Third Edition

# Introduction to CRYPTOGRAPHY with Coding Theory

Wade Trappe

Lawrence C. Washington

# Introduction to Cryptography

with Coding Theory

3rd edition

Wade Trappe

Wireless Information Network Laboratory and the
Electrical and Computer Engineering Department
Rutgers University

Lawrence C. Washington

Department of Mathematics University of Maryland

**Portfolio Manager:** *Chelsea Kharakozoua*

**Content Manager:** *Jeff Weidenaar*

**Content Associate:** *Jonathan Krebs*

**Content Producer:** *Tara Corpuz*

**Managing Producer:** *Scott Disanno*

**Producer:** *Jean Choe*

**Manager, Courseware QA:** *Mary Durnwald*

**Product Marketing Manager:** *Stacey Sveum*

**Product and Solution Specialist:** *Rosemary Morten*

**Senior Author Support/Technology Specialist:** *Joe Vetere*

**Manager, Rights and Permissions:** *Gina Cheselka*

**Text and Cover Design, Production Coordination, Composition, and Illustrations:** *Integra Software Services Pvt. Ltd*

**Manufacturing Buyer:** *Carol Melville, LSC Communications*

**Cover Image:** *Photographer is my life/Getty Images*

**Library of Congress Cataloging-in-Publication Data**

Names: Trappe, Wade, author. | Washington, Lawrence C., author.

Title: Introduction to cryptography : with coding theory / Wade Trappe, Lawrence Washington.

Description: 3rd edition. | [Hoboken, New Jersey] : [Pearson Education], [2020] | Includes bibliographical references and index. | Summary: "This book is based on a course in cryptography at the upper-level undergraduate and beginning graduate level that has been given at the University of Maryland since 1997, and a course that has been taught at Rutgers University since 2003"— Provided by publisher.

Identifiers: LCCN 2019029691 | ISBN 9780134860992 (paperback)

Subjects: LCSH: Coding theory. | Cryptography.

Classification: LCC QA268.T73 2020 | DDC 005.8/24—dc23

LC record available at https://lccn.loc.gov/2019029691

ScoutAutomatedPrintCode

ISBN-13: 978-0-13-485906-4

ISBN-10: 0-13-485906-5

Pearson

# Contents

# Preface

This book is based on a course in cryptography at the upper-level undergraduate and beginning graduate level that has been given at the University of Maryland since 1997, and a course that has been taught at Rutgers University since 2003. When designing the courses, we decided on the following requirements:

- The courses should be up-to-date and cover a broad selection of topics from a mathematical point of view.
- The material should be accessible to mathematically mature students having little background in number theory and computer programming.
- There should be examples involving numbers large enough to demonstrate how the algorithms really work.

We wanted to avoid concentrating solely on RSA and discrete logarithms, which would have made the courses mostly about number theory. We also did not want to focus on protocols and how to hack into friends' computers. That would have made the courses less mathematical than desired.

There are numerous topics in cryptology that can be discussed in an introductory course. We have tried to include many of them. The chapters represent, for the most part, topics that were covered during the different semesters we taught the course. There is certainly more material here than could be treated in most one-semester courses. The first thirteen chapters represent the core of the material. The choice of which of the remaining chapters are used depends on the level of the students and the objectives of the lecturer.

The chapters are numbered, thus giving them an ordering. However, except for Chapter 3 on number theory, which pervades the subject, the chapters are fairly independent of each other and can be covered in almost any reasonable order. Since students have varied backgrounds in number theory, we have collected the basic number theory facts together in Chapter 3 for ease of reference; however, we recommend introducing these concepts gradually throughout the course as they are needed.

The chapters on information theory, elliptic curves, quantum cryptography, lattice methods, and error correcting codes are somewhat more mathematical than the others. The chapter on error correcting codes was included, at the suggestion of several reviewers, because courses that include introductions to both cryptology and coding theory are fairly common.

## Computer Examples

Suppose you want to give an example for RSA. You could choose two one-digit primes and pretend to be working with fifty-digit primes, or you could use your favorite software package to do an actual example with large primes. Or perhaps you are working with shift ciphers and are trying to decrypt a message by trying all 26 shifts of the ciphertext. This should also be done on a computer.

Additionally, at the end of the book are appendices containing computer examples written in each of Mathematica®, Maple®, MATLAB®, and Sage that show how to do such calculations. These languages were chosen because they are user friendly and do not require prior programming experience. Although the course has been taught successfully without computers, these examples are an integral part of the book and should be studied, if at all possible. Not only do they contain numerical examples of how to do certain computations but also they demonstrate important ideas and issues that arise. They were placed at the end of the book because of the logistic and aesthetic problems of including extensive computer examples in these languages at the ends of chapters.

Additionally, programs available in Mathematica, Maple, and MATLAB can be downloaded from the Web site (`bit.ly/2JbcS6p`). Homework problems (the computer problems in various chapters) based on the software allow students to play with examples individually. Of course, students having more programming background could write their own programs instead. In a classroom, all that is needed is a computer (with one of the languages installed) and a projector in order to produce meaningful examples as the lecture is being given.

## New to the Third Edition

Two major changes have informed this edition: Changes to the field of cryptography and a change in the format of the text. We address these issues separately, although there is an interplay between the two:

## Content Changes

Cryptography is a quickly changing field. We have made many changes to the text since the last edition:

- Reorganized content previously in two chapters to four separate chapters on Stream Ciphers (including RC4), Block Ciphers, DES and AES (Chapters 5–8, respectively). The RC4 material, in particular, is new.
- Heavily revised the chapters on hash functions. Chapter 11 (Hash functions) now includes sections on SHA-2 and SHA-3. Chapter 12 (Hash functions: Attacks and Applications) now includes material on message authentication codes, password protocols, and blockchains.

- The short section on the one-time pad has been expanded to become Chapter 4, which includes sections on multiple use of the one-time pad, perfect secrecy, and ciphertext indistinguishability.
- Added Chapter 14, “What Can Go Wrong,” which shows what can happen when cryptographic algorithms are used or designed incorrectly.
- Expanded Chapter 16 on digital cash to include Bitcoin and cryptocurrencies.
- Added Chapter 22, which gives an introduction to Pairing-Based Cryptography.
- Updated the exposition throughout the book to reflect recent developments.
- Added references to the Maple, Mathematica, MATLAB, and Sage appendices in relevant locations in the text.
- Added many new exercises.
- Added a section at the back of the book that contains answers or hints to a majority of the odd-numbered problems.

## Format Changes

A focus of this revision was transforming the text from a print-based learning tool to a digital learning tool. The eText is therefore filled with content and tools that will help bring the content of the course to life for students in new ways and help improve instruction. Specifically, the following are features that are available only in the eText:

- Interactive Examples. We have added a number of opportunities for students to interact with content in a dynamic manner in order to build or enhance understanding. Interactive examples allow students to explore concepts in ways that are not possible without technology.
- Quick Questions. These questions, built into the narrative, provide opportunities for students to check and clarify understanding. Some help address potential misconceptions.
- Notes, Labels, and Highlights. Notes can be added to the eText by instructors. These notes are visible to all students in the course, allowing instructors to add their personal observations or directions to important topics, call out need-to-know information, or clarify difficult concepts. Students can add their own notes,

labels, and highlights to the eText, helping them focus on what they need to study. The customizable Notebook allows students to filter, arrange, and group their notes in a way that makes sense to them.

- Dashboard. Instructors can create reading assignments and see the time spent in the eText so that they can plan more effective instruction.
- Portability. Portable access lets students read their eText whenever they have a moment in their day, on Android and iOS mobile phones and tablets. Even without an Internet connection, offline reading ensures students never miss a chance to learn.
- Ease-of-Use. Straightforward setup makes it easy for instructors to get their class up and reading quickly on the first day of class. In addition, Learning Management System (LMS) integration provides institutions, instructors, and students with single sign-on access to the eText via many popular LMSs.
- Supplements. An Instructors' Solutions Manual can be downloaded by qualified instructors from the textbook's webpage at `www.pearson.com`.

## Acknowledgments

Many people helped and provided encouragement during the preparation of this book. First, we would like to thank our students, whose enthusiasm, insights, and suggestions contributed greatly. We are especially grateful to many people who have provided corrections and other input, especially Bill Gasarch, Jeff Adams, Jonathan Rosenberg, and Tim Strobell. We would like to thank Wenyuan Xu, Qing Li, and Pandurang Kamat, who drew several of the diagrams and provided feedback on the new material for the second edition. We have enjoyed working with the staff at Pearson, especially Jeff Weidenaar and Tara Corpuz.

The reviewers deserve special thanks: their suggestions on the exposition and the organization of the topics greatly enhanced the final result. The reviewers marked with an asterisk (*) provided input for this edition.

- * Anurag Agarwal, Rochester Institute of Technology

- * Pradeep Atrey, University at Albany

  Eric Bach, University of Wisconsin

  James W. Brewer, Florida Atlantic University

  Thomas P. Cahill, NYU

  Agnes Chan, Northeastern University

- * Nathan Chenette, Rose-Hulman Institute of Technology
- * Claude Crépeau, McGill University
- * Reza Curtmola, New Jersey Institute of Technology
- * Ahmed Desoky, University of Louisville

  Anthony Ephremides, University of Maryland, College Park

- * David J. Fawcett, Lawrence Tech University
- * Jason Gibson, Eastern Kentucky University
- * K. Gopalakrishnan, East Carolina University

  David Grant, University of Colorado, Boulder

  Jugal K. Kalita, University of Colorado, Colorado Springs

- * Saroja Kanchi, Kettering University
- * Andrew Klapper, University of Kentucky
- * Amanda Knecht, Villanova University

  Edmund Lamagna, University of Rhode Island

- * Aihua Li, Montclair State University
- * Spyros S. Magliveras, Florida Atlantic University
- * Nathan McNew, Towson University
- * Nick Novotny, IUPUI

  David M. Pozar, University of Massachusetts, Amherst

- * Emma Previato, Boston University
- * Hamzeh Roumani, York University
- * Bonnie Saunders, University of Illinois, Chicago
- * Ravi Shankar, University of Oklahoma
- * Ernie Stitzinger, North Carolina State

- * Armin Straub, University of South Alabama

  J. Felipe Voloch, University of Texas, Austin

  Daniel F. Warren, Naval Postgraduate School

- * Simon Whitehouse, Alfred State College

  Siman Wong, University of Massachusetts, Amherst

- * Huapeng Wu, University of Windsor

Wade thanks Nisha Gilra, who provided encouragement and advice; Sheilagh O’Hare for introducing him to the field of cryptography; and K. J. Ray Liu for his support. Larry thanks Susan Zengerle and Patrick Washington for their patience, help, and encouragement during the writing of this book.

Of course, we welcome suggestions and corrections. An errata page can be found at (`bit.ly/2J8nN0w`) or at the link on the book’s general Web site (`bit.ly/2T544yu`).

Wade Trappe

trappe@winlab.rutgers.edu

Lawrence C. Washington

lcw@math.umd.edu

# Chapter 1 Overview of Cryptography and Its Applications

People have always had a fascination with keeping information away from others. As children, many of us had magic decoder rings for exchanging coded messages with our friends and possibly keeping secrets from parents, siblings, or teachers. History is filled with examples where people tried to keep information secret from adversaries. Kings and generals communicated with their troops using basic cryptographic methods to prevent the enemy from learning sensitive military information. In fact, Julius Caesar reportedly used a simple cipher, which has been named after him.

As society has evolved, the need for more sophisticated methods of protecting data has increased. Now, with the information era at hand, the need is more pronounced than ever. As the world becomes more connected, the demand for information and electronic services is growing, and with the increased demand comes increased dependency on electronic systems. Already the exchange of sensitive information, such as credit card numbers, over the Internet is common practice. Protecting data and electronic systems is crucial to our way of living.

The techniques needed to protect data belong to the field of cryptography. Actually, the subject has three names, **cryptography**, **cryptology**, and **cryptanalysis**, which are often used interchangeably. Technically, however, cryptology is the all-inclusive term for the study of communication over nonsecure channels, and related problems. The process of designing systems to do this is called cryptography. Cryptanalysis deals with breaking such systems. Of course, it is essentially impossible to do either cryptography or cryptanalysis without having a good understanding of the methods of both areas.

Often the term **coding theory** is used to describe cryptography; however, this can lead to confusion. Coding theory deals with representing input information symbols by output symbols called code symbols. There are three basic applications that coding theory covers: compression, secrecy, and error correction. Over the past few decades, the term coding theory has become associated predominantly with error correcting codes. Coding theory thus studies communication over noisy channels and how to ensure that the message received is the correct message, as opposed to cryptography, which protects communication over nonsecure channels.

Although error correcting codes are only a secondary focus of this book, we should emphasize that, in any real-world system, error correcting codes are used in conjunction with encryption, since the change of a single bit is enough to destroy the message completely in a well-designed cryptosystem.

Modern cryptography is a field that draws heavily upon mathematics, computer science, and cleverness. This book provides an introduction to the mathematics and protocols needed to make data transmission and electronic systems secure, along with techniques such as electronic signatures and secret sharing.

## 1.1 Secure Communications

In the basic communication scenario, depicted in Figure 1.1, there are two parties, we'll call them Alice and Bob, who want to communicate with each other. A third party, Eve, is a potential eavesdropper.

Figure 1.1 The Basic Communication Scenario for Cryptography.

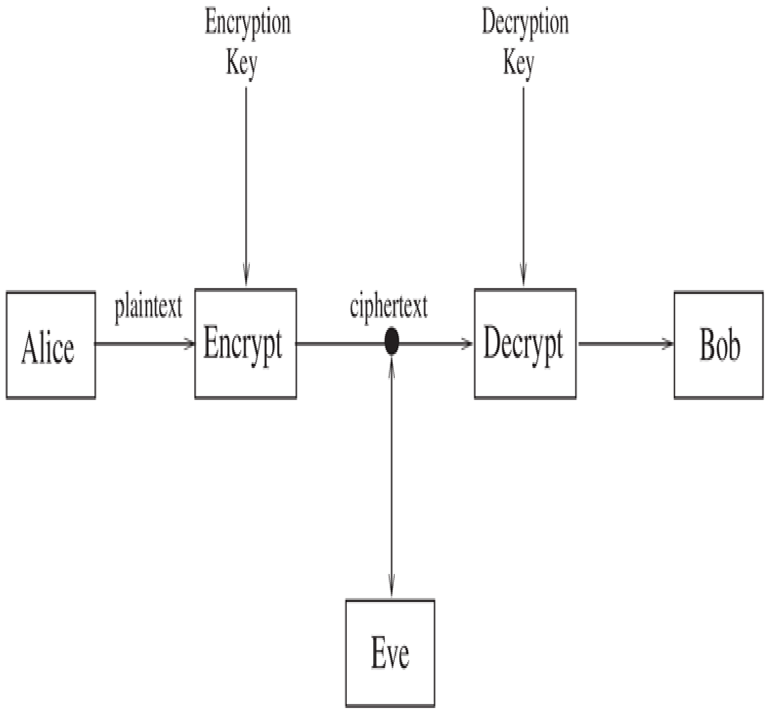

Figure 1.1 Full Alternative Text

When Alice wants to send a message, called the **plaintext**, to Bob, she encrypts it using a method prearranged with Bob. Usually, the encryption method is assumed to be known to Eve; what keeps the message secret is a **key**. When Bob receives the encrypted message, called the **ciphertext**, he changes it back to the plaintext using a decryption key.

Eve could have one of the following goals:

1. Read the message.
2. Find the key and thus read all messages encrypted with that key.
3. Corrupt Alice's message into another message in such a way that Bob will think Alice sent the altered message.
4. Masquerade as Alice, and thus communicate with Bob even though Bob believes he is communicating with Alice.

Which case we're in depends on how evil Eve is. Cases (3) and (4) relate to issues of integrity and authentication, respectively. We'll discuss these shortly. A more active and malicious adversary, corresponding to cases (3) and (4), is sometimes called Mallory in the literature. More passive observers (as in cases (1) and (2)) are sometimes named Oscar. We'll generally use only Eve, and assume she is as bad as the situation allows.

### 1.1.1 Possible Attacks

There are four main types of attack that Eve might be able to use. The differences among these types of attacks are the amounts of information Eve has available to her when trying to determine the key. The four attacks are as follows:

1. Ciphertext only: Eve has only a copy of the ciphertext.
2. Known plaintext: Eve has a copy of a ciphertext and the corresponding plaintext. For example, suppose Eve intercepts an encrypted press release, then sees the decrypted release the next day. If she can deduce the decryption key, and if Alice doesn't change the key, Eve can read all future messages. Or, if Alice always starts her messages with "Dear Bob," then Eve has a small piece of ciphertext and corresponding plaintext. For many weak cryptosystems, this suffices to find the key. Even for stronger

systems such as the German Enigma machine used in World War II, this amount of information has been useful.

3. Chosen plaintext: Eve gains temporary access to the encryption machine. She cannot open it to find the key; however, she can encrypt a large number of suitably chosen plaintexts and try to use the resulting ciphertexts to deduce the key.

4. Chosen ciphertext: Eve obtains temporary access to the decryption machine, uses it to “decrypt” several strings of symbols, and tries to use the results to deduce the key.

A chosen plaintext attack could happen as follows. You want to identify an airplane as friend or foe. Send a random message to the plane, which encrypts the message automatically and sends it back. Only a friendly airplane is assumed to have the correct key. Compare the message from the plane with the correctly encrypted message. If they match, the plane is friendly. If not, it’s the enemy. However, the enemy can send a large number of chosen messages to one of your planes and look at the resulting ciphertexts. If this allows them to deduce the key, the enemy can equip their planes so they can masquerade as friendly.

An example of a known plaintext attack reportedly happened in World War II in the Sahara Desert. An isolated German outpost every day sent an identical message saying that there was nothing new to report, but of course it was encrypted with the key being used that day. So each day the Allies had a plaintext-ciphertext pair that was extremely useful in determining the key. In fact, during the Sahara campaign, General Montgomery was carefully directed around the outpost so that the transmissions would not be stopped.

One of the most important assumptions in modern cryptography is **Kerckhoffs’s principle:** In assessing the security of a cryptosystem, one should always assume the enemy knows the method being used. This principle was enunciated by Auguste Kerckhoffs in 1883 in his classic treatise *La Cryptographie Militaire*. The enemy can obtain this information in many ways. For example, encryption/decryption machines can be captured and analyzed. Or people can defect or be captured. The security of the system should therefore be based on the key and not on the obscurity of the algorithm used. Consequently, we always assume that Eve has knowledge of the algorithm that is used to perform encryption.

### 1.1.2 Symmetric and Public Key Algorithms

Encryption/decryption methods fall into two categories: **symmetric key** and **public key**. In symmetric key algorithms, the encryption and decryption keys are known to both Alice and Bob. For example, the encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption key and the decryption key are the same. All of the classical (pre-1970) cryptosystems are symmetric, as are the more recent Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

Public key algorithms were introduced in the 1970s and revolutionized cryptography. Suppose Alice wants to communicate securely with Bob, but they are hundreds of kilometers apart and have not agreed on a key to use. It seems almost impossible for them to do this without first getting together to agree on a key, or using a trusted courier to carry the key from one to the other. Certainly Alice cannot send a message over open channels to tell Bob the key, and then send the ciphertext encrypted with this key. The amazing fact is that this problem has a solution, called public key cryptography. The encryption key is made public, but it is computationally infeasible to find the decryption key without information known only to Bob. The most popular implementation is RSA (see Chapter 9), which is based on the difficulty of factoring large integers. Other versions (see Chapters 10, 23, and 24) are the ElGamal system (based on the discrete log problem), NTRU (lattice based) and the McEliece system (based on error correcting codes).

Here is a nonmathematical way to do public key communication. Bob sends Alice a box and an unlocked padlock. Alice puts her message in the box, locks Bob's lock on it, and sends the box back to Bob. Of course, only Bob can open the box and read the message. The public key methods mentioned previously are mathematical realizations of this idea. Clearly there are questions of authentication that must be dealt with. For example, Eve could intercept the first transmission and substitute her own lock. If she then intercepts the locked box when Alice sends it back to Bob, Eve can unlock her lock and read Alice's message. This is a general problem that must be addressed with any such system.

Public key cryptography represents what is possibly the final step in an interesting historical progression. In the earliest years of cryptography, security depended on keeping the encryption method secret. Later, the method was assumed known, and the security depended on keeping the (symmetric) key private or unknown to adversaries. In public key cryptography, the method and the encryption key are made public, and everyone knows what must be done to find the decryption key. The security rests on the fact (or hope) that this is computationally infeasible. It's rather paradoxical that an increase in the power of cryptographic algorithms over the years has corresponded to an increase in the amount of information given to an adversary about such algorithms.

Public key methods are very powerful, and it might seem that they make the use of symmetric key cryptography obsolete. However, this added flexibility is not free and comes at a computational cost. The amount of computation needed in public key algorithms is typically several orders of magnitude more than the amount of computation needed in algorithms such as DES or AES/Rijndael. The rule of thumb is that public key methods should not be used for encrypting large quantities of data. For this reason, public key methods are used in applications where only small amounts of data must be processed (for example, digital signatures and sending keys to be used in symmetric key algorithms).

Within symmetric key cryptography, there are two types of ciphers: stream ciphers and block ciphers. In stream ciphers, the data are fed into the algorithm in small pieces (bits or characters), and the output is produced in corresponding small pieces. We discuss stream ciphers in Chapter 5. In block ciphers, however, a block of input bits is collected and fed into the algorithm all at once, and the output is a block of bits. Mostly we shall be concerned with block ciphers. In particular, we cover two very significant examples. The first is DES, and the second is AES, which was selected in the year 2000 by the National Institute for Standards and Technology as the replacement for DES. Public key methods such as RSA can also be regarded as block ciphers.

Finally, we mention a historical distinction between different types of encryption, namely **codes** and **ciphers**. In a code, words or certain letter combinations are replaced by codewords (which may be strings of symbols). For example, the British navy in World War I used 03680C, 36276C, and 50302C to represent *shipped at, shipped by*, and *shipped from*, respectively. Codes have the disadvantage that unanticipated words cannot be used. A cipher, on the other hand, does not use the linguistic structure of the message but rather encrypts every string of characters, meaningful or not, by some algorithm. A cipher is therefore more versatile than a code. In the early days of cryptography, codes were commonly used, sometimes in conjunction with ciphers. They are still used today; covert operations are often given code names. However, any secret that is to remain secure needs to be encrypted with a cipher. In this book, we'll deal exclusively with ciphers.

## 1.1.3 Key Length

The security of cryptographic algorithms is a difficult property to measure. Most algorithms employ keys, and the security of the algorithm is related to how difficult it is for an adversary to determine the key. The most obvious approach is to try every possible key and see which ones yield meaningful decryptions. Such an attack is called a **brute force attack**. In a brute force attack, the length of the key is directly related to how long it will take to search the entire keyspace. For example, if a key is $16 \text{ bits}$ long, then there are $2^{16} = 65536 \text{ possible keys.}$ The DES algorithm has a $56\text{-bit key}$ and thus has $2^{56} \approx 7.2 \times 10^{16} \text{ possible keys.}$

In many situations we'll encounter in this book, it will seem that a system can be broken by simply trying all possible keys. However, this is often easier said than done. Suppose you need to try $10^{30}$ possibilities and you have a computer that can do $10^{9}$ such calculations each second. There are around $3 \times 10^{7} \text{ seconds}$ in a year, so it would take a little more than $3 \times 10^{13} \text{ years}$ to complete the task, longer than the predicted life of the universe.

Longer keys are advantageous but are not guaranteed to make an adversary's task difficult. The algorithm itself also plays a critical role. Some algorithms might be able to be attacked by means other than brute force, and some algorithms just don't make very efficient use of their keys' bits. This is a very important point to keep in mind. Not all $128\text{-bit}$ algorithms are created equal!

For example, one of the easiest cryptosystems to break is the substitution cipher, which we discuss in Section 2.4. The number of possible keys is $26! \approx 4 \times 10^{26}$. In contrast, DES (see Chapter 7) has only $2^{56} \approx 7.2 \times 10^{16}$ keys. But it typically takes over a day on a specially designed computer to find a DES key. The difference is that an attack on a substitution cipher uses the underlying structure of the language, while the attack on DES is by brute force, trying all possible keys.

A brute force attack should be the last resort. A cryptanalyst always hopes to find an attack that is faster. Examples we'll meet are frequency analysis (for the substitution and Vigenère ciphers) and birthday attacks (for discrete logs).

We also warn the reader that just because an algorithm seems secure now, we can't assume that it will remain so. Human ingenuity has led to creative attacks on cryptographic protocols. There are many examples in modern cryptography where an algorithm or protocol was successfully attacked because of a loophole presented by poor implementation, or just because of advances in technology. The DES algorithm, which withstood $20$ years of cryptographic scrutiny, ultimately succumbed to attacks by a well-designed parallel computer. Even as you read this book, research in quantum computing is underway, which could dramatically alter the terrain of future cryptographic algorithms.

For example, the security of several systems we'll study depends on the difficulty of factoring large integers, say of around 600 digits. Suppose you want to factor a number $n$ of this size. The method used in elementary school is to divide $n$ by all of the primes up to the square root of $n$. There are approximately $1.4 \times 10^{297}$ primes less than $10^{300}$. Trying each one is impossible. The number of electrons in the universe is estimated to be less than $10^{90}$. Long before you finish your calculation, you'll get a call from the electric company asking you to stop. Clearly, more sophisticated factoring algorithms must be used, rather than this brute force type of attack. When RSA was invented, there were some good factoring algorithms available, but it was predicted that a 129-digit number such as the RSA challenge number (see Chapter 9) would not be factored within the foreseeable future. However, advances in algorithms and computer architecture have made such factorizations fairly routine (although they still require substantial computing resources), so now numbers of several hundred digits are recommended for security. But if a full-scale quantum computer is ever built, factorizations of even these numbers will be easy, and the whole RSA scheme (along with many other methods) will need to be reconsidered.

A natural question, therefore, is whether there are any unbreakable cryptosystems, and, if so, why aren't they used all the time?

The answer is yes; there is a system, known as the one-time pad, that is unbreakable. Even a brute force attack will not yield the key. But the unfortunate truth is that the expense of using a one-time pad is enormous. It requires exchanging a key that is as long as the plaintext, and even then the key can only be used once. Therefore, one opts for algorithms that, when implemented correctly with the appropriate key size, are unbreakable in any reasonable amount of time.

An important point when considering key size is that, in many cases, one can mathematically increase security by a slight increase in key size, but this is not always practical. If you are working with chips that can handle words of 64 bits, then an increase in the key size from 64 to 65 bits could mean redesigning your hardware, which could be expensive. Therefore, designing good cryptosystems involves both mathematical and engineering considerations.

Finally, we need a few words about the size of numbers. Your intuition might say that working with a 20-digit number takes twice as long as working with a 10-digit number. That is true in some algorithms. However, if you count up to $10^{10}$, you are not even close to $10^{20}$; you are only one 10 billionth of the way there. Similarly, a brute force attack against a 60-bit key takes a billion times longer than one against a 30-bit key.

There are two ways to measure the size of numbers: the actual magnitude of the number $n$, and the number of digits in its decimal representation (we could also use its binary representation), which is approximately $\log_{10}(n)$. The number of single-digit multiplications needed to square a $k\text{-digit number } n$, using the standard algorithm from elementary school, is $k^2$, or approximately $(\log_{10} n)^2$. The number of divisions needed to factor a number $n$ by dividing by all primes up to the square root of $n$ is around $n^{1/2}$. An algorithm that runs in time a power of $\log n$ is much more desirable than one that runs in time a power of $n$. In the present example, if we double the number of digits in $n$, the time it takes to square $n$ increases by a factor of 4, while the time it takes to factor $n$ increases enormously. Of course, there are better algorithms available for both of these operations, but, at present, factorization takes significantly longer than multiplication.

We'll meet algorithms that take time a power of $\log\ n$ to perform certain calculations (for example, finding greatest common divisors and doing modular exponentiation). There are other computations for which the best known algorithms run only slightly better than a power of $n$ (for example, factoring and finding discrete logarithms). The interplay between the fast algorithms and the slower ones is the basis of several cryptographic algorithms that we’ll encounter in this book.

## 1.2 Cryptographic Applications

Cryptography is not only about encrypting and decrypting messages, it is also about solving real-world problems that require information security. There are four main objectives that arise:

1. Confidentiality: Eve should not be able to read Alice's message to Bob. The main tools are encryption and decryption algorithms.

2. Data integrity: Bob wants to be sure that Alice's message has not been altered. For example, transmission errors might occur. Also, an adversary might intercept the transmission and alter it before it reaches the intended recipient. Many cryptographic primitives, such as hash functions, provide methods to detect data manipulation by malicious or accidental adversaries.

3. Authentication: Bob wants to be sure that only Alice could have sent the message he received. Under this heading, we also include identification schemes and password protocols (in which case, Bob is the computer). There are actually two types of authentication that arise in cryptography: entity authentication and data-origin authentication. Often the term *identification* is used to specify entity authentication, which is concerned with proving the identity of the parties involved in a communication. Data-origin authentication focuses on tying the information about the origin of the data, such as the creator and time of creation, with the data.

4. Non-repudiation: Alice cannot claim she did not send the message. Non-repudiation is particularly important in electronic commerce applications, where it is important that a consumer cannot deny the authorization of a purchase.

Authentication and non-repudiation are closely related concepts, but there is a difference. In a symmetric key cryptosystem, Bob can be sure that a message comes from Alice (or someone who knows Alice's key) since no one else could have encrypted the message that Bob decrypts successfully. Therefore, authentication is automatic. However, he cannot prove to anyone else that Alice sent the message, since he could have sent the message himself. Therefore, non-repudiation is essentially impossible. In a public key cryptosystem, both authentication and non-repudiation can be achieved (see Chapters 9, 13, and 15).

Much of this book will present specific cryptographic applications, both in the text and as exercises. Here is an overview.

Digital signatures: One of the most important features of a paper and ink letter is the signature. When a document is signed, an individual's identity is tied to the message. The assumption is that it is difficult for another person to forge the signature onto another document. Electronic messages, however, are very easy to copy exactly. How do we prevent an adversary from cutting the signature off one document and attaching it to another electronic document? We shall study cryptographic protocols that allow for electronic messages to be signed in such a way that everyone believes that the signer was the person who signed the document, and such that the signer cannot deny signing the document.

Identification: When logging into a machine or initiating a communication link, a user needs to identify herself or himself. But simply typing in a user name is not sufficient as it does not prove that the user is really who he or she claims to be. Typically a password is used. We shall touch upon various methods for identifying oneself. In the chapter on DES we discuss password files. Later, we present the Feige-Fiat-Shamir identification scheme, which is a zero-knowledge method for proving identity without revealing a password.

Key establishment: When large quantities of data need to be encrypted, it is best to use symmetric key encryption algorithms. But how does Alice give the secret key to Bob when she doesn't have the opportunity to meet him personally? There are various ways to do this. One way uses public key cryptography. Another method is the Diffie-Hellman key exchange algorithm. A different approach to this problem is to have a trusted third party give keys to Alice and Bob. Two examples are Blom’s key generation scheme and Kerberos, which is a very popular symmetric cryptographic protocol that provides authentication and security in key exchange between users on a network.

Secret sharing: In Chapter 17, we introduce secret sharing schemes. Suppose that you have a combination to a bank safe, but you don’t want to trust any single person with the combination to the safe. Rather, you would like to divide the combination among a group of people, so that at least two of these people must be present in order to open the safe. Secret sharing solves this problem.

Security protocols: How can we carry out secure transactions over open channels such as the Internet, and how can we protect credit card information from fraudulent merchants? We discuss various protocols, such as SSL and SET.

Electronic cash: Credit cards and similar devices are convenient but do not provide anonymity. Clearly a form of electronic cash could be useful, at least to some people. However, electronic entities can be copied. We give an example of an electronic cash system that provides anonymity but catches counterfeiters, and we discuss cryptocurrencies, especially Bitcoin.

Games: How can you flip coins or play poker with people who are not in the same room as you? Dealing the cards, for example, presents a problem. We show how cryptographic ideas can solve these problems.