eight-round version of DES requires 2^{14} chosen plaintexts.

Differential Cryptanalysis Attack

The differential cryptanalysis attack is complex; [BIHA93] provides a complete description. The rationale behind differential cryptanalysis is

to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block. Here, we provide a brief overview so that you can get the flavor of the attack.

We begin with a change in notation for DES. Consider the original plaintext block m to consist of two halves m_0,m_1. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round. So, at each round, only one new 32-bit block is created. If we label each new block m_1($2 \leq i \leq 17$), then the intermediate message halves are related as follows:

$m_{i+1} = m_{i-1} \oplus f(m_i, K_i)$, $i = 1, 2, ..., 16$

In differential cryptanalysis, we start with two messages, m and m', with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $m_i = m_i \oplus m_i'$ Then we have:

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)]\end{aligned}$$

Now, suppose that many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used. To put this more precisely, let us say that *X may cause Y with probability p*, if for a fraction p of the pairs in which the input XOR is X, the output XOR equals Y. We want to suppose that there are a number of values of X that have high probability of causing a particular output difference. Therefore, if we know Δm_{i-1} and Δm_i with high probability, then we know Δm_{i+1} with high probability. Furthermore, if a number of such differences are determined, it is feasible to determine the subkey used in the function f.

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages m and m' with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the ciphertext. Actually, there are two probable patterns of differences for the two 32-bit halves: ($\Delta m_{17} || m_{16}$). Next, we submit m and m' for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match,

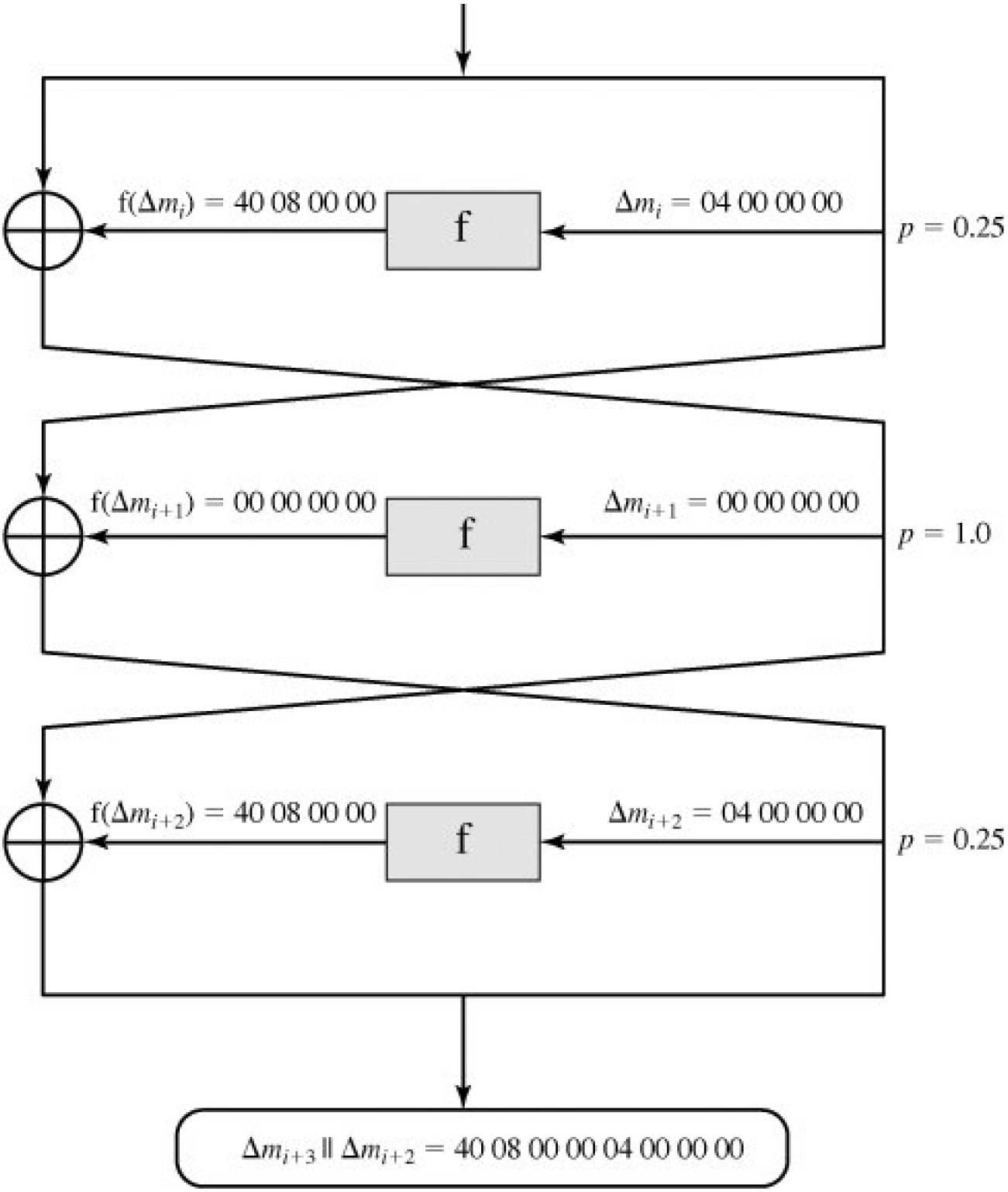
$E(K, m) \oplus E(K, m') = (\Delta m_{17} || m_{16})$

---

then we suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, we can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

Figure 3.7, based on a figure in [BIHA93], illustrates the propagation of differences through three rounds of DES. The probabilities shown on the right refer to the probability that a given set of intermediate differences will appear as a function of the input differences. Overall, after three rounds the probability that the output difference is as shown is equal to 0.25 x 1 x 0.25 = 0.0625.

Figure 3.7. Differential Propagation through Three Round of DES (numbers in hexadecimal)



Linear Cryptanalysis

A more recent development is linear cryptanalysis, described in [MATS93]. This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given 2^{43} known plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by other groups to

validate the linear cryptanalytic approach.

We now give a brief summary of the principle on which linear cryptanalysis is based. For a cipher with n-bit plaintext and ciphertext blocks and an *m*-bit key, let the plaintext block be labeled P[1], ... P[*n*], the cipher text block C[1], ... C[*n*], and the key K[1], ... K[*m*]. Then define

$A[i, j, ..., k] = A[i] \oplus A[j] \oplus ... \oplus A[k]$

The objective of linear cryptanalysis is to find an effective *linear* equation of the form:

$P[a_1, a_2, ..., a_a] \oplus C[b_1, b_2, ..., b_b] = K[g_1, g_2, ..., g_c]$

(where $x = 0$ or 1; $1 \leq a, b \leq n$, $1 \leq c \leq m$, and where the a, b and g terms represent fixed, unique bit locations) that holds with probability $p \neq 0.5$. The further p is from 0.5, the more effective the equation. Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext-ciphertext pairs. If the result is 0 more than half the time, assume $K[g_1, g_2, ..., g_c] = 0$. If it is 1 most of the time, assume $K[g_1, g_2, ..., g_c] = 1$. This gives us a linear equation on the key bits. Try to get more such relations so that we can solve for the key bits. Because we are dealing with linear equations, the problem can be approached one round of the cipher at a time, with the results combined.

PREV NEXT

3.5. Block Cipher Design Principles

Although much progress has been made in designing block ciphers that are cryptographically strong, the basic principles have not changed all that much since the work of Feistel and the DES design team in the early 1970s. It is useful to begin this discussion by looking at the published design criteria used in the DES effort. Then we look at three critical aspects of block cipher design: the number of rounds, design of the function F, and key scheduling.

DES Design Criteria

The criteria used in the design of DES, as reported in [COPP94], focused on the design of the S-boxes and on the P function that takes the output of the S boxes (Figure 3.6). The criteria for the S-boxes are as follows:

1. No output bit of any S-box should be too close a linear function of the input bits. Specifically, if we select any output bit and any subset of the six input bits, the fraction of inputs for which this output bit equals the XOR of these input bits should not be close to 0 or 1, but rather should be near 1/2.
2. Each row of an S-box (determined by a fixed value of the leftmost and rightmost input bits) should include all 16 possible output bit combinations.
3. If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.

4. If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.
5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.
6. For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
7. This is a criterion similar to the previous one, but for the case of three S-boxes.

Coppersmith pointed out that the first criterion in the preceding list was needed because the S-boxes are the only nonlinear part of DES. If the S-boxes were linear (i.e., each output bit is a linear combination of the input bits), the entire algorithm would be linear and easily broken. We have seen this phenomenon with the Hill cipher, which is linear. The remaining criteria were primarily aimed at thwarting differential cryptanalysis and at providing good confusion properties.

The criteria for the permutation P are as follows:

1. The four output bits from each S-box at round i are distributed so that two of them affect (provide input for) "middle bits" of round $(i + 1)$ and the other two affect end bits. The two middle bits of input to an S-box are not shared with adjacent S-boxes. The end bits are the two left-hand bits and the two right-hand bits, which are shared with adjacent S-boxes.
2. The four output bits from each S-box affect six different S-boxes on the next round, and no two affect the same S-box.
3. For two S-boxes j, k, if an output bit from S_j affects a middle bit of S_k on the next round, then an output bit from S_k cannot affect

a middle bit of S_j. This implies that for $j = k$, an output bit from S_j must not affect a middle bit of S_j

These criteria are intended to increase the diffusion of the algorithm.

Number of Rounds

The cryptographic strength of a Feistel cipher derives from three aspects of the design: the number of rounds, the function F, and the key schedule algorithm. Let us look first at the choice of the number of rounds.

The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F. In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. This criterion was certainly used in the design of DES. Schneier [SCHN96] observes that for 16-round DES, a differential cryptanalysis attack is slightly less efficient than brute force: the differential cryptanalysis attack requires $2^{55.1}$ operations,[9] whereas brute force requires 2^{55}. If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than brute-force key search.

[9] Recall that differential cryptanalysis of DES requires 2^{47} *chosen* plaintext. If all you have to work with is known plaintext, then you must sort through a large quantity of known plaintext-ciphertext pairs looking for the useful ones. This brings the level of effort up to $2^{55.1}$.

This criterion is attractive because it makes it easy to judge the strength of an algorithm and to compare different algorithms. In the absence of a cryptanalytic breakthrough, the strength of any algorithm that satisfies the criterion can be judged solely on key length.

Design of Function F

The heart of a Feistel block cipher is the function F. As we have seen, in DES, this function relies on the use of S-boxes. This is also the case for most other symmetric block ciphers, as we shall see in Chapter 4. However, we can make some general comments about the criteria for designing F. After that, we look specifically at S-box design.

Design Criteria for F

The function F provides the element of confusion in a Feistel cipher. Thus, it must be difficult to "unscramble" the substitution performed by F. One obvious criterion is that F be nonlinear, as we discussed previously. The more nonlinear F, the more difficult any type of cryptanalysis will be. There are several measures of nonlinearity, which are beyond the scope of this book. In rough terms, the more difficult it is to approximate F by a set of linear equations, the more nonlinear F is.

Several other criteria should be considered in designing F. We would like the algorithm to have good avalanche properties. Recall that, in general, this means that a change in one bit of the input should produce a change in many bits of the output. A more stringent version of this is the **strict avalanche criterion (SAC)** [WEBS86], which states that any output bit j of an S-box should change with probability 1/2 when any single input bit i is inverted for all i, j. Although SAC is expressed in terms of S-boxes, a similar criterion could be applied to F as a whole. This is important when considering designs that do not include S-boxes.

Another criterion proposed in [WEBS86] is the **bit independence criterion (BIC)**, which states that output bits j and k should change

independently when any single input bit i is inverted, for all i, j, and k. The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function.

S-Box Design

One of the most intense areas of research in the field of symmetric block ciphers is that of S-box design. The papers are almost too numerous to count.[10] Here we mention some general principles. In essence, we would like any change to the input vector to an S-box to result in random-looking changes to the output. The relationship should be nonlinear and difficult to approximate with linear functions.

[10] A good summary of S-box design studies through early 1996 can be found in [SCHN96].

One obvious characteristic of the S-box is its size. An $n \times m$ S-box has n input bits and m output bits. DES has 6 x 4 S-boxes. Blowfish, described in Chapter 6, has 8 x 32 S-boxes. Larger S-boxes, by and large, are more resistant to differential and linear cryptanalysis [SCHN96]. On the other hand, the larger the dimension n, the (exponentially) larger the lookup table. Thus, for practical reasons, a limit of n equal to about 8 to 10 is usually imposed. Another practical consideration is that the larger the S-box, the more difficult it is to design it properly.

S-boxes are typically organized in a different manner than used in DES. An $n \times m$ S-box typically consists of 2^n rows of m bits each. The n bits of input select one of the rows of the S-box, and the m bits in that row are the output. For example, in an 8 x 32 S-box, if the input is 00001001, the output consists of the 32 bits in row 9 (the first row is labeled row 0).

Mister and Adams [MIST96] propose a number of criteria for S-box design. Among these are that the S-box should satisfy both SAC and BIC. They also suggest that all linear combinations of S-box columns should be *bent*. Bent functions are a special class of Boolean functions that are highly nonlinear according to certain mathematical criteria [ADAM90]. There has been increasing interest in designing and analyzing S-boxes using bent functions.

A related criterion for S-boxes is proposed and analyzed in [HEYS95]. The authors define the **guaranteed avalanche (GA)** criterion as follows: An S-box satisfies GA of order p if, for a 1-bit input change, at least p output bits change. The authors conclude that a GA in the range of order 2 to order 5 provides strong diffusion characteristics for the overall encryption algorithm.

For larger S-boxes, such as 8 x 32, the question arises as to the best method of selecting the S-box entries in order to meet the type of criteria we have been discussing. Nyberg, who has written a lot about the theory and practice of S-box design, suggests the following approaches (quoted in [ROBS95b]):

- **Random:** Use some pseudorandom number generation or some table of random digits to generate the entries in the S-boxes. This may lead to boxes with undesirable characteristics for small sizes (e.g., 6 x 4) but should be acceptable for large S-boxes (e.g., 8 x 32).
- **Random with testing:** Choose S-box entries randomly, then test the results against various criteria, and throw away those that do not pass.
- **Human-made:** This is a more or less manual approach with only simple mathematics to support it. It is apparently the technique used in the DES design. This approach is difficult to carry through for large S-boxes.
- **Math-made:** Generate S-boxes according to mathematical principles. By using mathematical construction, S-boxes can be constructed that offer proven security against linear and differential cryptanalysis, together with good diffusion.

A variation on the first technique is to use S-boxes that are both random and key dependent. An example of this approach is Blowfish, described in Chapter 6, which starts with S-boxes filled with pseudorandom digits and then alters the contents using the key. A tremendous advantage of key-dependent S-boxes is that, because they are not fixed, it is impossible to analyze the S-boxes ahead of time to look for weaknesses.

Key Schedule Algorithm

A final area of block cipher design, and one that has received less attention than S-box design, is the key schedule algorithm. With any Feistel block cipher, the key is used to generate one subkey for each round. In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key. No general principles for this have yet been promulgated.

Hall suggests [ADAM94] that, at minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

PREV NEXT

3.6. Recommended Reading

There is a wealth of information on symmetric encryption. Some of the more worthwhile references are listed here. An essential reference work is [SCHN96]. This remarkable work contains descriptions of virtually every cryptographic algorithm and protocol published up to the time of the writing of the book. The author pulls together results from journals, conference proceedings, government publications, and standards documents and organizes these into a comprehensive and comprehensible survey. Another worthwhile and detailed survey is [MENE97]. A rigorous mathematical treatment is [STIN02].

The foregoing references provide coverage of public-key as well as symmetric encryption.

Perhaps the most detailed description of DES is [SIMO95]; the book also contains an extensive discussion of differential and linear cryptanalysis of DES. [BARK91] provides a readable and interesting analysis of the structure of DES and of potential cryptanalytic approaches to DES. [EFF98] details the most effective brute-force attack on DES. [COPP94] looks at the inherent strength of DES and its ability to stand up to cryptanalysis.

BARK91 Barker, W. *Introduction to the Analysis of the Data Encryption Standard (DES).* Laguna Hills, CA: Aegean Park Press, 1991.

COPP94 Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." *IBM Journal of Research and Development*, May 1994.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design.* Sebastopol, CA: O'Reilly, 1998

MENE97 Menezes, A.; van Oorschot, P.; and Vanstone, S. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press, 1997.

SCHN96 Schneier, B. *Applied Cryptography.* New York: Wiley, 1996.

SIMO95 Simovits, M. *The DES: An Extensive Documentation and Evaluation*. Laguna Hills, CA: Aegean Park Press, 1995.

STIN02 Stinson, D. *Cryptography: Theory and Practice.* Boca Raton, FL: CRC Press, 2002.

3.7. Key Terms, Review Questions, and Problems

Key Terms

avalanche effect

block cipher

confusion

Data Encryption Standard (DES)

differential cryptanalysis

diffusion

Feistel cipher

irreversible mapping

key

linear cryptanalysis

permutation

product cipher

reversible mapping

round

round function

subkey

substitution

Review Questions

3.1 Why is it important to study the Feistel cipher?

3.2 What is the difference between a block cipher and a stream cipher?

3.3 Why is it not practical to use an arbitrary reversible substitution cipher of the kind shown in Table 3.1?

3.4 What is a product cipher?

3.5 What is the difference between diffusion and confusion?

3.6 Which parameters and design choices determine the actual algorithm of a Feistel cipher?

3.7 What is the purpose of the S-boxes in DES?

3.8 Explain the avalanche effect.

3.9 What is the difference between differential and linear cryptanalysis?

Problems

3.1

a. In Section 3.1, under the subsection on the motivation for the Feistel cipher structure, it was stated that, for a block of n bits, the number of different reversible mappings for the ideal block cipher is $2^n!$. Justify.

b. In that same discussion, it was stated that for the ideal block cipher, which allows all possible reversible mappings, the size of the key is $n \times 2^n$ bits. But, if there are $2^n!$ possible mappings, it should take $\log_2 2^n!$ bits to discriminate among the different mappings, and so the key length should be $\log_2 2^n!$. However, $\log_2 2^n! < n \times 2^n$. Explain the discrepancy.

3.2 Consider a Feistel cipher composed of 16 rounds with block length 128 bits and key length 128 bits. Suppose that, for a given k, the key scheduling algorithm determines values for the first 8 round keys $k_1, k_2, ..., k_8$, and then sets

$k_9 = k_8, k_{10} = k_7, k_{11} = k_6, ..., k_{16} = k_1$

Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack. (An encryption oracle can be thought of as a device that, when given a plaintext, returns the corresponding ciphertext. The internal details of the device are not known to you and you cannot break open

the device. You can only gain information from the oracle by making queries to it and observing its responses.)

3.3 Consider a block encryption algorithm that encrypts blocks of length n, and let $N = 2^n$. Say we have t plaintext-ciphertext pairs $P_i, C_i = E(K, P_i)$, where we assume that the key K selects one of the $N!$ possible mappings. Imagine that we wish to find K by exhaustive search. We could generate key K' and test whether $C_i = E(K', P_i)$ for $1 \neq i \neq t$. If K' encrypts each P_i to its proper C_i then we have evidence that $K = K'$. However, it may be the case that the mappings $E(K, \cdot)$ and $E(K', \cdot)$ exactly agree on the t plaintext-ciphertext pairs P_i, C_i and agree on no other pairs.

a. What is the probability that $E(K, \cdot)$ and $E(K', \cdot)$ are in fact distinct mappings?

b. What is the probability that $E(K, \cdot)$ and $E(K', \cdot)$ agree on another t' plaintext-ciphertext pairs where $0 \neq t' \neq N - t$?

3.4 Let p be a permutation of the integers 0, 1, 2, ... $(2^n - 1)$ such that $p(m)$ gives the permuted value of m, $0 \neq m \leq 2^n$. Put another way, p maps the set of n-bit integers into itself and no two integers map into the same integer. DES is such a permutation for 64-bit integers. We say that p has a fixed point at m if $p(m) = m$. That is, if p is an encryption mapping, then a fixed point corresponds to a message that encrypts to itself. We are interested in the probability that p has no fixed points. Show the somewhat unexpected result that over 60% of mappings will have at least one fixed point.

3.5 Consider the substitution defined by row 1 of S-box S_1 in Table 3.3. Show a block diagram similar to Figure 3.1 that corresponds to this substitution.

3.6 Compute the bits number 1, 16, 33, and 48 at the output of the first round of the DES decryption, assuming that the ciphertext block is composed of all ones and the external key is composed of all ones.

3.7 Suppose the DES F function mapped every 32-bit input R, regardless of the value of the input K, to

a. 32-bit string of ones,

b. bitwise complement of R.

Hint: Use the following properties of the XOR operation:

1. What function would DES then compute?

2. What would the decryption look like?

$(A \otimes B) \otimes C = A \otimes (B \otimes C)$

$A \otimes A = 0$

$A \otimes 0 = A$

$A \otimes 1$ = bitwise complement of A

where

A, B, C are n-bit strings of bits

0 is an n-bit string of zeros

1 is an n-bit string of one

3.8 This problem provides a numerical example of encryption using a one-round version of DES. We start with the same bit pattern for the key K and the plaintext, namely:

in hexadecimal notation:	0 1 2 3 4 5 6 7 8 9 A B C D E F
in binary notation:	0000 0001 0010 0011 0100 0101 0110 0111
	1000 1001 1010 1011 0100 1101 1110 1111

a. Derive K_1, the first-round subkey.

b. Derive L_0, R_0.

c. Expand R_0 to get E[R_0], where E[·] is the expansion function of Figure 3.8.

d. Calculate $A = \text{E}[R_0] \otimes K_1$.

e. Group the 48-bit result of (d) into sets of 6 bits and evaluate the corresponding S-box substitutions.

f. Concatenate the results of (e) to get a 32-bit result, B.

g. Apply the permutation to get P(B).

h. Calculate $R_1 = \text{P}(B) \otimes L_0$.

i. Write down the ciphertext.

3.9 Show that DES decryption is, in fact, the inverse of DES encryption.

3.10 The 32-bit swap after the sixteenth iteration of the DES algorithm is needed to make the encryption process invertible by simply running the ciphertext back through the algorithm with the key order reversed. This was demonstrated in Problem 3.7. However, it still may not be entirely clear why the 32-bit swap is needed. To demonstrate why, solve the following exercises. First, some notation:

$A\|\|B$	= the concatenation of the bit strings A and B
$T_i(R\|\|L)$	= the transformation defined by the ith iteration of the encryption algorithm, for $1 \leq i \leq 16$
$TD_i(R\|\|L)$	= the transformation defined by the ith iteration of the decryption algorithm, for $1 \leq i \leq 16$
$T_{17}(R\|\|L)$	= $L\|\|R$. This transformation occurs after the sixteenth iteration of the encryption algorithm.

---

a. Show that the composition $TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15}||R_{15})))))$ is equivalent to the transformation that interchanges the 32-bit halves, L_{15} and R_{15}. That is, show that

$$TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15}||R_{15}))))) = R_{15}||L_{15}$$

b. Now suppose that we did away with the final 32-bit swap in the encryption algorithm. Then we would want the following equality to hold:

$$TD_1(IP(IP^{-1}(T_{16}(L_{15}||R_{15}))))) = R_{15}||L_{15}$$

Does it?

3.11 Compare the initial permutation table (Table 3.2a) with the permuted choice one table (Table 3.4b). Are the structures similar? If so, describe the similarities. What conclusions can you draw from this analysis?

3.12 When using the DES algorithm for decryption, the 16 keys ($K_1, K_2, ..., K_{16}$) are used in reverse order. Therefore, the right-hand side of Figure 3.5 is no longer valid. Design a key-generation scheme with the appropriate shift schedule (analogous to Table 3.4d) for the decryption process.

3.13

a. Let X' be the bitwise complement of X. Prove that if the complement of the plaintext block is taken and the complement of an encryption key is taken, then the result of DES encryption with these values is the complement of the original ciphertext. That is,

If $Y = E(K, X)$

Then $Y' = E(K', X')$

Hint: Begin by showing that for any two bit strings of equal length A and B, $(A \otimes B)' = A \otimes B$.

b. It has been said that a brute-force attack on DES requires searching a key space of 2^{56} keys. Does the result of part (a) change that?

3.14 Show that in DES the first 24 bits of each subkey come from the same subset of 28 bits of the initial key and that the second 24 bits of each subkey come from a disjoint subset of 28 bits of the initial key.

3.15 For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a linear block cipher EL that encrypts 128-bit blocks of plaintext into 128-bit blocks of ciphertext. Let EL(k, m) denote the encryption of a 128-bit message m under a key k (the actual bit length of k is irrelevant). Thus

$$EL(k, [m_1 \otimes m_2]) = EL(k, m_1) \otimes EL(k, m_1) \text{ for all 128-bit patterns } m_1, m_2$$

Describe how, with 128 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 128 plaintext/ciphertext pairs to work with and you have the ability to chose the value of the ciphertexts.)

Note: The following problems refer to simplified DES, described in Appendix C.

3.16 Refer to Figure C.2, which depicts key generation for S-DES.

a. How important is the initial P10 permutation function?

b. How important are the two LS-1 shift functions?

3.17 The equations for the variables q and r for S-DES are defined in the section on S-DES analysis. Provide the equations for s and t.

3.18 Using S-DES, decrypt the string (10100010) using the key (0111111101) by hand. Show intermediate results after each function (IP, F_K, SW, F_K, IP^{-1}). Then decode the first 4 bits of the plaintext string to a letter and the second 4 bits to another letter where we encode A through P in base 2 (i.e., A = 0000, B = 0001,..., P = 1111).

Hint: As a midway check, after the application of SW, the string should be (00010011).

Programming Problems

3.19 Create software that can encrypt and decrypt using a general substitution block cipher.

3.20 Create software that can encrypt and decrypt using S-DES. Test data: Use plaintext, ciphertext, and key of Problem 3.15.

PREV NEXT