INTRODUCTION TO CRYPTOGRAPHY AND NETWORK SECURITY

McGraw-Hill Forouzan Networking Series

Titles by Behrouz A. Forouzan:

Cryptography and Network Security
Data Communications and Networking
TCP/IP Protocol Suite
Local Area Networks
Business Data Communications

INTRODUCTION TO CRYPTOGRAPHY AND NETWORK SECURITY

Behrouz A. Forouzan

McGraw Hill Higher Education

Boston Burr Ridge, IL Dubuque, IA New York San Francisco St. Louis
Bangkok Bogotá Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto

The McGraw·Hill Companies

Higher Education

INTRODUCTION TO CRYPTOGRAPHY AND NETWORK SECURITY

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7

ISBN 978–0–07–287022–0
MHID 0–07–287022–2

Publisher: *Alan R. Apt*
Executive Marketing Manager: *Michael Weitz*
Senior Project Manager: *Sheila M. Frank*
Senior Production Supervisor: *Kara Kudronowicz*
Associate Media Producer: *Christina Nelson*
Senior Designer: *David W. Hash*
Cover/Interior Designer: *Rokusek Design*
(USE) Cover Image: *©John Still/Photonica–Getty Images*
Compositor: *ICC Macmillan Inc.*
Typeface: *10/12 Times Roman*
Printer: *R. R. Donnelley Crawfordsville, IN*

Library of Congress Cataloging-in-Publication Data

Forouzan, Behrouz A.
Introduction to cryptography and network security / Behrouz A. Forouzan.—1st ed.
p. cm.
Includes index.
ISBN 978-0-07-287022-0—ISBN 0-07-287022-2
1. Computer networks–Security measures. 2. Cryptography. I. Title.

TK5105.59.F672 2008
005.8–dc22

2006052665

www.mhhe.com

To my beloved daughter and son-in-law, Satara and Shane.

B. Forouzan

CONTENTS

Preface

The Internet, as a worldwide communication network, has changed our daily life in many ways. A new paradigm of commerce allows individuals to shop online. The World Wide Web (WWW) allows people to share information. The E-mail technology connect people in far-flung corners of the world. This inevitable evolution has also created dependency on the Internet.

The Internet, as an open forum, has created some security problems. Confidentiality, integrity, and authentication are needed. People need to be sure that their Internet communication is kept confidential. When they shop online, they need to be sure that the vendors are authentic. When they send their transactions request to their banks, they want to be certain that the integrity of the message is preserved.

Network security is a set of protocols that allow us to use the Internet comfortably—without worrying about security attacks. The most common tool for providing network security is cryptography, an old technique that has been revived and adapted to network security. This book first introduces the reader to the principles of cryptography and then applies those principles to describe network security protocols.

Features of the Book

Several features of this text are designed to make it particularly easy for readers to understand cryptography and network security.

Structure

This text uses an incremental approach to teaching cryptography and network security. It assumes no particular mathematical knowledge, such as number theory or abstract algebra. However, because cryptography and network security cannot be discussed without some background in these areas of mathematics, these topics are discussed in Chapters 2, 4, and 9. Readers who are familiar with these areas of mathematics can ignore these chapters. Chapters 1 through 15 discuss cryptography. Chapters 16 through 18 discuss network security.

Visual Approach

This text presents highly technical subject matters without complex formulas by using a balance of text and figures. More than 400 figures accompanying the text provide a visual and intuitive opportunity for understanding the materials. Figures are particularly important in explaining difficult cryptographic concepts and complex network security protocols.

Algorithms

Algorithms play an important role in teaching cryptography. To make the presentation independent from any computer language, the algorithms have been given in pseudocode that can be easily programmed in a modern language. At the website for this text, the corresponding programs are available for download.

Highlighted Points

Important concepts are emphasized in highlighted boxes for quick reference and immediate attention.

Examples

Each chapter presents a large number of examples that apply concepts discussed in the chapter. Some examples merely show the immediate use of concepts and formulae; some show the actual input/output relationships of ciphers; others give extra information to better understand some difficult ideas.

Recommended Reading

At the end of each chapter, the reader will find a list of books for further reading.

Key Terms

Key terms appear in bold in the chapter text, and a list of key terms appear at the end of each chapter. All key terms are also defined in the glossary at the end of the book.

Summary

Each chapter ends with a summary of the material covered in that chapter. The summary provides a brief overview of all the important points in the chapter.

Practice Set

At the end of each chapter, the students will find a practice set designed to reinforce and apply salient concepts. The practice set consists of two parts: review questions and exercises. The review questions are intended to test the reader's first-level understanding of the material presented in the chapter. The exercises require deeper understanding of the material.

Appendices

The appendices provide quick reference material or a review of materials needed to understand the concepts discussed in the book. Some discussions of mathematical topics are also presented in the appendices to avoid distracting those readers who are already familiar with these materials.

Proofs

Mathematical facts are mentioned in the chapters without proofs to emphasize the results of applying the facts. For those interested reader the proofs are given in Appendix Q.

Glossary and Acronyms

At the end of the text, the reader will find an extensive glossary and a list of acronyms.

Contents

After the introductory Chapter 1, the book is divided into four parts:

Part One: Symmetric-Key Encipherment

Part One introduces the symmetric-key cryptography, both traditional and modern. The chapters in this part emphasize the use of symmetric-key cryptography in providing secrecy. Part One includes Chapters 2 through 8.

Part Two: Asymmetric-Key Encipherment

Part Two discusses asymmetric-key cryptography. The chapters in this part show how asymmetric-key cryptography can provide security. Part Two includes Chapters 9 and 10.

Part Three: Integrity, Authentication, and Key Management

Part Three shows how cryptographic hashing functions can provide other security services, such as message integrity and authentication. The chapters in this part also show how asymmetric-key and symmetric-key cryptography can complement each other. Part Three includes Chapters 11 through 15.

Part Four: Network Security

Part Four shows how the cryptography discussed in Part One through Three can be used to create network security protocols at three levels of the Internet networking model. Part Four includes Chapters 16 to 18.

How to Use this Book

This book is written for both an academic and a professional audience. Interested professionals can use it for self-guidance study. As a textbook, it can be used for a one-semester or one-quarter course. The following are some guidelines.

- Parts one to three are strongly recommended.
- Part four is recommended if the course needs to move beyond cryptography and enter the domain of network security. A course in networking is a prerequisite for Part four.

Online Learning Center

The McGraw-Hill Online Learning Center contains much additional material related to *Cryptography and Network Security*. Readers can access the site at www.mhhe.com/forouzan. Professors and students can access lecture materials, such as Power Point slides. The solutions to odd-numbered problems are provided to students, and professors can use a password to access the complete set of solutions. Additionally, McGraw-Hill makes it easy to create a website for the course with an exclusive McGraw-Hill product called PageOut. It requires no prior knowledge of HTML, no long hours, and no design skills on your part. Instead, PageOut offers a series of templates. Simply fill them with your course information and click on one of 16 designs. The process takes under an hour and leaves you with a professionally designed website. Although PageOut offers "instant" development, the finished website provides powerful features. An interactive course syllabus allows you to post content to coincide with your lectures, so when students visit your PageOut website, your syllabus will direct them to components of Forouzan's Online Learning Center, or specific material of your own.

Acknowledgments

It is obvious that the development of a book of this scope needs the support of many people.

Peer Review

The most important contribution to the development of a book such as this comes from peer reviews. I cannot express my gratitude in words to the many reviewers who spent numerous hours reading the manuscript and providing me with helpful comments and ideas. I would especially like to acknowledge the contributions of the following reviewers:

Kaufman, Robert, *University of Texas, San Antonio*
Kesidis, George, *Penn State*
Stephens, Brooke, *U. of Maryland, Baltimore County*
Koc, Cetin, *Oregon State University*
Uminowicz, Bill, *Westwood College*
Wang, Xunhua, *James Madison University*
Kak, Subhash, *Louisiana State U.*
Dunigan, Tom, *U. of Tennessee, Knoxville*

McGraw-Hill Staff

Special thanks go to the staff of McGraw-Hill. Alan Apt, publisher, proved how a proficient publisher can make the impossible possible. Melinda Bilecki, the developmental editor, gave me help whenever I needed it. Sheila Frank, project manager, guided me through the production process with enormous enthusiasm. I also thank David Hash in design, Kara Kudronowicz in production, and Wendy Nelson, the copy editor.

Behrouz A. Forouzan

CHAPTER 1

Introduction

Objectives

This chapter has several objectives:

- To define three security goals
- To define security attacks that threaten security goals
- To define security services and how they are related to the three security goals
- To define security mechanisms to provide security services
- To introduce two techniques, cryptography and steganography, to implement security mechanisms.

We are living in the information age. We need to keep information about every aspect of our lives. In other words, information is an asset that has a value like any other asset. As an asset, information needs to be secured from attacks.

To be secured, information needs to be hidden from unauthorized access (*confidentiality*), protected from unauthorized change (*integrity*), and available to an authorized entity when it is needed (*availability*).

Until a few decades ago, the information collected by an organization was stored on physical files. The confidentiality of the files was achieved by restricting the access to a few authorized and trusted people in the organization. In the same way, only a few authorized people were allowed to change the contents of the files. Availability was achieved by designating at least one person who would have access to the files at all times.

With the advent of computers, information storage became electronic. Instead of being stored on physical media, it was stored in computers. The three security requirements, however, did not change. The files stored in computers require confidentiality, integrity, and availability. The implementation of these requirements, however, is different and more challenging.

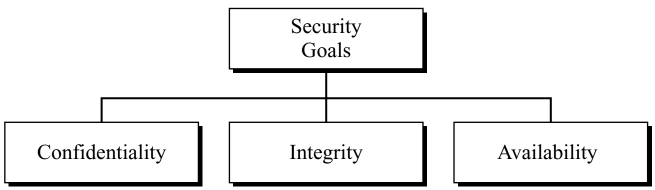
During the last two decades, computer networks created a revolution in the use of information. Information is now distributed. Authorized people can send and retrieve information from a distance using computer networks. Although the three above-mentioned requirements—confidentiality, integrity, and availability—have not changed, they now have some new dimensions. Not only should information be confidential when it is stored in a computer; there should also be a way to maintain its confidentiality when it is transmitted from one computer to another.

In this chapter, we first discuss the three major goals of information security. We then see how attacks can threaten these three goals. We then discuss the security services in relation to these security goals. Finally we define mechanisms to provide security services and introduce techniques that can be used to implement these mechanisms.

1.1 SECURITY GOALS

Let us first discuss three **security goals: confidentiality, integrity,** and **availability** (Figure 1.1).

Figure 1.1 *Taxonomy of security goals*



Confidentiality

Confidentiality is probably the most common aspect of information security. We need to protect our confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information. In the military, concealment of sensitive information is the major concern. In industry, hiding some information from competitors is crucial to the operation of the organization. In banking, customers' accounts need to be kept secret.

As we will see later in this chapter, confidentiality not only applies to the storage of the information, it also applies to the transmission of information. When we send a piece of information to be stored in a remote computer or when we retrieve a piece of information from a remote computer, we need to conceal it during transmission.

Integrity

Information needs to be changed constantly. In a bank, when a customer deposits or withdraws money, the balance of her account needs to be changed. **Integrity** means that changes need to be done only by authorized entities and through authorized mechanisms. Integrity violation is not necessarily the result of a malicious act; an interruption in the system, such as a power surge, may also create unwanted changes in some information.

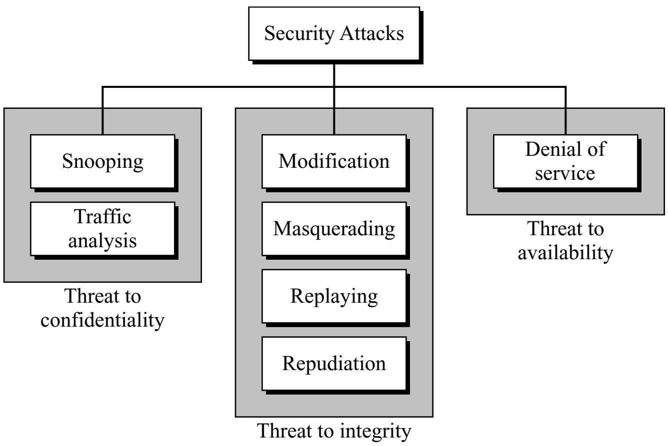
Availability

The third component of information security is **availability.** The information created and stored by an organization needs to be available to authorized entities. Information is useless if it is not available. Information needs to be constantly changed, which means it must be accessible to authorized entities. The unavailability of information is just as harmful for an organization as the lack of confidentiality or integrity. Imagine what would happen to a bank if the customers could not access their accounts for transactions.

1.2 ATTACKS

Our three goals of security—confidentiality, integrity, and availability—can be threatened by security **attacks.** Although the literature uses different approaches to categorizing the attacks, we will first divide them into three groups related to the security goals. Later, we will divide them into two broad categories based on their effects on the system. Figure 1.2 shows the first taxonomy.

Figure 1.2 *Taxonomy of attacks with relation to security goals*



Attacks Threatening Confidentiality

In general, two types of attacks threaten the confidentiality of information: **snooping** and **traffic analysis.**

Snooping

Snooping refers to unauthorized access to or interception of data. For example, a file transferred through the Internet may contain confidential information. An unauthorized entity may intercept the transmission and use the contents for her own benefit. To prevent snooping, the data can be made nonintelligible to the intercepter by using encipherment techniques discussed in this book.

Traffic Analysis

Although encipherment of data may make it nonintelligible for the intercepter, she can obtain some other type information by monitoring online traffic. For example, she can find the electronic address (such as the e-mail address) of the sender or the receiver. She can collect pairs of requests and responses to help her guess the nature of transaction.

Attacks Threatening Integrity

The integrity of data can be threatened by several kinds of attacks: **modification, masquerading, replaying,** and **repudiation.**

Modification

After intercepting or accessing information, the attacker modifies the information to make it beneficial to herself. For example, a customer sends a message to a bank to do some transaction. The attacker intercepts the message and changes the type of transaction to benefit herself. Note that sometimes the attacker simply deletes or delays the message to harm the system or to benefit from it.

Masquerading

Masquerading, or spoofing, happens when the attacker impersonates somebody else. For example, an attacker might steal the bank card and PIN of a bank customer and pretend that she is that customer. Sometimes the attacker pretends instead to be the receiver entity. For example, a user tries to contact a bank, but another site pretends that it is the bank and obtains some information from the user.

Replaying

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

Repudiation

This type of attack is different from others because it is performed by one of the two parties in the communication: the sender or the receiver. The sender of the message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.

An example of denial by the sender would be a bank customer asking her bank to send some money to a third party but later denying that she has made such a request. An example of denial by the receiver could occur when a person buys a product from a manufacturer and pays for it electronically, but the manufacturer later denies having received the payment and asks to be paid.

Attacks Threatening Availability

We mention only one attack threatening availability: **denial of service.**

Denial of Service

Denial of service (DoS) is a very common attack. It may slow down or totally interrupt the service of a system. The attacker can use several strategies to achieve this. She might send so many bogus requests to a server that the server crashes because of the heavy load. The attacker might intercept and delete a server's response to a client, making the client to believe that the server is not responding. The attacker may also intercept requests from the clients, causing the clients to send requests many times and overload the system.

Passive Versus Active Attacks

Let us now categorize the attacks into two groups: passive and active. Table 1.1 shows the relationship between this and the previous categorization.

Table 1.1 *Categorization of passive and active attacks*

Attacks	*Passive/Active*	*Threatening*
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

Passive Attacks

In a **passive attack,** the attacker's goal is just to obtain information. This means that the attack does not modify data or harm the system. The system continues with its normal operation. However, the attack may harm the sender or the receiver of the message. Attacks that threaten confidentiality—snooping and traffic analysis—are passive attacks. The revealing of the information may harm the sender or receiver of the message, but the system is not affected. For this reason, it is difficult to detect this type of attack until the sender or receiver finds out about the leaking of confidential information. Passive attacks, however, can be prevented by encipherment of the data.

Active Attacks

An **active attack** may change the data or harm the system. Attacks that threaten the integrity and availability are active attacks. Active attacks are normally easier to detect than to prevent, because an attacker can launch them in a variety of ways.

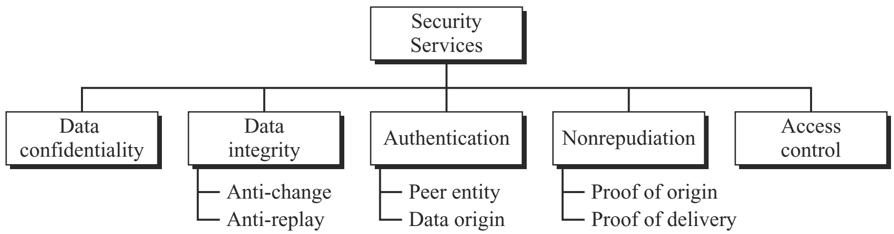
1.3 SERVICES AND MECHANISMS

The **International Telecommunication Union-Telecommunication Standardization Sector (ITU-T)** (see Appendix B) provides some security services and some mechanisms to implement those services. Security services and mechanisms are closely related because a mechanism or combination of mechanisms are used to provide a service. Also, a mechanism can be used in one or more services. We briefly discuss them here to give the general idea; we will discuss them in detail in later chapters devoted to specific services or mechanisms.

Security Services

ITU-T (X.800) has defined five services related to the security goals and attacks we defined in the previous sections. Figure 1.3 shows the taxonomy of those five common services.

Figure 1.3 *Security services*



It is easy to relate one or more of these services to one or more of the security goals. It is also easy to see that these services have been designed to prevent the security attacks that we have mentioned.

Data Confidentiality

Data confidentiality is designed to protect data from disclosure attack. The service as defined by X.800 is very broad and encompasses confidentiality of the whole message or part of a message and also protection against traffic analysis. That is, it is designed to prevent snooping and traffic analysis attack.

Data Integrity

Data integrity is designed to protect data from modification, insertion, deletion, and replaying by an adversary. It may protect the whole message or part of the message.

Authentication

This service provides the **authentication** of the party at the other end of the line. In connection-oriented communication, it provides authentication of the sender or receiver during the connection establishment (peer entity authentication). In connectionless communication, it authenticates the source of the data (data origin authentication).

Nonrepudiation

Nonrepudiation service protects against repudiation by either the sender or the receiver of the data. In nonrepudiation with proof of the origin, the receiver of the data can later prove the identity of the sender if denied. In nonrepudiation with proof of delivery, the sender of data can later prove that data were delivered to the intended recipient.

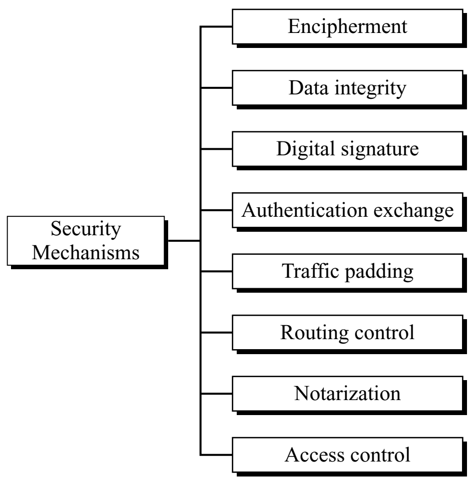
Access Control

Access control provides protection against unauthorized access to data. The term *access* in this definition is very broad and can involve reading, writing, modifying, executing programs, and so on.

Security Mechanisms

ITU-T (X.800) also recommends some **security mechanisms** to provide the security services defined in the previous section. Figure 1.4 gives the taxonomy of these mechanisms.

Figure 1.4 *Security mechanisms*



Encipherment

Encipherment, hiding or covering data, can provide confidentiality. It can also be used to complement other mechanisms to provide other services. Today two techniques—cryptography and steganography—are used for enciphering. We will discuss these shortly.

Data Integrity

The **data integrity** mechanism appends to the data a short checkvalue that has been created by a specific process from the data itself. The receiver receives the data and the checkvalue. He creates a new checkvalue from the received data and compares the newly created checkvalue with the one received. If the two checkvalues are the same, the integrity of data has been preserved.

Digital Signature

A **digital signature** is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature. The sender uses a process that involves showing that she owns a private key related to the public key that she has announced publicly. The receiver uses the sender's public key to prove that the message is indeed signed by the sender who claims to have sent the message.

Authentication Exchange

In **authentication exchange,** two entities exchange some messages to prove their identity to each other. For example, one entity can prove that she knows a secret that only she is supposed to know.

Traffic Padding

Traffic padding means inserting some bogus data into the data traffic to thwart the adversary's attempt to use the traffic analysis.

Routing Control

Routing control means selecting and continuously changing different available routes between the sender and the receiver to prevent the opponent from eavesdropping on a particular route.

Notarization

Notarization means selecting a third trusted party to control the communication between two entities. This can be done, for example, to prevent repudiation. The receiver can involve a trusted party to store the sender request in order to prevent the sender from later denying that she has made such a request.

Access Control

Access control uses methods to prove that a user has access right to the data or resources owned by a system. Examples of proofs are passwords and PINs.

Relation between Services and Mechanisms

Table 1.2 shows the relationship between the security services and the security mechanisms. The table shows that three mechanisms (encipherment, digital signature, and authentication exchange) can be used to provide authentication. The table also shows

Table 1.2 *Relation between security services and security mechanisms*

Security Service	*Security Mechanism*
Data confidentiality	Encipherment and routing control
Data integrity	Encipherment, digital signature, data integrity
Authentication	Encipherment, digital signature, authentication exchanges
Nonrepudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

that encipherment mechanism may be involved in three services (data confidentiality, data integrity, and authentication)

1.4 TECHNIQUES

Mechanisms discussed in the previous sections are only theoretical recipes to implement security. The actual implementation of security goals needs some techniques. Two techniques are prevalent today: one is very general (cryptography) and one is specific (steganography).

Cryptography

Some security mechanisms listed in the previous section can be implemented using cryptography. **Cryptography,** a word with Greek origins, means "secret writing." However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks. Although in the past *cryptography* referred only to the **encryption** and **decryption** of messages using secret keys, today it is defined as involving three distinct mechanisms: symmetric-key encipherment, asymmetric-key encipherment, and hashing. We will briefly discuss these three mechanisms here.

Symmetric-Key Encipherment

In **symmetric-key encipherment** (sometimes called secret-key encipherment or secret-key cryptography), an entity, say Alice, can send a message to another entity, say Bob, over an insecure channel with the assumption that an adversary, say Eve, cannot understand the contents of the message by simply eavesdropping over the channel. Alice encrypts the message using an encryption algorithm; Bob decrypts the message using a decryption algorithm. Symmetric-key encipherment uses a single **secret key** for both encryption and decryption. Encryption/decryption can be thought of as electronic locking. In symmetric-key enciphering, Alice puts the message in a box and locks the box using the shared secret key; Bob unlocks the box with the same key and takes out the message.

Asymmetric-Key Encipherment

In **asymmetric-key encipherment** (sometimes called public-key encipherment or public-key cryptography), we have the same situation as the symmetric-key encipherment, with a few exceptions. First, there are two keys instead of one: one **public key** and one **private key.** To send a secured message to Bob, Alice first encrypts the message using Bob's public key. To decrypt the message, Bob uses his own private key.

Hashing

In **hashing,** a fixed-length message digest is created out of a variable-length message. The digest is normally much smaller than the message. To be useful, both the message and the digest must be sent to Bob. Hashing is used to provide checkvalues, which were discussed earlier in relation to providing data integrity.

Steganography

Although this book is based on cryptography as a technique for implementing security mechanisms, another technique that was used for secret communication in the past is being revived at the present time: steganography. The word **steganography,** with origin in Greek, means "covered writing," in contrast with cryptography, which means "secret writing." Cryptography means concealing the contents of a message by enciphering; steganography means concealing the message itself by covering it with something else.

Historical Use

History is full of facts and myths about the use of steganography. In China, war messages were written on thin pieces of silk and rolled into a small ball and swallowed by the messenger. In Rome and Greece, messages were carved on pieces of wood, that were later dipped into wax to cover the writing. Invisible inks (such as onion juice or ammonia salts) were also used to write a secret message between the lines of the covering message or on the back of the paper; the secret message was exposed when the paper was heated or treated with another substance.

In recent times other methods have been devised. Some letters in an innocuous message might be overwritten in a pencil lead that is visible only when exposed to light at an angle. Null ciphers were used to hide a secret message inside an innocuous simple message. For example, the first or second letter of each word in the covering message might compose a secret message. Microdots were also used for this purpose. Secret messages were photographed and reduced to a size of a dot (period) and inserted into simple cover messages in place of regular periods at the end of sentences.

Modern Use

Today, any form of data, such as text, image, audio, or video, can be digitized, and it is possible to insert secret binary information into the data during digitization process. Such hidden information is not necessarily used for secrecy; it can also be used to protect copyright, prevent tampering, or add extra information.

Text Cover The cover of secret data can be text. There are several ways to insert binary data into an innocuous text. For example, we can use single space between words to represent the binary digit 0 and double space to represent binary digit 1. The following short message hides the 8-bit binary representation of the letter A in ASCII code (01000001).

This book is mostly about cryptography, not steganography.

□ □□ □ □ □ □ □□

0 1 0 0 0 0 1

In the above message there are two spaces between the “book” and “is” and between the “not” and “steganography”. Of course, sophisticated software can insert spaces that differ only slightly to hide the code from immediate recognition.

Another, more efficient method, is to use a dictionary of words organized according to their grammatical usages. We can have a dictionary containing 2 articles, 8 verbs, 32 nouns, and 4 prepositions. Then we agree to use cover text that always use sentences with the pattern *article-noun-verb-article-noun*. The secret binary data can be divided into 16-bit chunks. The first bit of binary data can be represented by an article (for example, 0 for *a* and 1 for *the*). The next five bits can be represented by a noun (subject of the sentence), the next four bits can be represented by a verb, the next bit by the second article, and the last five bits by another noun (object). For example, the secret data “Hi”, which is 01001000 01001001 in ASCII, could be a sentence like the following:

A	**friend**	**called**	**a**	**doctor.**
0	10010	0001	0	01001

This is a very trivial example. The actual approach uses more sophisticated design and a variety of patterns.

Image Cover Secret data can also be covered under a color image. Digitized images are made of pixels (picture elements), in which normally each pixel uses 24 bits (three bytes). Each byte represents one of the primary colors (red, green, or blue). We can therefore have 2^8 different shades of each color. In a method called LSB (least significant bit), the least significant bit of each byte is set to zero. This may make the image a little bit lighter in some areas, but this is not normally noticed. Now we can hide a binary data in the image by keeping or changing the least significant bit. If our binary digit is 0, we keep the bit; if it is 1, we change the bit to 1. In this way, we can hide a character (eight ASCII bits) in three pixels. For example, the following three pixels can represent the letter M.

0101001**1**	1011110**0**	0101010**1**
0101111**0**	1011110**0**	0110010**1**
0111111**0**	0100101**0**	0001010**1**

Of course, more sophisticated approaches are used these days.

Other Covers Other covers are also possible. The secret message, for example, can be covered under audio (sound and music) and video. Both audio and video are compressed today; the secret data can be embedded during or before the compression. We leave the discussion of these techniques to more specialized books in steganography.

1.5 THE REST OF THE BOOK

The rest of this book is divided into four parts.

Part One: Symmetric-Key Encipherment

The chapters in Part One discuss encipherment, both classic and modern, using symmetric-key cryptography. These chapters show how the first goal of security can be implemented using this technique.

Part Two: Asymmetric-Key Encipherment

The chapters in Part Two discuss encipherment using asymmetric-key cryptography. These chapters also show how the first goal of the security can be implemented using this technique.

Part Three: Integrity, Authentication, and Key Management

The chapters in Part Three introduce the third application of cryptography—hashing—and show how it can be combined with the materials discussed in Part I and II for implementing the second goal of security.

Part Four: Network Security

The chapters in Part Four show how the methods learned in the first three parts of the book can be combined to create network security using the Internet model.

1.6 RECOMMENDED READING

For more details about subjects discussed in this chapter, the following books and websites are good places to start. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books discuss security goals, attacks, and mechanisms. We recommend [Bis05] and [Sta06].

WebSites

The following websites give more information about topics discussed in this chapter.

http://www.faqs.org/rfcs/rfc2828.html
fag.grm.hia.no/IKT7000/litteratur/paper/x800.pdf

1.7 KEY TERMS

access control
active attack
asymmetric-key encipherment
authentication
authentication exchange
availability
confidentiality
cryptography
data confidentiality
data integrity
decryption
denial of service
digital signature
encipherment
encryption
hashing
integrity
International Telecommunication Union-Telecommunication Standardization Sector (ITU-T)
masquerading
modification
nonrepudiation
notarization
passive attack
private key
public key
replaying
repudiation
routing control
secret key
security attacks
security goals
security mechanisms
snooping
steganography
symmetric-key encipherment
traffic analysis
traffic padding

1.8 SUMMARY

- Three general goals have been defined for security: confidentiality, integrity, and availability.
- Two types of attacks threaten the confidentiality of information: snooping and traffic analysis. Four types of attacks can threaten the integrity of information: modification, masquerading, replaying, and repudiation. Denial-of-service attacks threaten the availability of information.
- Some organizations involved in data communication and networking, such as ITU-T or the Internet, have defined several security services that are related to the security goals and security attacks. This chapter discussed five common security services: data confidentiality, data integrity, authentication, nonrepudiation, and access control.
- ITU-T also recommends some mechanisms to provide security. We discussed eight of these mechanisms: encipherment, data integrity, digital signature, authentication exchange, traffic padding, routing control, notarization, and access control.

- ❏ There are two techniques—cryptography and steganography—that can implement some or all of the mechanisms. Cryptography or "secret writing" involves scrambling a message or creating a digest of the message. Steganography or "covered writing" means concealing the message by covering it with something else.

1.9 PRACTICE SET

Review Questions

1. Define the three security goals.
2. Distinguish between passive and active security attacks. Name some passive attacks. Name some active attacks.
3. List and define five security services discussed in this chapter.
4. Define eight security mechanisms discussed in this chapter.
5. Distinguish between cryptography and steganography.

Exercises

6. Which security service(s) are guaranteed when using each of the following methods to send mail at the post office?
   a. Regular mail
   b. Regular mail with delivery confirmation
   c. Regular mail with delivery and recipient signature
   d. Certified mail
   e. Insured mail
   f. Registered mail
7. Define the type of security attack in each of the following cases:
   a. A student breaks into a professor's office to obtain a copy of the next day's test.
   b. A student gives a check for $10 to buy a used book. Later she finds that the check was cashed for $100.
   c. A student sends hundreds of e-mails per day to another student using a phony return e-mail address.
8. Which security mechanism(s) are provided in each of the following cases?
   a. A school demands student identification and a password to let students log into the school server.
   b. A school server disconnects a student if she is logged into the system for more than two hours.
   c. A professor refuses to send students their grades by e-mail unless they provide student identification they were preassigned by the professor.
   d. A bank requires the customer's signature for a withdrawal.

9. Which technique (cryptography or steganography) is used in each of the following cases for confidentiality?
   a. A student writes the answers to a test on a small piece of paper, rolls up the paper, and inserts it in a ball-point pen, and passes the pen to another student.
   b. To send a message, a spy replaces each character in the message with a symbol that was agreed upon in advance as the character's replacement.
   c. A company uses special ink on its checks to prevent forgeries.
   d. A graduate student uses watermarks to protect her thesis, which is posted on her website.
10. What type of security mechanism(s) are provided when a person signs a form he has filled out to apply for a credit card?

PART 1

Symmetric-Key Encipherment

In Chapter 1, we saw that cryptography provides three techniques: symmetric-key ciphers, asymmetric-key ciphers, and hashing. Part One is devoted to symmetric-key ciphers. Chapters 2 and 4 review the mathematical background necessary for understanding the rest of the chapters in this part. Chapter 3 explores the traditional ciphers used in the past. Chapters 5, 6, and 7 explain modern block ciphers that are used today. Chapter 8 shows how modern block and stream ciphers can be used to encipher long messages.

Chapter 2: Mathematics of Cryptography: Part I

Chapter 2 reviews some mathematical concepts needed to understand the next few chapters. It discusses integer and modular arithmetic, matrices, and congruence relations.

Chapter 3: Traditional Symmetric-Key Ciphers

Chapter 3 introduces traditional symmetric-key ciphers. Although these ciphers are not used today, they are the foundation of modern symmetric-key ciphers. This chapter emphasizes the two categories of traditional ciphers: substitution ciphers and transposition ciphers. It also introduces the concepts of stream ciphers and block ciphers.

Chapter 4: Mathematics of Cryptography: Part II

Chapter 4 is another review of mathematics needed to understand the contents of the subsequent chapters. It reviews some algebraic structures, such as groups, rings, and finite fields, which are used in modern block ciphers.

Chapter 5: Introduction to Modern Symmetric-Key Ciphers

Chapter 5 is an introduction to modern symmetric-key ciphers. Understanding the individual elements used in modern symmetric-key ciphers paves the way to a better understanding and analysis of modern ciphers. This chapter introduces components of block ciphers such as P-boxes and S-boxes. It also distinguishes between two classes of product ciphers: Feistel and non-Feistel ciphers.

Chapter 6: Data Encryption Standard (DES)

Chapter 6 uses the elements defined in Chapter 5 to discuss and analyze one of the common symmetric-key ciphers used today, the Data Encryption Standard (DES). The emphasis is on how DES uses 16 rounds of Feistel ciphers.

Chapter 7: Advanced Encryption Standard (AES)

Chapter 7 shows how some algebraic structures discussed in Chapter 4 and some elements discussed in Chapter 5 can create a very strong cipher, the Advanced Encryption Standard (AES). The emphasis is on how the algebraic structures discussed in Chapter 4 achieve the AES security goals.

Chapter 8: Encipherment Using Modern Symmetric-Key Ciphers

Chapter 8 shows how modern block and stream ciphers can actually be used to encipher long messages. It explains five modes of operation designed to be used with modern block ciphers. It also introduces two stream ciphers used for real-time processing of data.

CHAPTER 2

Mathematics of Cryptography

Part I: Modular Arithmetic, Congruence, and Matrices

Objectives

This chapter is intended to prepare the reader for the next few chapters in cryptography. The chapter has several objectives:

- To review integer arithmetic, concentrating on divisibility and finding the greatest common divisor using the Euclidean algorithm
- To understand how the extended Euclidean algorithm can be used to solve linear Diophantine equations, to solve linear congruent equations, and to find the multiplicative inverses
- To emphasize the importance of modular arithmetic and the modulo operator, because they are extensively used in cryptography
- To emphasize and review matrices and operations on residue matrices that are extensively used in cryptography
- To solve a set of congruent equations using residue matrices

Cryptography is based on some specific areas of mathematics, including number theory, linear algebra, and algebraic structures. In this chapter, we discuss only the topics in the above areas that are needed to understand the contents of the next few chapters. Readers who are familiar with these topics can skip this chapter entirely or partially. Similar chapters are provided throughout the book when needed. Proofs of theorems and algorithms have been omitted, and only their applications are shown. The interested reader can find proofs of the theorems and algorithms in Appendix Q.

Proofs of theorems and algorithms discussed in this chapter can be found in Appendix Q.

2.1 INTEGER ARITHMETIC

In **integer arithmetic,** we use a set and a few operations. You are familiar with this set and the corresponding operations, but they are reviewed here to create a background for modular arithmetic.

Set of Integers

The **set of integers,** denoted by **Z,** contains all integral numbers (with no fraction) from negative infinity to positive infinity (Figure 2.1).

Figure 2.1 *The set of integers*

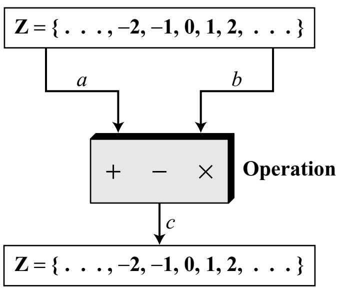
$$\mathbf{Z} = \{\ \ldots,\ -2,\ -1,\ 0,\ 1,\ 2,\ \ldots\ \}$$

Binary Operations

In cryptography, we are interested in three binary operations applied to the set of integers. A **binary operation** takes two inputs and creates one output. Three common binary operations defined for integers are *addition, subtraction,* and *multiplication*. Each of these operations takes two inputs (a and b) and creates one output (c) as shown in Figure 2.2. The two inputs come from the set of integers; the output goes into the set of integers.

Note that *division* does not fit in this category because, as we will see shortly, it produces two outputs instead of one.

Figure 2.2 *Three binary operations for the set of integers*



Example 2.1

The following shows the results of the three binary operations on two integers. Because each input can be either positive or negative, we can have four cases for each operation.

Add:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
Subtract:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
Multiply:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

Integer Division

In integer arithmetic, if we divide a by n, we can get q and r. The relationship between these four integers can be shown as

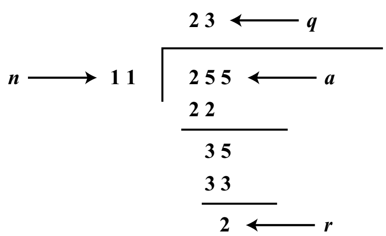
$$a = q \times n + r$$

In this relation, a is called the *dividend;* q, the *quotient;* n, the *divisor;* and r, the *remainder*. Note that this is not an operation, because the result of dividing a by n is two integers, q and r. We can call it *division relation*.

Example 2.2

Assume that $a = 255$ and $n = 11$. We can find $q = 23$ and $r = 2$ using the division algorithm we have learned in arithmetic as shown in Figure 2.3.

Figure 2.3 *Example 2.2, finding the quotient and the remainder*

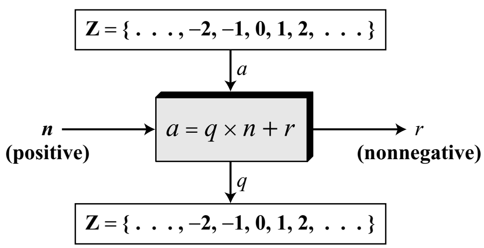


Most computer languages can find the quotient and the remainder using language-specific operators. For example, in the C language, the operator / can find the quotient and the operator % can find the remainder.

Two Restrictions

When we use the above division relationship in cryptography, we impose two restrictions. First, we require that the divisor be a positive integer ($n > 0$). Second, we require that the remainder be a nonnegative integer ($r \geq 0$). Figure 2.4 shows this relationship with the two above-mentioned restrictions.

Figure 2.4 *Division algorithm for integers*



Example 2.3

When we use a computer or a calculator, r and q are negative when a is negative. How can we apply the restriction that r needs to be positive? The solution is simple, we decrement the value of q by 1 and we add the value of n to r to make it positive.

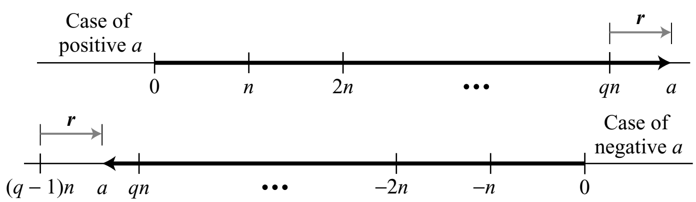
$$-255 = (\mathbf{-23} \times 11) + (\mathbf{-2}) \qquad \leftrightarrow \qquad -255 = (\mathbf{-24} \times 11) + \mathbf{9}$$

We have decremented −23 to become −24 and added 11 to −2 to make it 9. The above relation is still valid.

The Graph of the Relation

We can show the above relation with the two restrictions on n and r using two graphs in Figure 2.5. The first one shows the case when a is positive; the second when a is negative.

Figure 2.5 *Graph of division algorithm*



Starting from zero, the graph shows how we can reach the point representing the integer a on the line. In case of a positive a, we need to move $q \times n$ units to the right and then move extra r units in the same direction. In case of a negative a, we need to move $(q - 1) \times n$ units to the left (q is negative in this case) and then move r units in the opposite direction. In both cases the value of r is positive.

Divisibility

Let us briefly discuss **divisibility,** a topic we often encounter in cryptography. If a is not zero and we let $r = 0$ in the division relation, we get

$$\boldsymbol{a = q \times n}$$

We then say that n divides a (or n is a divisor of a). We can also say that a is divisible by n. When we are not interested in the value of q, we can write the above relationship as $\boldsymbol{a|n}$. If the remainder is not zero, then n does not divide a and we can write the relationship as $\boldsymbol{a \nmid n}$.

Example 2.4

a. The integer 4 divides the integer 32 because $\mathbf{32} = 8 \times \mathbf{4}$. We show this as $4|32$.
b. The number 8 does not divide the number 42 because $\mathbf{42} = 5 \times \mathbf{8} + 2$. There is a remainder, the number 2, in the equation. We show this as $8 \nmid 42$.

Example 2.5

a. We have $13|78$, $7|98$, $-6|24$, $4|44$, and $11|(-33)$.
b. We have $13 \nmid 27$, $7 \nmid 50$, $-6 \nmid 23$, $4 \nmid 41$, and $11 \nmid (-32)$.

Properties

Following are several properties of divisibility. The interested reader can check Appendix Q for proofs.

Property 1: if $a|1$, then $a = \pm 1$.
Property 2: if $a|b$ and $b|a$, then $a = \pm b$.
Property 3: if $a|b$ and $b|c$, then $a|c$.
Property 4: if $a|b$ and $a|c$, then $a|(m \times b + n \times c)$, where m and n are arbitrary integers.

Example 2.6

a. Since $3|15$ and $15|45$, according to the third property, $3|45$.
b. Since $3|15$ and $3|9$, according to the fourth property, $3|(15 \times 2 + 9 \times 4)$, which means $3|66$.

All Divisors

A positive integer can have more than one divisor. For example, the integer 32 has six divisors: 1, 2, 4, 8, 16, and 32. We can mention two interesting facts about divisors of positive integers:

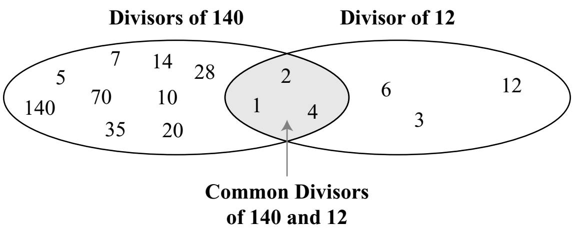
Fact 1: The integer 1 has only one divisor, itself.

Fact 2: Any positive integer has at least two divisors, 1 and itself (but it can have more).

Greatest Common Divisor

One integer often needed in cryptography is the **greatest common divisor** of two positive integers. Two positive integers may have many common divisors, but only one greatest common divisor. For example, the common divisors of 12 and 140 are 1, 2, and 4. However, the greatest common divisor is 4. See Figure 2.6.

Figure 2.6 *Common divisors of two integers*



The greatest common divisor of two positive integers is the largest integer that can divide both integers.

Euclidean Algorithm

Finding the greatest common divisor (gcd) of two positive integers by listing all common divisors is not practical when the two integers are large. Fortunately, more than 2000 years ago a mathematician named Euclid developed an algorithm that can find the greatest common divisor of two positive integers. The **Euclidean algorithm** is based on the following two facts (see Appendix Q for the proof):

Fact 1: gcd (a, 0) = a

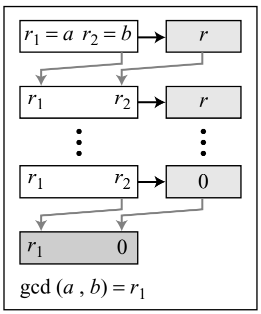
Fact 2: gcd (a, b) = gcd (b, r), where r is the remainder of dividing a by b

The first fact tells us that if the second integer is 0, the greatest common divisor is the first one. The second fact allows us to change the value of a, b until b becomes 0. For example, to calculate the gcd (36, 10), we can use the second fact several times and the first fact once, as shown below.

$$\gcd(36, 10) = \gcd(10, 6) = \gcd(6, 4) = \gcd(4, 2) = \gcd(2, 0) = 2$$

In other words, gcd (36, 10) = 2, gcd (10, 6) = 2, and so on. This means that instead of calculating gcd (36, 10), we can find gcd (2, 0). Figure 2.7 shows how we use the above two facts to calculate gcd (a, b).

Figure 2.7 *Euclidean algorithm*



a. Process

```
r1 ← a; r2 ← b;   (Initialization)
while (r2 > 0)
 {
  q ← r1 / r2;
  r ← r1 − q × r2;
  r1 ← r2; r2 ← r;
 }

  gcd (a, b) ← r1
```

b. Algorithm

We use two variables, r_1 and r_2, to hold the changing values during the process of reduction. They are initialized to a and b. In each step, we calculate the remainder of r_1 divided by r_2 and store the result in the variable r. We then replace r_1 by r_2 and r_2 by r. The steps are continued until r_2 becomes 0. At this moment, we stop. The gcd (a, b) is r_1.

When gcd (a, b) = 1, we say that a and b are relatively prime.

Example 2.7

Find the greatest common divisor of 2740 and 1760.

Solution

We apply the above procedure using a table. We initialize r_1 to 2740 and r_2 to 1760. We have also shown the value of q in each step. We have gcd (2740, 1760) = 20.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

Example 2.8

Find the greatest common divisor of 25 and 60.

Solution

We chose this particular example to show that it does not matter if the first number is smaller than the second number. We immediately get our correct ordering. We have gcd (25, 65) = 5.

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

The Extended Euclidean Algorithm

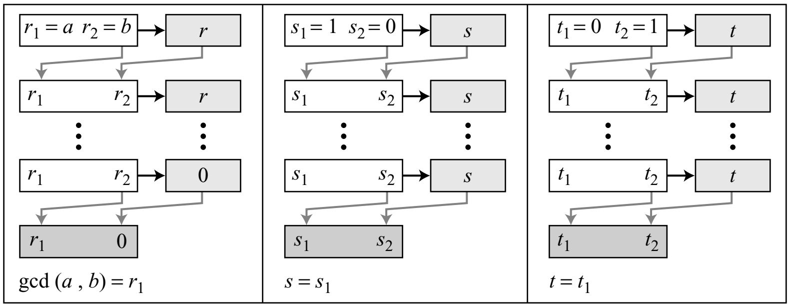
Given two integers a and b, we often need to find other two integers, s and t, such that

$$s \times a + t \times b = \text{gcd}(a, b)$$

The **extended Euclidean algorithm** can calculate the gcd (a, b) and at the same time calculate the value of s and t. The algorithm and the process is shown in Figure 2.8.

As shown in Figure 2.8, the extended Euclidean algorithm uses the same number of steps as the Euclidean algorithm. However, in each step, we use three sets of calculations and exchanges instead of one. The algorithm uses three sets of variables, r's, s's, and t's.

Figure 2.8 *Extended Euclidean algorithm*



a. Process

```
r1 ← a; r2 ← b;
s1 ← 1; s2 ← 0;        (Initialization)
t1 ← 0; t2 ← 1;
while (r2 > 0)
{
  q ← r1 / r2;
    r  ← r1 − q × r2;
    r1 ← r2;   r2 ← r;      (Updating r's)
    s  ← s1 − q × s2;
    s1 ← s2;   s2 ← s;      (Updating s's)
    t  ← t1 − q × t2;
    t1 ← t2;   t2 ← t;      (Updating t's)
}
  gcd (a , b) ← r1;   s ← s1;   t ← t1
```

b. Algorithm

In each step, r_1, r_2, and r have the same values in the Euclidean algorithm. The variables r_1 and r_2 are initialized to the values of a and b, respectively. The variables s_1 and s_2 are initialized to 1 and 0, respectively. The variables t_1 and t_2 are initialized to 0 and 1, respectively. The calculations of r, s, and t are similar, with one warning. Although r is the remainder of dividing r_1 by r_2, there is no such relationship between the other two sets. There is only one quotient, q, which is calculated as r_1/r_2 and used for the other two calculations.

Example 2.9

Given $a = 161$ and $b = 28$, find gcd (a, b) and the values of s and t.

Solution

$$r = r_1 - q \times r_2 \qquad s = s_1 - q \times s_2 \qquad t = t_1 - q \times t_2$$

We use a table to follow the algorithm.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	−5
1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		**−1**	4		**6**	−23	

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$. The answers can be tested because we have

$$(-1) \times 161 + 6 \times 28 = 7$$

Example 2.10

Given $a = 17$ and $b = 0$, find gcd (a, b) and the values of s and t.

Solution

We use a table to follow the algorithm.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
	17	0		**1**	**0**		**0**	**1**	

Note that we need no calculation for q, r, and s. The first value of r_2 meets our termination condition. We get gcd (17, 0) = 17, $s = 1$, and $t = 0$. This indicates why we should initialize s_1 to 1 and t_1 to 0. The answers can be tested as shown below:

$$(1 \times 17) + (0 \times 0) = 17$$

Example 2.11

Given $a = 0$ and $b = 45$, find gcd (a, b) and the values of s and t.

Solution

We use a table to follow the algorithm.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
0	0	45	0	1	0	1	0	1	0
	45	0		0	1		**1**	0	

We get gcd (0, 45) = 45, $s = 0$, and $t = 1$. This indicates why we should initialize s_2 to 0 and t_2 to 1. The answer can be tested as shown below:

$$(0 \times 0) + (1 \times 45) = 45$$

Linear Diophantine Equations

Although we will see a very important application of the extended Euclidean algorithm in the next section, one immediate application is to find the solutions to the **linear Diophantine equations** of two variables, an equation of type $ax + by = c$. We need to find integer values for x and y that satisfy the equation. This type of equation has either no solution or an infinite number of solutions. Let $d = \gcd(a, b)$. If $d \nmid c$, then the equation has no solution. If $d \mid c$, then we have an infinite number of solutions. One of them is called the particular; the rest, general.

A linear Diophantine equation of two variables is $ax + by = c$.

Particular Solution

If $d \mid c$, a particular solution to the above equation can be found using the following steps:

1. Reduce the equation to $a_1x + b_1y = c_1$ by dividing both sides of the equation by d. This is possible because d divides a, b, and c by the assumption.
2. Solve for s and t in the relation $a_1s + b_1t = 1$ using the extended Euclidean algorithm.
3. The particular solution can be found:

Particular solution: $x_0 = (c/d)s$ ***and*** $y_0 = (c/d)t$

General Solutions

After finding the particular solution, the general solutions can be found:

General solutions: $x = x_0 + k\,(b/d)$ ***and*** $y = y_0 - k\,(a/d)$ **where k is an integer**

Example 2.12

Find the particular and general solutions to the equation $21x + 14y = 35$.

Solution

We have $d = \gcd(21, 14) = 7$. Since $7 \mid 35$, the equation has an infinite number of solutions. We can divide both sides by 7 to find the equation $3x + 2y = 5$. Using the extended Euclidean algorithm, we find s and t such as $3s + 2t = 1$. We have $s = 1$ and $t = -1$. The solutions are

Particular: $x_0 = 5 \times 1 = 5$ and $y_0 = 5 \times (-1) = -5$ since $35/7 = 5$
General: $x = 5 + k \times 2$ and $y = -5 - k \times 3$ where k is an integer

Therefore, the solutions are $(5, -5)$, $(7, -8)$, $(9, -11)$, . . . We can easily test that each of these solutions satisfies the original equation.

Example 2.13

A very interesting application in real life is when we want to find different combinations of objects having different values. For example, imagine we want to cash a \$100 check and get some \$20 and some \$5 bills. We have many choices, which we can find by solving the corresponding Diophantine equation $20x + 5y = 100$. Since $d = \gcd(20, 5) = 5$ and $5 \mid 100$, the equation has an infinite number of solutions, but only a few of them are acceptable in this case (only answers in which both x and y are nonnegative integers). We divide both sides by 5 to get $4x + y = 20$. We then solve the equation $4s + t = 1$. We can find $s = 0$ and $t = 1$ using the extended Euclidean algorithm. The particular solutions are $x_0 = 0 \times 20 = 0$ and $y_0 = 1 \times 20 = 20$. The general solutions with x and y nonnegative are (0, 20), (1, 16), (2, 12), (3, 8), (4, 4), (5, 0). The rest of the solutions are not acceptable because y becomes negative. The teller at the bank needs to ask which of the above combinations we want. The first has no \$20 bills; the last has no \$5 bills.

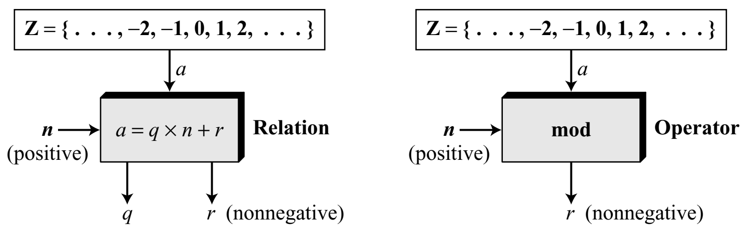
2.2 MODULAR ARITHMETIC

The division relationship ($a = q \times n + r$) discussed in the previous section has two inputs (a and n) and two outputs (q and r). In **modular arithmetic,** we are interested in only one of the outputs, the remainder r. We don't care about the quotient q. In other words, we want to know what is the value of r when we divide a by n. This implies that we can change the above relation into a binary operator with two inputs a and n and one output r.

Modulo Operator

The above-mentioned binary operator is called the **modulo operator** and is shown as *mod*. The second input (n) is called the **modulus.** The output r is called the **residue.** Figure 2.9 shows the division relation compared with the modulo operator.

Figure 2.9 *Division relation and modulo operator*



As Figure 2.9 shows, the modulo operator (**mod**) takes an integer (a) from the set **Z** and a positive modulus (n). The operator creates a nonnegative residue (r). We can say

$$a \bmod n = r$$

Example 2.14

Find the result of the following operations:

a. 27 mod 5
b. 36 mod 12
c. −18 mod 14
d. −7 mod 10

Solution

We are looking for the residue r. We can divide the a by n and find q and r. We can then disregard q and keep r.

a. Dividing 27 by 5 results in $r = 2$. This means that 27 mod 5 = 2.
b. Dividing 36 by 12 results in $r = 0$. This means that 36 mod 12 = 0.
c. Dividing −18 by 14 results in $r = -4$. However, we need to add the modulus (14) to make it nonnegative. We have $r = -4 + 14 = 10$. This means that −18 mod 14 = 10.
d. Dividing −7 by 10 results in $r = -7$. After adding the modulus to −7, we have $r = 3$. This means that −7 mod 10 = 3.

Set of Residues: $\mathbf{Z}_n$

The result of the modulo operation with modulus n is always an integer between 0 and $n - 1$. In other words, the result of a mod n is always a nonnegative integer less than n. We can say that the modulo operation creates a set, which in modular arithmetic is referred to as the **set of least residues modulo *n***, or $\mathbf{Z}_n$. However, we need to remember that although we have only one set of integers ($\mathbf{Z}$), we have infinite instances of the set of residues ($\mathbf{Z}_n$), one for each value of n. Figure 2.10 shows the set $\mathbf{Z}_n$ and three instances, $\mathbf{Z}_2$, $\mathbf{Z}_6$, and $\mathbf{Z}_{11}$.

Figure 2.10 *Some $\mathbf{Z}_n$ sets*

$\mathbf{Z}_n = \{ 0, 1, 2, 3, \ldots, (n-1) \}$

$\mathbf{Z}_2 = \{ 0, 1 \}$ $\mathbf{Z}_6 = \{ 0, 1, 2, 3, 4, 5 \}$ $\mathbf{Z}_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$

Congruence

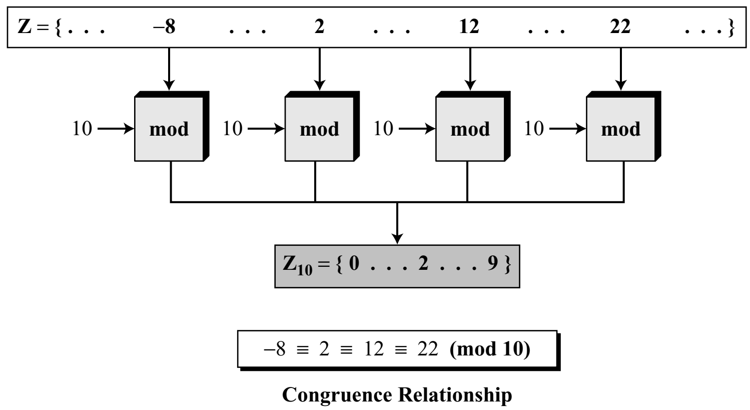
In cryptography, we often used the concept of **congruence** instead of equality. Mapping from $\mathbf{Z}$ to $\mathbf{Z}_n$ is not one-to-one. Infinite members of $\mathbf{Z}$ can map to one member of $\mathbf{Z}_n$. For example, the result of 2 mod 10 = 2, 12 mod 10 = 2, 22 mod 2 = 2, and so on. In modular arithmetic, integers like 2, 12, and 22 are called congruent mod 10. To show that two integers are congruent, we use the **congruence operator** (≡). We add the phrase (mod n) to the right side of the congruence to define the value of modulus that makes the relationship valid. For example, we write:

$2 \equiv 12 \pmod{10}$	$13 \equiv 23 \pmod{10}$	$34 \equiv 24 \pmod{10}$	$-8 \equiv 12 \pmod{10}$
$3 \equiv 8 \pmod{5}$	$8 \equiv 13 \pmod{5}$	$23 \equiv 33 \pmod{5}$	$-8 \equiv 2 \pmod{5}$

Figure 2.11 shows the idea of congruence. We need to explain several points.

a. The congruence operator looks like the equality operator, but there are differences. First, an equality operator maps a member of $\mathbf{Z}$ to itself; the congruence operator maps a member from $\mathbf{Z}$ to a member of $\mathbf{Z}_n$. Second, the equality operator is one-to-one; the congruence operator is many-to-one.

Figure 2.11 *Concept of congruence*



b. The phrase (mod n) that we insert at the right-hand side of the congruence operator is just an indication of the destination set ($\mathbf{Z}_n$). We need to add this phrase to show what modulus is used in the mapping. The symbol *mod* used here does not have the same meaning as the binary operator. In other words, the symbol *mod* in 12 mod 10 is an operator; the phrase (mod 10) in $2 \equiv 12 \pmod{10}$ means that the destination set is $\mathbf{Z}_{10}$.

Residue Classes

A **residue class** $[a]$ or $[a]_n$ is the set of integers congruent modulo n. In other words, it is the set of all integers such that $x = a \pmod{n}$. For example, if $n = 5$, we have five sets [0], [1], [2], [3], and [4] as shown below:

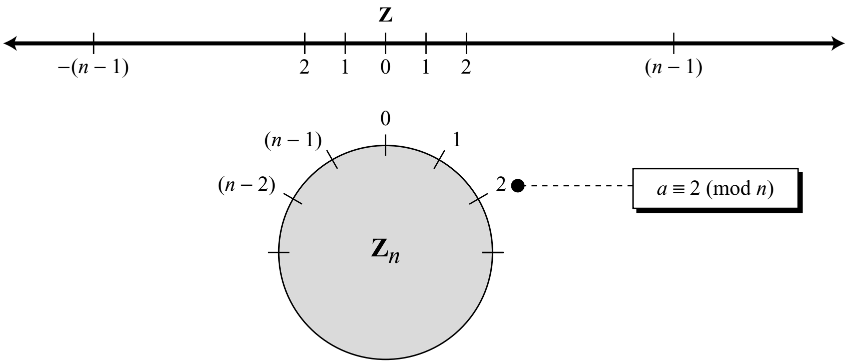
$$
\begin{aligned}
[0] &= \{\ldots, -15, -10, -5, 0, 5, 10, 15, \ldots\}\\
[1] &= \{\ldots, -14, -9, -4, 1, 6, 11, 16, \ldots\}\\
[2] &= \{\ldots, -13, -8, -3, 2, 7, 12, 17, \ldots\}\\
[3] &= \{\ldots, -12, -7, -5, 3, 8, 13, 18, \ldots\}\\
[4] &= \{\ldots, -11, -6, -1, 4, 9, 14, 19, \ldots\}
\end{aligned}
$$

The integers in the set [0] are all reduced to 0 when we apply the modulo 5 operation on them. The integers in the set [1] are all reduced to 1 when we apply the modulo 5 operation, and so on. In each set, there is one element called the least (nonnegative) residue. In the set [0], this element is 0; in the set [1], this element is 1; and so on. The set of all of these least residues is what we have shown as $\mathbf{Z}_5 = \{0, 1, 2, 3, 4\}$. In other words, the set $\mathbf{Z}_n$ is the set of all **least residue** modulo n.

Circular Notation

The concept of congruence can be better understood with the use of a circle. Just as we use a line to show the distribution of integers in $\mathbf{Z}$, we can use a circle to show the

Figure 2.12 *Comparison of $\mathbf{Z}$ and $\mathbf{Z}_n$ using graphs*



distribution of integers in $\mathbf{Z}_n$. Figure 2.12 shows the comparison between the two. Integers 0 to $n - 1$ are spaced evenly around a circle. All congruent integers modulo n occupy the same point on the circle. Positive and negative integers from $\mathbf{Z}$ are mapped to the circle in such a way that there is a symmetry between them.

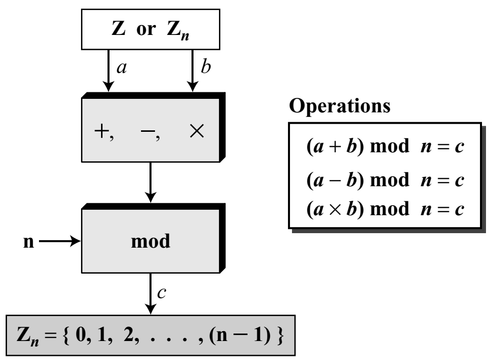
Example 2.15

We use modular arithmetic in our daily life; for example, we use a clock to measure time. Our clock system uses modulo 12 arithmetic. However, instead of a 0 we use the number 12. So our clock system starts with 0 (or 12) and goes until 11. Because our days last 24 hours, we navigate around the circle two times and denote the first revolution as A.M. and the second as P.M.

Operations in $\mathbf{Z}_n$

The three binary operations (*addition, subtraction,* and *multiplication*) that we discussed for the set $\mathbf{Z}$ can also be defined for the set $\mathbf{Z}_n$. The result may need to be mapped to $\mathbf{Z}_n$ using the mod operator as shown in Figure 2.13.

Figure 2.13 *Binary operations in $\mathbf{Z}_n$*



Actually, two sets of operators are used here. The first set is one of the binary operators (+, −, ×); the second is the mod operator. We need to use parentheses to emphasize the order of operations. As Figure 2.13 shows, the inputs (a and b) can be members of $\mathbf{Z}_n$ or $\mathbf{Z}$.

Example 2.16

Perform the following operations (the inputs come from $\mathbf{Z}_n$):

a. Add 7 to 14 in $\mathbf{Z}_{15}$.
b. Subtract 11 from 7 in $\mathbf{Z}_{13}$.
c. Multiply 11 by 7 in $\mathbf{Z}_{20}$.

Solution

The following shows the two steps involved in each case:

$$(14 + 7) \bmod 15 \;\rightarrow\; (21) \bmod 15 = 6$$
$$(7 - 11) \bmod 13 \;\rightarrow\; (-4) \bmod 13 = 9$$
$$(7 \times 11) \bmod 20 \;\rightarrow\; (77) \bmod 20 = 17$$

Example 2.17

Perform the following operations (the inputs come from either $\mathbf{Z}$ or $\mathbf{Z}_n$):

a. Add 17 to 27 in $\mathbf{Z}_{14}$.
b. Subtract 43 from 12 in $\mathbf{Z}_{13}$.
c. Multiply 123 by −10 in $\mathbf{Z}_{19}$.

Solution

The following shows the two steps involved in each case:

$$(17 + 27) \bmod 14 \;\rightarrow\; (44) \bmod 14 = 2$$
$$(12 - 43) \bmod 13 \;\rightarrow\; (-31) \bmod 13 = 8$$
$$(123 \times (-10)) \bmod 19 \;\rightarrow\; (-1230) \bmod 19 = 5$$

Properties

We mentioned that the two inputs to the three binary operations in the modular arithmetic can come from $\mathbf{Z}$ or $\mathbf{Z}_n$. The following properties allow us to first map the two inputs to $\mathbf{Z}_n$ (if they are coming from $\mathbf{Z}$) before applying the three binary operations (+, −, ×). Interested readers can find proofs for these properties in Appendix Q.

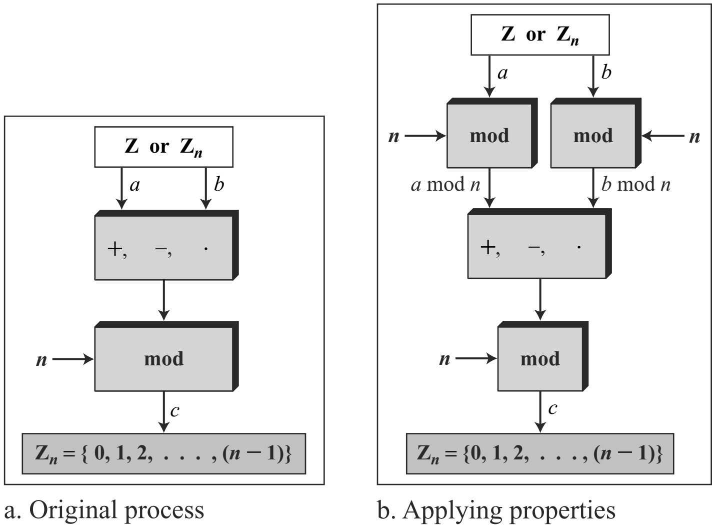
First Property: $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$

Second Property: $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$

Third Property: $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

Figure 2.14 shows the process before and after applying the above properties. Although the figure shows that the process is longer if we apply the above properties, we should remember that in cryptography we are dealing with very large integers. For example, if we multiply a very large integer by another very large integer, we

Figure 2.14 *Properties of mod operator*



may have an integer that is too large to be stored in the computer. Applying the above properties make the first two operands smaller before the multiplication operation is applied. In other words, the properties allow us to work with smaller numbers. This fact will manifest itself more clearly in discussion of the exponential operation in later chapters.

Example 2.18

The following shows the application of the above properties:

1. (1,723,345 + 2,124,945) mod 11 = (8 + 9) mod 11 = 6
2. (1,723,345 − 2,124,945) mod 16 = (8 − 9) mod 11 = 10
3. (1,723,345 × 2,124,945) mod 16 = (8 × 9) mod 11 = 6

Example 2.19

In arithmetic, we often need to find the remainder of powers of 10 when divided by an integer. For example, we need to find 10 mod 3, 10^2 mod 3, 10^3 mod 3, and so on. We also need to find 10 mod 7, 10^2 mod 7, 10^3 mod 7, and so. The third property of the mod operator mentioned above makes life much easier.

$10^n \bmod x = (10 \bmod x)^n \bmod x$ Applying the third property n times.

We have

$10 \bmod 3 = 1 \quad \rightarrow \quad 10^n \bmod 3 = (10 \bmod 3)^n = 1$
$10 \bmod 9 = 1 \quad \rightarrow \quad 10^n \bmod 9 = (10 \bmod 9)^n = 1$
$10 \bmod 7 = 3 \quad \rightarrow \quad 10^n \bmod 7 = (10 \bmod 7)^n = 3^n \bmod 7$

Example 2.20

We have been told in arithmetic that the remainder of an integer divided by 3 is the same as the remainder of the sum of its decimal digits. In other words, the remainder of dividing 6371 by 3 is the same as dividing 17 by 3 because 6 + 3 + 7 + 1 = 17. We can prove this claim using the properties of the mod operator. We write an integer as the sum of its digits multiplied by the powers of 10.

$$a = a_n \times 10^n + \cdots + a_1 \times 10^1 + a_0 \times 10^0$$
For example: $6371 = 6 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 1 \times 10^0$

Now we can apply the mod operator to both sides of the equality and use the result of the previous example that 10^n mod 3 is 1.

$$\begin{aligned} a \bmod 3 &= (a_n \times 10^n + \cdots + a_1 \times 10^1 + a_0 \times 10^0) \bmod 3 \\ &= (a_n \times 10^n) \bmod 3 + \cdots + (a_1 \times 10^1) \bmod 3 + (a_0 \times 10^0) \bmod 3 \\ &= (a_n \bmod 3) \times (10^n \bmod 3) + \cdots + (a_1 \bmod 3) \times (10^1 \bmod 3) + \\ &\quad (a_0 \bmod 3) \times (10^0 \bmod 3) \\ &= a_n \bmod 3 + \cdots + a_1 \bmod 3 + a_0 \bmod 3 \\ &= (a_n + \cdots + a_1 + a_0) \bmod 3 \end{aligned}$$

Inverses

When we are working in modular arithmetic, we often need to find the inverse of a number relative to an operation. We are normally looking for an **additive inverse** (relative to an addition operation) or a **multiplicative inverse** (relative to a multiplication operation).

Additive Inverse

In $\mathbf{Z}_n$, two numbers a and b are additive inverses of each other if

$$a + b \equiv 0 \pmod{n}$$

In $\mathbf{Z}_n$, the additive inverse of a can be calculated as $b = n - a$. For example, the additive inverse of 4 in $\mathbf{Z}_{10}$ is $10 - 4 = 6$.

In modular arithmetic, each integer has an additive inverse.

The sum of an integer and its additive inverse is congruent to 0 modulo *n*.

Note that in modular arithmetic, each number has an additive inverse and the inverse is unique; each number has one and only one additive inverse. However, the inverse of the number may be the number itself.

Example 2.21

Find all additive inverse pairs in $\mathbf{Z}_{10}$.

Solution

The six pairs of additive inverses are (0, 0), (1, 9), (2, 8), (3, 7), (4, 6), and (5, 5). In this list, 0 is the additive inverse of itself; so is 5. Note that the additive inverses are reciprocal; if 4 is the additive inverse of 6, then 6 is also the additive inverse of 4.

Multiplicative Inverse

In $\mathbf{Z}_n$, two numbers a and b are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

For example, if the modulus is 10, then the multiplicative inverse of 3 is 7. In other words, we have $(3 \times 7) \bmod 10 = 1$.

In modular arithmetic, an integer may or may not have a multiplicative inverse.

When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo *n*.

It can be proved that a has a multiplicative inverse in $\mathbf{Z}_n$ if and only if $\gcd(n, a) = 1$. In this case, a and n are said to be **relatively prime.**

Example 2.22

Find the multiplicative inverse of 8 in $\mathbf{Z}_{10}$.

Solution

There is no multiplicative inverse because $\gcd(10, 8) = 2 \neq 1$. In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Example 2.23

Find all multiplicative inverses in $\mathbf{Z}_{10}$.

Solution

There are only three pairs: (1, 1), (3, 7) and (9, 9). The numbers 0, 2, 4, 5, 6, and 8 do not have a multiplicative inverse. We can see that

$$(1 \times 1) \bmod 10 = 1 \qquad (3 \times 7) \bmod 10 = 1 \qquad (9 \times 9) \bmod 10 = 1$$

Example 2.24

Find all multiplicative inverse pairs in $\mathbf{Z}_{11}$.

Solution

We have seven pairs: (1, 1), (2, 6), (3, 4), (5, 9), (7, 8), (9, 9), and (10, 10). In moving from $\mathbf{Z}_{10}$ to $\mathbf{Z}_{11}$, the number of pairs doubles. The reason is that in $\mathbf{Z}_{11}$, $\gcd(11, a)$ is 1 (relatively prime) for all values of a except 0. It means all integers 1 to 10 have multiplicative inverses.

The integer a in $\mathbf{Z}_n$ has a multiplicative inverse if and only if $\gcd(n, a) \equiv 1 \pmod{n}$

The extended Euclidean algorithm we discussed earlier in the chapter can find the multiplicative inverse of b in $\mathbf{Z}_n$ when n and b are given and the inverse exists. To show this, let us replace the first integer a with n (the modulus). We can say that the algorithm can find s and t such $s \times n + b \times t = \gcd(n, b)$. However, if the multiplicative inverse of b exists, gcd (n, b) must be 1. So the relationship is

$$(s \times n) + (b \times t) = 1$$

Now we apply the modulo operator to both sides. In other words, we map each side to $\mathbf{Z}_n$. We will have

$$(s \times n + b \times t) \bmod n = 1 \bmod n$$
$$[(s \times n) \bmod n] + [(b \times t) \bmod n] = 1 \bmod n$$
$$0 + [(b \times t) \bmod n] = 1$$
$$(b \times t) \bmod n = 1 \quad \rightarrow \text{This means } t \text{ is the multiplicative inverse of } b \text{ in } Z_n$$

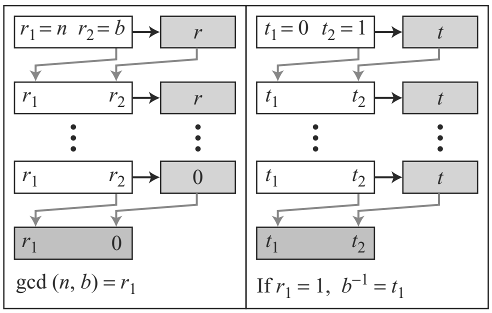
Note that $[(s \times n) \bmod n]$ in the third line is 0 because if we divide $(s \times n)$ by n, the quotient is s but the remainder is 0.

The extended Euclidean algorithm finds the multiplicative inverses of b in $\mathbf{Z}_n$ when n and b are given and gcd $(n, b) = 1$.

The multiplicative inverse of b is the value of t after being mapped to $\mathbf{Z}_n$.

Figure 2.15 shows how we find the multiplicative inverse of a number using the extended Euclidean algorithm.

Figure 2.15 *Using the extended Euclidean algorithm to find the multiplicative inverse*



a. Process

```
r1 ← n; r2 ← b;
t1 ← 0; t2 ← 1;

while (r2 > 0)
{
  q ← r1 / r2;

  r  ← r1 − q × r2;
  r1 ← r2;   r2 ← r;

  t  ← t1 − q × t2;
  t1 ← t2;   t2 ← t;
}
if (r1 = 1) then b^-1 ← t1
```

b. Algorithm

Example 2.25

Find the multiplicative inverse of 11 in $\mathbf{Z}_{26}$.

Solution

We use a table similar to the one we used before with $r_1 = 26$ and $r_2 = 11$. We are interested only in the value of t.

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	−2
2	11	4	3	1	−2	5
1	4	3	1	−2	5	−7
3	3	1	0	5	−7	26
	1	0		−7	26	

The gcd (26, 11) is 1, which means that the multiplicative inverse of 11 exists. The extended Euclidean algorithm gives $t_1 = -7$. The multiplicative inverse is $(-7) \bmod 26 = 19$. In other words, 11 and 19 are multiplicative inverse in $\mathbf{Z_{26}}$. We can see that $(11 \times 19) \bmod 26 = 209 \bmod 26 = 1$.

Example 2.26

Find the multiplicative inverse of 23 in $\mathbf{Z_{100}}$.

Solution

We use a table similar to the one we used before with $r_1 = 100$ and $r_2 = 23$. We are interested only in the value of t.

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	−4
2	23	8	7	1	−4	19
1	8	7	1	−4	9	−13
7	7	1	0	9	−13	100
	1	0		−13	100	

The gcd (100, 23) is 1, which means the inverse of 23 exists. The extended Euclidean algorithm gives $t_1 = -13$. The inverse is $(-13) \bmod 100 = 87$. In other words, 13 and 87 are multiplicative inverses in $\mathbf{Z_{100}}$. We can see that $(23 \times 87) \bmod 100 = 2001 \bmod 100 = 1$.

Example 2.27

Find the inverse of 12 in $\mathbf{Z_{26}}$.

Solution

We use a table similar to the one we used before, with $r_1 = 26$ and $r_2 = 12$.

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	−2
6	12	2	0	1	−2	13
	2	0		−2	13	

The gcd $(26, 12) = 2 \neq 1$, which means there is no multiplicative inverse for 12 in $\mathbf{Z_{26}}$.

Addition and Multiplication Tables

Figure 2.16 shows two tables for addition and multiplication. In the addition table, each integer has an additive inverse. The inverse pairs can be found when the result of addition is zero. We have (0, 0), (1, 9), (2, 8), (3, 7), (4, 6), and (5, 5). In the multiplication table we have only three multiplicative pairs (1, 1), (3, 7) and (9, 9). The pairs can be found whenever the result of multiplication is 1. Both tables are symmetric with respect to the diagonal of elements that moves from the top left to the bottom right, revealing the commutative property for addition and multiplication ($a + b = b + a$ and $a \times b = b \times a$). The addition table also shows that each row or column is a permutation of another row or column. This is not true for the multiplication table.

Figure 2.16 *Addition and multiplication tables for* $\mathbf{Z}_{10}$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Addition Table in $\mathbf{Z}_{10}$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	0	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Multiplication Table in $\mathbf{Z}_{10}$

Different Sets for Addition and Multiplication

In cryptography we often work with inverses. If the sender uses an integer (as the encryption key), the receiver uses the inverse of that integer (as the decryption key). If the operation (encryption/decryption algorithm) is addition, $\mathbf{Z}_n$ can be used as the set of possible keys because each integer in this set has an additive inverse. On the other hand, if the operation (encryption/decryption algorithm) is multiplication, $\mathbf{Z}_n$ cannot be the set of possible keys because only some members of this set have a multiplicative inverse. We need another set. The new set, which is a subset of $\mathbf{Z}_n$ includes only integers in $\mathbf{Z}_n$ that have a unique multiplicative inverse. This set is called $\mathbf{Z}_{n^*}$. Figure 2.17 shows some instances of two sets. Note that $\mathbf{Z}_{n^*}$ can be made from multiplication tables, such as the one shown in Figure 2.16.

Each member of $\mathbf{Z}_n$ has an additive inverse, but only some members have a multiplicative inverse. Each member of $\mathbf{Z}_{n^*}$ has a multiplicative inverse, but only some members have an additive inverse.

We need to use $\mathbf{Z}_n$ when additive inverses are needed; we need to use $\mathbf{Z}_n{}^*$ when multiplicative inverses are needed.

Figure 2.17 *Some $\mathbf{Z}_n$ and $\mathbf{Z}_n*$ sets*

$\mathbf{Z}_6 = \{0, 1, 2, 3, 4, 5\}$	$\mathbf{Z}_6^* = \{1, 5\}$
$\mathbf{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$	$\mathbf{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$
$\mathbf{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	$\mathbf{Z}_{10}^* = \{1, 3, 7, 9\}$

Two More Sets

Cryptography often uses two more sets: $\mathbf{Z}_p$ and $\mathbf{Z}_{p}*$. The modulus in these two sets is a prime number. Prime numbers will be discussed in later chapters; suffice it to say that a prime number has only two divisors: integer 1 and itself.

The set $\mathbf{Z}_p$ is the same as $\mathbf{Z}_n$ except that n is a prime. $\mathbf{Z}_p$ contains all integers from 0 to $p - 1$. Each member in $\mathbf{Z}_p$ has an additive inverse; each member except 0 has a multiplicative inverse.

The set $\mathbf{Z}_{p}*$ is the same as $\mathbf{Z}_{n}*$ except that n is a prime. $\mathbf{Z}_{p}*$ contains all integers from 1 to $p - 1$. Each member in $\mathbf{Z}_{p}*$ has an additive and a multiplicative inverse. $\mathbf{Z}_{p}*$ is a very good candidate when we need a set that supports both additive and multiplicative inverse.

The following shows these two sets when $p = 13$.

$Z_{13} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
$Z_{13}* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

2.3 MATRICES

In cryptography we need to handle matrices. Although this topic belongs to a special branch of algebra called linear algebra, the following brief review of matrices is necessary preparation for the study of cryptography. Readers who are familiar with this topic can skip part or all of this section. The section begins with some definitions and then shows how to use matrices in modular arithmetic.

Definitions

A **matrix** is a rectangular array of $l \times m$ elements, in which l is the number of rows and m is the number of columns. A matrix is normally denoted with a boldface uppercase letter such as **A**. The element a_{ij} is located in the ith row and jth column. Although the elements can be a set of numbers, we discuss only matrices with elements in **Z**. Figure 2.18 shows a matrix.

If a matrix has only one row ($l = 1$), it is called a **row matrix;** if it has only one column ($m = 1$), it is called a **column matrix.** In a **square matrix,** in which there is the

Figure 2.18 *A matrix of size $l \times m$*

$$\textbf{Matrix A:}\quad \begin{array}{c c} & m \text{ columns} \\ l \text{ rows} & \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix} \end{array}$$

same number of rows and columns ($l = m$), the elements $a_{11}, a_{22}, \ldots, a_{mm}$ make the **main diagonal.** An additive identity matrix, denoted as **0**, is a matrix with all rows and columns set to 0's. An **identity matrix,** denoted as **I,** is a square matrix with 1s on the main diagonal and 0s elsewhere. Figure 2.19 shows some examples of matrices with elements from **Z**.

Figure 2.19 *Example of matrices*

$$\underset{\text{Row matrix}}{\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix}} \quad \underset{\substack{\text{Column}\\\text{matrix}}}{\begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}} \quad \underset{\substack{\text{Square}\\\text{matrix}}}{\begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix}} \quad \underset{\mathbf{0}}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}} \quad \underset{\mathbf{I}}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$$

Operations and Relations

In linear algebra, one relation (equality) and four operations (addition, subtraction, multiplication, and scalar multiplication) are defined for matrices.

Equality

Two matrices are equal if they have the same number of rows and columns and the corresponding elements are equal. In other words, **A** = **B** if we have $a_{ij} = b_{ij}$ for all i's and j's.

Addition and Subtraction

Two matrices can be added if they have the same number of columns and rows. This addition is shown as **C** = **A** + **B**. In this case, the resulting matrix **C** has also the same number of rows and columns as **A** or **B**. Each element of **C** is the sum of the two corresponding elements of **A** and **B**: $c_{ij} = a_{ij} + b_{ij}$. Subtraction is the same except that each element of **B** is subtracted from the corresponding element of **A**: $d_{ij} = a_{ij} - b_{ij}$.

Example 2.28

Figure 2.20 shows an example of addition and subtraction.

Figure 2.20 *Addition and subtraction of matrices*

$$\underbrace{\begin{bmatrix} 12 & 4 & 4 \\ 11 & 12 & 30 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} + \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}}_{\mathbf{C} = \mathbf{A} + \mathbf{B}} \qquad \underbrace{\begin{bmatrix} -2 & 0 & -2 \\ -5 & -8 & 10 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} - \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}}_{\mathbf{D} = \mathbf{A} - \mathbf{B}}$$

Multiplication

We can multiply two matrices of different sizes if the number of columns of the first matrix is the same as the number of rows of the second matrix. If **A** is an $l \times m$ matrix and **B** is an $m \times p$ matrix, the product of the two is a matrix **C** of size $l \times p$. If each element of matrix **A** is called a_{ij}, each element of matrix **B** is called b_{jk}, then each element of matrix **C**, c_{ik}, can be calculated as

$$c_{ik} = \sum a_{ij} \times b_{jk} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \cdots + a_{im} \times b_{mj}$$

Example 2.29

Figure 2.21 shows the product of a row matrix (1×3) by a column matrix (3×1). The result is a matrix of size 1×1.

Figure 2.21 *Multiplication of a row matrix by a column matrix*

$$\begin{array}{ccccc} \mathbf{C} & & \mathbf{A} & & \mathbf{B} \\ \begin{bmatrix} 53 \end{bmatrix} & = & \begin{bmatrix} 5 & 2 & 1 \end{bmatrix} & \times & \begin{bmatrix} 7 \\ 8 \\ 2 \end{bmatrix} \end{array}$$

In which: $\boxed{53 = 5 \times 7 + 2 \times 8 + 1 \times 2}$

Example 2.30

Figure 2.22 shows the product of a 2×3 matrix by a 3×4 matrix. The result is a 2×4 matrix.

Figure 2.22 *Multiplication of a 2×3 matrix by a 3×4 matrix*

$$\begin{array}{ccccc} \mathbf{C} & & \mathbf{A} & & \mathbf{B} \\ \begin{bmatrix} 52 & 18 & 14 & 9 \\ 41 & 21 & 22 & 7 \end{bmatrix} & = & \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix} & \times & \begin{bmatrix} 7 & 3 & 2 & 1 \\ 8 & 0 & 0 & 2 \\ 1 & 3 & 4 & 0 \end{bmatrix} \end{array}$$

Scalar Multiplication

We can also multiply a matrix by a number (called a **scalar**). If **A** is an $l \times m$ matrix and x is a scalar, $\mathbf{C} = x\mathbf{A}$ is a matrix of size $l \times m$, in which $c_{ij} = x \times a_{ij}$.

Example 2.31

Figure 2.23 shows an example of scalar multiplication.

Figure 2.23 *Scalar multiplication*

$$\underset{\mathbf{B}}{\begin{bmatrix} 15 & 6 & 3 \\ 9 & 6 & 12 \end{bmatrix}} = 3 \times \underset{\mathbf{A}}{\begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix}}$$

Determinant

The **determinant** of a square matrix **A** of size $m \times m$ denoted as det (**A**) is a scalar calculated recursively as shown below:

1. If $m = 1$, det $(\mathbf{A}) = a_{11}$
2. If $m > 1$, det $(\mathbf{A}) = \sum_{i=1...m} (-1)^{i+j} \times a_{ij} \times \det(\mathbf{A}_{ij})$

   Where $\mathbf{A}_{ij}$ is a matrix obtained from **A** by deleting the *i*th row and *j*th column.

The determinant is defined only for a square matrix.

Example 2.32

Figure 2.24 shows how we can calculate the determinant of a 2×2 matrix based on the determinant of a 1×1 matrix using the above recursive definition. The example shows that when *m* is 1 or 2, it is very easy to find the determinant of a matrix.

Figure 2.24 *Calculating the determinant of a 2 × 2 matrix*

$$\det\begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det[\,4\,] + (-1)^{1+2} \times 2 \times \det[\,3\,] \longrightarrow 5 \times 4 - 2 \times 3 = 14$$

or

$$\boxed{\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}}$$

Example 2.33

Figure 2.25 shows the calculation of the determinant of a 3×3 matrix.

Figure 2.25 *Calculating the determinant of a 3 × 3 matrix*

$$\det\begin{bmatrix} 5 & 2 & 1 \\ 3 & 0 & -4 \\ 2 & 1 & 6 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det\begin{bmatrix} 0 & -4 \\ 1 & 6 \end{bmatrix} + (-1)^{1+2} \times 2 \times \det\begin{bmatrix} 3 & -4 \\ 2 & 6 \end{bmatrix} + (-1)^{1+3} \times 1 \times \det\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$= (+1) \times 5 \times (+4) \quad + \quad (-1) \times 2 \times (24) \quad + \quad (+1) \times 1 \times (3) = -25$$

Inverses

Matrices have both additive and multiplicative inverses.

Additive Inverse

The additive inverse of matrix **A** is another matrix **B** such that **A** + **B** = **0**. In other words, we have $b_{ij} = -a_{ij}$ for all values of i and j. Normally the additive inverse of **A** is defined by −**A**.

Multiplicative Inverse

The multiplicative inverse is defined only for square matrices. The multiplicative inverse of a square matrix **A** is a square matrix **B** such that **A** × **B** = **B** × **A** = **I**. Normally the multiplicative inverse of **A** is defined by $\mathbf{A}^{-1}$. The multiplicative inverse exists only if the det(**A)** has a multiplicative inverse in the corresponding set. Since no integer has a multiplicative inverse in **Z**, there is no multiplicative inverse of a matrix in **Z**. However, matrices with real elements have inverses only if det (**A)** ≠ 0.

Multiplicative inverses are only defined for square matrices.

Residue Matrices

Cryptography uses residue matrices: matrices with all elements are in $\mathbf{Z}_n$. All operations on residue matrices are performed the same as for the integer matrices except that the operations are done in modular arithmetic. One interesting result is that a residue matrix has a multiplicative inverse if the determinant of the matrix has a multiplicative inverse in $\mathbf{Z}_n$. In other words, a residue matrix has a multiplicative inverse if gcd (det(**A**), n) = 1.

Example 2.34

Figure 2.26 shows a residue matrix **A** in $\mathbf{Z}_{26}$ and its multiplicative inverse $\mathbf{A}^{-1}$. We have det(**A**) = 21 which has the multiplicative inverse 5 in $\mathbf{Z}_{26}$. Note that when we multiply the two matrices, the result is the multiplicative identity matrix in $\mathbf{Z}_{26}$.

Figure 2.26 *A residue matrix and its multiplicative inverse*

$$\mathbf{A} = \begin{bmatrix} 3 & 5 & 7 & 2 \\ 1 & 4 & 7 & 2 \\ 6 & 3 & 9 & 17 \\ 13 & 5 & 4 & 16 \end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix} 15 & 21 & 0 & 15 \\ 23 & 9 & 0 & 22 \\ 15 & 16 & 18 & 3 \\ 24 & 7 & 15 & 3 \end{bmatrix}$$

$$\det(\mathbf{A}) = 21 \qquad \det(\mathbf{A}^{-1}) = 5$$

Congruence

Two matrices are congruent modulo n, written as $\mathbf{A} \equiv \mathbf{B} \pmod{n}$, if they have the same number of rows and columns and all corresponding elements are congruent modulo n. In other words, $\mathbf{A} \equiv \mathbf{B} \pmod{n}$ if $a_{ij} \equiv b_{ij} \pmod{n}$ for all i's and j's.

2.4 LINEAR CONGRUENCE

Cryptography often involves solving an equation or a set of equations of one or more variables with coefficient in $\mathbf{Z}_n$. This section shows how to solve equations when the power of each variable is 1 (linear equation).

Single-Variable Linear Equations

Let us see how we can solve equations involving a single variable—that is, equations of the form $ax \equiv b \pmod{n}$. An equation of this type might have no solution or a limited number of solutions. Assume that the gcd $(a, n) = d$. If $d \nmid b$, there is no solution. If $d|b$, there are d solutions.

If $d|b$, we use the following strategy to find the solutions:

1. Reduce the equation by dividing both sides of the equation (including the modulus) by d.
2. Multiply both sides of the reduced equation by the multiplicative inverse of a to find the particular solution x_0.
3. The general solutions are $x = x_0 + k\,(n/d)$ for $k = 0, 1, \ldots, (d - 1)$.

Example 2.35

Solve the equation $10x \equiv 2 \pmod{15}$.

Solution

First we find the gcd (10 and 15) = 5. Since 5 does not divide 2, we have no solution.

Example 2.36

Solve the equation $14x \equiv 12 \pmod{18}$.

Solution

Note that gcd (14 and 18) = 2. Since 2 divides 12, we have exactly two solutions, but first we reduce the equation.

$$14x \equiv 12 \pmod{18} \rightarrow \quad 7x \equiv 6 \pmod{9} \quad \rightarrow \; x \equiv 6\,(7^{-1}) \pmod{9}$$
$$x_0 = (6 \times 7^{-1}) \bmod 9 = (6 \times 4) \pmod{9} = 6$$
$$x_1 = x_0 + 1 \times (18/2) = 15$$

Both solutions, 6 and 15 satisfy the congruence relation, because $(14 \times 6) \bmod 18 = 12$ and also $(14 \times 15) \bmod 18 = 12$.

Example 2.37

Solve the equation $3x + 4 \equiv 6 \pmod{13}$.

Solution

First we change the equation to the form $ax \equiv b \pmod{n}$. We add -4 (the additive inverse of 4) to both sides, which give $3x \equiv 2 \pmod{13}$. Because gcd $(3, 13) = 1$, the equation has only one solution, which is $x_0 = (2 \times 3^{-1}) \bmod 13 = 18 \bmod 13 = 5$. We can see that the answer satisfies the original equation: $3 \times 5 + 4 \equiv 6 \pmod{13}$.

Set of Linear Equations

We can also solve a set of linear equations with the same modulus if the matrix formed from the coefficients of the variables is invertible. We make three matrices. The first is the square matrix made from the coefficients of variables. The second is a column matrix made from the variables. The third is a column matrix made from the values at the right-hand side of the congruence operator. We can interpret the set of equations as matrix multiplication. If both sides of congruence are multiplied by the multiplicative inverse of the first matrix, the result is the variable matrix at the right-hand side, which means the problem can be solved by a matrix multiplication as shown in Figure 2.27.

Figure 2.27 *Set of linear equations*

$$\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \equiv & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & \equiv & b_2 \\
\vdots & & \vdots & & & & \vdots & & \vdots \\
a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & \equiv & b_n
\end{array}$$

a. Equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \equiv \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

b. Interpretation

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \equiv \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

c. Solution

Example 2.38

Solve the set of following three equations:

$3x + 5y + 7z \equiv 3 \pmod{16}$
$x + 4y + 13z \equiv 5 \pmod{16}$
$2x + 7y + 3z \equiv 4 \pmod{16}$

Solution

Here x, y, and z play the roles of x_1, x_2, and x_3. The matrix formed by the set of equations is invertible. We find the multiplicative inverse of the matrix and multiply it by the column matrix formed from 3, 5, and 4. The result is $x \equiv 15 \pmod{16}$, $y \equiv 4 \pmod{16}$, and $z \equiv 14 \pmod{16}$. We can check the answer by inserting these values into the equations.

2.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books give an easy but thorough coverage of number theory including [Ros06], [Sch99], [Cou99], and [BW00]. Matrices are discussed in any book about linear algebra; [LEF04], [DF04], and [Dur05] are good texts to start with.

WebSites

The following websites give more information about topics discussed in this chapter.

http://en.wikipedia.org/wiki/Euclidean_algorithm

http://en.wikipedia.org/wiki/Multiplicative_inverse

http://en.wikipedia.org/wiki/Additive_inverse

2.6 KEY TERMS

additive inverse
binary operation
column matrix
congruence
congruence operator
determinant
divisibility
Euclidean algorithm
extended Euclidean algorithm
greatest common divisor
identity matrix
integer arithmetic
least residue
linear congruence
linear Diophantine equation
main diagonal
matrix
modular arithmetic
modulo operator (mod)
modulus
multiplicative inverse
relatively prime
residue
residue class
row matrix
scalar
set of integers, $\mathbf{Z}$
set of residues, $\mathbf{Z}_n$
square matrix

2.7 SUMMARY

- The set of integers, denoted by **Z**, contains all integral numbers from negative infinity to positive infinity. Three common binary operations defined for integers are addition, subtraction, and multiplication. Division does not fit in this category because it produces two outputs instead of one.
- In integer arithmetic, if we divide a by n, we can get q and r. The relationship between these four integers can be shown as $a = q \times n + r$. We say a|b if $a = q \times n$. We mentioned four properties of divisibility in this chapter.
- Two positive integers can have more than one common divisor. But we are normally interested in the greatest common divisor. The Euclidean algorithm gives an efficient and systematic way to calculation of the greatest common divisor of two integer.
- The extended Euclidean algorithm can calculate gcd (a, b) and at the same time calculate the value of s and t to satisfy the equation $as + bt = \gcd(a, b)$.
- A linear Diophantine equation of two variables is $ax + by = c$. It has a particular and general solution.
- In modular arithmetic, we are interested only in remainders; we want to know the value of r when we divide a by n. We use a new operator called modulo operator (mod) so that a mod $n = r$. Now n is called the modulus; r is called the residue.
- The result of the modulo operation with modulus n is always an integer between 0 and. We can say that the modulo operation creates a set, which in modular arithmetic is referred to as the set of least residues modulo n, or $\mathbf{Z}_n$.
- Mapping from **Z** to $\mathbf{Z}_n$ is not one-to-one. Infinite members of **Z** can map to one member of $\mathbf{Z}_n$. In modular arithmetic, all integers in **Z** that map to one integer in $\mathbf{Z}_n$ are called congruent modulo n. To show that two integers are congruent, we use the congruence operator (≡).
- A residue class $[a]$ is the set of integers congruent modulo n. It is the set of all integers such that $x = a \pmod{n}$.
- The three binary operations (addition, subtraction, and multiplication) defined for the set **Z** can also be defined for the set $\mathbf{Z}_n$. The result may need to be mapped to $\mathbf{Z}_n$ using the mod operator.
- Several properties were defined for the modulo operation in this chapter.
- In $\mathbf{Z}_n$, two numbers a and b are additive inverses of each other if $a + b \equiv 0 \pmod{n}$. They are the multiplicative inverse of each other if $a \times b \equiv 1 \pmod{n}$. The integer a has a multiplicative inverse in $\mathbf{Z}_n$ if and only if gcd $(n, a) = 1$ (a and n are relatively prime).
- The extended Euclidean algorithm finds the multiplicative inverses of b in $\mathbf{Z}_n$ when n and b are given and gcd $(n, b) = 1$. The multiplicative inverse of b is the value of t after being mapped to $\mathbf{Z}_n$.
- A matrix is a rectangular array of $l \times m$ elements, in which l is the number of rows and m is the number of columns. We show a matrix with a boldface uppercase letter such as **A**. The element a_{ij} is located in the ith row and jth column.

- ❑ Two matrices are equal if they have the same number of rows and columns and the corresponding elements are equal.
- ❑ Addition and subtraction are done only on matrices of equal sizes. We can multiply two matrices of different sizes if the number of columns of the first matrix is the same as the number of rows of the second matrix.
- ❑ In residue matrices, all elements are in $\mathbf{Z}_n$. All operations on residue matrices are done in modular arithmetic. A residue matrix has an inverse if the determinant of the matrix has an inverse.
- ❑ An equation of the form $ax \equiv b \pmod{n}$ may have no solution or a limited number of solutions. If $\gcd(a, n) | b$, there is a limited number of solutions.
- ❑ A set of linear equations with the same modulus can be solved if the matrix formed from the coefficients of variables has an inverse.

2.8 PRACTICE SET

Review Questions

1. Distinguish between $\mathbf{Z}$ and $\mathbf{Z}_n$. Which set can have negative integers? How can we map an integer in $\mathbf{Z}$ to an integer in $\mathbf{Z}_n$?
2. List four properties of divisibility discussed in this chapter. Give an integer with only one divisor. Give an integer with only two divisors. Give an integer with more than two divisors.
3. Define the greatest common divisor of two integers. Which algorithm can effectively find the greatest common divisor?
4. What is a linear Diophantine equation of two variables? How many solutions can such an equation have? How can the solution(s) be found?
5. What is the modulo operator, and what is its application? List all properties we mentioned in this chapter for the modulo operation.
6. Define congruence and compare with equality.
7. Define a residue class and a least residue.
8. What is the difference between the set $\mathbf{Z}_n$ and the set $\mathbf{Z}_n*$? In which set does each element have an additive inverse? In which set does each element have a multiplicative inverse? Which algorithm is used to find the multiplicative inverse of an integer in $\mathbf{Z}_n$?
9. Define a matrix. What is a row matrix? What is a column matrix? What is a square matrix? What type of matrix has a determinant? What type of matrix can have an inverse?
10. Define linear congruence. What algorithm can be used to solve an equation of type $ax \equiv b \pmod{n}$? How can we solve a set of linear equations?

Exercises

11. Which of the following relations are true and which are false?

$$5|26 \qquad 3|123 \qquad 27\nmid 127 \qquad 15\nmid 21 \qquad 23|96 \qquad 8|5$$

12. Using the Euclidean algorithm, find the greatest common divisor of the following pairs of integers.
    a. 88 and 220
    b. 300 and 42
    c. 24 and 320
    d. 401 and 700
13. Solve the following.
    a. Given gcd (a, b) = 24, find gcd (a, b, 16).
    b. Given gcd (a, b, c) = 12, find gcd (a, b, c, 16)
    c. Find gcd (200, 180, and 450).
    d. Find gcd (200, 180, 450, 610).
14. Assume that n is a nonnegative integer.
    a. Find gcd ($2n + 1$, n).
    b. Using the result of part a, find gcd (201, 100), gcd (81, 40), and gcd (501, 250).
15. Assume that n is a nonnegative integer.
    a. Find gcd ($3n + 1$, $2n + 1$).
    b. Using the result of part a, find gcd (301, 201) and gcd (121, 81).
16. Using the extended Euclidean algorithm, find the greatest common divisor of the following pairs and the value of s and t.
    a. 4 and 7
    b. 291 and 42
    c. 84 and 320
    d. 400 and 60
17. Find the results of the following operations.
    a. 22 mod 7
    b. 140 mod 10
    c. −78 mod 13
    d. 0 mod 15
18. Perform the following operations using reduction first.
    a. (273 + 147) mod 10
    b. (4223 + 17323) mod 10
    c. (148 + 14432) mod 12
    d. (2467 + 461) mod 12
19. Perform the following operations using reduction first.
    a. (125×45) mod 10
    b. (424×32) mod 10
    c. (144×34) mod 12
    d. (221×23) mod 22

20. Use the properties of the mod operator to prove the following:
    a. The remainder of any integer when divided by 10 is the rightmost digit.
    b. The remainder of any integer when divided by 100 is the integer made of the two rightmost digits.
    c. The remainder of any integer when divided by 1000 is the integer made of the three rightmost digits.
21. We have been told in arithmetic that the remainder of an integer divided by 5 is the same as the remainder of division of the rightmost digit by 5. Use the properties of the mod operator to prove this claim.
22. We have been told in arithmetic that the remainder of an integer divided by 2 is the same as the remainder of division of the rightmost digit by 2. Use the properties of the mod operator to prove this claim.
23. We have been told in arithmetic that the remainder of an integer divided by 4 is the same as the remainder of division of the two rightmost digits by 4. Use the properties of the mod operator to prove this claim.
24. We have been told in arithmetic that the remainder of an integer divided by 8 is the same as the remainder of division of the rightmost three digits by 8. Use the properties of the mod operator to prove this claim.
25. We have been told in arithmetic that the remainder of an integer divided by 9 is the same as the remainder of division of the sum of its decimal digits by 9. In other words, the remainder of dividing 6371 by 9 is the same as dividing 17 by 9 because $6 + 3 + 7 + 1 = 17$. Use the properties of the mod operator to prove this claim.
26. The following shows the remainders of powers of 10 when divided by 7. We can prove that the pattern will be repeated for higher powers.

$$\begin{array}{lll} 10^0 \bmod 7 = 1 & 10^1 \bmod 7 = 3 & 10^2 \bmod 7 = 2 \\ 10^3 \bmod 7 = -1 & 10^4 \bmod 7 = -3 & 10^5 \bmod 7 = -2 \end{array}$$

    Using the above information, find the remainder of an integer when divided by 7. Test your method with 631453672.
27. The following shows the remainders of powers of 10 when divided by 11. We can prove that the pattern will be repeated for higher powers.

$$10^0 \bmod 11 = 1 \qquad 10^1 \bmod 11 = -1 \qquad 10^2 \bmod 11 = 1 \qquad 10^3 \bmod 11 = -1$$

    Using the above information, find the remainder of an integer when divided by 11. Test your method with 631453672.
28. The following shows the remainders of powers of 10 when divided by 13. We can prove that the pattern will be repeated for higher powers.

$$\begin{array}{lll} 10^0 \bmod 13 = 1 & 10^1 \bmod 13 = -3 & 10^2 \bmod 13 = -4 \\ 10^3 \bmod 13 = -1 & 10^4 \bmod 13 = 3 & 10^5 \bmod 13 = 4 \end{array}$$

    Using the above information, find the remainder of an integer when divided by 13. Test your method with 631453672.

29. Let us assign numeric values to the uppercase alphabet (A = 0, B = 1, . . . Z = 25). We can now do modular arithmetic on the system using modulo 26.
    a. What is (A + N) mod 26 in this system?
    b. What is (A + 6) mod 26 in this system?
    c. What is (Y − 5) mod 26 in this system?
    d. What is (C −10) mod 26 in this system?
30. List all additive inverse pairs in modulus 20.
31. List all multiplicative inverse pairs in modulus 20.
32. Find the multiplicative inverse of each of the following integers in $\mathbf{Z}_{180}$ using the extended Euclidean algorithm.
    a. 38
    b. 7
    c. 132
    d. 24
33. Find the particular and the general solutions to the following linear Diophantine equations.
    a. $25x + 10y = 15$
    b. $19x + 13y = 20$
    c. $14x + 21y = 77$
    d. $40x + 16y = 88$
34. Show that there are no solutions to the following linear Diophantine equations:
    a. $15x + 12y = 13$
    b. $18x + 30y = 20$
    c. $15x + 25y = 69$
    d. $40x + 30y = 98$
35. A post office sells only 39-cent and 15-cent stamps. Find the number of stamps a customer needs to buy to put $2.70 postage on a package. Find a few solutions.
36. Find all solutions to each of the following linear equations:
    a. $3x \equiv 4 \pmod 5$
    b. $4x \equiv 4 \pmod 6$
    c. $9x \equiv 12 \pmod 7$
    d. $256x \equiv 442 \pmod{60}$
37. Find all solutions to each of the following linear equations:
    a. $3x + 5 \equiv 4 \pmod 5$
    b. $4x + 6 \equiv 4 \pmod 6$
    c. $9x + 4 \equiv 12 \pmod 7$
    d. $232x + 42 \equiv 248 \pmod{50}$
38. Find $(\mathbf{A} \times \mathbf{B})$ mod 16 using the matrices in Figure 2.28.

Figure 2.28 *Matrices for Exercise 38*

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & 7 & 10 \end{bmatrix}} \times \underset{\mathbf{B}}{\begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}} \qquad \underset{\mathbf{A}}{\begin{bmatrix} 3 & 4 & 6 \\ 1 & 1 & 8 \\ 5 & 8 & 3 \end{bmatrix}} \times \underset{\mathbf{B}}{\begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 5 & 2 & 4 \end{bmatrix}}$$

39. In Figure 2.29, find the determinant and the multiplicative inverse of each residue matrix over $\mathbf{Z}_{10}$.

Figure 2.29 *Matrices for Exercise 39*

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}} \qquad \underset{\mathbf{B}}{\begin{bmatrix} 4 & 2 \\ 1 & 1 \end{bmatrix}} \qquad \underset{\mathbf{C}}{\begin{bmatrix} 3 & 4 & 6 \\ 1 & 1 & 8 \\ 5 & 8 & 3 \end{bmatrix}}$$

40. Find all solutions to the following sets of linear equations:

a. $3x + 5y \equiv 4 \pmod 5$
$2x + y \equiv 3 \pmod 5$

b. $3x + 2y \equiv 5 \pmod 7$
$4x + 6y \equiv 4 \pmod 7$

c. $7x + 3y \equiv 3 \pmod 7$
$4x + 2y \equiv 5 \pmod 7$

d. $2x + 3y \equiv 5 \pmod 8$
$x + 6y \equiv 3 \pmod 8$

CHAPTER 3

Traditional Symmetric-Key Ciphers

Objectives

This chapter presents a survey of traditional symmetric-key ciphers used in the past. By explaining the principles of such ciphers, it prepares the reader for the next few chapters, which discuss modern symmetric-key ciphers. This chapter has several objectives:

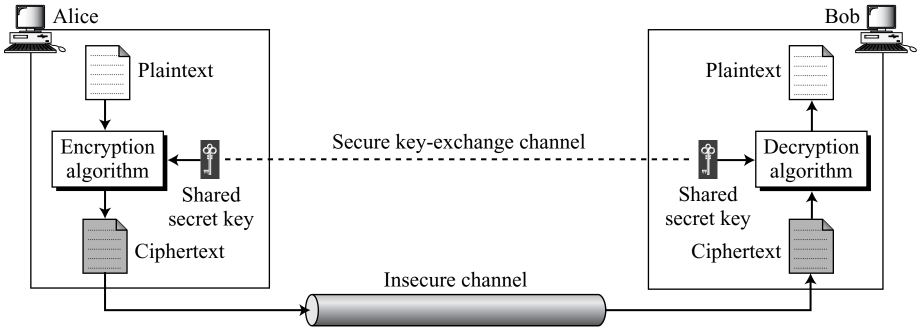
- To define the terms and the concepts of symmetric-key ciphers
- To emphasize the two categories of traditional ciphers: substitution ciphers and transposition ciphers
- To describe the categories of cryptanalysis used to break the symmetric ciphers
- To introduce the concepts of the stream ciphers and block ciphers
- To discuss some very dominant ciphers used in the past, such as the Enigma machine

The general idea behind symmetric-key ciphers will be introduced here using examples from cryptography. The terms and definitions presented are used in all later chapters on symmetric-key ciphers. We then discuss traditional symmetric-key ciphers. These ciphers are not used today, but we study them for several reasons. First, they are simpler than modern ciphers and easier to understand. Second, they show the basic foundation of cryptography and encipherment: This foundation can be used to better understand modern ciphers. Third, they provide the rationale for using modern ciphers, because the traditional ciphers can be easily attacked using a computer. Ciphers that were secure in earlier eras are no longer secure in this computer age.

3.1 INTRODUCTION

Figure 3.1 shows the general idea behind a symmetric-key cipher.

Figure 3.1 *General idea of symmetric-key cipher*



In Figure 3.1, an entity, Alice, can send a message to another entity, Bob, over an insecure channel with the assumption that an adversary, Eve, cannot understand the contents of the message by simply eavesdropping over the channel.

The original message from Alice to Bob is called **plaintext;** the message that is sent through the channel is called the **ciphertext.** To create the ciphertext from the plaintext, Alice uses an **encryption algorithm** and a **shared secret key.** To create the plaintext from ciphertext, Bob uses a **decryption algorithm** and the same secret key. We refer to encryption and decryption algorithms as **ciphers.** A **key** is a set of values (numbers) that the cipher, as an algorithm, operates on.

Note that the symmetric-key encipherment uses a single key (the key itself may be a set of values) for both encryption and decryption. In addition, the encryption and decryption algorithms are inverses of each other. If P is the plaintext, C is the ciphertext, and K is the key, the encryption algorithm $E_k(x)$ creates the ciphertext from the plaintext; the decryption algorithm $D_k(x)$ creates the plaintext from the ciphertext. We assume that $E_k(x)$ and $D_k(x)$ are inverses of each other: they cancel the effect of each other if they are applied one after the other on the same input. We have

Encryption: $C = E_k(P)$ Decryption: $P = D_k(C)$

In which, $D_k(E_k(x)) = E_k(D_k(x)) = x$

We can prove that the plaintext created by Bob is the same as the one originated by Alice. We assume that Bob creates P_1; we prove that $P_1 = P$:

Alice: $C = E_k(P)$ **Bob:** $P_1 = D_k(C) = D_k(E_k(P)) = P$

We need to emphasize that, according to Kerckhoff's principle (described later), it is better to make the encryption and decryption public but keep the shared key secret.

This means that Alice and Bob need another channel, a secured one, to exchange the secret key. Alice and Bob can meet once and exchange the key personally. The secured channel here is the face-to-face exchange of the key. They can also trust a third party to give them the same key. They can create a temporary secret key using another kind of cipher—asymmetric-key ciphers—which will be described in later chapters. The concern will be dealt with in future chapters. In this chapter, we assume that there is an established secret key between Alice and Bob.

Using symmetric-key encipherment, Alice and Bob can use the same key for communication on the other direction, from Bob to Alice. This is why the method is called symmetric.

Another element in symmetric-key encipherment is the number of keys. Alice needs another secret key to communicate with another person, say David. If there are m people in a group who need to communicate with each other, how many keys are needed? The answer is $(m \times (m - 1))/2$ because each person needs $m - 1$ keys to communicate with the rest of the group, but the key between A and B can be used in both directions. We will see in later chapters how this problem is being handled.

Encryption can be thought of as locking the message in a box; decryption can be thought of as unlocking the box. In symmetric-key encipherment, the same key locks and unlocks as shown in Figure 3.2. Later chapters show that the asymmetric-key encipherment needs two keys, one for locking and one for unlocking.

Figure 3.2 *Symmetric-key encipherment as locking and unlocking with the same key*

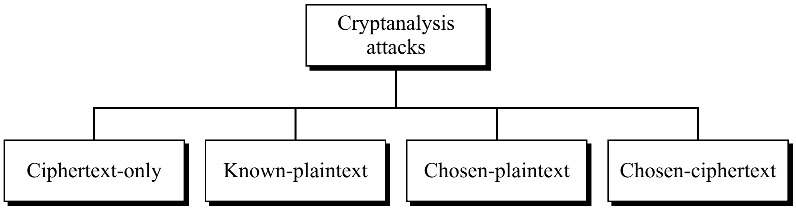
Kerckhoff's Principle

Although it may appear that a cipher would be more secure if we hide both the encryption/decryption algorithm and the secret key, this is not recommended. Based on **Kerckhoff's principle,** one should always assume that the adversary, Eve, knows the encryption/decryption algorithm. The resistance of the cipher to attack must be based only on the secrecy of the key. In other words, guessing the key should be so difficult that there is no need to hide the encryption/decryption algorithm. This principle manifests itself more clearly when we study modern ciphers. There are only a few algorithms for modern ciphers today. The **key domain** for each algorithm, however, is so large that it makes it difficult for the adversary to find the key.

Cryptanalysis

As cryptography is the science and art of creating secret codes, **cryptanalysis** is the science and art of breaking those codes. In addition to studying cryptography techniques, we also need to study cryptanalysis techniques. This is needed, not to break other people's codes, but to learn how vulnerable our cryptosystem is. The study of cryptanalysis helps us create better secret codes. There are four common types of cryptanalysis attacks, as shown in Figure 3.3. We will study some of these attacks on particular ciphers in this and future chapters.

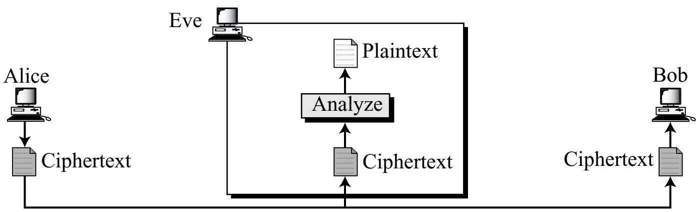
Figure 3.3 *Cryptanalysis attacks*



Ciphertext-Only Attack

In a **ciphertext-only attack,** Eve has access to only some ciphertext. She tries to find the corresponding key and the plaintext. The assumption is that Eve knows the algorithm and can intercept the ciphertext. The ciphertext-only attack is the most probable one because Eve needs only the ciphertext for this attack. To thwart the decryption of a message by an adversary, a cipher must be very resisting to this type of attack. Figure 3.4 shows the process.

Figure 3.4 *Ciphertext-only attack*



Various methods can be used in ciphertext-only attack. We mention some common ones here.

Brute-Force Attack

In the **brute-force method** or **exhaustive-key-search method,** Eve tries to use all possible keys. We assume that Eve knows the algorithm and knows the key domain (the list of all possible keys). Using the intercepted cipher, Eve decrypts the ciphertext with every possible key until the plaintext makes sense. Using brute-force attack was a difficult task in the past; it is easier today using a computer. To prevent this type of attack, the number of possible keys must be very large.

Statistical Attack

The cryptanalyst can benefit from some inherent characteristics of the plaintext language to launch a **statistical attack.** For example, we know that the letter E is the most-frequently used letter in English text. The cryptanalyst finds the mostly-used character in the ciphertext and assumes that the corresponding plaintext character is E. After finding a few pairs, the analyst can find the key and use it to decrypt the message. To prevent this type of attack, the cipher should hide the characteristics of the language.

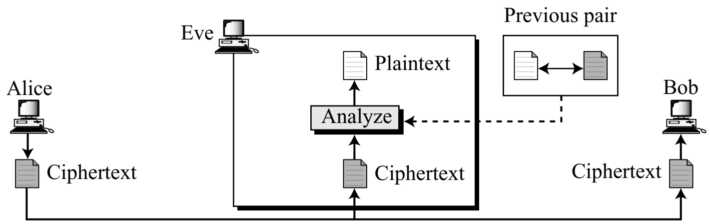
Pattern Attack

Some ciphers may hide the characteristics of the language, but may create some patterns in the ciphertext. A cryptanalyst may use a **pattern attack** to break the cipher. Therefore, it is important to use ciphers that make the ciphertext look as random as possible.

Known-Plaintext Attack

In a **known-plaintext attack,** Eve has access to some plaintext/ciphertext pairs in addition to the intercepted ciphertext that she wants to break, as shown in Figure 3.5.

Figure 3.5 *Known-plaintext attack*

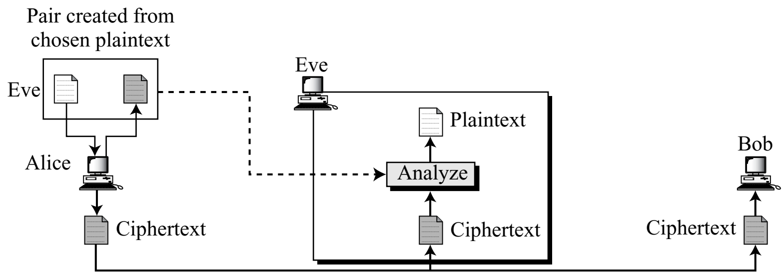


The plaintext/ciphertext pairs have been collected earlier. For example, Alice has sent a secret message to Bob, but she has later made the contents of the message public. Eve has kept both the ciphertext and the plaintext to use them to break the next secret message from Alice to Bob, assuming that Alice has not changed her key. Eve uses the relationship between the previous pair to analyze the current ciphertext. The same methods used in a ciphertext-only attack can be applied here. This attack is easier to implement because Eve has more information to use for analysis. However, it is less likely to happen because Alice may have changed her key or may have not disclosed the contents of any previous messages.

Chosen-Plaintext Attack

The **chosen-plaintext attack** is similar to the known-plaintext attack, but the plaintext/ciphertext pairs have been chosen by the attacker herself. Figure 3.6 shows the process.

Figure 3.6 *Chosen-plaintext attack*

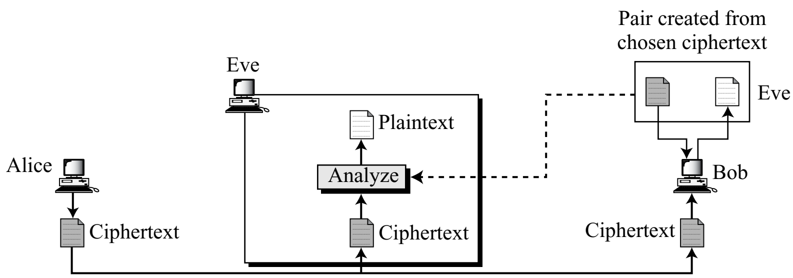


This can happen, for example, if Eve has access to Alice's computer. She can choose some plaintext and intercept the created ciphertext. Of course, she does not have the key because the key is normally embedded in the software used by the sender. This type of attack is much easier to implement, but it is much less likely to happen.

Chosen-Ciphertext Attack

The **chosen-ciphertext attack** is similar to the chosen-plaintext attack, except that Eve chooses some ciphertext and decrypts it to form a ciphertext/plaintext pair. This can happen if Eve has access to Bob's computer. Figure 3.7 shows the process.

Figure 3.7 *Chosen-ciphertext attack*



Categories of Traditional Ciphers

We can divide traditional symmetric-key ciphers into two broad categories: substitution ciphers and transposition ciphers. In a substitution cipher, we replace one symbol in the ciphertext with another symbol; in a transposition cipher, we reorder the position of symbols in the plaintext.

3.2 SUBSTITUTION CIPHERS

A **substitution cipher** replaces one symbol with another. If the symbols in the plaintext are alphabetic characters, we replace one character with another. For example, we can replace letter A with letter D, and letter T with letter Z. If the symbols are digits (0 to 9), we can replace 3 with 7, and 2 with 6. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.

A substitution cipher replaces one symbol with another.

Monoalphabetic Ciphers

We first discuss a group of substitution ciphers called the **monoalphabetic ciphers.** In monoalphabetic substitution, a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text. For example, if the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D. In other words, the relationship between letters in the plaintext and the ciphertext is one-to-one.

In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.

Example 3.1

The following shows a plaintext and its corresponding ciphertext. We use lowercase characters to show the plaintext; we use uppercase characters to show the ciphertext. The cipher is probably monoalphabetic because both l's (els) are encrypted as O's.

Plaintext: hello   **Ciphertext:** KHOOR

Example 3.2

The following shows a plaintext and its corresponding ciphertext. The cipher is not monoalphabetic because each l (el) is encrypted by a different character. The first l (el) is encrypted as N; the second as Z.

Plaintext: hello   **Ciphertext:** ABNZF

Additive Cipher

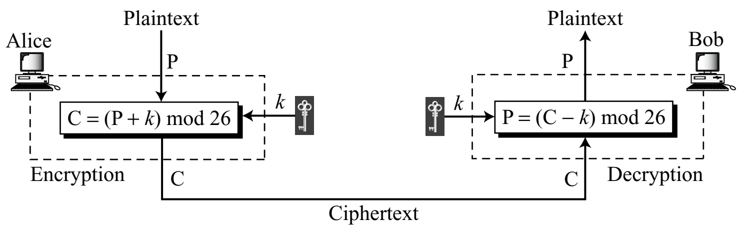
The simplest monoalphabetic cipher is the **additive cipher.** This cipher is sometimes called a **shift cipher** and sometimes a **Caesar cipher,** but the term *additive cipher* better reveals its mathematical nature. Assume that the plaintext consists of lowercase letters (a to z), and that the ciphertext consists of uppercase letters (A to Z). To be able to apply mathematical operations on the plaintext and ciphertext, we assign numerical values to each letter (lower- or uppercase), as shown in Figure 3.8.

Figure 3.8 *Representation of plaintext and ciphertext characters in* $\mathbf{Z}_{26}$

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

In Figure 3.8 each character (lowercase or uppercase) is assigned an integer in $\mathbf{Z}_{26}$. The secret key between Alice and Bob is also an integer in $\mathbf{Z}_{26}$. The encryption algorithm adds the key to the plaintext character; the decryption algorithm subtracts the key from the ciphertext character. All operations are done in $\mathbf{Z}_{26}$. Figure 3.9. shows the process.

Figure 3.9 *Additive cipher*



We can easily prove that the encryption and decryption are inverse of each other because plaintext created by Bob (P_1) is the same as the one sent by Alice (P).

$$P_1 = (C - k) \text{ mod } 26 = (P + k - k) \text{ mode } 26 = P$$

When the cipher is additive, the plaintext, ciphertext, and key are integers in $\mathbf{Z}_{26}$.

Example 3.3

Use the additive cipher with key = 15 to encrypt the message “hello”.

Solution

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

The result is "WTAAD". Note that the cipher is monoalphabetic because two instances of the same plaintext character (l's) are encrypted as the same character (A).

Example 3.4

Use the additive cipher with key = 15 to decrypt the message "WTAAD".

Solution

We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22	Decryption: (22 − 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 − 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 − 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 − 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 − 15) mod 26	Plaintext: 14 → o

The result is "hello". Note that the operation is in modulo 26 (see Chapter 2), which means that a negative result needs to be mapped to $\mathbf{Z}_{26}$ (for example −15 becomes 11).

Shift Cipher

Historically, additive ciphers are called shift ciphers. The reason is that the encryption algorithm can be interpreted as "shift *key* characters down" and the encryption algorithm can be interpreted as "shift *key* character up". For example, if the key = 15, the encryption algorithm shifts 15 characters down (toward the end of the alphabet). The decryption algorithm shifts 15 characters up (toward the beginning of the alphabet). Of course, when we reach the end or the beginning of the alphabet, we wrap around (manifestation of modulo 26).

Caesar Cipher

Julius Caesar used an additive cipher to communicate with his officers. For this reason, additive ciphers are sometimes referred to as the **Caesar cipher.** Caesar used a key of 3 for his communications.

Additive ciphers are sometimes referred to as shift ciphers or Caesar cipher.

Cryptanalysis

Additive ciphers are vulnerable to ciphertext-only attacks using exhaustive key searches (brute-force attacks). The key domain of the additive cipher is very small; there are only 26 keys. However, one of the keys, zero, is useless (the ciphertext is the same as the plaintext). This leaves only 25 possible keys. Eve can easily launch a brute-force attack on the ciphertext.

Example 3.5

Eve has intercepted the ciphertext "UVACLYFZLJBYL". Show how she can use a brute-force attack to break the cipher.

Solution

Eve tries keys from 1 to 7. With a key of 7, the plaintext is "not very secure", which makes sense.

Ciphertext: UVACLYFZLJBYL

K = 1 → **Plaintext:** tuzbkxeykiaxk
K = 2 → **Plaintext:** styajwdxjhzwj
K = 3 → **Plaintext:** rsxzivcwigyvi
K = 4 → **Plaintext:** qrwyhubvhfxuh
K = 5 → **Plaintext:** pqvxgtaugewtg
K = 6 → **Plaintext:** opuwfsztfdvsf
K = 7 → **Plaintext:** notverysecure

Additive ciphers are also subject to statistical attacks. This is especially true if the adversary has a long ciphertext. The adversary can use the frequency of occurrence of characters for a particular language. Table 3.1 shows the frequency for an English text of 100 characters.

Table 3.1 *Frequency of occurrence of letters in an English text*

Letter	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

However, sometimes it is difficult to analyze a ciphertext based only on information about the frequency of a single letter; we may need to know the occurrence of specific letter combinations. We need to know the frequency of two-letter or three-letter strings in the ciphertext and compare them with the frequency of two-letter or three-letter strings in the underlying language of the plaintext.

The most common two-letter groups (**digrams**) and three-letter groups (**trigrams**) for the English text are shown in Table 3.2.

Table 3.2 *Grouping of digrams and trigrams based on their frequency in English*

Digram	TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI, OF
Trigram	THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR, DTH

Example 3.6

Eve has intercepted the following ciphertext. Using a statistical attack, find the plaintext.

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILY-
VVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW

Solution

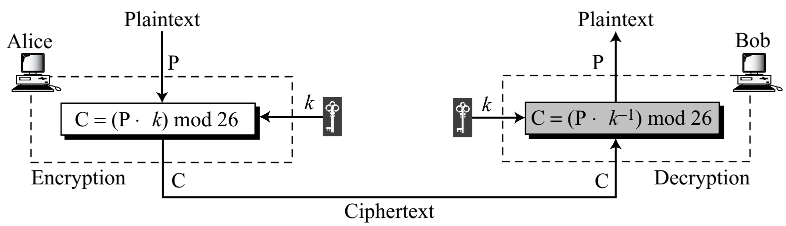
When Eve tabulates the frequency of letters in this ciphertext, she gets: I =14, V =13, S =12, and so on. The most common character is I with 14 occurrences. This shows that character I in the ciphertext probably corresponds to the character e in plaintext. This means key = 4. Eve deciphers the text to get

the house is now for sale for four million dollars it is worth more hurry before the seller receives more offers

Multiplicative Ciphers

In a **multiplicative cipher,** the encryption algorithm specifies multiplication of the plaintext by the key and the decryption algorithm specifies division of the ciphertext by the key as shown in Figure 3.10. However, since operations are in $\mathbf{Z}_{26}$, decryption here means multiplying by the multiplicative inverse of the key. Note that the key needs to belong to the set $\mathbf{Z}_{26}$* to guarantee that the encryption and decryption are inverses of each other.

Figure 3.10 *Multiplicative cipher*



In a multiplicative cipher, the plaintext and ciphertext are integers in $\mathbf{Z}_{26}$; the key is an integer in $\mathbf{Z}_{26}$*.

Example 3.7

What is the key domain for any multiplicative cipher?

Solution

The key needs to be in $\mathbf{Z}_{26}$*. This set has only 12 members: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.

Example 3.8

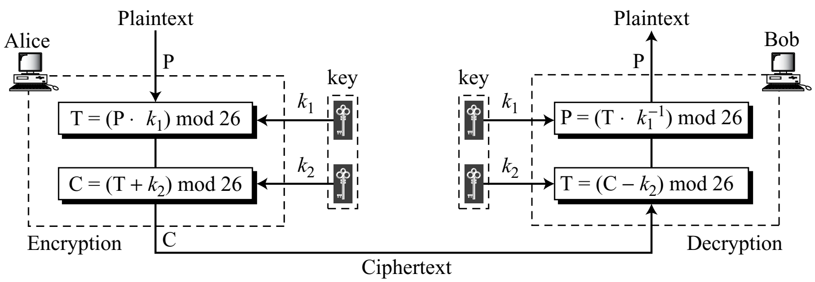
We use a multiplicative cipher to encrypt the message "hello" with a key of 7. The ciphertext is "XCZZU".

Plaintext: h → 07	Encryption: (07×07) mod 26	ciphertext: 23 → X
Plaintext: e → 04	Encryption: (04×07) mod 26	ciphertext: 02 → C
Plaintext: l → 11	Encryption: (11×07) mod 26	ciphertext: 25 → Z
Plaintext: l → 11	Encryption: (11×07) mod 26	ciphertext: 25 → Z
Plaintext: o → 14	Encryption: (14×07) mod 26	ciphertext: 20 → U

Affine Cipher

We can combine the additive and multiplicative ciphers to get what is called the **affine cipher**—a combination of both ciphers with a pair of keys. The first key is used with the multiplicative cipher; the second key is used with the additive cipher. Figure 3.11 shows that the affine cipher is actually two ciphers, applied one after another. We could have shown only one complex operation for the encryption or decryption such as $C = (P \times k_1 + k_2) \bmod 26$ and $P = ((C - k_2) \times k_1^{-1}) \bmod 26$. However, we have used a temporary result (T) and have indicated two separate operations to show that whenever we use a combination of ciphers we should be sure that each one has an inverse at the other side of the line and that they are used in reverse order in the encryption and decryption. If addition is the last operation in encryption, then subtraction should be the first in decryption.

Figure 3.11 *Affine cipher*



In the affine cipher, the relationship between the plaintext P and the ciphertext C is

$$C = (P \times k_1 + k_2) \bmod 26 \qquad P = ((C - k_2) \times k_1^{-1}) \bmod 26$$

where k_1^{-1} is the multiplicative inverse of k_1 and $-k_2$ is the additive inverse of k_2

Example 3.9

The affine cipher uses a pair of keys in which the first key is from $\mathbf{Z}_{26}*$ and the second is from $\mathbf{Z}_{26}$. The size of the key domain is $26 \times 12 = 312$.

Example 3.10

Use an affine cipher to encrypt the message "hello" with the key pair (7, 2).

Solution

We use 7 for the multiplicative key and 2 for the additive key. We get "ZEBBW".

P: h $\rightarrow$ 07	Encryption: $(07 \times 7 + 2) \bmod 26$	C: 25 $\rightarrow$ Z
P: e $\rightarrow$ 04	Encryption: $(04 \times 7 + 2) \bmod 26$	C: 04 $\rightarrow$ E
P: l $\rightarrow$ 11	Encryption: $(11 \times 7 + 2) \bmod 26$	C: 01 $\rightarrow$ B
P: l $\rightarrow$ 11	Encryption: $(11 \times 7 + 2) \bmod 26$	C: 01 $\rightarrow$ B
P: o $\rightarrow$ 14	Encryption: $(14 \times 7 + 2) \bmod 26$	C: 22 $\rightarrow$ W

Example 3.11

Use the affine cipher to decrypt the message "ZEBBW" with the key pair (7, 2) in modulus 26.

Solution

Add the additive inverse of $-2 \equiv 24 \pmod{26}$ to the received ciphertext. Then multiply the result by the multiplicative inverse of $7^{-1} \equiv 15 \pmod{26}$ to find the plaintext characters. Because 2 has an additive inverse in $\mathbf{Z}_{26}$ and 7 has a multiplicative inverse in $\mathbf{Z}_{26}*$, the plaintext is exactly what we used in Example 3.10.

C: Z $\rightarrow$ 25	Decryption: $((25 - 2) \times 7^{-1}) \bmod 26$	P:07 $\rightarrow$ h
C: E $\rightarrow$ 04	Decryption: $((04 - 2) \times 7^{-1}) \bmod 26$	P:04 $\rightarrow$ e
C: B $\rightarrow$ 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 $\rightarrow$ l
C: B $\rightarrow$ 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 $\rightarrow$ l
C: W $\rightarrow$ 22	Decryption: $((22 - 2) \times 7^{-1}) \bmod 26$	P:14 $\rightarrow$ o

Example 3.12

The additive cipher is a special case of an affine cipher in which $k_1 = 1$. The multiplicative cipher is a special case of affine cipher in which $k_2 = 0$.

Cryptanalysis of Affine Cipher

Although the brute-force and statistical method of ciphertext-only attack can be used, let us try a chosen-plaintext attack. Assume that Eve intercepts the following ciphertext:

PWUFFOGWCHFDWIWEJOUUNJORSMDWRHVCMWJUPVCCG

Eve also very briefly obtains access to Alice's computer and has only enough time to type a two-letter plaintext: "et". She then tries to encrypt the short plaintext using two different algorithms, because she is not sure which one is the affine cipher.

Algorithm 1:	Plaintext: et	ciphertext: $\rightarrow$ WC
Algorithm 2:	Plaintext: et	ciphertext: $\rightarrow$ WF

To find the key, Eve uses the following strategy:

a. Eve knows that if the first algorithm is affine, she can construct the following two equations based on the first data set.

e $\rightarrow$ W	04 $\rightarrow$ 22	$(04 \times k_1 + k_2) \equiv 22 \pmod{26}$
t $\rightarrow$ C	19 $\rightarrow$ 02	$(19 \times k_1 + k_2) \equiv 02 \pmod{26}$

As we learned in Chapter 2, these two congruence equations can be solved and the values of k_1 and k_2 can be found. However, this answer is not acceptable because $k_1 = 16$ cannot be the first part of the key. Its value, 16, does not have a multiplicative inverse in $\mathbf{Z}_{26*}$.

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 19 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 19 & 7 \\ 3 & 24 \end{bmatrix} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 16 \\ 10 \end{bmatrix} \longrightarrow k_1 = 16 \quad k_2 = 10$$

b. Eve now tries the result of the second set of data.

e $\rightarrow$ W	04 $\rightarrow$ 22	$(04 \times k_1 + k_2) \equiv 22 \pmod{26}$
t $\rightarrow$ F	19 $\rightarrow$ 05	$(19 \times k_1 + k_2) \equiv 05 \pmod{26}$

The square matrix and its inverse are the same. Now she has $k_1 = 11$ and $k_2 = 4$. This pair is acceptable because k_1 has a multiplicative inverse in $\mathbf{Z}_{26*}$. She tries the pair of keys (19, 22), which are the inverse of the pair (11, 4), to decipher the message. The plaintext is

best time of the year is spring when flowers bloom

Monoalphabetic Substitution Cipher

Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to brute-force attack. After Alice and Bob agreed to a single key, that key is used to encrypt each letter in the plaintext or decrypt each letter in the ciphertext. In other words, the key is independent from the letters being transferred.

A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. Alice and Bob can agree on a table showing the mapping for each character. Figure 3.12 shows an example of such a mapping.

Figure 3.12 *An example key for monoalphabetic substitution cipher*

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

Example 3.13

We can use the key in Figure 3.12 to encrypt the message

this message is easy to encrypt but hard to find the key

The ciphertext is

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

Cryptanalysis

The size of the key space for the monoalphabetic substitution cipher is 26! (almost 4×10^{26}). This makes a brute-force attack extremely difficult for Eve even if she is using a powerful computer. However, she can use statistical attack based on the frequency of characters. The cipher does not change the frequency of characters.

The monoalphabetic ciphers do not change the frequency of characters in the ciphertext, which makes the ciphers vulnerable to statistical attack.

Polyalphabetic Ciphers

In **polyalphabetic substitution,** each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. For example, “a” could be enciphered as “D” in the beginning of the text, but as “N” at the middle. Polyalphabetic ciphers have the advantage of hiding the letter frequency of the underlying language. Eve cannot use single-letter frequency statistic to break the ciphertext.

To create a polyalphabetic cipher, we need to make each ciphertext character dependent on both the corresponding plaintext character and the position of the plaintext character in the message. This implies that our key should be a stream of subkeys, in which each subkey depends somehow on the position of the plaintext character that uses that subkey for encipherment. In other words, we need to have a key stream $k = (k_1, k_2, k_3, \ldots)$ in which k_i is used to encipher the ith character in the plaintext to create the ith character in the ciphertext.

Autokey Cipher

To see the position dependency of the key, let us discuss a simple polyalphabetic cipher called the **autokey cipher.** In this cipher, the key is a stream of subkeys, in which each subkey is used to encrypt the corresponding character in the plaintext. The first subkey is a predetermined value secretly agreed upon by Alice and Bob. The second subkey is the value of the first plaintext character (between 0 and 25). The third subkey is the value of the second plaintext. And so on.

$P = P_1P_2P_3 \ldots$ $\quad C = C_1C_2C_3\ldots$ $\quad k = (k_1, P_1, P_2, \ldots)$

Encryption: $C_i = (P_i + k_i) \bmod 26$ $\quad$ Decryption: $P_i = (C_i - k_i) \bmod 26$

The name of the cipher, *autokey*, implies that the subkeys are automatically created from the plaintext cipher characters during the encryption process.

Example 3.14

Assume that Alice and Bob agreed to use an autokey cipher with initial key value $k_1 = 12$. Now Alice wants to send Bob the message "Attack is today". Enciphering is done character by character. Each character in the plaintext is first replaced by its integer value as shown in Figure 3.8. The first subkey is added to create the first ciphertext character. The rest of the key is created as the plaintext characters are read. Note that the cipher is polyalphabetic because the three occurrences of "a" in the plaintext are encrypted differently. The three occurrences of the "t" are enciphered differently.

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	*12*	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	**M**	**T**	**M**	**T**	**C**	**M**	**S**	**A**	**L**	**H**	**R**	**D**	**Y**

Cryptanalysis

The autokey cipher definitely hides the single-letter frequency statistics of the plaintext. However, it is still as vulnerable to the brute-force attack as the additive cipher. The first subkey can be only one of the 25 values (1 to 25). We need polyalphabetic ciphers that not only hide the characteristics of the language but also have large key domains.

Playfair Cipher

Another example of a polyalphabetic cipher is the **Playfair cipher** used by the British army during World War I. The secret key in this cipher is made of 25 alphabet letters arranged in a 5 × 5 matrix (letters I and J are considered the same when encrypting). Different arrangements of the letters in the matrix can create many different secret keys. One of the possible arrangements is shown in Figure 3.13. We have dropped the letters in the matrix diagonally starting from the top right-hand corner.

Figure 3.13 *An example of a secret key in the Playfair cipher*

Secret Key =

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

Before encryption, if the two letters in a pair are the same, a bogus letter is inserted to separate them. After inserting bogus letters, if the number of characters in the plaintext is odd, one extra bogus character is added at the end to make the number of characters even.

The cipher uses three rules for encryption:

a. If the two letters in a pair are located in the same row of the secret key, the corresponding encrypted character for each letter is the next letter to the right in the same row (with wrapping to the beginning of the row if the plaintext letter is the last character in the row).
b. If the two letters in a pair are located in the same column of the secret key, the corresponding encrypted character for each letter is the letter beneath it in the same column (with wrapping to the beginning of the column if the plaintext letter is the last character in the column).
c. If the two letters in a pair are not in the same row or column of the secret, the corresponding encrypted character for each letter is a letter that is in its own row but in the same column as the other letter.

The Playfair cipher meets our criteria for a polyalphabetic cipher. The key is a stream of subkeys in which the subkeys are created two at a time. In Playfair cipher, the key stream and the cipher stream are the same. This means that the above-mentioned rules can be thought of as the rules for creating the key stream. The encryption algorithm takes a pair of characters from the plaintext and creates a pair of subkeys by following the above-mentioned rules. We can say that the key stream depends on the position of the character in the plaintext. Position dependency has a different interpretation here: the subkey for each plaintext character depends on the next or previous neighbor. Looking at the Playfair cipher in this way, the ciphertext is actually the key stream.

$\mathrm{P} = \mathrm{P}_1\mathrm{P}_2\mathrm{P}_3 \ldots$ $\mathrm{C} = \mathrm{C}_1\mathrm{C}_2\mathrm{C}_3 \ldots$ $k = [(k_1, k_2), (k_3, k_4), \ldots]$

Encryption: $\mathrm{C}_\mathrm{i} = k_\mathrm{i}$ Decryption: $\mathrm{P}_\mathrm{i} = k_\mathrm{i}$

Example 3.15

Let us encrypt the plaintext "hello" using the key in Figure 3.13. When we group the letters in two-character pairs, we get "he, ll, o". We need to insert an x between the two l's (els), giving "he, lx, lo". We have

he → EC lx → QZ lo → BX
Plaintext: hello Ciphertext: ECQZBX

We can see from this example that the cipher is actually a polyalphabetic cipher: the two occurrences of the letter "l" (el) are encrypted as "Q" and "B".

Cryptanalysis of a Playfair Cipher

Obviously a brute-force attack on a Playfair cipher is very difficult. The size of the key domain is 25! (factorial 25). In addition, the encipherment hides the single-letter frequency of the characters. However, the frequencies of diagrams are preserved (to some extent because of filler insertion), so a cryptanalyst can use a ciphertext-only attack based on the digram frequency test to find the key.

Vigenere Cipher

One interesting kind of polyalphabetic cipher was designed by Blaise de Vigenere, a sixteenth-century French mathematician. A **Vigenere cipher** uses a different strategy to create the key stream. The key stream is a repetition of an initial secret key stream of length *m,* where we have $1 \leq m \leq 26$. The cipher can be described as follows where $(k_1, k_2, \ldots, k_m)$ is the initial secret key agreed to by Alice and Bob.

$$P = P_1P_2P_3 \ldots \qquad C = C_1C_2C_3 \ldots \qquad K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$$

$$\text{Encryption: } C_i = P_i + k_i \qquad \text{Decryption: } P_i = C_i - k_i$$

One important difference between the Vigenere cipher and the other two polyalphabetic ciphers we have looked at, is that the Vigenere key stream does not depend on the plaintext characters; it depends only on the position of the character in the plaintext. In other words, the key stream can be created without knowing what the plaintext is.

Example 3.16

Let us see how we can encrypt the message "She is listening" using the 6-character keyword "*PASCAL*". The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***
C's values:	**07**	**07**	**22**	**10**	**18**	**22**	**23**	**18**	**11**	**6**	**13**	**19**	**02**	**06**
Ciphertext:	**H**	**H**	**W**	**K**	**S**	**W**	**X**	**S**	**L**	**G**	**N**	**T**	**C**	**G**

Example 3.17

Vigenere cipher can be seen as combinations of *m* additive ciphers. Figure 3.14 shows how the plaintext of the previous example can be thought of as six different pieces, each encrypted separately. The figure helps us later understand the cryptanalysis of Vigenere ciphers. There are *m* pieces of the plaintext, each encrypted with a different key, to make *m* pieces of ciphertext.

Example 3.18

Using Example 3.18, we can say that the additive cipher is a special case of Vigenere cipher in which $m = 1$.

Vigenere Tableau

Another way to look at Vigenere ciphers is through what is called a **Vigenere tableau** shown in Table 3.3.

Figure 3.14 *A Vigenere cipher as a combination of m additive ciphers*

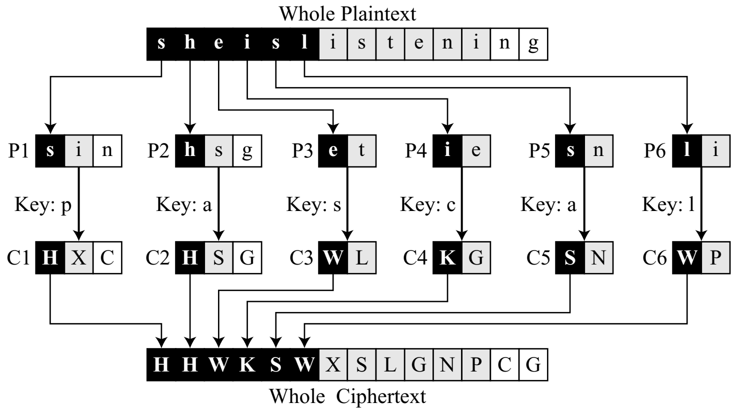


Table 3.3 *A Vigenere tableau*

	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	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

The first row shows the plaintext character to be encrypted. The first column contains the characters to be used by the key. The rest of the tableau shows the ciphertext characters.To find the ciphertext for the plaintext "she is listening" using the word "*PASCAL*" as the key, we can find "s" in the first row, "*P*" in the first column, the cross section is the ciphertext character "H". We can find "h" in the first row and "*A*" in the second column, the cross section is the ciphertext character "H". We do the same until all ciphertext characters are found.

Cryptanalysis of Vigenere Ciphers

Vigenere ciphers, like all polyalphabetic ciphers, do not preserve the frequency of characters. However, Eve still can use some techniques to decipher an intercepted ciphertext. The cryptanalysis here consists of two parts: finding the length of the key and finding the key itself.

1. Several methods have been devised to find the length of the key. One method is discussed here. In the so-called **Kasiski test,** the cryptanalyst searches for repeated text segments, of at least three characters, in the ciphertext. Suppose that two of these segments are found and the distance between them is d. The cryptanalyst assumes that $d|m$ where m is the key length. If more repeated segments can be found with distances $d_1, d_2, \ldots, d_n$, then gcd $(d_1, d_2, \ldots, d_n)/m$. This assumption is logical because if two characters are the same and are $k \times m$ (k = 1, 2, …) characters apart in the plaintext, they are the same and $k \times m$ characters apart in the ciphertext. Cryptanalyst uses segments of at least three characters to avoid the cases where the characters in the key are not distinct. Example 3.20 may help us to understand the reason.
2. After the length of the key has been found, the cryptanalyst uses the idea shown in Example 3.18. She divides the ciphertext into m different pieces and applies the method used to cryptanalyze the additive cipher, including frequency attack. Each ciphertext piece can be decrypted and put together to create the whole plaintext. In other words, the whole ciphertext does not preserve the single-letter frequency of the plaintext, but each piece does.

Example 3.19

Let us assume we have intercepted the following ciphertext:

LIOMWGFEGGDVWGHHCQUCRHRWAGWIOWQLKGZETKKMEVLWPCZVGTH-
VTSGXQOVGCSVETQLTJSUMVWVEUVLXEWSLGFZMVVWLGYHCUSWXQH-
KVGSHEEVFLCFDGVSUMPHKIRZDMPHHBVWVWJWIXGFWLTSHGJOUEEHH-
VUCFVGOWICQLTJSUXGLW

The Kasiski test for repetition of three-character segments yields the results shown in Table 3.4.

Table 3.4 *Kasiski test for Example 3.19*

String	*First Index*	*Second Index*	*Difference*
JSU	68	168	100
SUM	69	117	48
VWV	72	132	60
MPH	119	127	8

The greatest common divisor of differences is 4, which means that the key length is multiple of 4. First try $m = 4$. Divide the ciphertext into four pieces. Piece C_1 is made of characters 1, 5, 9, …; piece C2 is made of characters 2, 6, 10, …; and so on. Use the statistical attack on each piece separately. Interleave the decipher pieces one character at a time to get the whole plaintext. If the plaintext does not make sense, try with another *m*.

```
C1: LWGWCRAOKTEPGTQCTJVUEGVGUQGECVPRPVJGTJEUGCJG
P1: jueuapymircneroarhtsthihytrahcieixsthcarrehe
C2: IGGGQHGWGKVCTSOSQSWVWFVYSHSVFSHZHWWFSOHCOQSL
P2: ussscts iswhofeaeceihcetesoecatnpntherhctecex
C3: OFDHURWQZKLZHGVVLUVLSZWHWKHFDUKDHVIWHUHFWLUW
P3: lcaerotnwhiwedssirsiirhketehretltiideatrairt
C4: MEVHCWILEMWVVXGETMEXLMLCXVELGMIMBWXLGEVVITX
P4: iardysehaisrrtcapiafpwtethecarhaesfterectpt
```

In this case, the plaintext makes sense.

Julius Caesar used a cryptosystem in his wars, which is now referred to as Caesar cipher. It is an additive cipher with the key set to three. Each character in the plaintext is shifted three characters to create ciphertext.

Hill Cipher

Another interesting example of a polyalphabetic cipher is the **Hill cipher** invented by Lester S. Hill. Unlike the other polyalphabetic ciphers we have already discussed, the plaintext is divided into equal-size blocks. The blocks are encrypted one at a time in such a way that each character in the block contributes to the encryption of other characters in the block. For this reason, the Hill cipher belongs to a category of ciphers called *block ciphers*. The other ciphers we studied so far belong to the category called *stream ciphers*. The differences between block and stream ciphers are discussed at the end of this chapter.

In a Hill cipher, the key is a square matrix of size $m \times m$ in which m is the size of the block. If we call the key matrix **K**, each element of the matrix is $k_{i,j}$ as shown in Figure 3.15.

Figure 3.15 *Key in the Hill cipher*

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1m} \\ k_{21} & k_{22} & \cdots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \cdots & k_{mm} \end{bmatrix}$$

Let us show how one block of the ciphertext is encrypted. If we call the m characters in the plaintext block $P_1, P_2, \ldots, P_m$, the corresponding characters in the ciphertext block are $C_1, C_2, \ldots, C_m$. Then we have

$$
\begin{aligned}
C_1 &= P_1\, k_{11} + P_2\, k_{21} + \cdots + P_m\, k_{m1} \\
C_2 &= P_1\, k_{12} + P_2\, k_{22} + \cdots + P_m\, k_{m2} \\
&\cdots \\
C_m &= P_1\, k_{1m} + P_2\, k_{2m} + \cdots + P_m\, k_{mm}
\end{aligned}
$$

The equations show that each ciphertext character such as C_1 depends on all plaintext characters in the block ($P_1, P_2, \ldots, P_m$). However, we should be aware that not all square matrices have multiplicative inverses in $\mathbf{Z}_{26}$, so Alice and Bob should be careful in selecting the key. Bob will not be able to decrypt the ciphertext sent by Alice if the matrix does not have a multiplicative inverse.

The key matrix in the Hill cipher needs to have a multiplicative inverse.

Example 3.20

Using matrices allows Alice to encrypt the whole plaintext. In this case, the plaintext is an $l \times m$ matrix in which l is the number of blocks. For example, the plaintext "code is ready" can make a 3×4 matrix when adding extra bogus character "z" to the last block and removing the spaces. The ciphertext is "OHKNIHGKLISS". Bob can decrypt the message using the inverse of the key matrix. Encryption and decryption are shown in Figure 3.16.

Figure 3.16 *Example 3.20*

$$
\overset{\mathbf{C}}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} = \overset{\mathbf{P}}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} \overset{\mathbf{K}}{\begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix}}
$$

a. Encryption

$$
\overset{\mathbf{P}}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} = \overset{\mathbf{C}}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} \overset{\mathbf{K}^{-1}}{\begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix}}
$$

b. Decryption

Cryptanalysis of Hill Ciphers

Ciphertext-only cryptanalysis of Hill ciphers is difficult. First, a brute-force attack on a Hill cipher is extremely difficult because the key is an $m \times m$ matrix. Each entry in the matrix can have one of the 26 values. At first glance, this means that the size of the key domain is $26^{m \times m}$. However, not all of the matrices have multiplicative inverses. The key domain is smaller, but still huge.

Second, Hill ciphers do not preserve the statistics of the plaintext. Eve cannot run frequency analysis on single letters, digrams, or trigrams. A frequency analysis of words of size m might work, but this is very rare that a plaintext has many strings of size m that are the same.

Eve, however, can do a known-plaintext attack on the cipher if she knows the value of m and knows the plaintext/ciphertext pairs for at least m blocks. The blocks can belong to the same message or different messages but should be distinct. Eve can create two $m \times m$ matrices, **P** (plaintext) and **C** (ciphertext) in which the corresponding rows represent the corresponding known plaintext/ciphertext pairs. Because $\mathbf{C} = \mathbf{PK}$, Eve can use the relationship $\mathbf{K} = \mathbf{CP}^{-1}$ to find the key if **P** is invertible. If **P** is not invertible, then Eve needs to use a different set of m plaintext/ciphertext pairs.

If Eve does not know the value of m, she can try different values provided that m is not very large.

Example 3.21

Assume that Eve knows that $m = 3$. She has intercepted three plaintext/ciphertext pair blocks (not necessarily from the same message) as shown in Figure 3.17.

Figure 3.17 *Example 3.22, forming the ciphertext cipher*

$$
\begin{array}{ccc}
\begin{bmatrix} 05 & 07 & 10 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 03 & 06 & 00 \end{bmatrix} \\
\begin{bmatrix} 13 & 17 & 07 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 14 & 16 & 09 \end{bmatrix} \\
\begin{bmatrix} 00 & 05 & 04 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 03 & 17 & 11 \end{bmatrix} \\
\mathbf{P} & & \mathbf{C}
\end{array}
$$

She makes matrices **P** and **C** from these pairs. Because P is invertible, she inverts the P matrix and multiplies it by C to get the K matrix as shown in Figure 3.18.

Figure 3.18 *Example 3.21, finding the key*

$$
\underset{\mathbf{K}}{\begin{bmatrix} 02 & 03 & 07 \\ 05 & 07 & 09 \\ 01 & 02 & 11 \end{bmatrix}} = \underset{\mathbf{P}^{-1}}{\begin{bmatrix} 21 & 14 & 01 \\ 00 & 08 & 25 \\ 13 & 03 & 08 \end{bmatrix}} \underset{\mathbf{C}}{\begin{bmatrix} 03 & 06 & 00 \\ 14 & 16 & 09 \\ 03 & 17 & 11 \end{bmatrix}}
$$

Now she has the key and can break any ciphertext encrypted with that key.

One-Time Pad

One of the goals of cryptography is perfect secrecy. A study by Shannon has shown that perfect secrecy can be achieved if each plaintext symbol is encrypted with a key randomly chosen from a key domain. For example, an additive cipher can be easily broken because the same key is used to encrypt every character. However, even this simple cipher can become a perfect cipher if the key that is used to encrypt each character is chosen randomly from the key domain (00, 01, 02, …, 25)—that is, if the first character is encrypted using the key 04, the second character is encrypted using the key 02, the third character is encrypted using the key 21; and so on. Ciphertext-only attack is impossible. Other types of attacks are also impossible if the sender changes the key each time she sends a message, using another random sequence of integers.

This idea is used in a cipher called **one-time pad,** invented by Vernam. In this cipher, the key has the same length as the plaintext and is chosen completely in random.

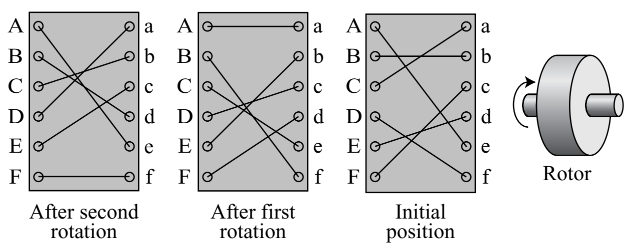
A one-time pad is a perfect cipher, but it is almost impossible to implement commercially. If the key must be newly generated each time, how can Alice tell Bob the new key each time she has a message to send? However, there are some occasions when a one-time pad can be used. For example, if the president of a country needs to send a completely secret message to the president of another country, she can send a trusted envoy with the random key before sending the message.

Some variations of the one-time pad cipher will be discussed in later chapters when modern use of cryptography is introduced.

Rotor Cipher

Although one-time pad ciphers are not practical, one step toward more secured encipherment is the **rotor cipher.** It uses the idea behind monoalphabetic substitution but changes the mapping between the plaintext and the ciphertext characters for each plaintext character. Figure 3.19 shows a simplified example of a rotor cipher.

Figure 3.19 *A rotor cipher*



The rotor shown in Figure 3.19 uses only 6 letters, but the actual rotors use 26 letters. The rotor is permanently wired, but the connection to encryption/decryption characters is provided by brushes. Note that the wiring is shown as though the rotor were transparent and one could see the inside.

The initial setting (position) of the rotor is the secret key between Alice and Bob. The first plaintext character is encrypted using the initial setting; the second character is encrypted after the first rotation (in Figure 3.19 at 1/6 turn, but the actual setting is 1/26 turn); and so on.

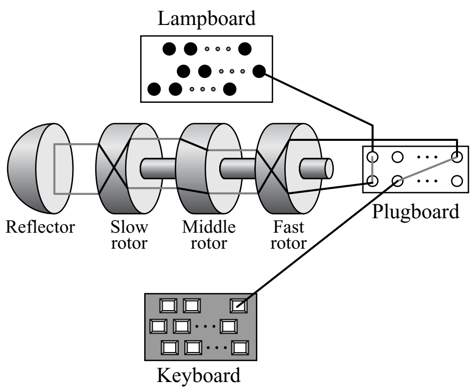
A three-letter word such as “bee” is encrypted as “BAA” if the rotor is stationary (the monoalphabetic substitution cipher), but it will encrypted as “BCA” if it is rotating (the rotor cipher). This shows that the rotor cipher is a polyalphabetic cipher because two occurrences of the same plaintext character are encrypted as different characters.

The rotor cipher is as resistant to a brute-force attack as the monoalphabetic substitution cipher because Eve still needs to find the first set of mappings among 26! possible ones. The rotor cipher is much more resistant to statistical attack than the monoalphabetic substitution cipher because it does not preserve letter frequency.

Enigma Machine

The **Enigma machine** was originally invented by Sherbius, but was modified by the German army and extensively used during World War II. The machine was based on the principle of rotor ciphers. Figure 3.20 shows a simple schematic diagram of the machine.

Figure 3.20 *A schematic of the Enigma machine*



The following lists the main components of the machine:

1. A keyboard with 26 keys used for entering the plaintext when encrypting and for entering the ciphertext when decrypting.
2. A lampboard with 26 lamps that shows the ciphertext characters in encrypting and the plaintext characters in decrypting.
3. A plugboard with 26 plugs manually connected by 13 wires. The configuration is changed every day to provide different scrambling.
4. Three wired rotors as described in the previous section. The three rotors were chosen daily out of five available rotors. The fast rotor rotates 1/26 of a turn for each character entered on the keyboard. The middle rotor makes 1/26 turn for each complete turn of the fast rotor. The slow rotor makes 1/26 turn for each complete turn of the middle rotor.
5. A reflector, which is stationary and prewired.

Code Book

To use the Enigma machine, a code book was published that gives several settings for each day, including:

a. The three rotors to be chosen, out of the five available ones.
b. The order in which the rotors are to be installed.
c. The setting for the plugboard.
d. A three-letter code of the day.

Procedure for Encrypting a Message

To encrypt a message, the operator followed these steps:

1. Set the starting position of the rotors to the code of the day. For example, if the code was “HUA”, the rotors were initialized to “H”, “U”, and “A”, respectively.
2. Choose a random three-letter code, such as “ACF”. Encrypt the text “ACFACF” (repeated code) using the initial setting of rotors in step 1. For example, assume the encrypted code is “OPNABT”.
3. Set the starting positions of the rotors to OPN (half of the encrypted code).
4. Append the encrypted six letters obtained from step 2 (“OPNABT”) to the beginning of the message.
5. Encrypt the message including the 6-letter code. Send the encrypted message.

Procedure for Decrypting a Message

To decrypt a message, the operator followed these steps:

1. Receive the message and separate the first six letters.
2. Set the starting position of the rotors to the code of the day.
3. Decrypt the first six letters using the initial setting in step 2.
4. Set the positions of the rotors to the first half of the decrypted code.
5. Decrypt the message (without the first six letters).

Cryptanalysis

We know that the Enigma machine was broken during the war, although the German army and the rest of the world did not hear about this until a few decades later. The question is how such a complicated cipher was attacked. Although the German army tried to hide the internal wiring of the rotors, the Allies somehow obtained some copies of the machines. The next step was to find the setting for each day and the code sent to initialize the rotors for every message. The invention of the first computer helped the Allies to overcome these difficulties. The full picture of the machine and its cryptanalysis can be found at some of the Enigma Websites.

3.3 TRANSPOSITION CIPHERS

A **transposition cipher** does not substitute one symbol for another, instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext may appear in the first position of the ciphertext. In other words, a transposition cipher reorders (transposes) the symbols.

A transposition cipher reorders symbols.

Keyless Transposition Ciphers

Simple transposition ciphers, which were used in the past, are keyless. There are two methods for permutation of characters. In the first method, the text is written into a table column by column and then transmitted row by row. In the second method, the text is written into the table row by row and then transmitted column by column.

Example 3.22

A good example of a keyless cipher using the first method is the **rail fence cipher.** In this cipher, the plaintext is arranged in two lines as a zigzag pattern (which means column by column); the ciphertext is created reading the pattern row by row. For example, to send the message “Meet me at the park” to Bob, Alice writes

m		**e**		**m**		**a**		**t**		**e**		**a**		**k**
	e		**t**		**e**		**t**		**h**		**p**		**r**	

She then creates the ciphertext “MEMATEAKETETHPR” by sending the first row followed by the second row. Bob receives the ciphertext and divides it in half (in this case the second half has one less character). The first half forms the first row; the second half, the second row. Bob reads the result in zigzag. Because there is no key and the number of rows is fixed (2), the cryptanalysis of the ciphertext would be very easy for Eve. All she needs to know is that the rail fence cipher is used.

Example 3.23

Alice and Bob can agree on the number of columns and use the second method. Alice writes the same plaintext, row by row, in a table of four columns.

m	**e**	**e**	**t**
m	**e**	**a**	**t**
t	**h**	**e**	**p**
a	**r**	**k**	

She then creates the ciphertext “MMTAEEHREAEKTTP” by transmitting the characters column by column. Bob receives the ciphertext and follows the reverse process. He writes the received message, column by column, and reads it row by row as the plaintext. Eve can easily decipher the message if she knows the number of columns.

Example 3.24

The cipher in Example 3.23 is actually a transposition cipher. The following shows the permutation of each character in the plaintext into the ciphertext based on the positions.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
01	05	09	13	02	06	10	13	03	07	11	15	04	08	12

The second character in the plaintext has moved to the fifth position in the ciphertext; the third character has moved to the ninth position; and so on. Although the characters are permuted, there is a pattern in the permutation: (01, 05, 09, 13), (02, 06, 10, 13), (03, 07, 11, 15), and (08, 12). In each section, the difference between the two adjacent numbers is 4.

Keyed Transposition Ciphers

The keyless ciphers permute the characters by using writing plaintext in one way (row by row, for example) and reading it in another way (column by column, for example). The permutation is done on the whole plaintext to create the whole ciphertext. Another method is to divide the plaintext into groups of predetermined size, called blocks, and then use a key to permute the characters in each block separately.

Example 3.25

Alice needs to send the message "Enemy attacks tonight" to Bob. Alice and Bob have agreed to divide the text into groups of five characters and then permute the characters in each group. The following shows the grouping after adding a bogus character at the end to make the last group the same size as the others.

e n e m y	a t t a c	k s t o n	i g h t z

The key used for encryption and decryption is a permutation key, which shows how the character are permuted. For this message, assume that Alice and Bob used the following key:

Encryption ↓	3	1	4	5	2	↑ Decryption
	1	2	3	4	5	

The third character in the plaintext block becomes the first character in the ciphertext block; the first character in the plaintext block becomes the second character in the ciphertext block; and so on. The permutation yields

E E M Y N	**T A A C T**	**T K O N S**	**H I T Z G**

Alice sends the ciphertext "EEMYNTAACTTKONSHITZG" to Bob. Bob divides the ciphertext into 5-character groups and, using the key in the reverse order, finds the plaintext.

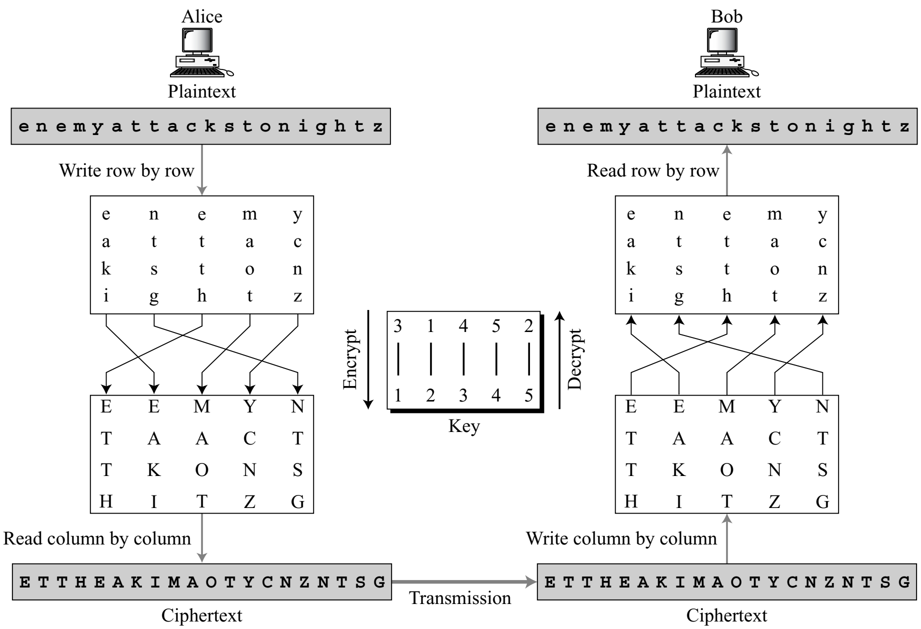
Combining Two Approaches

More recent transposition ciphers combine the two approaches to achieve better scrambling. Encryption or decryption is done in three steps. First, the text is written into a table row by row. Second, the permutation is done by reordering the columns. Third, the new table is read column by column. The first and third steps provide a keyless global reordering; the second step provides a blockwise keyed reordering. These types of ciphers are often referred to as keyed columnar transposition ciphers or just columnar transposition ciphers.

Example 3.26

Suppose Alice again enciphers the message in Example 3.25, this time using the combined approach. The encryption and decryption is shown in Figure 3.21.

Figure 3.21 *Example 3.27*

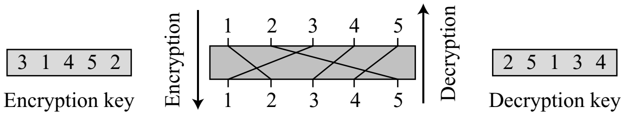


The first table is created by Alice writing the plaintext row by row. The columns are permuted using the same key as in the previous example. The ciphertext is created by reading the second table column by column. Bob does the same three steps in the reverse order. He writes the ciphertext column by column into the first table, permutes the columns, and then reads the second table row by row.

Keys

In Example 3.27, a single key was used in two directions for the column exchange: downward for encryption, upward for decryption. It is customary to create two keys from this graphical representation: one for encryption and one for direction. The keys are stored in tables with one entry for each column. The entry shows the source column number; the destination column number is understood from the position of the entry. Figure 3.22 shows how the two tables can be made from the graphical representation of the key.

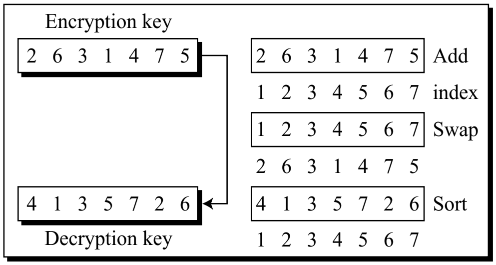
Figure 3.22 *Encryption/decryption keys in transpositional ciphers*



The encryption key is (3 1 4 5 2). The first entry shows that column 3 (contents) in the source becomes column 1 (position or index of the entry) in the destination. The decryption key is (2 5 1 3 4). The first entry shows that column 2 in the source becomes column 1 in the destination.

How can the decryption key be created if the encryption key is given, or vice versa? The process can be done manually in a few steps, as shown in Figure 3.23. First add indices to the key table, then swap the contents and indices, finally sort the pairs according to the index.

Figure 3.23 *Key inversion in a transposition cipher*



a. Manual process

```
Given: EncKey [index]
index ← 1
while (index ≤ Column)
 {
  DecKey[EncKey[index]] ← index
  index ← index + 1
 }
Return : DecKey [index]
```

b. Algorithm

Using Matrices

We can use matrices to show the encryption/decryption process for a transposition cipher. The plaintext and ciphertext are $l \times m$ matrices representing the numerical values of the characters; the keys are square matrices of size $m \times m$. In a permutation matrix, every row or column has exactly one 1 and the rest of the values are 0s. Encryption is performed by multiplying the plaintext matrix by the key matrix to get the ciphertext matrix; decryption is performed by multiplying the ciphertext by the inverse key matrix to get the plaintext matrix. A very interesting point is that the decryption matrix in this case is the inverse of the encryption matrix. However, there is no need to invert the matrix, the encryption key matrix can simply be transposed (swapping the rows and columns) to get the decryption key matrix.

Example 3.27

Figure 3.24 shows the encryption process. Multiplying the 4×5 plaintext matrix by the 5×5 encryption key gives the 4×5 ciphertext matrix. Matrix manipulation requires changing the characters in Example 3.27 to their numerical values (from 00 to 25). Note that the matrix multiplication provides only the column permutation of the transposition; reading and writing into the matrix should be provided by the rest of the algorithm.

Figure 3.24 *Representation of the key as a matrix in the transposition cipher*

$$\underset{\text{Plaintext}}{\begin{bmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{bmatrix}} \cdot \underset{\text{Encryption key}}{\begin{array}{c} \begin{array}{ccccc} \boxed{3} & \boxed{1} & \boxed{4} & \boxed{5} & \boxed{2} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}} = \underset{\text{Ciphertext}}{\begin{bmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{bmatrix}}$$

Cryptanalysis of Transposition Ciphers

Transposition ciphers are vulnerable to several kinds of ciphertext-only attacks.

Statistical Attack

A transposition cipher does not change the frequency of letters in the ciphertext; it only reorders the letters. So the first attack that can be applied is single-letter frequency analysis. This method can be useful if the length of the ciphertext is long enough. We have seen this attack before. However, transposition ciphers do not preserve the frequency of digrams and trigrams. This means that Eve cannot use these tools. In fact, if a cipher does not preserve the frequency of digrams and trigrams, but does preserve the frequency of single letters, it is probable that the cipher is a transposition cipher.

Brute-Force Attack

Eve can try all possible keys to decrypt the message. However, the number of keys can be huge $(1! + 2! + 3! + \cdots + L!)$, where L is the length of the ciphertext. A better approach is to guess the number of columns. Eve knows that the number of columns divides L. For example, if the length of the cipher is 20 characters, then $20 = 1 \times 2 \times 2 \times 5$.

This means the number of columns can be a combination of these factors (1, 2, 4, 5, 10, 20). However, the first (only one column) is out of the question and the last (only one row) is unlikely.

Example 3.28

Suppose that Eve has intercepted the ciphertext message "EEMYNTAACTTKONSHITZG". The message length $L = 20$ means the number of columns can be 1, 2, 4, 5, 10, or 20. Eve ignores the first value because it means only one column and no permutation.

a. If the number of columns is 2, the only two permutations are (1, 2) and (2, 1). The first one means there would be no permutation. Eve tries the second one. Eve divides the ciphertext into two-character units: "EE MY NT AA CT TK ON SH IT ZG". She then tries to permute each of these getting "ee ym nt aa tc kt no hs ti gz", which does not make sense.
b. If the number of columns is 4, there are 4! = 24 permutations. The first one (1 2 3 4) means there would be no permutation. Eve needs to try the rest. After trying all 23 possibilities, Eve finds no plaintext that makes sense.
c. If the number of columns is 5, there are 5! = 120 permutations. The first one (1 2 3 4 5) means there would be no permutation. Eve needs to try the rest. The permutation (2 5 1 3 4) yields a plaintext "enemyattackstonightz" that makes sense after removing the bogus letter z and adding spaces.

Pattern Attack

Another attack on the transposition cipher can be called pattern attack. The ciphertext created from a keyed transposition cipher has some repeated patterns. The following show where each character in the ciphertext in Example 3.28 comes from.

03	**08**	**13**	**18**	*01*	*06*	*11*	*16*	**04**	**09**	**14**	**19**	*05*	*10*	*15*	*20*	**02**	**07**	**12**	**17**

The 1st character in the ciphertext comes from the 3rd character in the plaintext. The 2nd character in the ciphertext comes from the 8th character in the plaintext. The 20th character in the ciphertext comes from the 17th character in the plaintext, and so on. There is a pattern in the above list. We have five groups: (3, 8, 13, 18), (1, 6, 11, 16), (4, 9, 14, 19), (5, 10, 15, 20), and (2, 7, 12, 17). In all groups, the difference between the two adjacent numbers is 5. This regularity can be used by the cryptanalyst to break the cipher. If Eve knows or can guess the number of columns (which is 5 in this case), she can organize the ciphertext in groups of four characters. Permuting the groups can provide the clue to finding the plaintext.

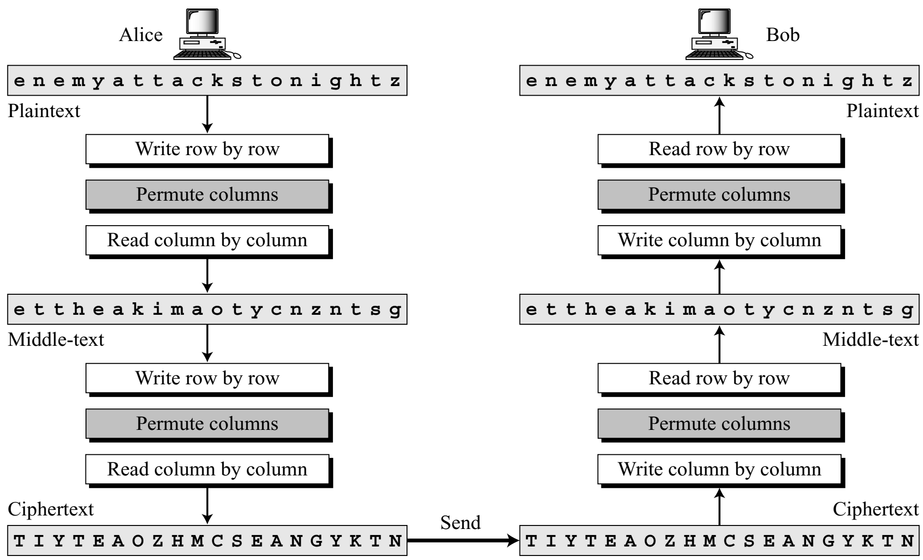
Double Transposition Ciphers

Double transposition ciphers can make the job of the cryptanalyst difficult. An example of such a cipher would be the one that repeats twice the algorithm used for encryption and decryption in Example 3.26. A different key can be used in each step, but normally the same key is used.

Example 3.29

Let us repeat Example 3.26 using double transposition. Figure 3.25 shows the process.

Figure 3.25 *Double transposition cipher*



Although, the cryptanalyst can still use the single-letter frequency attack on the ciphertext, a pattern attack is now much more difficult. The pattern analysis of the text shows

13	16	05	07	03	06	10	20	18	04	10	12	01	09	15	17	08	11	19	02

Comparing the above set with the result in Example 3.28, we see that there is no repetitive pattern. Double transposition removes the regularities we have seen before.

3.4 STREAM AND BLOCK CIPHERS

The literature divides the symmetric ciphers into two broad categories: stream ciphers and block ciphers. Although the definitions are normally applied to modern ciphers, this categorization also applies to traditional ciphers.

Stream Ciphers

In a **stream cipher,** encryption and decryption are done one symbol (such as a character or a bit) at a time. We have a plaintext stream, a ciphertext stream, and a key stream. Call the plaintext stream P, the ciphertext stream C, and the key stream K.

$$P = P_1P_2P_3, \ldots \qquad C = C_1C_2C_3, \ldots \qquad K = (k_1, k_2, k_3, \ldots)$$

$$C_1 = E_{k1}(P_1) \qquad C_2 = E_{k2}(P_2) \qquad C_3 = E_{k3}(P_3) \ldots$$

Figure 3.26 shows the idea behind a stream cipher. Characters in the plaintext are fed into the encryption algorithm, one at a time; the ciphertext characters are also created one at a time. The key stream, can be created in many ways. It may be a stream of predetermined values; it may be created one value at a time using an algorithm. The values may depend on the plaintext or ciphertext characters. The values may also depend on the previous key values.

Figure 3.26 *Stream cipher*

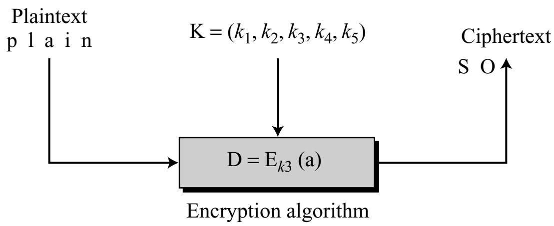


Figure 3.26 shows the moment where the third character in the plaintext stream is being encrypted using the third value in the key stream. The result creates the third character in the ciphertext stream.

Example 3.30

Additive ciphers can be categorized as stream ciphers in which the key stream is the repeated value of the key. In other words, the key stream is considered as a predetermined stream of keys or K = (k, k, …, k). In this cipher, however, each character in the ciphertext depends only on the corresponding character in the plaintext, because the key stream is generated independently.

Example 3.31

The monoalphabetic substitution ciphers discussed in this chapter are also stream ciphers. However, each value of the key stream in this case is the mapping of the current plaintext character to the corresponding ciphertext character in the mapping table.

Example 3.32

Vigenere ciphers are also stream ciphers according to the definition. In this case, the key stream is a repetition of m values, where m is the size of the keyword. In other words,

$$K = (k_1, k_2, \ldots k_m, k_1, k_2, \ldots k_m, \ldots)$$

Example 3.33

We can establish a criterion to divide stream ciphers based on their key streams. We can say that a stream cipher is a monoalphabetic cipher if the value of k_i does not depend on the position of the plaintext character in the plaintext stream; otherwise, the cipher is polyalphabetic.

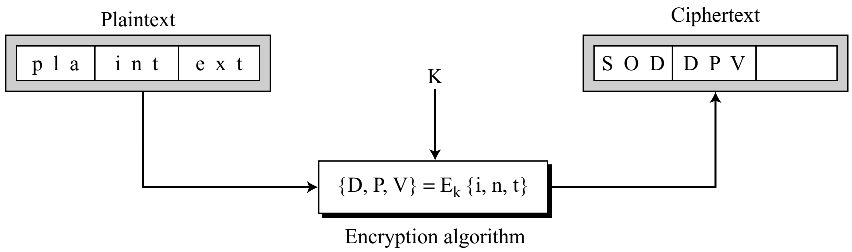
- Additive ciphers are definitely monoalphabetic because k_i in the key stream is fixed; it does not depend on the position of the character in the plaintext.
- Monoalphabetic substitution ciphers are definitely *monoalphabetic* because k_i does not depend on the position of the corresponding character in the plaintext stream; it depends only on the value of the plaintext character.

- Vigenere ciphers are polyalphabetic ciphers because k_i definitely depends on the position of the plaintext character. However, the dependency is cyclic. The key is the same for two characters m positions apart.

Block Ciphers

In a **block cipher,** a group of plaintext symbols of size m ($m > 1$) are encrypted together creating a group of ciphertext of the same size. Based on the definition, in a block cipher, a single key is used to encrypt the whole block even if the key is made of multiple values. Figure 3.27 shows the concept of a block cipher.

Figure 3.27 *Block cipher*



In a block cipher, a ciphertext block depends on the whole plaintext block.

Example 3.34

Playfair ciphers are block ciphers. The size of the block is $m = 2$. Two characters are encrypted together.

Example 3.35

Hill ciphers are block ciphers. A block of plaintext, of size 2 or more is encrypted together using a single key (a matrix). In these ciphers, the value of each character in the ciphertext depends on all the values of the characters in the plaintext. Although the key is made of $m \times m$ values, it is considered as a single key.

Example 3.36

From the definition of the block cipher, it is clear that every block cipher is a polyalphabetic cipher because each character in a ciphertext block depends on all characters in the plaintext block.

Combination

In practice, blocks of plaintext are encrypted individually, but they use a stream of keys to encrypt the whole message block by block. In other words, the cipher is a block cipher when looking at the individual blocks, but it is a stream cipher when looking at the whole message considering each block as a single unit. Each block uses a different key that may be generated before or during the encryption process. Examples of this will appear in later chapters.

3.5 RECOMMENDED READING

The following books and websites give more details about subjects discussed in this chapter. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books discuss classic symmetric-key ciphers. [Kah96] and [Sin99] give a thorough history of these ciphers. [Sti06], [Bar02], [TW06], [Cou99], [Sta06], [Sch01], [Mao03], and [Gar01] provide good accounts of the technical details.

WebSites

The following websites give more information about topics discussed in this chapter.

http://www.cryptogram.org
http://www.cdt.org/crypto/
http://www.cacr.math.uwaterloo.ca/
http://www.acc.stevens.edu/crypto.php
http://www.crypto.com/
http://theory.lcs.mit.edu/~rivest/crypto-security.html
http://www.trincoll.edu/depts/cpsc/cryptography/substitution.html
http://hem.passagen.se/tan01/transpo.html
http://www.strangehorizons.com/2001/20011008/steganography.shtml

3.6 KEY TERMS

additive cipher
affine cipher
autokey cipher
block cipher
brute-force attack
Caesar cipher
chosen-ciphertext attack
chosen-plaintext attack
cipher
ciphertext
ciphertext-only attack
cryptanalysis
decryption algorithm
digram
double transposition cipher
encryption algorithm
Enigma machine
exhaustive-key-search method
Hill cipher
Kasiski test
Kerckhoff’s principle
key
key domain
known-plaintext attack