COMPUTATIONAL NUMBER THEORY AND MODERN CRYPTOGRAPHY

INFORMATION SECURITY SERIES

The Wiley-HEP Information Security Series systematically introduces the fundamentals of information security design and application. The goals of the Series are:

- to provide fundamental and emerging theories and techniques to stimulate more research in cryptology, algorithms, protocols, and architectures
- to inspire professionals to understand the issues behind important security problems and the ideas behind the solutions
- to give references and suggestions for additional reading and further study

The Series is a joint project between Wiley and Higher Education Press (HEP) of China. Publications consist of advanced textbooks for graduate students as well as researcher and practitioner references covering the key areas, including but not limited to:

– Modern Cryptography
– Cryptographic Protocols and Network Security Protocols
– Computer Architecture and Security
– Database Security
– Multimedia Security
– Computer Forensics
– Intrusion Detection

COMPUTATIONAL NUMBER THEORY AND MODERN CRYPTOGRAPHY

Song Y. Yan

College of Sciences
North China University of Technology
Beijing, China

&

Department of Mathematics
Harvard University
Cambridge, USA

WILEY

高等教育出版社
HIGHER EDUCATION PRESS

This edition first published 2013

Published by John Wiley & Sons Singapore Pte. Ltd., 1 Fusionopolis Walk, #07-01 Solaris South Tower, Singapore 138628, under exclusive license by Higher Education Press in all media and all languages throughout the world excluding Mainland China and excluding Simplified and Traditional Chinese languages.

Library of Congress Cataloging-in-Publication Data

Yan, Song Y.
Computational number theory and modern cryptography / Song Y. Yan.
pages cm
Includes bibliographical references and index.
ISBN 978-1-118-18858-3 (hardback)
1. Data encryption (Computer science) 2. Number theory–Data processing. I. Title.
QA76.9.A25Y358 2012
005.8′2–dc23

2012032708

ISBN: 9781118188583

Typeset in 10/12pt Times by Aptara Inc., New Delhi, India

CONTENTS

ABOUT THE AUTHOR

Professor Song Y. Yan majored in both Computer Science and Mathematics, and obtained a PhD in Number Theory in the Department of Mathematics at the University of York, England. His current research interests include Computational Number Theory, Computational Complexity Theory, Algebraic Coding Theory, Public-Key Cryptography and Information/Network Security. He published, among others, the following five well-received and popular books in computational number theory and public-key cryptography:

[1] *Perfect, Amicable and Sociable Numbers: A Computational Approach*, World Scientific, 1996.

[2] *Number Theory for Computing*, Springer, First Edition, 2000, Second Edition, 2002. (Polish Translation, Polish Scientific Publishers PWN, Warsaw, 2006; Chinese Translation, Tsinghua University Press, Beijing, 2007.)

[3] *Cryptanalytic Attacks on RSA*, Springer, 2007. (Russian Translation, Moscow, 2010.)

[4] *Primality Testing and Integer Factorization in Public-Key Cryptography*, Springer, First Edition, 2004; Second Edition, 2009.

[5] *Quantum Attacks on Public-Key Cryptosystems*, Springer, 2012.

Song can be reached by email address songyuanyan@gmail.com anytime.

PREFACE

The book is about number theory and modern cryptography. More specically, it is about *computational* number theory and modern *public-key* cryptography based on number theory. It consists of four parts. The first part, consisting of two chapters, provides some preliminaries. Chapter 1 provides some basic concepts of number theory, computation theory, computational number theory, and modern public-key cryptography based on number theory. In chapter 2, a complete introduction to some basic concepts and results in abstract algebra and elementary number theory is given.

The second part is on computational number theory. There are three chapters in this part. Chapter 3 deals with algorithms for primality testing, with an emphasis on the Miller-Rabin test, the elliptic curve test, and the AKS test. Chapter 4 treats with algorithms for integer factorization, including the currently fastest factoring algorithm NFS (Number Field Sieve), and the elliptic curve factoring algorithm ECM (Elliptic Curve Method). Chapter 5 discusses various modern algorithms for discrete logarithms and for elliptic curve discrete logarithms. It is well-known now that primality testing can be done in polynomial-time on a digital computer, however, integer factorization and discrete logarithms still cannot be performed in polynomial-time. From a computational complexity point of view, primality testing is feasible (tractable, easy) on a digital computer, whereas integer factorization and discrete logarithms are infeasible (intractable, hard, difficult). Of course, no-one has yet been able to prove that the integer factorization and the discrete logarithm problems must be infeasible on a digital computer.

Building on the results in the first two parts, the third part of the book studies the modern cryptographic schemes and protocols whose security relies exactly on the infeasibility of the integer factorization and discrete logarithm problems. There are four chapters in this part. Chapter 6 presents some basic concepts and ideas of secret-key cryptography. Chapter 7 studies the integer factoring based public-key cryptography, including, among others, the most famous and widely used RSA cryptography, the Rabin cryptosystem, the probabilistic encryption and the zero-knowledge proof protocols. Chapter 8 studies the discrete logarithm based cryptography, including the DHM key-exchange protocol (the world's first public-key system), the ElGamal cryptosystem, and the US Government's Digital Signature Standard (DSS), Chapter 9 discusses various cryptographic systems and digital signature schemes based on the infeasibility of the elliptic curve discrete logarithm problem, some of them are just the elliptic curve analogues of the ordinary public-key cryptography such as elliptic curve DHM, elliptic curve ElGamal, elliptic curve RSA, and elliptic curve DSA/DSS.

It is interesting to note that although integer factorization and discrete logarithms cannot be solved in polynomial-time on a classical *digital* computer, they all can be solved in polynomial-time on a quantum computer, provided that a practical quantum computer with several thousand quantum bits can be built. So, the last part of the book is on quantum computational number theory and quantum-computing resistant cryptography. More speciffically, in Chapter 10, we shall study efficient quantum algorithms for solving the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP). Since IFP, DLP and ECDLP can be solved efficiently on a quantum computer, the IFP, DLP and ECDLP based cryptographic systems and protocols can be broken efficiently on a quantum computer. However, there are many infeasible problems such as the coding-based problems and the lattice-based problems that cannot be solved in polynomial-time even on a quantum computer. That is, a quantum computer is basically a special type of computing device using a different computing paradigm, it is only suitable or good for some special problems such as the IFP, DLP and ECDLP problems. Thus, in chapter 11, the last chapter of the book, we shall discuss some quantum-computing resistant cryptographic systems, including the coding-based and lattice-based cryptographic systems, that resist all known quantum attacks. Note that quantum-computing resistant cryptography is still classic cryptography, but quantum resistant. We shall, however, also introduce a truly quantum cryptographic scheme, based on ideas of quantum mechanics and some DNA cryptographic schemes based on idea of DNA molecular computation.

The materials presented in the book are based on the author's many years teaching and research experience in the field, and also based on the author's other books published in the past ten years or so, particularly the following three books, all by Springer:

[1] Number Theory for Computing, 2nd Edition, 2002.
[2] Cryptanalytic Attacks on RSA, 2007.
[3] Primality Testing and Integer Factorization in Public-Key Cryptography, 2nd Edition, 2009.

The book is suited as a text for final year undergraduate or first year postgraduate courses in computational number theory and modern cryptography, or as a basic research reference in the field.

Corrections, comments and suggestions from readers are very welcomed and can be sent via email to songyuanyan@gmail.com.

Song Y. Yan
London, England
June 2012

ACKNOWLEDGMENTS

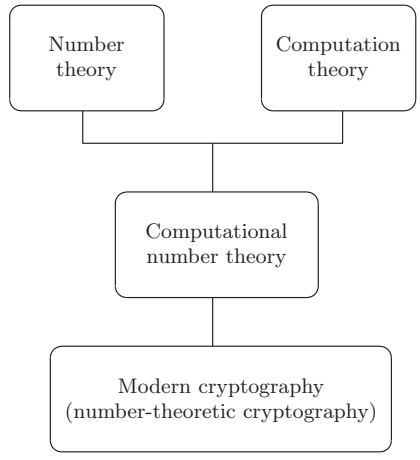
The author would like to thank the editors at Wiley and HEP, particularly Hongying Chen, Shelley Chow, James Murphy, Clarissa Lim, and Shalini Sharma, for their encouragement, assistance, and proof-reading. Special thanks must also be given to the three anonymous referees for their very helpful and constructive comments and suggestions.

The work was supported in part by the Royal Society London, the Royal Academy of Engineering London, the Recruitment Program of Global Experts of Hubei Province, the Funding Project for Academic Human Resources Development in Institutions of Higher Learning under the Jurisdiction of the Beijing Municipality (PHR/IHLB), the Massachusetts Institute of Technology and Harvard University.

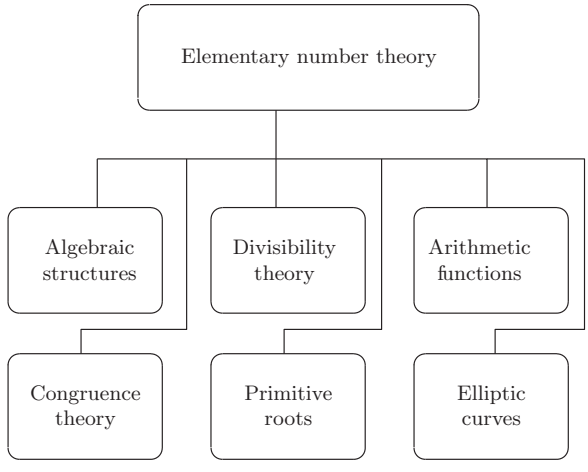
Part I

Preliminaries

In this part, we shall first explain what are number theory, computation theory, computational number theory, and modern (number-theoretic) cryptography are. The relationship betweeen them may be shown in the following figure:



Then we shall present an introduction to the elementary theory of numbers from an algebraic perspective (see the following figure), that shall be used throughout the book.



Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

1

Introduction

In this chapter, we present some basic concepts and ideas of number theory, computation theory, computational number theory, and modern (number-theoretic) cryptography. More specifically, we shall try to answer the following typical questions in the field:

- What is number theory?
- What is computation theory?
- What is computational number theory?
- What is modern (number-theoretic) cryptography?

1.1 What is Number Theory?

Number theory is concerned mainly with the study of the properties (e.g., the divisibility) of the integers

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\},$$

particularly the positive integers

$$\mathbb{Z}^+ = \{1, 2, 3, \ldots\}.$$

For example, in divisibility theory, all positive integers can be classified into three classes:

1. Unit: 1.
2. Prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, $\cdots$.
3. Composite numbers: 4, 6, 8, 9, 10, 12, 14, 15, $\cdots$.

Recall that a positive integer $n > 1$ is called a prime number, if its only divisors are 1 and n, otherwise, it is a composite number. 1 is neither prime number nor composite number. Prime numbers play a central role in number theory, as any positive integer $n > 1$ can be written uniquely into the following standard prime factorization form:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} \tag{1.1}$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

Table 1.1 $\pi(x)$ for some large x

x	$\pi(x)$
10^{15}	29844570422669
10^{16}	279238341033925
10^{17}	2623557157654233
10^{18}	24739954287740860
10^{19}	234057667276344607
10^{20}	2220819602560918840
10^{21}	21127269486018731928
10^{22}	201467286689315906290
10^{23}	1925320391606803968923
10^{24}	18435599767349200867866

where $p_1 < p_2 < \cdots < p_k$ are primes and $\alpha_1, \alpha_2, \cdots, \alpha_k$ positive integers. Although prime numbers have been studied for more than 2000 years, there are still many open problems about their distribution. Let us investigate some of the most interesting problems about prime numbers.

1. The distribution of prime numbers.
   Euclid proved 2000 years ago in his *Elements* that there were infinitely many prime numbers. That is, the sequence of prime numbers

   $$2, 3, 5, 7, 11, 13, 17, 19, \cdots$$

   is endless. For example, 2, 3, 5 are the first three prime numbers, whereas $2^{43112609} - 1$ is the largest prime number to date, it has 12978189 digits and was found on 23 August 2008. Let $\pi(x)$ denote the prime numbers up to x (Table 1.1 gives some values of $\pi(x)$ for some large x), then Euclid's theorem of infinitude of primes actually says that

   $$\pi(x) \to \infty, \quad \text{as } x \to \infty.$$

   A much better result about the distribution of prime numbers is the Prime Number theorem, stating that

   $$\pi(x) \sim x/\log x. \tag{1.2}$$

   In other words,

   $$\lim_{x\to\infty} \frac{\pi(x)}{x/\log x} = 1. \tag{1.3}$$

   Note that the log is the natural logarithm $\log_e$ (normally denoted by ln), where $e = 2.7182818\ldots$. However, if the Riemann Hypothesis [3] is true, then there is a refinement of the Prime Number theorem

   $$\pi(x) = \int_2^x \frac{\mathrm{d}t}{\log t} + \mathcal{O}\left(xe^{-c\sqrt{\log x}}\right) \tag{1.4}$$

to the effect that

$$\pi(x) = \int_2^x \frac{\mathrm{d}t}{\log t} + \mathcal{O}\left(\sqrt{x}\log x\right). \tag{1.5}$$

Of course we do not know if the Riemann Hypothesis is true. Whether or not the Riemann Hypothesis is true is one of the most important open problems in mathematics, and in fact it is one of the seven Millennium Prize Problems proposed by the Clay Mathematics Institute in Boston in 2000, each with a one million US dollars prize [4]. The Riemann hypothesis states that all the nontrivial (complex) zeros ρ of the ζ function

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}, \quad s = \sigma + it, \quad \sigma,\ t \in \mathbb{R},\ i = \sqrt{-1} \tag{1.6}$$

lying in the critical strip $0 < \mathrm{Re}(s) < 1$ must lie on the critical line $\mathrm{Re}(s) = \frac{1}{2}$, that is, $\rho = \frac{1}{2} + it$, where ρ denotes a nontrivial zero of $\zeta(s)$. Riemann calculated the first five nontrivial zeros of $\zeta(s)$ and found that they all lie on the critical line (see Figure 1.1), he then conjectured that all the nontrivial zeros of $\zeta(s)$ are on the critical line.

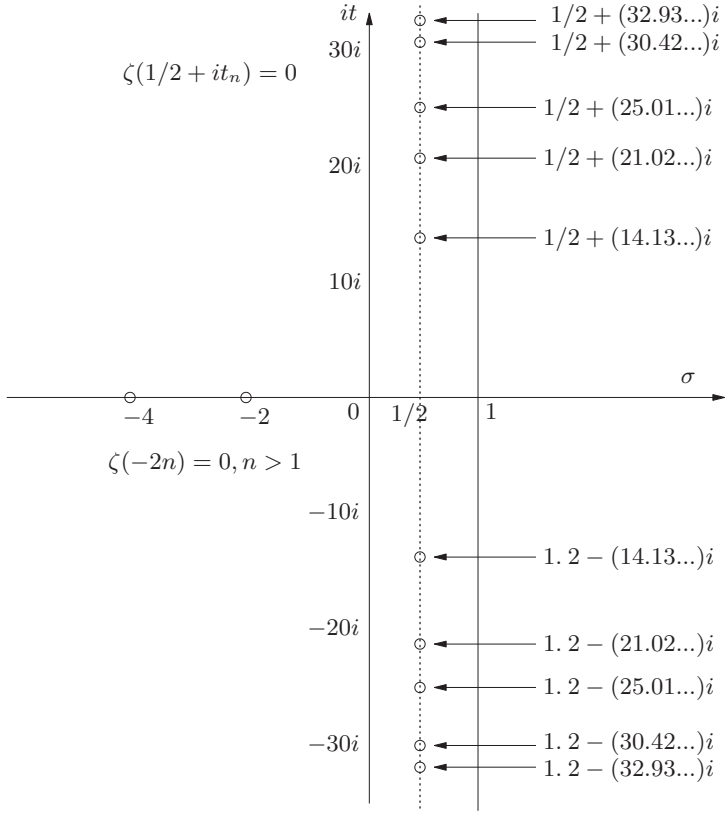


Figure 1.1 Riemann hypothesis

Table 1.2 Ten large twin prime pairs

Rank	Twin primes	Digits	Discovery date
1	$65516468355 \cdot 2^{333333} \pm 1$	100355	Aug 2009
2	$2003663613 \cdot 2^{195000} \pm 1$	58711	Jan 2007
3	$194772106074315 \cdot 2^{171960} \pm 1$	51780	Jun 2007
4	$100314512544015 \cdot 2^{171960} \pm 1$	51780	Jun 2006
5	$16869987339975 \cdot 2^{171960} \pm 1$	51779	Sep 2005
6	$33218925 \cdot 2^{169690} \pm 1$	51090	Sep 2002
7	$22835841624 \cdot 7^{54321} \pm 1$	45917	Nov 2010
8	$12378188145 \cdot 2^{140002} \pm 1$	42155	Dec 2010
9	$23272426305 \cdot 2^{140001} \pm 1$	42155	Dec 2010
10	$8151728061 \cdot 2^{125987} \pm 1$	37936	May 2010

2. The distribution of twin prime numbers.
Twin prime numbers are of the form $n \pm 1$, where both numbers are prime. For example, (3, 5), (5, 7), (11, 13) are the first three smallest twin prime pairs, whereas the largest twin primes so far are $65516468355 \cdot 2^{333333} \pm 1$, discovered in August 2009, both numbers having 100355 digits. Table 1.2 gives 10 large twin prime pairs. Let $\pi_2(x)$ be the number of twin primes up to x (Table 1.3 gives some values of $\pi_2(x)$ for different x), then the twin prime conjecture states that

$$\pi_2(x) \to \infty, \quad \text{as } x \to \infty.$$

If the probability of a random integer x and the integer $x + 2$ being prime were statistically independent, then it would follow from the prime number theorem that

$$\pi_2(x) \sim \frac{x}{(\log x)^2}, \tag{1.7}$$

or more precisely,

$$\pi_2(x) \sim c\frac{x}{(\log x)^2}, \tag{1.8}$$

with

$$c = 2\prod_{p \geq 3}\left(1 - \frac{1}{(p-1)^2}\right). \tag{1.9}$$

Table 1.3 $\pi_2(x)$ for some large values

x	10^6	10^7	10^8	10^9	10^{10}	10^{11}
$\pi_2(x)$	8169	58980	440312	3424506	27412679	224376048

As these probabilities are not independent, so Hardy and Littlewood conjectured that

$$\pi_2(x) = 2\prod_{p\geq 3}\frac{p(p-2)}{(p-1)^2}\int_2^x \frac{\mathrm{d}t}{(\log t)^2}$$
$$\approx 1.320323632\int_2^x \frac{\mathrm{d}t}{(\log t)^2}. \tag{1.10}$$

The infinite product in the above formula is the twin prime constant; this constant was estimated to be approximately 0.6601618158 . . .. Using very complicated arguments based on sieve methods, in his work on the Goldbach conjecture, the Chinese mathematician Chen showed that *there are infinitely many pairs of integers* $(n, n+2)$*, with n prime and* $n+2$ *a product of at most two primes*. The famous Goldbach conjecture states that *every even number greater than 4 is the sum of two odd prime numbers*. It was conjectured by Goldbach in a letter to Euler in 1742. It remains unsolved to this day. The best result for this conjecture is due to Chen, who announced it in 1966, but the full proof was not given until 1973 due to the chaotic Cultural Revolution, that *every sufficiently large even number is the sum of one prime number and the product of at most two prime numbers*, that is, $E = p_1 + p_2p_3$, where E is a sufficiently large even number and p_1, p_2, p_3 are prime numbers. As a consequence, there are infinitely many such twin numbers (p_1, $p_1 + 2 = p_2p_3$). Extensions relating to the twin prime numbers have also been considered. For example, are there infinitely many triplet primes (p, q, r) with $q = p + 2$ and $r = p + 6$? The first five triplets of this form are as follows: (5, 7, 11), (11, 13, 17), (17, 19, 23), (41, 43, 47), (101, 103, 107). The triplet prime problem is much harder than the twin prime problem. It is amusing to note that there is only one triplet prime (p, q, r) with $q = p + 2$ and $r = p + 4$. That is, (3, 5, 7). The Riemann Hypothesis, the Twin Prime Problem, and the Goldbach conjecture form the famous Hilbert's 8th Problem.

3. The distribution of arithmetic progressions of prime numbers.
   An arithmetic progression of prime numbers is defined to be the sequence of primes satisfying:

$$p,\ p + d,\ p + 2d, \cdots,\ p + (k-1)d \tag{1.11}$$

where p is the first term, d the common difference, and $p + (k-1)d$ the last term of the sequence. For example, the following are some sequences of the arithmetic progression of primes:

3	5	7		
5	11	17	23	
5	11	17	23	29

The longest arithmetic progression of primes is the following sequence with 23 terms: $56211383760397 + k\cdot 44546738095860$ with $k = 0, 1, \cdots, 22$. Thanks to Green and Tao who proved in 2007 that *there are arbitrary long arithmetic progressions of primes* (i.e., k can be any arbitrary large natural number), which enabled, among others, Tao to receive a Field Prize in 2006, the equivalent to a Nobel Prize for Mathematics. However, their result is not about consecutive primes; we still do not know

if there are arbitrary long arithmetic progressions of consecutive primes, although Chowa proved in 1944 that there exists an infinity of three consecutive primes of arithmetic progressions. Note that an arithmetic progression of consecutive primes is a sequence of consecutive primes in the progression. In 1967, Jones, Lal, and Blundon found an arithmetic progression of five consecutive primes $10^{10} + 24493 + 30k$ with $k = 0, 1, 2, 3, 4$. In the same year, Lander and Parkin discovered six in an arithmetic progression $121174811 + 30k$ with $k = 0, 1, 2, 3, 4, 5$. The longest arithmetic progression of consecutive primes, discovered by Manfred Toplic in 1998, is $507618446770482 \cdot 193\# + x77 + 210k$, where $193\#$ is the product of all primes ≤ 193, that is, $193\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdots 193$, $x77$ is a 77-digit number 54538241683887582668189703590110659057865934764604873840781923513421103495579 and $k = 0, 1, 2, \cdots, 9$.

It should be noted that problems in number theory are easy to state, because they are mainly concerned with integers with which we are very familiar, but often very hard to solve!

Problems for Section 1.1

1. Show that there are infinitely many prime numbers.
2. Prove or disprove there are infinitely many twin prime numbers.
3. Are there infinitely many triple prime numbers of the form p, $p + 2$, $p + 4$, where p, $p + 2$, $p + 4$ are all prime numbers? For example, $3, 5, 7$ are such triple prime numbers.
4. Are there infinitely many triple prime numbers of the form p, $p + 2$, $p + 6$, where p, $p + 2$, $p + 6$ are all prime numbers? For example, $5, 7, 11$ are such triple prime numbers.
5. (Prime Number Theorem) Show that

$$\lim_{x \to \infty} \frac{\pi(x)}{x/\log x} = 1.$$

6. The Riemann ζ-function is defined as follows:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

where $s = \sigma + it$ is a complex number. Riemann conjectured that all zeroes of $\zeta(s)$ in the critical strip $0 \leq \sigma \leq 1$ must lie on the critical line $\sigma = \frac{1}{2}$. That is,

$$\zeta\left(\frac{1}{2} + it\right) = 0.$$

Prove or disprove the Riemann Hypothesis.

7. Andrew Beal in 1993 conjectured that the equation $x^a + y^b = z^c$ has no positive integer solutions in x, y, z, a, b, c, where $a, b, c \geq 3$ and $\gcd(x, y) = (y, z) = (x, z) = 1$. Beal has offered \$100 000 for a proof or a disproof of this conjecture.

8. Prove or disprove the Goldbach conjecture that any even number greater than 6 is the sum of two odd prime numbers.
9. A positive integer n is perfect if $\sigma(n) = 2n$, where $\sigma(n)$ is the sum of all divisors of n. For example, 6 is perfect since $\sigma(6) = 1 + 2 + 3 + 6 = 2 \cdot 6 = 12$. Show n is perfect if and only if $n = 2^{p-1}(2^p - 1)$, where $2^p - 1$ is a Mersenne prime.
10. All known perfect numbers are even perfect. Recent research shows that if there exists an odd perfect number, it must be greater than 10^{300} and must have at least 29 prime factors (not necessarily distinct). Prove or disprove that there exists at least one odd perfect number.
11. Show that there are arbitrary long arithmetic progressions of prime numbers

$$p, p + d, p + 2d, \cdots, p + (k - 1)d$$

where p is the first term, d the common difference, and $p + (k - 1)d$ the last term of the sequence, and furthermore, all the terms in the sequence are prime numbers and k can be any arbitrary large positive integer.
12. Prove or disprove that there are arbitrary long arithmetic progressions of consecutive prime numbers.

1.2 What is Computation Theory?

Computation theory, or the theory of computation, is a branch that deals with whether and how efficiently problems can be solved on a model of computation, using an algorithm. It may be divided into two main branches: Computability theory and computational complexity theory. Generally speaking, computability theory deals with what a computer can or cannot do theoretically (i.e., without any restrictions), whereas complexity theory deals with what computer can or cannot do practically (with e.g., time or space limitations). Feasibility or infeasibility theory is a subfield of complexity theory, which concerns itself with what a computer can or cannot do efficiently in polynomial-time. A reasonable model of computation is the Turing machine, first studied by the great British logician and mathematician Alan Turing in 1936, we shall first introduce the basic concepts of Turing machines, then discuss complexity, feasibility, and infeasiblity theories based on Turing machines.

Definition 1.1 A standard multitape *Turing machine*, M (see Figure 1.2), is an algebraic system defined by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F) \tag{1.12}$$

where

1. Q is a finite set of *internal states*;
2. Σ is a finite set of symbols called the *input alphabet*. We assume that $\Sigma \subseteq \Gamma - \{\square\}$;
3. Γ is a finite set of symbols called the *tape alphabet*;

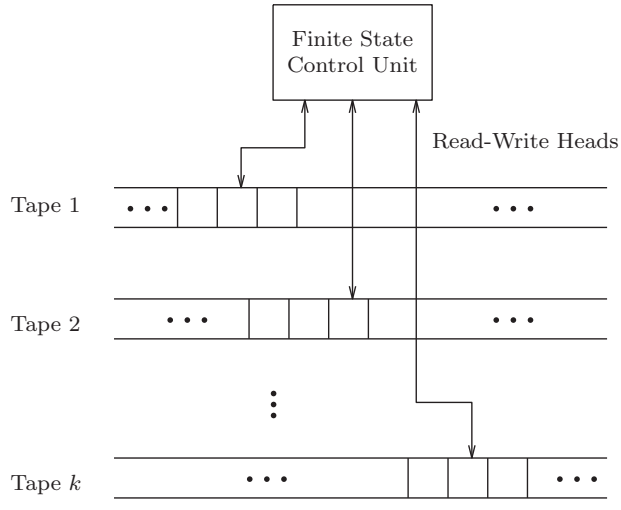


Figure 1.2 k-tape ($k \geq 1$) Turing machine

4. δ is the transition function, which is defined by
   (i) if M is a deterministic Turing machine (DTM), then

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k \tag{1.13}$$

   (ii) if M is a nondeterministic Turing machine (NDTM), then

$$\delta : Q \times \Gamma^k \to 2^{Q \times \Gamma^k \times \{L,R\}^k} \tag{1.14}$$

   where L and R specify the movement of the read-write head *left* or *right*. When $k = 1$, it is just a standard one-tape Turing machine;
5. $\square \in \Gamma$ is a special symbol called the *blank*;
6. $q_0 \in Q$ is the *initial state*;
7. $F \subseteq Q$ is the set of *final states*.

Thus, Turing machines provide us with the simplest possible abstract model of computation for modern digital (even quantum) computers.

Any *effectively* computable function can be computed by a Turing machine, and there is no effective procedure that a Turing machine cannot perform. This leads naturally to the following famous Church–Turing thesis, named after Alonzo Church (1903–1995) and Alan Turing (1912–1954):

> **The Church–Turing thesis**: Any effectively computable function can be computed by a Turing machine.

The Church–Turing thesis thus provides us with a powerful tool to distinguish what is computation and what is not computation, what function is computable and what function

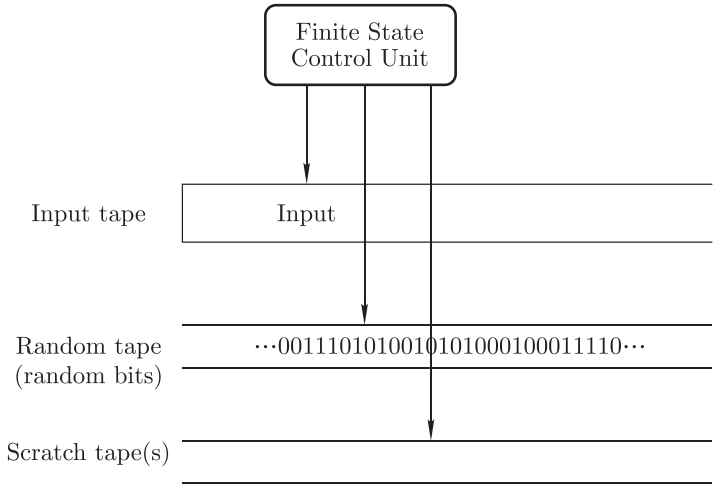


Figure 1.3 Probabilistic k-tape ($k \geq 1$) Turing machine

is not computable, and more generally, what computers can do and what computers cannot do. From a computer science and particularly a cryptographic point of view, we are not just interested in what computers can do, but in what computers can do efficiently. That is, in cryptography we are more interested in practical computable rather than just theoretical computable; this leads to the Cook–Karp thesis.

Definition 1.2 A *probabilistic Turing machine* is a type of nondeterministic Turing machine with distinct states called *coin-tossing states*. For each coin-tossing state, the finite control unit specifies two possible legal next states. The computation of a probabilistic Turing machine is deterministic except that in coin-tossing states the machine tosses an unbiased coin to decide between the two *possible legal* next states.

A probabilistic Turing machine can be viewed as a *randomized Turing machine*, as described in Figure 1.3. The first tape, holding input, is just the same as conventional multitape Turing machine. The second tape is referred to as random tape, containing randomly and independently chosen bits, with probability $1/2$ of a 0 and the same probability $1/2$ of a 1. The third and subsequent tapes are used, if needed, as scratch tapes by the Turing machine.

Definition 1.3 $\mathcal{P}$ is the class of problems solvable in polynomial-time by a deterministic Turing machine (DTM). Problems in this class are classified to be tractable (feasible) and easy to solve on a computer. For example, additions of any two integers, no matter how big they are, can be performed in polynomial-time, and hence are is in $\mathcal{P}$.

Definition 1.4 $\mathcal{NP}$ is the class of problems solvable in polynomial-time on a nondeterministic Turing machine (NDTM). Problems in this class are classified to be intractable

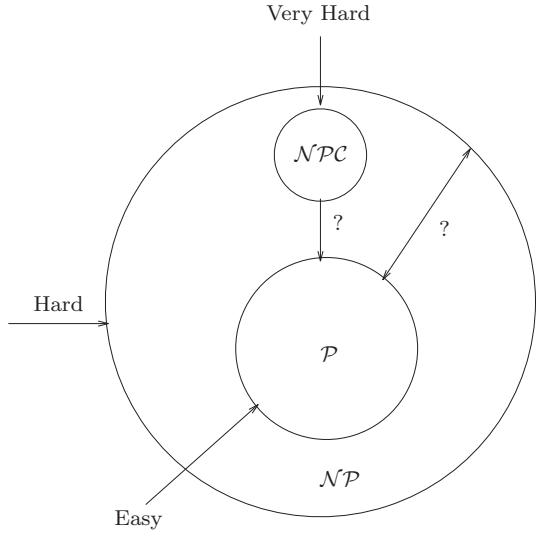


Figure 1.4 The $\mathcal{P}$ Versus $\mathcal{NP}$ problem

(infeasible) and hard to solve on a computer. For example, the Traveling Salesman Problem (TSP) is in $\mathcal{NP}$, and hence it is hard to solve.

In terms of formal languages, we may also say that $\mathcal{P}$ is the class of languages where the membership in the class can be decided in polynomial-time, whereas $\mathcal{NP}$ is the class of languages where the membership in the class can be verified in polynomial-time. It seems that the power of polynomial-time verifiable is greater than that of polynomial-time decidable, but no proof has been given to support this statement (see Figure 1.4). The question of whether or not $\mathcal{P} = \mathcal{NP}$ is one of the greatest unsolved problems in computer science and mathematics, and in fact it is one of the seven Millennium Prize Problems proposed by the Clay Mathematics Institute in Boston in 2000, each with one-million US dollars.

Definition 1.5 $\mathcal{EXP}$ is the class of problems solvable by a deterministic Turing machine (DTM) in time bounded by 2^{n^i}.

Definition 1.6 A function f is polynomial-time computable if for any input w, $f(w)$ will halt on a Turing machine in polynomial-time. A language A is polynomial-time reducible to a langauge B, denoted by $A \leq_{\mathcal{P}} B$, if there exists a polynomial-time computable function such that for every input w,

$$w \in A \Longleftrightarrow f(w) \in B.$$

The function f is called the polynomial-time reduction of A to B.

Definition 1.7 A language/problem L is $\mathcal{NP}$-complete, denoted by $\mathcal{NPC}$, if it satisfies the following two conditions:

(1) $L \in \mathcal{NP}$,
(2) $\forall A \in \mathcal{NP}$, $A \leq_{\mathcal{P}} L$.

Definition 1.8 A problem D is $\mathcal{NP}$-hard, denoted by $\mathcal{NPH}$, if it satisfies the following condition:

$$\forall A \in \mathcal{NP}, \quad A \leq_{\mathcal{P}} D$$

where D may be in $\mathcal{NP}$, or may not be in $\mathcal{NP}$. Thus, $\mathcal{NP}$-hard means *at least as hard as any $\mathcal{NP}$-problem*, although it might, in fact, be harder.

Definition 1.9 $\mathcal{RP}$ is the class of problems solvable in expected polynomial-time with *one-sided error* by a probabilistic (randomized) Turing machine (PTM). By "one-sided error" we mean that the machine will answer "yes" when the answer is "yes" with a probability of error $< 1/2$, and will answer "no" when the answer is "no" with zero probability of error.

Definition 1.10 $\mathcal{ZPP}$ is the class of problems solvable in expected polynomial-time with *zero error* on a probabilistic Turing machine (PTM). It is defined by $\mathcal{ZPP} = \mathcal{RP} \cap \text{co-}\mathcal{RP}$, where co-$\mathcal{RP}$ is the complement of $\mathcal{RP}$. By "zero error" we mean that the machine will answer "yes" when the answer is "yes" (with zero probability of error), and will answer "no" when the answer is "no" (also with zero probability of error). But note that the machine may also answer "?", which means that the machine does not know if the answer is "yes" or "no." However, it is guaranteed that in at most half of simulation cases the machine will answer "?." $\mathcal{ZPP}$ is usually referred to as an *elite class*, because it also equals to the class of problems that can be solved by randomized algorithms that always give the correct answer and run in expected polynomial-time.

Definition 1.11 $\mathcal{BPP}$ is the class of problems solvable in expected polynomial-time with *two-sided error* on a probabilistic Turing machine (PTM), in which the answer always has probability at least $\frac{1}{2} + \delta$, for some fixed $\delta > 0$ of being correct. The "$\mathcal{B}$" in $\mathcal{BPP}$ stands for "bounded away the error probability from $\frac{1}{2}$"; for example, the error probability could be $\frac{1}{3}$.

It is widely believed, although no proof has been given, that problems in $\mathcal{P}$ are computationally tractable, whereas problems not in (beyond) $\mathcal{P}$ are computationally intractable. This is the famous Cook–Karp thesis, named after Stephen Cook and Richard Karp:

> **The Cook–Karp thesis**. Any computationally tractable problem can be computed by a Turing machine in deterministic polynomial-time.

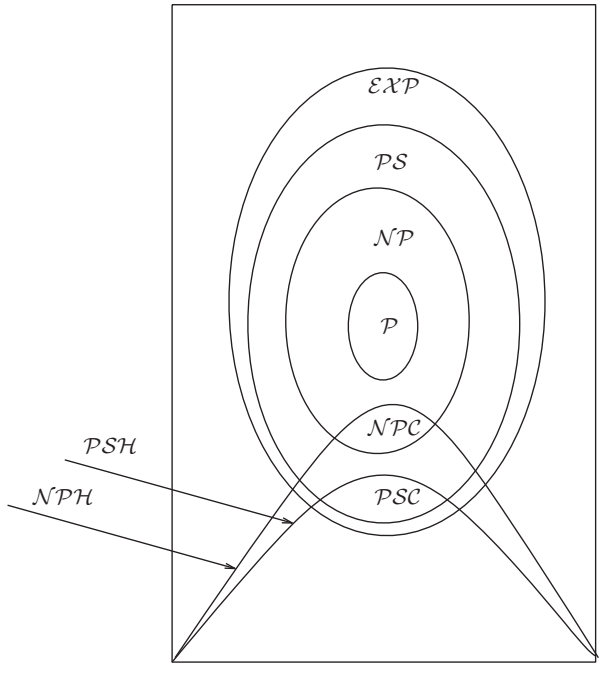


Figure 1.5 Conjectured relationships among classes $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{NPC}$, etc.

Thus, problems in $\mathcal{P}$ are tractable whereas problems in $\mathcal{NP}$ are intractable. However, there is not a clear cut line between the two types of problems. This is exactly the $\mathcal{P}$ versus $\mathcal{NP}$ problem, mentioned earlier.

Similarly, one can define the classes of problems of $\mathcal{P}$-Space, $\mathcal{NP}$-Space, $\mathcal{P}$-Space Complete, and $\mathcal{P}$-Space Hard. We shall use $\mathcal{NPC}$ to denote the set of $\mathcal{NP}$-Complete problems, $\mathcal{PSC}$ the set of $\mathcal{P}$-Space Complete problems, $\mathcal{NPH}$ the set of $\mathcal{NP}$-Hard problems, and $\mathcal{PSH}$ the set of $\mathcal{P}$-Space Hard problems. The relationships among the classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NPC}$, $\mathcal{PSC}$, $\mathcal{NPH}$, $\mathcal{PSH}$, and $\mathcal{EXP}$ may be described as in Figure 1.5.

It is clear that a time class is included in the corresponding space class since one unit is needed for the space by one square. Although it is not known whether or not $\mathcal{P} = \mathcal{NP}$, it is known that $\mathcal{PSPACE} = \mathcal{NPSPACE}$. It is generally believed that

$$\mathcal{P} \subseteq \mathcal{ZPP} \subseteq \mathcal{RP} \subseteq \binom{\mathcal{BPP}}{\mathcal{NP}} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXP}. \tag{1.15}$$

Besides the proper inclusion $\mathcal{P} \subset \mathcal{EXP}$, it is not known whether any of the other inclusions in the above hierarchy is proper. Note that the relationship of $\mathcal{BPP}$ and $\mathcal{NP}$ is not known, although it is believed that $\mathcal{NP} \not\subseteq \mathcal{BPP}$.

Problems for Section 1.2

1. Explain why the Church–Turing thesis cannot be proved to be true.
2. Explain why, if any one of the problems in $\mathcal{NPC}$ can be solved in $\mathcal{P}$, then all the problems in $\mathcal{NP}$ can be solved in $\mathcal{P}$.
3. Show that $\mathcal{PSPACE} = \mathcal{NPSPACE}$.
4. Prove or disprove $\mathcal{P} \neq \mathcal{NP}$.
5. In number theoretic computation, it is reasonable to measure how many bit operations a number theoretic algorithm requires, rather than just how many arithmetic operation it requires. Let a and b both have β (or at least one of them has β) bits. Show that the bit operations for the multiplication of two β numbers can be as follows:
   1. $\mathcal{O}(\beta^2)$ if an ordinary method is used;
   2. $\mathcal{O}(\beta^{\log_2 3})$ if a simple divide-and-conquer method is used;
   3. $\mathcal{O}(\beta \log \beta \log\log \beta) = \mathcal{O}(\beta^{1+\epsilon})$ if a fast method (e.g., Fourier transforms) is used.
6. Show that the addition, subtraction, and division of two integers can be done in polynomial-time.
7. Show that the polynomial factorization (not integer factorization) can be done in polynomial-time.
8. Show that matrix multiplications can be done in polynomial-time.

1.3 What is Computational Number Theory?

Computational number theory is a new branch of mathematics. Informally, it can be regarded as a combined and disciplinary subject of number theory and computer science, particularly computation theory, including the theory of classical electronic computing, quantum computing, and biological computing:

Computational Number Theory := Number Theory $\oplus$ Computation Theory

$\Downarrow$	$\Downarrow$	$\Downarrow$
Primality Testing	Elementary Number Theory	Computability Theory
Integer Factorization	Algebraic Number Theory	Complexity Theory
Discrete Logarithms	Combinatorial Number Theory	Infeasibility Theory
Elliptic Curves	Analytic Number Theory	Computer Algorithms
Conjecture Verification	Arithmetic Algebraic Geometry	Computer Architectures
Theorem Proving	Probabilistic Number Theory	Quantum Computing
	Applied Number Theory	Biological Computing
$\vdots$	$\vdots$	$\vdots$

Basically, any topic in number theory where computation plays a central role can be regarded as a topic in computational number theory. Computational number theory aims at either using computing techniques to solve number-theoretic problems, or using number-theoretic techniques to solve computer science problems. We concentrate in this book on using computing techniques to solve number-theoretic problems that have connections and applications in

modern public-key cryptography. Typical questions or problems in this category of computational number theory include:

1. **Primality Testing Problem (PTP)**. PTP can be formally defined as follows:

$$\text{PTP} \stackrel{\text{def}}{=} \begin{cases} \text{Input :} & n > 1 \\ \text{Output :} & \begin{cases} \text{Yes} : n \in \text{Primes} \\ \text{No :} \ \ \text{Otherwise} \end{cases} \end{cases} \tag{1.16}$$

Theoretically speaking, PTP can be solved in polynomial-time, that is, PTP can be solved efficiently on a computer. However, it may still be difficult to decide whether or not a large number is prime. Call a number a Mersenne prime if it is of the form

$$M_p = 2^p - 1 \tag{1.17}$$

where p is prime and $2^p - 1$ is also prime. To date, only 47 such p have been found (see Table 1.4); the first 4 were found 2500 years ago. Note that $2^{43112609} - 1$ is not only the largest known Mersenne prime, but also the largest known prime in the world to date. The

Table 1.4 The 47 known Mersenne primes $M_p = 2^p - 1$

No	p	Digits(M_p)	Time	No	p	Digits(M_p)	Time
1	2	1	–	2	3	1	–
3	5	2	–	4	7	3	–
5	13	4	1461	6	17	6	1588
7	19	6	1588	8	31	10	1750
9	61	19	1883	10	89	27	1911
11	107	33	1913	12	127	39	1876
13	521	157	1952	14	607	183	1952
15	1279	386	1952	16	2203	664	1952
17	2281	687	1952	18	3217	969	1957
19	4253	1281	1961	20	4423	1332	1961
21	9689	2917	1963	22	9941	2993	1963
23	11213	3376	1963	24	19937	6002	1971
25	21701	6533	1978	26	23209	6987	1979
27	44497	13395	1979	28	86243	25962	1982
29	110503	33265	1988	30	132049	39751	1983
31	216091	65050	1985	32	756839	227832	1992
33	859433	258716	1994	34	1257787	378632	1996
35	1398269	420921	1996	36	2976221	895932	1997
37	3021377	909526	1998	38	6972593	2098960	1999
39	13466917	4053946	2001	40	20996011	6320430	2003
41	24036583	7235733	2004	42	25964951	7816230	2005
43	30402457	9152052	2005	44	32582657	9808358	2006
45	37156667	11185272	2008	46	42643801	12837064	2009
47	43112609	12978189	2008	48	?	?	?

search for the largest Mersenne prime and/or the largest prime has always been a hot topic in computational number theory. EFF (Electronic Frontier Foundation) has offered in total 550 000 US dollars to the first individual or organization who can find the following large primes:

Prizes	Conditions for the New Primes
\$50000	at least 1000000 digits
\$100000	at least 10000000 digits
\$150000	at least 100000000 digits
\$250000	at least 1000000000 digits

The first prize was claimed by Nayan Hajratwala in Michigan in 1996, who found the 38th Mersenne prime $2^{6972593} - 1$ with 2098960 digits, the second prize was claimed by Edson Smith at UCLA in 2008, who found the 46th Mersenne prime $2^{42643801} - 1$ with 12837064 digits. The remaining two prizes remain unclaimed. Of course, we still do not know if there are infinitely many Mersenne primes.

2. **Integer Factorization Problem (IFP)**. IFP can be formally defined as follows:

$$\text{IFP} \stackrel{\text{def}}{=} \begin{cases} \text{Input}: \quad n > 1 \\ \text{Output}: a \mid n, \quad 1 < a < n. \end{cases} \tag{1.18}$$

The IFP assumption is that given the positive integer $n > 1$, it is hard to find its nontrivial factor(s), that is,

$$\{n = ab\} \xrightarrow{\text{hard}} \{a,\ 1 < a < n\}.$$

Note that in IFP, we aim at finding just one nontrivial factor a (not necessarily a prime factor) of n. The Fundamental Theorem of Arithmetic asserts that any positive integer $n > 1$ can be uniquely written into the following standard prime factorization form:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}, \tag{1.19}$$

where $p_1 < p_2 < \cdots < p_k$ are primes, and $\alpha_1, \alpha_2, \cdots, \alpha_k$ are positive integers. Clearly, recursively performing the operations of primality testing and integer factorization, n can be eventually written in its standard prime factorization form, say, if we wish to factor 123457913315, the recursive process can be shown in Figure 1.6. So, if we define the Prime Factorization Problem (PFP) as follows:

$$\text{PFP} \stackrel{\text{def}}{=} \begin{cases} \text{Input}: \quad n > 1 \\ \text{Output}: p_1^{\alpha_1}, p_2^{\alpha_2}, \cdots, p_k^{\alpha_k} \end{cases} \tag{1.20}$$

then

$$\text{PFP} \stackrel{\text{def}}{=} \text{PTP} \oplus \text{IFP}.$$

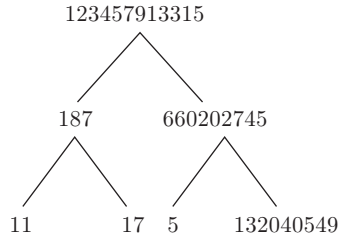


Figure 1.6 Prime factorization of 123457913315

Although PTP can be solved efficiently in polynomial-time, IFP cannot be solved in polynomial-time. Finding polynomial-time factoring algorithms is one of the most important research topics in computational number theory. At present, no polynomial-time algorithm for factoring has been found and no one has yet proved that such an algorithm exists. The current world record for integer factorization is the RSA-768 (a number with 768 bits and 232 digits):

123018668453011775513049495838496272077285356959533479219732245
215172640050726365751874520219978646938995647494277406384592519
255732630345373154826850791702612214291346167042921431160222124
047927473779408066535141959745985690214341 3
=
334780716989568987860441698482126908177047949837137685689124 31
388982883793878002287614711652531743087737814467999489
×
367460436667995904282446337996279526322 7 915816434308764267603 2
283815739666511279 233373417143396810270092798736308917.

It was factored on 9 Dec 2009. The factoring process requires about 10^{20} operations and would need about 2000 years of computation on a single core 2.2 GHz AMD Opteron.

3. **Discrete Logarithm Problem (DLP)**. According to historical records, logarithms over the set of real numbers $\mathbb{R}$ were first invented in the 16th century by the Scottish mathematician John Napier (1550–1617). We define k to be the logarithm to the base x of y

$$k = \log_x y \tag{1.21}$$

if and only if

$$x^k = y. \tag{1.22}$$

So the Logarithm Problem (LP) over $\mathbb{R}$ may be defined as follows:

$$\text{LP} \stackrel{\text{def}}{=} \begin{cases} \text{Input}: \quad x, y \\ \text{Output}: k \text{ such that } y = x^k. \end{cases} \tag{1.23}$$

For example, $\log_3 19683 = 9$, since $3^9 = 19683$. LP over $\mathbb{R}$ is easy to solve, since

$$\log_x y = \frac{\ln y}{\ln x} \tag{1.24}$$

where the natural logarithms can be calculated efficiently by the following formula (of course, depending on their accuracy):

$$\ln x = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^n}{n}. \tag{1.25}$$

For example,

$$\log_2 5 = \frac{\ln 5}{\ln 2} \approx \frac{1.609437912}{0.692147106} \approx 2.321928095.$$

We can always get a result at certain level of accuracy. The Discrete Logarithm Problem over the multiplicative group $\mathbb{Z}_n^*$, discussed in this book, is completely different from the traditional one we just discussed. Let

$$\mathbb{Z}_n^* = \{a : 1 \leq a \leq n, \gcd(a, n) = 1\}, x \in \mathbb{Z}_n^*. \tag{1.26}$$

DLP may be defined as follows:

$$\text{DLP} \stackrel{\text{def}}{=} \begin{cases} \text{Input}: \quad x, n, y \\ \text{Output}: k \text{ such that } y \equiv x^k \pmod{n}. \end{cases} \tag{1.27}$$

The DLP assumption is that

$$\{x, n, y \equiv x^k \pmod{n}\} \stackrel{\text{hard}}{\longrightarrow} \{k\}.$$

The following are some small and simple examples of DLP:

$$\begin{gathered} \log_3 57 \equiv k \pmod{1009} \Longrightarrow k \text{ does not exist;} \\ \log_{11} 57 \equiv k \pmod{1009} \Longrightarrow k = 375; \\ \log_3 20 \equiv k \pmod{1009} \Longrightarrow k = \{165, 333, 501, 669, 837, 1005\}. \end{gathered}$$

As can be seen, in the first example, the required discrete logarithm does not exist, whereas in the last example, the required discrete logarithms are not unique. In what follows, we

give a somewhat larger example of DLP: Let

$$p = (739 \cdot 7^{149} - 736)/3,$$
$$7^a \equiv 12740218011997394682426924433432284974938204258693162165455773529032291467909599868186097881304659516645545814428058807676603378 1 \pmod{p},$$
$$7^b \equiv 18016228528745310244478283483679989501596704669534669731302512173405995377205847595817691062538069210165184866236213793402680304 9 \pmod{p}.$$

Find 7^{ab}. To compute 7^{ab}, we need either to find a from $7^a \bmod p$ or b from $7^b \bmod p$, so that we can calculate $7^{ab} = (7^a)^b = (7^b)^a$. This problem was proposed by McCurley in 1990 and solved by Weber in 1998. The answer to 7^{ab} is:

$$7^{ab} = 38127280411190014138078391507929634193998643551018670285056375615045523966929403922102172514053270928872639426370063532797740808.$$

It is obtained by first finding

$$a = 61858690859651883273593331652037904267987643069521713459146222184952599815614487782075749218290977740833879185045794674973 4 \pmod{p-1}.$$

4. **Elliptic Curve Discrete Logarithm Problem (ECDLP)**. Elliptic Curve Discrete Logarithm Problem (ECDLP) is a very natural generalization of the Discrete Logarithm Problem (DLP) from multiplication group $\mathbb{Z}_n^*$ to the elliptic curve groups $E(\mathbb{Q})$, $E(\mathbb{Z}_n)$, or $E(\mathbb{F}_p)$. Let E be an elliptic curve

$$E : y^2 = x^3 + ax + b \tag{1.28}$$

over a field $\mathcal{K}$, denoted by $E\backslash\mathcal{K}$. A straight line (nonvertical) L connecting points P and Q intersects the elliptic curve E at a third point R, and the point $P \oplus Q$ is the reflection of R in the X-axis. That is, if $R = (x_3, y_3)$, then $P \oplus Q = (x_3, -y_3)$ is the reflection of R in the X-axis. Note that a vertical line, such as L' or L'', meets the curve at two points (not necessarily distinct), and also at the point at infinity $\mathcal{O}_E$ (we may think of the point at infinity as lying far off in the direction of the Y-axis). The line at infinity meets the curve at the point $\mathcal{O}_E$ three times. Of course, the nonvertical line meets the curve at three points in the XY plane. Thus, every line meets the curve at three points. The algebraic formula for computing $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$ on E is as follows:

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2,\ \lambda(x_1 - x_3) - y_1), \tag{1.29}$$

where

$$\lambda = \begin{cases} \dfrac{3x_1^2 + a}{2y_1} & \text{if } P_1 = P_2 \\ \dfrac{y_2 - y_1}{x_2 - x_1} & \text{otherwise.} \end{cases} \tag{1.30}$$

Given E and $P \in E$, it is easy to find $Q = kP$, which is of course also in E. For example, to compute $Q = 105P$, we first let

$$k = 105 = (1101001)_2$$

and then perform the operations as follows:

$$\begin{aligned}
&1\text{: } Q \leftarrow P + 2Q \Rightarrow Q \leftarrow P && \Rightarrow Q = P\\
&1\text{: } Q \leftarrow P + 2Q \Rightarrow c \leftarrow P + 2P && \Rightarrow Q = 3P\\
&0\text{: } Q \leftarrow 2Q \quad\ \Rightarrow Q \leftarrow 2(P + 2P) && \Rightarrow Q = 6P\\
&1\text{: } Q \leftarrow P + 2Q \Rightarrow Q \leftarrow P + 2(2(P + 2P)) && \Rightarrow Q = 13P\\
&0\text{: } Q \leftarrow 2Q \quad\ \Rightarrow Q \leftarrow 2(P + 2(2(P + 2P))) && \Rightarrow Q = 26P\\
&0\text{: } Q \leftarrow 2Q \quad\ \Rightarrow Q \leftarrow 2(2(P + 2(2(P + 2P)))) && \Rightarrow Q = 52P\\
&1\text{: } Q \leftarrow P + 2Q \Rightarrow Q \leftarrow P + 2(2(2(P + 2(2(P + 2P))))) && \Rightarrow Q = 105P.
\end{aligned}$$

This gives the required result $Q = P + 2(2(2(P + 2(2(P + 2P))))) = 105P$. As can be seen, given $(E\backslash\mathcal{K}, k, P)$ it is easy to compute

$$Q = kP. \tag{1.31}$$

However, it is hard to find k given $(E\backslash\mathcal{K}, P, Q)$. This is the Elliptic Curve Discrete Logarithm Problem (ECDLP), which may be formally defined as follows (let E be an elliptic curve over finite field $\mathbb{F}_p$):

$$\text{ECDLP} \stackrel{\text{def}}{=} \begin{cases} \text{Input}: \quad E\backslash\mathbb{F}_p, (P, Q) \in E(\mathbb{F}_p) \\ \text{Output}: k > 1 \text{ such that } Q \equiv kP \pmod{p}. \end{cases} \tag{1.32}$$

The ECDLP assumption asserts that

$$\{(P, Q \equiv kP \pmod{p}) \in E(\mathbb{F}_p)\} \xrightarrow{\text{hard}} \{k\}.$$

Suppose that we are given

$$(190, 271) \equiv k(1, 237) \pmod{1009}$$

with

$$E : y^2 \equiv x^3 + 71x + 602 \pmod{1009}$$

then it is easy to find

$$k = 419$$

since the finite field $\mathbb{F}_p$ is small. However, when the finite field is large, such as

$$Q(x_Q, y_Q) \equiv kP(x_P, y_P) \pmod{p}$$

Table 1.5 Some Certicom ECDLP challenge problems

Curves	Bits	Operations	Prizes (US Dollars)	Status
ECCp-97	97	$3.0 \cdot 10^{14}$	\$5000	1998
ECCp-109	109	$2.1 \cdot 10^{16}$	\$10 000	2002
ECCp-131	131	3.5×10^{19}	\$20 000	?
ECCp-163	163	2.4×10^{24}	\$30 000	?
ECCp-191	191	4.9×10^{28}	\$40 000	?
ECCp-239	239	8.2×10^{35}	\$50 000	?
ECC2p-359	359	9.6×10^{53}	\$100 000	?

on $E\backslash\mathbb{F}_p$, where

$p = 155003179783434785924857641481313994241 1,$
$a = 139926757376357881587790523597115331671 0,$
$b = 100929654219153246407626036752581629397 6,$
$x_P = 131795376323959588846552414558987269569 0,$
$y_P = 434829348619031278460656303481105428081,$
$x_Q = 124739221131790715130324772148964069924 0,$
$y_Q = 207534858442090452193999571026315995117.$

In this case, it is very hard to find k. Certicom Canada offered 20000 US dollars to the first individual or organization to find the correct value of k. More Certicom prize problems, along with this line, may be found in Table 1.5 (the above-mentioned \$20000 prize curve corresponds to ECCp-131, as p has 131-bits in this example):

5. **The Root Finding Problem (RFP)**. The k-th Root Finding Problem (RFP), or RFP for short, may be defined as follows:

$$k\text{RFP} \stackrel{\text{def}}{=} \{k, N, y \equiv x^k \pmod N\} \stackrel{\text{find}}{\longrightarrow} \{x \equiv \sqrt[k]{y} \pmod N\}. \tag{1.33}$$

If the prime factorization of N is known, one can compute the Euler function $\phi(N)$ and solve the linear Diophantine equation $ku - \phi(N)v = 1$ in u and v, and the computation $x \equiv y^u \pmod N$ gives the required value. Thus, if IFP can be solved in polynomial-time, then RFP can also be solved in polynomial-time:

$$\text{IFP} \stackrel{\mathcal{P}}{\Longrightarrow} \text{RFP}.$$

The security of RSA relies on the intractability of IFP, and also on RFP; if any one of the problems can be solved in polynomial-time, RSA can be broken in polynomial-time.

6. **The Square Root Problem (SQRT)** Let $y \in \text{QR}_N$, where QR_N denotes the set of quadratic residues modulo N, which should be introduced later. The SQRT Problem is to find an x such that

$$x^2 \equiv y \pmod N \text{ or } x \equiv \sqrt{y} \pmod N. \tag{1.34}$$

That is,

$$\text{SQRT} \stackrel{\text{def}}{=} \{N \in \mathbb{Z}^+_{>1}, y \in \text{QR}_N, y \equiv x^2 \pmod N\} \stackrel{\text{find}}{\longrightarrow} \{x\}. \tag{1.35}$$

When N is prime, the *SQRT Problem* can be solved in polynomial-time. However, when N is composite one needs to factor N first. Thus, if IFP can be solved in polynomial-time, SQRT can also be solved in polynomial-time:

$$\text{IFP} \overset{\mathcal{P}}{\Longrightarrow} \text{SQRT}.$$

On the other hand, if SQRT can be solved in polynomial-time, IFP can also be solved in polynomial-time:

$$\text{SQRT} \overset{\mathcal{P}}{\Longrightarrow} \text{IFP}.$$

That is,

$$\text{SQRT} \overset{\mathcal{P}}{\Longleftrightarrow} \text{IFP}.$$

It is precisely this intractability of SQRT Problem that Rabin used to construct his cryptosystem in 1979.

7. **Modular Polynomial Root Finding Problem (MPRFP)**. It is easy to compute the integer roots of a polynomial in one variable over $\mathbb{Z}$:

$$p(x) = 0 \tag{1.36}$$

but the following Modular Polynomial Root Finding Problem (MPRFP), or the MPRFP for short, can be hard:

$$p(x) \equiv 0 \pmod{N}, \tag{1.37}$$

which aims at finding integer roots (solutions) of the modular polynomial in one variable. This problem can, of course, be extended to find integer roots (solutions) of the modular polynomial in several variables as follows:

$$p(x, y, \cdots) \equiv 0 \pmod{N}. \tag{1.38}$$

Coppersmith, in 1997, developed a powerful method to find all small solutions x_0 of the modular polynomial equations in one or two variables of degree δ using the lattice reduction algorithm LLL (we shall discuss Coppersmith's method later). Of course, for LLL to be run in a reasonable amount of time for finding such x_0's, the values of δ cannot be large.

8. **The Quadratic Residuosity Problem (QRP)**. Let $N \in \mathbb{Z}^+_{>1}$, $\gcd(y, N) = 1$. Then y is a quadratic residue modulo N, denoted by $y \in \text{QR}_\text{N}$, if the quadratic congruence

$$x^2 \equiv y \pmod{N}, \tag{1.39}$$

has a solution in x. If the congruence has no solution in x, then y is a quadratic nonresidue modulo N, denoted by $y \in \overline{\text{QR}}_N$. The Quadratic Residuosity Problem (QRP), or *QRP* for

short, is to decide whether or not $y \in \mathrm{QR}_N$:

$$\mathrm{QRP} \stackrel{\text{def}}{=} \{n \in \mathbb{Z}^+_{>1}, x^2 \equiv y \pmod N\} \xrightarrow{\text{decide}} \{y \in \mathrm{QR}_N\}. \tag{1.40}$$

If N is prime, or the prime factorization of N is known, then QRP can be solved simply by evaluating the Legendre symbol $L(y, N)$. If n is not a prime then one evaluates the Jacobi symbol $J(y, N)$ which, unfortunately, does not reveal if $y \in \mathrm{QR}_N$, that is, $J(y, N) = 1$ does not imply $y \in \mathrm{QR}_N$ (it does if N is prime). For example, $L(15, 17) = 1$, so $x^2 \equiv 15 \pmod{17}$ is soluble, with $x = \pm 21$ being the two solutions. However, although $J(17, 21) = 1$ there is no solution for $x^2 \equiv 17 \pmod{21}$. Thus, when N is composite, the only way to decide whether or not $y \in \mathrm{QR}_N$ is to factor N. Thus, if IFP can be solved in polynomial-time, QRP can also be solved in polynomial-time:

$$\mathrm{IFP} \stackrel{\mathcal{P}}{\Longrightarrow} \mathrm{QRP}.$$

The security of the Goldwasser–Micali probabilistic encryption scheme is based on the intractability of QRP.

9. **Shortest Vector Problem (SVP)**. Problems related to lattices are also often hard to solve. Let $\mathbb{R}^n$ denote the space of n-dimensional real vectors $a = \{a_1, a_2, \cdots, a_n\}$ with usual dot product $a \cdot b$ and Euclidean Norm or length $||a|| = (a \cdot a)^{1/2}$. $\mathbb{Z}^n$ is the set of vectors in $\mathbb{R}^n$ with integer coordinates. If $A = \{a_1, a_2, \cdots, a_n\}$ is a set of linear independent vectors in $\mathbb{R}^n$, then the set of vectors

$$\left\{ \sum_{i=1}^{n} k_i a_i : k_1, k_2, \cdots, k_n \in \mathbb{Z} \right\} \tag{1.41}$$

is a lattice in $\mathbb{R}^n$, denoted by $L(A)$ or $L(a_1, a_2, \cdots, a_n)$. A is called a basis of the lattice. A set of vectors in $\mathbb{R}^n$ is a n-dimensional lattice if there is a basis V of n linear independent vectors such that $L = L(V)$. If $A = \{a_1, a_2, \cdots, a_n\}$ is a set of vectors in a lattice L, then the length of the set A is defined by $\max(||a_i||)$. A fundamental theorem, due to Minkowski, is as follows.

Theorem 1.1 (Minkowski) There is a universal constant γ, such that for any lattice L of dimension n, $\exists v \in L$, $v \neq 0$, such that

$$||v|| = \gamma \sqrt{n} \det(L)^{1/n}. \tag{1.42}$$

The determinant $\det(L)$ of a lattice is the volume of the n-dimensional fundamental parallelepiped, and the absolute constant γ is known as Hermite's constant.

A natural problem concerned with lattices is the *Shortest Vector Problem (SVP)*, or the *SVP* for short:

> Find the shortest nonzero vector in a high dimensional lattice.

Minkowski's theorem is just an existence-type theorem and offers no clue on how to find a short or the shortest nonzero vector in a high dimensional lattice. There is no efficient algorithm for finding the shortest nonzero vector, or finding an approximate short nonzero vector. The lattice reduction algorithm LLL can be used to find short vectors, but it is not effective in finding short vectors when the dimension n is large, say, for example, $n \geq 100$. This allows lattices to be used in the design of cryptographic systems and in fact, several cryptographic systems, such as NTRU and the Ajtai–Dwork system, are based on the intractability of finding the shortest nonzero vector in a high dimensional lattice.

In this book, we are more interested in those number-theoretic problems that are computationally intractable, since the security of modern public-key cryptography relies on the intractability of these problems. A problem is computationally intractable if it cannot be solved in polynomial-time. Thus, from a computational complexity point of view, any problem beyond $\mathcal{P}$ is intractable. There are, however, different types of intractable problems (see Figure 1.7).

(1) Provable intractable problems: Problems that are Turing computable but can be shown in $\mathcal{PS}$ ($\mathcal{P}$-$\mathcal{S}$pace), $\mathcal{NPS}$ ($\mathcal{NP}$-$\mathcal{S}$pace), $\mathcal{EXP}$ (exponential time) and so on, of course outside $\mathcal{NP}$, are provably and certainly intractable. Note that although we do not know if $\mathcal{P} = \mathcal{NPS}$, we know $\mathcal{PS} = \mathcal{NPS}$.

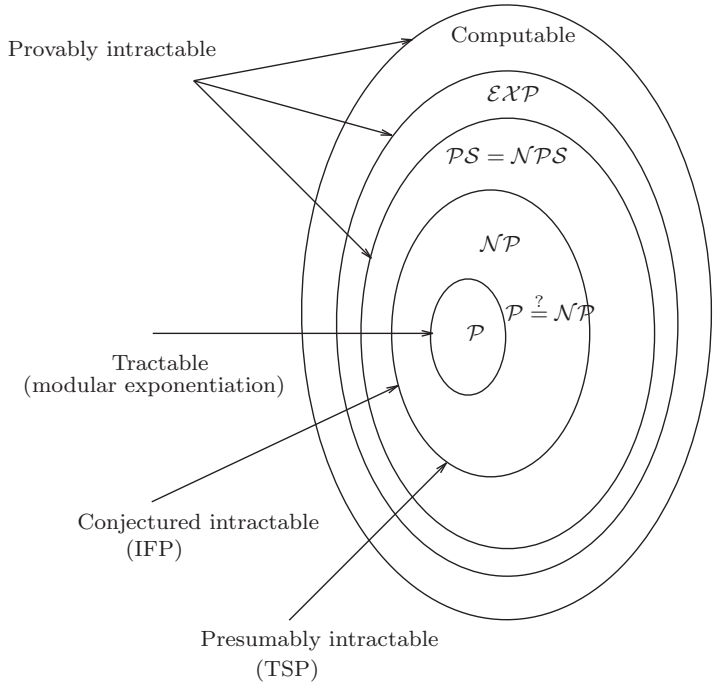


Figure 1.7 Tractable and intractable problems

(2) Presumably intractable problems: Problems in $\mathcal{NP}$ but outside of $\mathcal{P}$, particularly those problems in $\mathcal{NPC}$ ($\mathcal{NP}$-complete) such as the Traveling Salesman Problem, the Knapsack Problem, and the Satisfiability Problem, are presumably intractable, since we do not know whether or not $\mathcal{P} = \mathcal{NP}$. If $\mathcal{P} = \mathcal{NP}$, then all problems in $\mathcal{NP}$ will no longer be intractable. However, it is more likely that $\mathcal{P} \neq \mathcal{NP}$. From a cryptographic point of view, it would be nice if encryption schemes could be designed to be based on some $\mathcal{NP}$-complete problems, since these types of schemes can be difficult to break. Experience, however, tells us that very few encryption schemes are based on $\mathcal{NP}$-complete problems.
(3) Conjectured intractable problems: By conjectured intractable problems we mean that the problems are currently in $\mathcal{NP}$-complete, but no-one can prove they must be in $\mathcal{NP}$-complete; they may be in $\mathcal{P}$ if efficient algorithms are invented for solving these problems. Typical problems in this category include the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP). Again, from a cryptographic point of view, we are more interested in this type of intractable problem and, in fact, the IFP, DLP, and ECDLP are essentially the only three intractable problems that are practical and widely used in commercial cryptography. For example, the most famous and widely used RSA cryptographic system relies for its security on the intractability of the IFP problem.

The difference between the presumably intractable problems and the conjectured intractable problems is important and should not be confused. For example, both TSP and IFP are intractable, but the difference between TSP and IFP is that TSP has been proved to be $\mathcal{NP}$-complete whereas IFP is only *conjectured* to be $\mathcal{NP}$-complete. IFP may be $\mathcal{NP}$-complete, but also may not be $\mathcal{NP}$-complete.

Finally, we present a complexity measure of number-theoretic problems in big-$\mathcal{O}$ notation.

Definition 1.12 Let

$$f, g : \mathbb{Z}^+ \to \mathbb{R}. \tag{1.43}$$

Define

$$f = \mathcal{O}(g) \tag{1.44}$$

if there exists $c \in \mathbb{R}_{>0}$ with

$$|f(n)| \leq cg(n), \quad \text{for all } n. \tag{1.45}$$

Definition 1.13 Let

$$L_n(\alpha, c) = \exp(c(\log n)^{\alpha}(\log\log n)^{1-\alpha}), \tag{1.46}$$

where $\alpha \in [0, 1]$, $c \in \mathbb{R}_{>0}$.

(1) If a problem can be solved by an algorithm in expected running time

$$T(n) = \mathcal{O}(L_n(0, c)), \tag{1.47}$$

then the algorithm is polynomial-time algorithm (or efficient algorithm), and the corresponding problem is an easy problem (i.e., the problem can be solved easily).

It is also often the case that $\mathcal{O}((\log n)^k)$ is used with k constant to represent polynomial-time complexity. For example, the multiplication of two $\log n$-bit numbers by ordinary method takes time $\mathcal{O}((\log n)^2)$, the fastest known multiplication method has a running time of

$$\mathcal{O}(\log n \log\log n \log\log\log n) = \mathcal{O}((\log n)^{1+\epsilon}).$$

(2) If a problem can be solved by an algorithm in expected running time

$$T(n) = \mathcal{O}(L_n(1, c)), \tag{1.48}$$

then the algorithm is an exponential-time algorithm (or an inefficient algorithm), and the corresponding problem is a hard problem (i.e., the problem is hard to solve).

Note that since $\log n$ is the length of input, $\mathcal{O}((\log n)^{12})$ is polynomial-time complexity, whereas $\mathcal{O}((n)^{0.1})$ is not, since $\mathcal{O}((n)^{0.1}) = \mathcal{O}(2^{0.1 \log n})$, an exponential complexity.

(3) An algorithm is of subexponential-time complexity if

$$T(n) = \mathcal{O}(L_n(\alpha, c)), \quad 0 < \alpha < 1. \tag{1.49}$$

Subexponential-time complexity is an important and interesting class between the two extremes, and in fact, many of the number theoretic algorithms discussed in this book, such as the algorithms for integer factorization and discrete logarithms, fall into this special class, which is slower than polynomial-time but faster than exponential-time.

Subexponential-time complexity is an important complexity class in computational number theory and, in fact, the best algorithms for IFP and DLP run in subexponential-time. For ECDLP, we even do not have a subexponential-time algorithm.

Problems for Section 1.3

1. Prove or disprove that
   (1) there are infinitely many Mersenne prime numbers;
   (2) there are infinitely many Mersenne composite numbers.
   Find the 48th Mersenne prime.
2. What is the difference between the Integer Factorization Problem and the Prime Factorization Problem?
3. What is the difference between the Discrete Logarithm Problem and the Elliptic Discrete Logarithm Problem.
4. Show that solving the Square Root Problem is equivalent to that of the Integer Factorization Problem.

5. Show that solving the Quadratic Residuosity Problem is equivalent to that of the Integer Factorization Problem.
6. Find all the prime factors of the following numbers:
   (1) 11111111111 (the number consisting of eleven 1)
   (2) 111111111111 (the number consisting of twelve 1)
   (3) 1111111111111 (the number consisting of thirteen 1)
   (4) 11111111111111 (the number consisting of fourteen 1)
   (5) 111111111111111 (the number consisting of fifteen 1)
   (6) 1111111111111111 (the number consisting of sixteen 1)
   (7) 11111111111111111 (the number consisting of seventeen 1)
   (8) Can you find any pattern for the prime factorization of the above numbers?
7. Do you think the Integer Factorization Problem, or more generally the Prime Factorization Problem, are hard to solve? Justify your answer.
8. Can you find some problems that have similar properties or difficulties to the Integer Factorization Problem (we shall explain this in detail in the next section)?
9. Find the discrete logarithm k

$$k \equiv \log_2 3 \pmod{11}$$

such that

$$2^k \equiv 3 \pmod{11},$$

and the discrete logarithm k

$$k \equiv \log_{123456789} 962 \pmod{9876543211}$$

such that

$$123456789^k \equiv 962 \pmod{9876543211}.$$

10. Find the square root y

$$y \equiv \sqrt{3} \pmod{11}$$

such that

$$y^2 \equiv 3 \pmod{11},$$

and the square root y

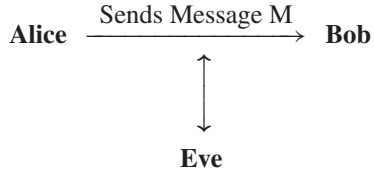
$$y \equiv \sqrt{123456789} \pmod{987654321}$$

such that

$$y^2 \equiv 123456789 \pmod{987654321}.$$

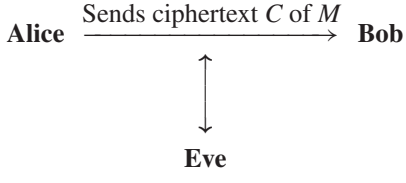
1.4 What is Modern Cryptography?

Cryptography, one of the main topics of this book, is the art and science of secure data communications over insecure channels. It is a very old subject, as old as our human civilization. The basic scenario of cryptography is that Alice wishes to send a message M to Bob over the insecure public channel, but Eve can eavesdrop on the communications from the public channel:



(Eve can easily get M)

To stop Eve to reading/understanding the message M (note that no one can stop Even eavesdroping M), Alice first encrypts the plaintext M to ciphertext C, and then sends C to Bob:



(Eve cannot easily recover M from C)

As we just mentioned, cryptography is an old subject, and in fact it has at least 5000 years of history, however, in this book we are more interested in *modern cryptography*. By modern cryptography, we mean the cryptography studied and invented mainly after the 1970s. Often these types of cryptography are based on advanced and sophisticated mathematics, so we call it mathematical cryptography. More specifically, we call it *number-theoretic cryptography* if its construction and security are based on the concepts and results in number theory. Of course, modern cryptography may also be based on, say for example, quantum physics and molecular biology, in which case, we may call it *quantum cryptography*, or *biological (DNA) cryptography*. Traditionally, cryptography is meant to be *secret-key cryptography*, in which the *encryption* and *decryption* use the same key. By the same key, we mean the encryption key, say, e and the decryption key, say d are polynomial-time computable. That is, given e, $d = 1/e$ can be computed easily in polynomial-time. In other words, e and d are *polynomial-time equivalent* but not physically equivalent. In public-key cryptography, however, e and d are different, as given e, $d = 1/e$ *cannot* be computed in polynomial-time. Of course, they can be computed in exponential-time. So, in public-key cryptography, e and d are *not* polynomial-time equivalent. Other significant difference between secret-key cryptography and public-key cryptography is that public-key cryptography is normally not

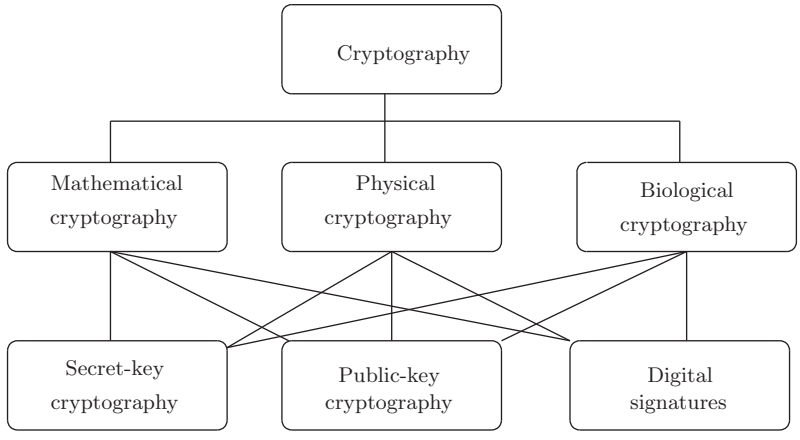


Figure 1.8 Types of cryptography

only useful for encryption, but is also useful for *digital signatures*. Figure 1.8 shows the types of cryptography and the relationships among the different types of cryptography.

Now let us take RSA as an example to illustrate the classification of different type of cryptography. First of all, RSA is a type of mathematical cryptography, more specifically it is a type of number-theoretic cryptography, as its construction and security are all based on the infeasible number-theoretic problem – the Integer Factorization Problem. Secondly, RSA is public-key cryptography and, in fact, it is the first practical, widely used, and still unbreakable public-key cryptography and was invented in 1977 by Rivet, Shamir, and Adleman, then all at MIT. Let M be a plaintext message. To encrypt the M, one computes

$$C \equiv M^e \pmod{n}, \tag{1.50}$$

where e is the encryption key, and both e and n are *public*. (The notation $a \equiv b \pmod{n}$ reads "a is congruent to b modulo n." Congruences will be studied in detail in Section 2.4.) To decrypt the encrypted message C, one computes

$$M \equiv C^d \pmod{n}, \tag{1.51}$$

where d is the *private* decryption key satisfying

$$ed \equiv 1 \pmod{\phi(n)}, \tag{1.52}$$

where $\phi(n)$ is Euler's ϕ-function ($\phi(n)$, for $n \geq 1$, is defined to be the number of positive integers not exceeding n which are relatively prime to n; see Definition 2.31). By (1.52), we have $ed = 1 + k\phi(n)$ for some integer k. By Euler's theorem (see Theorem 2.112), $M^{\phi(n)} \equiv 1 \pmod{n}$, we have $M^{k\phi(n)} \equiv 1 \pmod{n}$. Thus,

$$C^d \equiv M^{ed} \equiv M^{1+k\phi(n)} \equiv M \pmod{n}. \tag{1.53}$$

For those who do not have the private key but can factor n, say for example, $n = pq$, they can find d by computing

$$d \equiv e^{-1} \pmod{\phi(n)} \equiv e^{-1} \pmod{(p-1)(q-1)}, \tag{1.54}$$

and hence, decrypt the message.

Remarkably enough, RSA can also be easily used to obtain digital signatures: Just use the encryption key e and decryption key d in the opposite direction:

1. Signature generation;

$$S \equiv M^d \pmod{n}, \tag{1.55}$$

   where S is the digital signature, which can only be generated by the one who has the private key d.
2. Signature verification;

$$M \equiv S^e \pmod{n}, \tag{1.56}$$

   Anyone can easily verify the signature S, as the verification key is the public key e.

Problems for Section 1.4

1. Why can secret-key cryptography not be used for obtaining digital signatures?
2. Can all the public-key cryptographic schemes be used for obtaining digital signatures?
3. In term of computational complexity, explain why d cannot be computed from e in polynomial-time, given $ed \equiv 1 \pmod{\phi(n)}$ where $n = pq$ with p, q prime numbers.
4. Explain why the security of RSA relies on the infeasibity of the prime factorization of the modulus n.
5. Explain why RSA is only computationally unbreakable, not absolutely unbreakable.
6. Can the RSA encryption exponent e be 2? Justify your answer.
7. Modify RSA to allow $e = 2$.
8. Is RSA is unbreakable? Justify your answer.
9. Although the cryptographic schemes discussed this book are mainly based on the hard number-theoretic probems, there are some other cryptographic schemes whose security is based on some other difficult problems. Write essays on
   (1) DNA-based cryptography
   (2) chaos-based cryptography
   (3) optical cryptography
   (4) quantum cryptography.

1.5 Bibliographic Notes and Further Reading

In this chapter, we have given a general picture of number theory, computation theory, computational number theory, and modern number-theoretic cryptography, each of them in their own right large, well-established, and important subjects.

For more information on number theory, readers may consult [1–3, 5–12, 14–23]. For more information on computation theory, particularly computability, complexity, and infeasibiity, see e.g., [24–39]. For more information on computational number theory, see for example, [13, 14, 23, 27, 38, 40–50]. For more information on cryptography, it is suggested readers consult [51–66].

References

1. T. M. Apostol, *Introduction to Analytic Number Theory*, Corrected 5th Printing, Undergraduate Texts in Mathematics, Springer, 1998.
2. A. Baker, *A Concise Introduction to the Theory of Numbers*, Cambridge University Press, 1984.
3. E. Bombieri, *The Riemann Hypothesis*, In: J. Carlson, A. Jaffe and A. Wiles, Editors, *The Millennium Prize Problems*, Clay Mathematics Institute and American Mathematical Society, 2006, pp. 107–128.
4. J. Carlson, A. Jaffe and A. Wiles, *The Millennium Prize Problems*, American Mathematical Society, 2006.
5. J. R. Chen, "On the representation of a larger even integer as the sum of a prime and the product of at most two primes", *Scientia Sinica* **16**, 1973, pp. 157–176.
6. H. Davenport, *The Higher Arithmetic*, 7th Edition, Cambridge University Press, 1999.
7. U. Dudley, *A Guide to Elementary Number Theory*, Mathematical Association of America, 2010.
8. H. M. Edwards, *Higher Arithmetic: Al Algorithmic Introduction to Number Theory*, American Mathematical Society, 2008.
9. B. Green and T. Tao, "The Primes Contain Arbitrarily Long Arithmetic Progressions", *Annual of Mathematics*, **167**, 2, 2008, pp. 481–547.
10. G. H. Hardy and E. M. Wright, *An Introduction to Theory of Numbers*, 6th Edition, Oxford University Press, 2008.
11. D. Husemöller, *Elliptic Curves*, Graduate Texts in Mathematics **111**, Springer, 1987.
12. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd Edition, Graduate Texts in Mathematics **84**, Springer, 1990.
13. D. E. Knuth, *The Art of Computer Programming II – Seminumerical Algorithms*, 3rd Edition, Addison-Wesley, 1998.
14. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer, 1994.
15. W. J. LeVeque, *Fundamentals of Number Theory*, Dover, 1977.
16. I. Niven, H. S. Zuckerman and H. L. Montgomery, *An Introduction to the Theory of Numbers*, 5th Edition, Wiley, 1991.
17. J. H. Silverman, *A Friendly Introduction to Number Theory*, 3rd Edition, Prentice-Hall, 2006.
18. J. H. Silverman, *The Arithmetic of Elliptic Curves*, 2nd Edition, Graduate Texts in Mathematics **106**, Springer, 2009.
19. J. H. Silverman and J. Tate, *Rational Points on Elliptic Curves*, Undergraduate Texts in Mathematics, Springer, 1992.
20. J. Stillwell, *Elements of Number Theory*, Springer, 2000.
21. L. C. Washinton, *Elliptic Curve: Number Theory and Cryptography*, 2nd Edition, CRC Press, 2008.
22. A. Wiles, "Modular Elliptic Curves and Fermat's last Theorem", *Annals of Mathematics*, **141**, 1994, pp. 443–551.
23. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer, 2002.
24. S. Arora and B. Barak, *Computational Complexity*, Cambridge University Press, 2009.

25. S. Cook, *The Complexity of Theorem-Proving Procedures*, *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing*, New York, 1971, pp. 151–158.
26. S. Cook, *The P versus NP Problem*, In: J. Carlson, A. Jaffe, and A. Wiles, Editors, *The Millennium Prize Problems*, Clay Mathematics Institute and American Mathematical Society, 2006, pp. 87–104.
27. T. H. Cormen, C. E. Ceiserson, and R. L. Rivest, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009.
28. M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
29. J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Theory of Computation*, Jones and Bartlett Publishers, 2008.
30. J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Edition, Addison-Wesley, 2007.
31. P. Linz, *An Introduction to Formal Languages and Automata*, 3rd Edition, Jones and Bartlett Publishers, 2000.
32. R. Karp, "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*. Edited by R. E. Miller and J. W. Thatcher, Plenum Press, New York, 1972, pp. 85–103.
33. C. Petzold, *The Annotated Turing: A Guide Tour through Alan Turing's Historical Paper on Computability and the Turing Machine*, Wiley, 2008.
34. G. Rozenberg and A. Salomaa, *Cornerstones of Undecidability*, Prentice-Hall, 1994.
35. R. Sedgewick and K. Wayne, *Algorithms*, 4th Edition, Cambridge University Press, 2011.
36. M. Sipser, *Introduction to the Theory of Computation*, 2nd Edition, Thomson, 2006.
37. A. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, Series 2 **42**, 1937, pp. 230–265 and **43**, 1937, pp. 544–546.
38. H. S. Wilf, *Algorithms and Complexity*, 2nd Edition, A. K. Peters, 2002.
39. S. Y. Yan, *An Introduction to Formal Languages and Machine Computation*, World-Scientific, 1998.
40. L. M. Adleman, "Algorithmic Number Theory – The Complexity Contribution", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1994, pp. 88–113.
41. E. Bach and J. Shallit, *Algorithmic Number Theory I – Efficient Algorithms*, MIT Press, 1996.
42. H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer, 1993.
43. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer, 2005.
44. M. F. Jones, M. Lal, and W. J. Blundon, "Statistics on Certain Large Primes", *Mathematics of Computation*, **21**, 97, 1967, pp. 103–107.
45. E. Kranakis, *Primality and Cryptography*, Wiley, 1986.
46. C. Pomerance, "Computational Number Theory", *Princeton Companion to Mathematics*. Edited by W. T. Gowers, Princeton University Press, 2008, pp. 348–362.
47. H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, Boston, 1990.
48. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
49. R. D. Silverman, "A Perspective on Computational Number Theory", *Notices of the American Mathematical Society*, **38**, 6, 1991, pp. 562–568.
50. J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 2nd Edition, Cambridge University Press, 2003.
51. M. Ajtai and C. Dwork, "A Pubic-Key Cryptosystem with Worst-Case/Average-Case Equivalence", *Proceedings of Annual ACM Symposium on Theory of Computing*, 1997, pp. 284–293.
52. F. L. Bauer, *Decrypted Secrets – Methods and Maxims of Cryptology*, 3rd Edition, Springer, 2002.
53. J.A. Buchmann, *Introduction to Cryptography*, Springer, 2004.
54. H. Cohen and G. Frey, *Elliptic and Hyperelliptic Curve Cryptography*, Chapman & Hall/CRC, 2006.
55. N. Ferguson, B. Schneier and T. Kohno, *Cryptography Engineering*, Wiley, 2010.
56. S. Goldwasser and S. Micali, "Probabilistic Encryption", *Journal of Computer and System Sciences*, **28**, 1984, pp. 270–299.
57. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman and W. Whyte, "NTRUEncrypt and NTRUSign: Efficient Public Key Algorithmd for a Post-Quantum World", *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto 2006)*, 23–26 May 2006, pp. 71–77.
58. N. Koblitz, *Algebraic Aspects of Cryptography*, Springer, 1998.
59. A. J. Menezes, P.C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
60. *Codes: The Guide to Secrecy from Ancient to Modern Times*, CRC Press, 2005.

61. J. Pieprzyk, Hardjono and J. Seberry, *Fundamentals of Computer Security*, Springer, 2003.
62. J. Rothe, *Complexity Theory and Cryptology*, Springer, 2005.
63. I. Shparlinski, *Number Theoretic Methods in Cryptography*, Birkhäuser, 1999.
64. I. Shparlinski, *Cryptographic Applications of Analytic Number Theory*, Birkhäuser, 2003.
65. M. Stamp and R. M. Low, *Applied Cryptanaysis*, IEEE Press and Wiley, 2007.
66. A. L. Young, *Mathematical Ciphers*, American Mathematical Society, 2006.

2

Fundamentals

In this chapter, we provide some basic concepts and results in elementary number theory that shall be used throughout the book, including

- Basic algebraic structures
- Divisibility theory
- Congruence theory
- Theory of elliptic curves

2.1 Basic Algebraic Structures

Definition 2.1 A *group*, denoted by G, is a nonempty set G of elements together with a binary operation $*$ (e.g., the ordinary addition or multiplication), such that the following axioms are satisfied:

(1) Closure: $a * b \in G, \quad \forall a, b \in G$.
(2) Associativity: $(a * b) * c = a * (b * c), \quad \forall a, b, c \in G$.
(3) Existence of identity: There is a unique element $e \in G$, called the identity, such that $e * a = a * e = a, \quad \forall a \in G$.
(4) Existence of inverse: For every $a \in G$ there is a unique element b such that $a * b = b * a = e$. This b is denoted by a^{-1} and called the inverse of a.

The group G is called a *commutative group* if it satisfies a further axiom:

(5) Commutativity: $a * b = b * a, \quad \forall a, b \in G$.

A commutative group is also called an *Abelian group*, in honor of the Norwegian mathematician N. H. Abel (1802–1829).

Example 2.1 The set of all positive integers (also called natural numbers), $\mathbb{Z}^+$, with operation $+$ is *not* a group, since there is no identity element for $+$ in $\mathbb{Z}^+$. The set $\mathbb{Z}^+$ with operation $\cdot$ is *not* a group; there is an identity element 1, but no inverse of 3.

Example 2.2 The set of all non-negative integers, $\mathbb{Z}_{\geq 0}$, with operation $+$ is *not* a group; there is an identity element 0, but no inverse for 2.

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

Example 2.3 The sets $\mathbb{Q}^+$ and $\mathbb{R}^+$ of positive numbers and the sets $\mathbb{Q}^*$, $\mathbb{R}^*$ and $\mathbb{C}^*$ of nonzero numbers with operation $\cdot$ are Abelian groups.

Definition 2.2 If the binary operation of a group is denoted by $+$, then the identity of a group is denoted by 0 and the inverse a by $-a$; this group is said to be an *additive group*. If the binary operation of a group is denoted by $*$, then the identity of a group is denoted by 1 or e; this group is said to be a *multiplicative group*.

Definition 2.3 A group is called a *finite group* if it has a finite number of elements; otherwise it is called an *infinite group*. The number of elements in G is called the order of G and is denoted by $|G|$ or $\#(G)$.

Example 2.4 The order of $\mathbb{Z}$ is infinite, that is, $|\mathbb{Z}| = \infty$. However, the order of $\mathbb{Z}_{11}$ is finite, since $|\mathbb{Z}_{11}| = 11$.

Definition 2.4 A nonempty set G' of a group G which is itself a group, under the same operation, is called a *subgroup* of G.

Definition 2.5 Let a be an element of a multiplicative group G. The elements a^r, where r is an integer, form a subgroup of G, called the subgroup generated by a. A group G is *cyclic* if there is an element $a \in G$ such that the subgroup generated by a is the whole of G. If G is a finite cyclic group with identity element e, the set of elements of G may be written $\{e, a, a^2, \ldots, a^{n-1}\}$, where $a^n = e$ and n is the smallest such positive integer. If G is an infinite cyclic group, the set of elements may be written $\{\ldots, a^{-2}, a^{-1}, e, a, a^2, \ldots\}$.

By making appropriate changes, a cyclic *additive group* can be defined. For example, the set $\{0, 1, 2, \ldots, n-1\}$ with addition modulo n is a cyclic group, and the set of all integers with addition is an infinite cyclic group.

Example 2.5 The congruences modulo n form a group. If we take $a + b \equiv c \pmod 6$, then we get the following complete addition table for the additive group modulo 6 (see Table 2.1):

Table 2.1 Additive group modulo 6

$\oplus$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Definition 2.6 A *ring*, denoted by $(R, \oplus, \odot)$, or simply R, is a set of at least two elements with *two* binary operations $\oplus$ and $\odot$, which we call addition and multiplication, defined on R such that the following axioms are satisfied:

(1) The set is *closed* under the operation $\oplus$:

$$a \oplus b \in R, \quad \forall a, b \in R, \tag{2.1}$$

(2) The associative law holds for $\oplus$:

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c, \quad \forall a, b, c \in R, \tag{2.2}$$

(3) The commutative law holds for $\oplus$:

$$a \oplus b = b \oplus a, \quad \forall a, b \in R, \tag{2.3}$$

(4) There is a special (zero) element $0 \in R$, called the additive identity of R, such that

$$a \oplus 0 = 0 \oplus a = a, \quad \forall a \in R, \tag{2.4}$$

(5) For each $a \in R$, there is a corresponding element $-a \in R$, called the additive inverse of a, such that:

$$a \oplus (-a) = 0, \quad \forall a \in R, \tag{2.5}$$

(6) The set is closed under the operation $\odot$:

$$a \odot b \in R, \quad \forall a, b \in R, \tag{2.6}$$

(7) The associative law holds for $\odot$:

$$a \odot (b \odot c) = (a \odot b) \odot c, \quad \forall a, b, c \in R, \tag{2.7}$$

(8) The operation $\odot$ is distributive with respect to $\oplus$:

$$a \odot (b \oplus c) = a \odot b \oplus a \odot c, \quad \forall a, b, c \in R, \tag{2.8}$$

$$(a \oplus b) \odot c = a \odot c \oplus b \odot c, \quad \forall a, b, c \in R. \tag{2.9}$$

From a group theoretic point of view, a ring is an Abelian group, with the additional properties that the closure, associative, and distributive laws hold for $\odot$.

Example 2.6 $(\mathbb{Z}, \oplus, \odot)$, $(\mathbb{Q}, \oplus, \odot)$, $(\mathbb{R}, \oplus, \odot)$, and $(\mathbb{C}, \oplus, \odot)$ are all rings.

Definition 2.7 A *commutative ring* is a ring that further satisfies:

$$a \odot b = b \odot a, \quad \forall a, b \in R. \tag{2.10}$$

Definition 2.8 A *ring with identity* is a ring that contains an element 1 satisfying:

$$a \odot 1 = a = 1 \odot a, \quad \forall a \in R. \tag{2.11}$$

Definition 2.9 An *integral domain* is a commutative ring with identity $1 \neq 0$ that satisfies:

$$a, b \in R \,\&\, ab = 0 \Longrightarrow a = 0 \,\text{or}\, b = 0. \tag{2.12}$$

Definition 2.10 A *division ring* is a ring R with identity $1 \neq 0$ that satisfies:

for each $a \neq 0 \in R$, the equations $ax = 1$ and $xa = 1$ have solutions in R.

Definition 2.11 A *field*, denoted by K, is a division ring with commutative multiplication.

Example 2.7 The integer set $\mathbb{Z}$, with the usual addition and multiplication, forms a commutative ring with identity, but is not a field.

It is clear that a field is a type of ring, which can be defined more generally as follows:

Definition 2.12 A *field*, denoted by $(K, \oplus, \odot)$, or simply K, is a set of at least two elements with *two* binary operations $\oplus$ and $\odot$, which we call addition and multiplication, defined on K such that the following axioms are satisfied:

(1) The set is *closed* under the operation $\oplus$:

$$a \oplus b \in K, \quad \forall a, b \in K, \tag{2.13}$$

(2) The associative law holds for $\oplus$:

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c, \quad \forall a, b, c \in K, \tag{2.14}$$

(3) The commutative law holds for $\oplus$:

$$a \oplus b = b \oplus a, \quad \forall a, b \in K, \tag{2.15}$$

(4) There is a special (zero) element $0 \in K$, called the additive identity of K, such that

$$a \oplus 0 = 0 \oplus a = a, \quad \forall a \in K, \tag{2.16}$$

(5) For each $a \in K$, there is a corresponding element $-a \in K$, called the additive inverse of a, such that:

$$a \oplus (-a) = 0, \quad \forall a \in K, \tag{2.17}$$

(6) The set is closed under the operation $\odot$:

$$a \odot b \in K, \quad \forall a, b \in K, \tag{2.18}$$

(7) The associative law holds for $\odot$:

$$a \odot (b \odot c) = (a \odot b) \odot c, \quad \forall a, b, c \in K, \tag{2.19}$$

(8) The operation $\odot$ is distributive with respect to $\oplus$:

$$a \odot (b \oplus c) = a \odot b \oplus a \odot c, \quad \forall a, b, c \in K, \tag{2.20}$$

$$(a \oplus b) \odot c = a \odot c \oplus b \odot c, \quad \forall a, b, c \in K. \tag{2.21}$$

(9) There is an element $1 \in K$, called the multiplicative identity of K, such that $1 \neq 0$ and

$$a \odot 1 = a, \quad \forall a \in K, \tag{2.22}$$

(10) For each nonzero element $a \in K$ there is a corresponding element $a^{-1} \in K$, called the multiplicative inverse of a, such that

$$a \odot a^{-1} = 1, \tag{2.23}$$

(11) The commutative law holds for $\odot$:

$$a \odot b = b \odot a, \quad \forall a, b \in K. \tag{2.24}$$

Again, from a group theoretic point of view, a field is an Abelian group with respect to addition and also the nonzero field elements form an Abelian group with respect to multiplication.

Remark 2.1 An alternative definition of a field is:

> If all the elements of a ring, other than the zero, form a commutative group under $\odot$, then it is called a *field*.

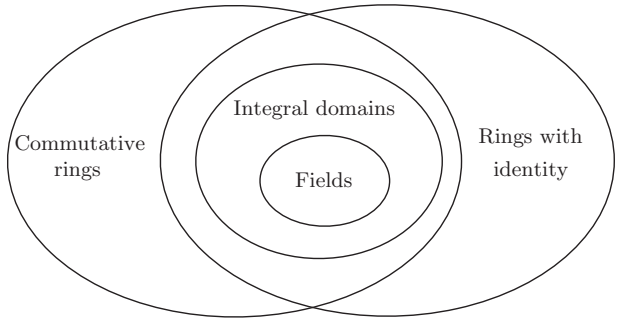


Figure 2.1 Containment of various rings

Example 2.8 The integer set $\mathbb{Z}$, with the usual addition and multiplication, forms a commutative ring with identity.

Figure 2.1 gives a Venn diagram view of containment for algebraic structures having two binary operations.

Example 2.9 Familiar examples of fields are the set of rational numbers, $\mathbb{Q}$, the set of real numbers, $\mathbb{R}$, and the set of complex numbers, $\mathbb{C}$; since $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ are all infinite sets, they are all infinite fields. The set of integers $\mathbb{Z}$ is a ring but *not* a field, since 2, for example, has no multiplicative inverse; 2 is not a unit in $\mathbb{Z}$. The only units (i.e., the invertible elements) in $\mathbb{Z}$ are 1 and -1. Another example of a ring which is not a field is the set $K[x]$ of polynomials in x with coefficients belonging to a field K.

Theorem 2.1 *$\mathbb{Z}/n\mathbb{Z}$ is a field if and only if n is prime.*

What this theorem says is that whenever n is prime, the set of congruence classes modulo n forms a field. This *prime field* $\mathbb{Z}/p\mathbb{Z}$ will be specifically denoted by $\mathbb{F}_p$.

Definition 2.13 A *finite field* is a field that has a finite number of elements in it; we call the number the *order* of the field.

The following fundamental result on finite fields was first proved by Évariste Galois (1811–1832):

Theorem 2.2 *There exists a field of order q if and only if q is a prime power (i.e., $q = p^r$) with p prime and $r \in \mathbb{N}$. Moreover, if q is a prime power, then there is, up to relabelling, only one field of that order.*

A finite field of order q with q a prime power is often called a *Galois field*, and is denoted by GF(q), or just $\mathbb{F}_q$.

Example 2.10 The finite field $\mathbb{F}_5$ has elements $\{0, 1, 2, 3, 4\}$ and is described by the following addition and multiplication tables (see Table 2.2):

Table 2.2 Addition and multiplication for $\mathbb{F}_5$

$\oplus$	0	1	2	3	4	$\odot$	0	1	2	3	4
0	0	1	2	3	4	0	0	0	0	0	0
1	1	2	3	4	0	0	1	1	2	3	4
2	2	3	4	0	1	0	2	2	4	1	3
3	3	4	0	1	2	0	3	3	1	4	2
4	4	0	1	2	3	0	4	4	3	2	1

Let F be a ring. A polynomial with coefficients in F is an expression

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

where $a_i \in F$ for $i = 0, 1, 2, \ldots, n$ and x is a variable. The set of all polynomials $f(x)$ with coefficients in F is denoted by $F[x]$. In particular, if F is taken to be $\mathbb{Z}_p$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$, then the corresponding polynomial sets are denoted by $\mathbb{Z}_p[x]$, $\mathbb{Z}[x]$, $\mathbb{Q}[x]$, $\mathbb{R}[x]$, $\mathbb{C}[x]$, respectively. The degree of the polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ is n if $a_n \neq 0$. a_n is called the leading coefficient, and if $a_n = 1$ then the polynomial is called monic. Two polynomials $f(x)$ and $g(x)$ in $F[x]$ are equal if they have the same degree and all their coefficients are identical. If $f(a) = 0$, then a is called a root of $f(x)$ or zero of $f(x)$. Two polynomials $f(x) = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$ and $g(x) = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0$, with $n > m$, can be added, subtracted, and multiplied as follows:

$$\begin{aligned} f(x) \pm g(x) &= (a_0 \pm b_0) + (a_1 \pm b_1)x + \cdots + (a_m \pm b_m)x^m \\ &\quad + b_{m+1}x^{m+1} + \cdots + b_n x^n \\ &= \sum_{i=1}^{m} (a_i \pm b_i)x^i + \sum_{j=m+1}^{n} b_j x^j. \end{aligned}$$

$$\begin{aligned} f(x)g(x) &= a_0 b_0 + (a_0 b_1 + a_1 b_0)x + \cdots + a_m b_n x^{m+n} \\ &= \sum_{i=0}^{m} \sum_{j=0}^{n} a_i b_j x^{i+j}. \end{aligned}$$

Example 2.11 Let $f(x) = 2x^5 + x - 1$ and $g(x) = 3x^2 + 2$. Then

$$\begin{aligned} f(x) + g(x) &= 2x^5 + 3x^2 + x + 1, \\ f(x) - g(x) &= 2x^5 - 3x^2 + x - 3. \end{aligned}$$

Let $f(x) = 1 + x - x^2$ and $g(x) = 2 + x^2 + x^3$. Then

$$f(x)g(x) = 2 + 2x - x^2 + 2x^3 - x^5.$$

The division algorithm and Euclid's algorithm for integers can be extended naturally to polynomials.

Theorem 2.3 (Division algorithm for polynomials) *Let F be a field, $f(x)$ and $p(x)$ ($p(x) \neq 0$) polynomials in $F[x]$. There are unique polynomials $q(x)$ and $r(x)$ such that*

$$f(x) = p(x)q(x) + r(x)$$

where either $r(x) = 0$ or $\deg(r(x)) < \deg(p(x))$.

Example 2.12 Let $f(x) = 2x^5 + x - 1$ and $p(x) = 3x^2 + 2$. Then

$$2x^5 + x - 1 = (3x^2 + 2)\left(\frac{2}{3}x^3 - \frac{4}{9}x\right) + \left(-1 + \frac{17}{9}x\right) \text{ in } \mathbb{Q}[x],$$

$$2x^5 + x - 1 = (3x^2 + 2)(3x^3 + 5x) + 5x + 6 \text{ in } \mathbb{Z}_7[x].$$

Theorem 2.4 (Euclid's algorithm for polynomials) *Let f and g be nonzero polynomials in $F[x]$. Euclid's algorithm for polynomials runs in exactly the same way as that for integers*

$$\begin{aligned}
f &= gq_0 + r_1 \\
g &= r_1q_1 + r_2 \\
r_1 &= r_2q_2 + r_3 \\
r_2 &= r_3q_3 + r_4 \\
&\vdots \\
r_{n-2} &= r_{n-1}q_{n-1} + r_n \\
r_{n-1} &= r_nq_n + 0
\end{aligned}$$

Then, $\gcd(f, g) = r_n$. Moreover, if $d(x)$ is the greatest common divisor of $f(x)$ and $g(x)$, then there are polynomials $s(x)$ and $t(x)$ such that

$$d(x) = s(x)f(x) + t(x)g(x).$$

Example 2.13 Let

$$\begin{aligned}
f(x) &= x^5 + x^3 - x^2 - 1, \\
g(x) &= x^3 + 3x^2 + x + 3.
\end{aligned}$$

Then

$$d(x) = x^2 + 1,$$
$$s(x) = -\frac{1}{28},$$
$$t(x) = \frac{1}{28}x^2 - \frac{3}{28}x + \frac{9}{28}.$$

For polynomials, the analog to *prime number* is that of *irreducible polynomials*. A polynomial $f(x)$ of degree at least one in $F[x]$ is called *irreducible* over F if it cannot be written as a product of two nonconstant polynomials in $F[x]$ of lower degree. For example, in $Q[x]$, $f(x) = x^2 + 1$ is irreducible, since there is no factorization of $f(x)$ into polynomials both of degree less than 2 (of course, $x^2 + 1 = \frac{1}{2}(2x^2 + 2)$, but $\frac{1}{2}$ is unit in $\mathbb{Q}$). $x^2 - 2$ is irreducible in $Q[x]$ since it has no rational root. However, $x^2 - 2$ is reducible in $\mathbb{R}[x]$ as $x^2 - 2 = (x - \sqrt{2})(x + \sqrt{2})$. Factoring polynomials over rings with zero divisors can lead to some strange behaviors. For example, in $\mathbb{Z}_6$, 3 is a zero divisor, not a unit, since $1/3 \bmod 6$ does not exist. So if we consider the polynomial $3x + 3$ in $\mathbb{Z}_6[x]$, then we can factor it in several ways

$$3x + 3 = 3(x + 1) = (2x + 1)(3x + 3) = (2x^2 + 1)(3x + 3).$$

However, if F is a field, say for example, $\mathbb{Z}_5$, then $3x + 3$ can be uniquely factored into reducible polynomials in $\mathbb{Z}_5[x]$.

Theorem 2.5 *(Unique factorization in $F[x]$) Every nonconstant polynomial $f(x)$ in $F[x]$ with F a field is the product of irreducible polynomials*

$$f(x) = c \prod_{i=1}^{k} p_i(x)$$

where c is the constant, $p_i(x)$ for $i = 1, 2, \ldots, k$ are irreducible polynomials in $F[x]$.

Definition 2.14 Let θ be a complex number and

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{Q}[x]. \tag{2.25}$$

If θ is the root of the polynomial $f(x)$, then θ is called an *algebraic number*. If $f(x)$ is irreducible over $\mathbb{Q}$ and $a_n \neq 0$, then θ is of degree n.

Example 2.14 $i = \sqrt{-1}, \sqrt{2}$ are the algebraic numbers of degree 2, since they are roots of the polynomials $x^2 + 1$ and $x^2 - 2$, whereas $\sqrt[5]{3}$ is an algebraic number with degree 5, since it is the root of the polynomial $x^5 - 3$.

Every rational number is an algebraic number since $\frac{a}{b}$ is the root of the linear polynomial $x - \frac{a}{b} \in \mathbb{Q}[x]$. The set of all algebraic numbers is a field with respect to the operations of complex addition and multiplication. In particular, if α and β are algebraic numbers, then $\alpha + \beta$, $\alpha - \beta$, and $\frac{\alpha}{\beta}$ with $\beta > 0$ are all algebraic numbers.

Requiring a number to be a root of a polynomial with *rational* coefficients is the same as asking for it to be a root of a polynomial with *integer* coefficients. The rational number $\frac{a}{b}$ is the root of $bx - a \in \mathbb{Z}[x]$ as well as of $x - \frac{a}{b} \in \mathbb{Q}[x]$. So every algebraic number α is a root of the same polynomial

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{Z}[x]. \tag{2.26}$$

If the leading coefficient of $f(x) \in \mathbb{Z}[x]$ is 1 (i.e., $a_n = 1$), then α is an *algebraic integer*.

Example 2.15 $\sqrt{2}$, $\frac{-1+\sqrt{-3}}{2}$ and $\sqrt{7}+\sqrt{11}$ are algebraic integers. Every ordinary integer a is an algebraic integer since it is a root of $x - a \in \mathbb{Z}[x]$.

Let $a, b \in \mathbb{Z}$, then $a + bi$ is an algebra integer of degree 2 as it is the root of $x^2 - 2ax + (a^2 + b^2)$. The set of all $a + bi$ is denoted by $\mathbb{Z}[i]$ and is called a *Gaussian integer*. Similarly, the elements in set $\mathbb{Z}$ are called *rational integers*, In $\mathbb{Z}$, the numbers 2, 3, 5, 7, 11, 13, 17 are primes. However, in $\mathbb{Z}[i]$, the numbers 2, 5, 13, 17 are not primes, since

$$\begin{aligned} 2 &= (1+i)(1-i) \\ 5 &= (2+i)(2-i) = (1+2i)(1-2i) = -i(2+i)(1-2i) \\ 13 &= (3+2i)(3-2i) \\ 17 &= (4+i)(4-i) \end{aligned}$$

In fact, any prime in $\mathbb{Z}$ of the form $p \equiv 1 \pmod 4$ can always be factored into the form $-i(a + bi)(b + ai)$. To distinguish these, we call the primes in $\mathbb{Z}$ *rational primes*, and primes in $\mathbb{Z}[i]$ *Gaussian primes*. Also we define the *norm* of $a + b\sqrt{m}$ to be $N(a + b\sqrt{m}) = a^2 + mb^2$, so $N(-22 + 19i) = 845$.

Every algebraic integer is an algebraic number, but not vice versa.

Definition 2.15 Let α be algebraic over a field F. The unique, monic, irreducible polynomial f in $F[x]$ with α as a zero is called *minimal polynomial* of α over F. The degree of α over F is defined to be the degree of f. For example, the minimal polynomial $\sqrt[3]{2} \in \mathbb{Q}(\sqrt[3]{2})$ over $\mathbb{Q}$ is $x^3 - 2$.

Theorem 2.6 *An algebraic number is an algebraic integer if and only if its minimal polynomial has integer coefficients.*

Example 2.16 The number $\sqrt[3]{\frac{5}{7}}$ is an algebraic number but not an algebraic integer since its minimal polynomial is $x^3 - \frac{5}{7}$.

Remark 2.2 The elements of $\mathbb{Z}$ are the only rational numbers that are algebraic integers, since $\frac{a}{b}$ has minimal polynomial $x - \frac{a}{b}$ and this only has integer coefficients if $\frac{a}{b} \in \mathbb{Z}$.

Theorem 2.7 *The set of algebraic numbers forms a field, and the set of algebraic integers forms a ring.*

Problems for Section 2.1

1. Let $G = \{a, b, c, d, e, f\}$ and let $\oplus$ be defined as follows:

$\oplus$	e	a	b	c	d	f
e	e	a	b	c	d	f
a	a	e	d	f	b	c
b	b	f	e	d	c	a
c	c	d	f	e	a	b
d	d	c	a	b	f	c
f	f	b	c	a	e	d

   Show that G is a noncommutative group.
2. Show that $\mathbb{Z}_n^* = \{a : a \in \mathbb{Z}_n, \gcd(a, n) = 1\}$ is a multiplicative group.
3. Let

$$G = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{R}, ad - bc = 1 \right\}$$

   Show that G is a group under the usual matrix multiplication. Note: This group is usually denoted by $\mathrm{SL}(2, \mathbb{R})$ and is called the special linear group of order 2.
4. Let

$$G = \left\{ \begin{pmatrix} 1 & n \\ 0 & 1 \end{pmatrix} : n \in \mathbb{Z} \right\}$$

   Show that $(G, *)$ is commutative group, where $*$ is the usual matrix multiplication.
5. Show that $\mathbb{Z} = \{0, \pm 1, \pm 2, \pm 3, \ldots\}$ is a ring.
6. Show that $\mathbb{Z}_n = \{0, 1, 2, 3, \ldots, n-1\}$ is a ring.
7. Let R be a multiplicative ring and $a, b \in R$. Show that for all $n \in \mathbb{Z}^+$,

$$(a+b)^n = a^n + \binom{n}{1} a^{n-1}b + \cdots + \binom{n}{r} a^{n-r}b^r + \cdots + \binom{n}{n-1} ab^{n-1} + b^n.$$

8. Show that the set of all rational numbers forms a field.
9. Show that if p is a prime, then $\mathbb{Z}_p$ is a field.
10. Show that the multiplicative group is isomorphic group modulo 9 to the additive group modulo 6.
11. Show that any two cyclic groups of order n are isomorphic.
12. Show that the set of all rational numbers forms a field.
13. Prove that for any prime $p > 2$, the sum

$$\frac{a}{b} = \frac{1}{1^3} + \frac{1}{2^3} + \frac{1}{3^3} + \cdots + \frac{1}{(p-1)^3}$$

has the property that

$$p \mid a.$$

14. Show that there exists an irreducible polynomial of arbitrary degree n over $\mathbb{Z}_p$ with p prime.
15. Show that if m and n are positive integers such that $m \mid n$, then $\mathbb{F}_{p^n}$ contains a unique subfield $\mathbb{F}_{p^m}$, $p^m - 1 \mid p^n - 1$, whence $x^{p^m-1} - 1 \mid x^{p^n-1} - 1$ and so $x^{p^m-1} - x \mid x^{p^n-1} - x$.
16. Let F be a field containing $\mathbb{Z}_p$ and $f(x)$ be a polynomial over $\mathbb{Z}_p$. Show that if $c \in F$ is a root of $f(x)$, then c^p is also a root of $f(x)$.

2.2 Divisibility Theory

Divisibility has been studied for at least 3000 years. The ancient Greeks considered problems about even and odd numbers, perfect and amicable numbers, and the prime numbers, among many others; even today a few of these problems are still unsolved (amazing!).

Definition 2.16 Let a and b be integers with $a \neq 0$. We say a divides b, denoted by $a \mid b$, if there exists an integer c such that $b = ac$. When a divides b, we say that a is a *divisor* (or *factor*) of b, and b is a *multiple* of a. If a does not divide b, we write $a \nmid b$. If $a \mid b$ and $0 < a < b$, then a is called a *proper divisor* of b.

Note that it is usually sufficient to consider only positive divisors of an integer.

Example 2.17 The integer 200 has the following divisors:

$$1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 200.$$

Thus, for example, we can write

$$8 \mid 200,\ 50 \mid 200,\ 7 \nmid 200,\ 35 \nmid 200.$$

Definition 2.17 A divisor of n is called a *trivial divisor* of n if it is either 1 or n itself. A divisor of n is called a *nontrivial divisor* if it is a divisor of n, but is neither 1, nor n.

Theorem 2.8 (Division algorithm) *For any integer a and any positive integer b, there exist unique integers q and r such that*

$$a = bq + r, \quad 0 \leq r < b, \tag{2.27}$$

where a is called the dividend*, q the* quotient*, and r the* remainder*. If $b \nmid a$, then r satisfies the stronger inequalities $0 < r < b$.*

Proof: Consider the arithmetic progression

$$\ldots, -3b, -2b, -b, 0, b, 2b, 3b, \ldots$$

then there must be an integer q such that

$$qb \leq a < (q+1)b.$$

Let $a - qb = r$, then $a = bq + r$ with $0 \leq r < b$. To prove the uniqueness of q and r, suppose there is another pair q_1 and r_1 satisfying the same condition in 2.27, then

$$a = bq_1 + r_1, \quad 0 \leq r_1 < b.$$

We first show that $r_1 = r$. For if not, we may presume that $r < r_1$, so that $0 < r_1 - r < b$, and then we see that $b(q - q_1) = r_1 - r$, and so $b \mid (r_1 - r)$, which is impossible. Hence, $r = r_1$, and also $q = q_1$. ∎

Definition 2.18 Consider the following equation

$$a = 2q + r, \quad a, q, r \in \mathbb{Z}, \ 0 \leq r < 2. \tag{2.28}$$

Then if $r = 0$, then a is an *even*, whereas if $r = 1$, then a is an *odd*.

Definition 2.19 A positive integer n greater than 1 is called a *prime* if its only divisors are n and 1. Otherwise, it is called a *composite*.

Example 2.18 The integer 23 is prime since its only divisors are 1 and 23, whereas 22 is composite since it is divisible by 2 and 11.

Prime numbers have many special and nice properties, and play a central role in the development of number theory. Mathematicians throughout history have been fascinated by primes. The first result on prime numbers is from Euclid:

Theorem 2.9 (Euclid) There are infinitely many primes.

Proof: Suppose that $p_1, p_2, \ldots, p_k$ are all the primes. Consider the number $N = p_1 p_2 \cdots p_k + 1$. If it is a prime, then it is a new prime. Otherwise, it has a prime factor q. If q were one of the primes p_i, $i = 1, 2, \ldots, k$, then $q \mid (p_1 p_2 \cdots p_k)$, and since $q \mid (p_1 p_2 \cdots p_k + 1)$, q would divide the difference of these numbers, namely 1, which is impossible. So q cannot be one of the p_i for $i = 1, 2, \ldots, k$, and must therefore be a new prime. This completes the proof. ∎

Theorem 2.10 *If n is a composite, then n has a prime divisor p such that $p \leq \sqrt{n}$.*

Proof: Let p be the smallest prime divisor of n. If $n = rs$, then $p \leq r$ and $p \leq s$. Hence, $p^2 \leq rs = n$. That is, $p \leq \sqrt{n}$. ∎

Theorem 2.10 can be used to find all the prime numbers up to a given positive integer x; this procedure is called the Sieve of Eratosthenes, attributed to the ancient Greek astronomer and mathematician Eratosthenes of Cyrene. To apply the sieve, list all the integers from 2 up to x in order:

$$2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \ldots, x.$$

Starting from 2, delete all the multiples $2m$ of 2 such that $2 < 2m \leq x$:

$$2, 3, 5, 7, 9, 11, 13, 15, \ldots, x.$$

Starting from 3, delete all the multiples $3m$ of 3 such that $3 < 3m \leq x$:

$$2, 3, 5, 7, 11, 13, \ldots, x.$$

In general, if the resulting sequence at the kth stage is

$$2, 3, 5, 7, 11, 13, \ldots, p, \ldots, x.$$

then delete all the multiples pm of p such that $p < pm \leq x$. Continue this exhaustive computation, until $p \leq \lfloor \sqrt{x} \rfloor$, where $\lfloor \sqrt{x} \rfloor$ denotes the greatest integer $\leq \sqrt{x}$, for example, $\lfloor 0.5 \rfloor = 0$ and $\lfloor 2.9 \rfloor = 2$. The remaining integers are all the primes between $\lfloor \sqrt{x} \rfloor$ and x and if we take care not to delete $2, 3, 5, \ldots, p \leq \lfloor \sqrt{x} \rfloor$, the sieve then gives all the primes less than or equal to x.

Table 2.3 Sieve of Eratosthenes for numbers up to 100

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Algorithm 2.1 (The Sieve of Eratosthenes) Given a positive integer $n > 1$, this algorithm will find all prime numbers up to n.

[1] Create a list of integers from 2 to n.
[2] For prime numbers p_i $(i = 1, 2, \ldots)$ from 2, 3, 5 up to $\lfloor\sqrt{n}\rfloor$, delete all the multiples mp_i from the list, with $p_i < mp_i \leq n, m = 1, 2, \ldots$.
[3] List the remaining integers.

Example 2.19 Suppose we want to find all primes up to 100. First note that up to $\sqrt{100} = 10$, there are only 4 primes 2, 3, 5, 7. Thus in a table containing all positive integers from 2 to 100: Retain 2, 3, 5, 7, but cross all the multiples of 2, 3, 5, 7. After the sieving steps, the remaining numbers are the primes up to 100, as shown in Table 2.3.

Theorem 2.11 *Every composite number has a prime factor.*

Proof: Let n be a composite number. Then

$$n = n_1 n_2$$

where n_1 and n_2 are positive integers with $n_1, n_2 < n$. If either n_1 or n_2 are prime, then the theorem is proved. If n_1 and n_2 are not prime, then

$$n_1 = n_3 n_4$$

where n_3 and n_4 are positive integers with $n_3, n_4 < n_1$. Again if n_3 or n_4 are prime, then the theorem is proved. If n_3 and n_4 are not prime, then we can write

$$n_3 = n_5 n_6$$

where n_5 and n_6 are positive integers with $n_5, n_6 < n_3$. In general, after k steps we write

$$n_{2k-1} = n_{2k+1} n_{2k+2}$$

where n_{2k+1} and n_{2k+2} are positive integers with $n_{2k+1}, n_{2k+1} < n_{2k-1}$. Since

$$n > n_1 > n_3 > n_5 > \cdots n_{2k-1} > 0$$

for any value k, the process must terminate. So there must exist an n_{2k-1} for some value of k, that is prime. Hence, every composite has a prime factor. ∎

Prime numbers are the building blocks of positive integers, as the following theorem shows:

Theorem 2.12 (Fundamental Theorem of Arithmetic) *Every positive integer n greater than 1 can be written uniquely as the product of primes:*

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} = \prod_{i=1}^{k} p_i^{\alpha_i} \tag{2.29}$$

where $p_1, p_2, \ldots, p_k$ are distinct primes, and $\alpha_1, \alpha_2, \ldots, \alpha_k$ are natural numbers.

Proof: We shall first show that a factorization exists. Starting from $n > 1$, if n is a prime, then it stands as a *product* with a single factor. Otherwise, n can be factored into, say, ab, where $a > 1$ and $b > 1$. Apply the same argument to a and b: Each is either a prime or a product of two numbers both > 1. The numbers other than primes involved in the expression for n are greater than 1 and decrease at every step; hence eventually all the numbers must be prime.

Now we come to uniqueness. Suppose that the theorem is false and let $n > 1$ be the smallest number having more than one expression as the product of primes, say

$$n = p_1 p_2 \ldots p_r = q_1 q_2 \ldots q_s$$

where each $p_i (i = 1, 2, \ldots, r)$ and each $q_j (j = 1, 2, \ldots, s)$ is prime. Clearly both r and s must be greater than 1 (otherwise n is prime, or a prime is equal to a composite). If for example p_1 were one of the $q_j (j = 1, 2, \ldots, s)$, then n/p_1 would have two expressions as a product of primes, but $n/p_1 < n$ so this would contradict the definition of n. Hence p_1 is not equal to any of the $q_j (j = 1, 2, \ldots, s)$, and similarly none of the $p_i (i = 1, 2, \ldots, r)$ equals any of the $q_j (j = 1, 2, \ldots, s)$. Next, there is no loss of generality in presuming that $p_1 < q_1$, and we define the positive integer N as

$$N = (q_1 - p_1) q_2 q_3 \cdots q_s = p_1 (p_2 p_3 \cdots p_r - q_2 q_3 \cdots q_s).$$

Certainly $1 < N < n$, so N is uniquely factorable into primes. However, $p_1 \nmid (q_1 - p_1)$, since $p_1 < q_1$ and q_1 is prime. Hence one of the above expressions for N contains p_1 and the other does not. This contradiction proves the result: There cannot be any exceptions to the theorem. ∎

Definition 2.20 Let a and b be integers, not both zero. The largest divisor d such that $d \mid a$ and $d \mid b$ is called the *greatest common divisor* (gcd) of a and b. The greatest common divisor of a and b is denoted by $\gcd(a, b)$.

Example 2.20 The sets of positive divisors of 111 and 333 are as follows:

$$1, 3, 37, 111$$
$$1, 3, 9, 37, 111, 333$$

so $\gcd(111, 333) = 111$. But $\gcd(91, 111) = 1$, since 91 and 111 have no common divisors other than 1.

The next theorem indicates that $\gcd(a, b)$ can be represented as a linear combination of a and b.

Theorem 2.13 *Let a and b be integers, not both zero. Then there exist integers x and y such that*

$$d = \gcd(a, b) = ax + by. \tag{2.30}$$

Proof: Consider the set of all linear combinations $au + bv$, where u and v range over all integers. Clearly this set of integers $\{au + bv\}$ includes positive, negative, as well as 0. It contains a smallest positive element, say, m, such that $m = ax + by$. Use the division algorithm, to write $a = mq + r$, with $0 \leq r < m$. Then

$$r = a - mq = a - q(ax + by) = (1 - qx)a + (-qy)b$$

and hence r is also a linear combination of a and b. But $r < m$, so it follows from the definition of m that $r = 0$. Thus $a = mq$, that is, $m \mid a$; similarly, $m \mid b$. Therefore, m is a common divisor of a and b. Since $d \mid a$ and $d \mid b$, d divides any linear combination of a and b. Since $d = \gcd(a, b)$, we must have $d = m$. ∎

Corollary 2.1 *If a and b are integers, not both zero, then the set*

$$S = \{ax + by : x, y \in \mathbb{Z}\}$$

is precisely the set of all multiples of $d = \gcd(a, b)$.

Proof: It follows from Theorem 2.13, because d is the smallest positive value of $ax + by$ where x and y range over all integers. ∎

Definition 2.21 Two integers a and b are called *relatively prime* if $\gcd(a, b) = 1$. We say that integers $n_1, n_2, ..., n_k$ are *pairwise relatively prime* if, whenever $i \neq j$, we have $\gcd(n_i, n_j) = 1$.

Example 2.21 91 and 111 are relatively prime, since $\gcd(91, 111) = 1$.

The following theorem characterizes relative primes in terms of linear combinations.

Theorem 2.14 *Let a and b be integers, not both zero, then a and b are relatively prime if and only if there exist integers x and y such that $ax + by = 1$.*

Proof: If a and b are relatively prime, so that $\gcd(a, b) = 1$, then Theorem 2.13 guarantees the existence of integers x and y satisfying $ax + by = 1$. As for the converse, suppose that $ax + by = 1$ and that $d = \gcd(a, b)$. Since $d \mid a$ and $d \mid b$, $d \mid (ax + by)$, that is, $d \mid 1$. Thus $d = 1$. The result follows. ∎

Theorem 2.15 *If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.*

Proof: By Theorem 2.13, we can write $ax + by = 1$ for some choice of integers x and y. Multiplying this equation by c we get

$$acx + bcy = c.$$

Since $a \mid ac$ and $a \mid bc$, it follows that $a \mid (acx + bcy)$. The result thus follows. ∎

For the greatest common divisor of more than two integers, we have the following result.

Theorem 2.16 *Let $a_1, a_2, \ldots, a_n$ be n integers. Let also*

$$\left.\begin{aligned} \gcd(a_1, a_2) &= d_2 \\ \gcd(d_2, a_3) &= d_3 \\ &\vdots \\ \gcd(d_{n-1}, a_n) &= d_n \end{aligned}\right\} \tag{2.31}$$

Then

$$\gcd(a_1, a_2, \ldots, a_n) = d_n. \tag{2.32}$$

Proof: By (2.31), we have $d_n \mid a_n$ and $d_n \mid d_{n-1}$. Since $d_{n-1} \mid a_{n-1}$ and $d_{n-1} \mid d_{n-2}$, $d_n \mid a_{n-1}$ and $d_n \mid d_{n-2}$. Continuing in this way, we finally have $d_n \mid a_n$, $d_n \mid a_{n-1}$, $\cdots$, $d_n \mid a_1$, so d_n is a common divisor of $a_1, a_2, \ldots, a_n$. Now suppose that d is any common divisor of $a_1, a_2, \ldots, a_n$, then $d \mid a_1$ and $d \mid d_2$. Observe the fact that the common divisor of a and b and the divisor of $\gcd(a, b)$ are the same, so $d \mid d_2$. Similarly, we have $d \mid d_3, \ldots, d \mid d_n$. Therefore, $d \leq |d| \leq d_n$. So, d_n is the greatest common divisor of $a_1, a_2, \ldots, a_n$. ∎

Definition 2.22 If d is a multiple of a and also a multiple of b, then d is a common multiple of a and b. The *least common multiple* (lcm) of two integers a and b, is the smallest of the common multiples of a and b. The least common multiple of a and b is denoted by $\text{lcm}(a, b)$.

Theorem 2.17 *Suppose a and b are not both zero (i.e., one of the a and b can be zero, but not both zero), and that $m = \text{lcm}(a, b)$. If x is a common multiple of a and b, then $m \mid x$. That is, every common multiple of a and b is a multiple of the least common multiple.*

Proof: If any one of a and b is zero, then all common multiples of a and b are zero, so the statement is trivial. Now we assume that both a and b are not zero. Dividing x by m, we get

$$x = mq + r, \quad \text{where } 0 \leq r < m.$$

Now $a \mid x$ and $b \mid x$ and also $a \mid m$ and $b \mid m$; so by Theorem 2.8, $a \mid r$ and $b \mid r$. That is, r is a common multiple of a and b. But m is the least common multiple of a and b, so $r = 0$. Therefore, $x = mq$ and the result follows. ∎

For the least common multiple of more than two integers, we have the following result.

Theorem 2.18 *Let $a_1, a_2, \ldots, a_n$ be n integers. Let also*

$$\left.\begin{aligned} \text{lcm}(a_1, a_2) &= m_2, \\ \text{lcm}(m_2, a_3) &= m_3, \\ &\vdots \\ \text{lcm}(m_{n-1}, a_n) &= m_n. \end{aligned}\right\} \tag{2.33}$$

Then

$$\text{lcm}(a_1, a_2, \ldots, a_n) = m_n. \tag{2.34}$$

Proof: By (2.33), we have $m_i \mid m_{i+1}$, $i = 2, 3, \ldots, n-1$, and $a_1 \mid m_2$, $a_i \mid m_i$, $i = 2, 3, \ldots, n$. So, m_n is a common multiple of $a_1, a_2, \ldots, a_n$. Now let m be any common multiple of $a_1, a_2, \ldots, a_n$, then $a_1 \mid m, a_2 \mid m$. Observe the result that all the common multiples of a and b are the multiples of $\text{lcm}(a, b)$. So $m_1 \mid m$ and $a_3 \mid m$. Continuing the process in this way, we finally have $m_n \mid m$. Thus, $m_n \leq |m|$. Therefore, $m_n = \text{lcm}(a_1, a_2, \ldots, a_n)$∎

One way to calculate the $\gcd(a, b)$ or the $\text{lcm}(a, b)$ is to use the standard prime factorizations of a and b. That is:

Theorem 2.19 *If*

$$a = \prod_{i=1}^{k} p_i^{\alpha_i}, \quad \alpha_i \geq 0,$$

and

$$b = \prod_{i=1}^{k} p_i^{\beta_i}, \quad \beta_i \geq 0,$$

then

$$\gcd(a, b) = \prod_{i=1}^{k} p_i^{\gamma_i} \tag{2.35}$$

$$\text{lcm}(a, b) = \prod_{i=1}^{k} p_i^{\delta_i} \tag{2.36}$$

where $\gamma_i = \min(\alpha_i, \beta_i)$ *and* $\delta_i = \max(\alpha_i, \beta_i)$ *for* $i = 1, 2, \ldots, k$.

Proof: It is easy to see that

$$\gcd(a, b) = \prod_{i=1}^{k} p_i^{\gamma_i}, \text{ where } \gamma_i \text{ is the lesser of } \alpha_i \text{ and } \beta_i,$$
$$\text{lcm}(a, b) = \prod_{i=1}^{k} p_i^{\delta_i}, \text{ where } \delta_i \text{ is the greater of } \alpha_i \text{ and } \beta_i.$$

The result thus follows. ∎

Corollary 2.2 *Suppose a and b are positive integers, then*

$$\text{lcm}(a, b) = \frac{ab}{\gcd(a, b)}. \tag{2.37}$$

Proof: Since $\gamma_i + \delta_i = \alpha_i + \beta_i$, it is now obvious that

$$\gcd(a, b) \cdot \text{lcm}(a, b) = ab.$$

The result thus follows. ∎

Example 2.22 Find gcd(240, 560) and lcm(240, 560). Since the prime factorizations of 240 and 560 are

$$240 = 2^4 \cdot 3 \cdot 5 = 2^4 \cdot 3^1 \cdot 5^1 \cdot 7^0$$
$$560 = 2^4 \cdot 5 \cdot 7 = 2^4 \cdot 3^0 \cdot 5^1 \cdot 7^1,$$

we get

$$\begin{aligned}\gcd(240, 560) &= 2^{\min(4,4)} \cdot 3^{\min(1,0)} \cdot 5^{\min(1,1)} \cdot 7^{\min(0,1)} \\ &= 2^4 \cdot 3^0 \cdot 5^1 \cdot 7^0 \\ \text{lcm}(240,\ 560) &= 2^{\max(4,4)} \cdot 3^{\max(1,0)} \cdot 5^{\max(1,1)} \cdot 7^{\max(0,1)} \\ &= 2^4 \cdot 3^1 \cdot 5^1 \cdot 7^1\end{aligned}$$

Of course, if we know $\gcd(240, 560) = 80$, then we can find $\text{lcm}(240,\ 560)$ by

$$\text{lcm}(240,\ 560) = 240 \cdot 560/80 = 1680.$$

Similarly, if we know $\text{lcm}(240,\ 560)$, we can find $\gcd(240, 560)$ by

$$\gcd(240, 560) = 240 \cdot 560/1680 = 80.$$

There is an efficient method, due to Euclid, for finding the greatest common divisor of two integers.

Theorem 2.20 (Division theorem) *Let a, b, q, r be integers with $b > 0$ and $0 \leq r < b$ such that $a = bq + r$. Then $\gcd(a, b) = \gcd(b, r)$.*

Proof: Let $X = \gcd(a, b)$ and $Y = \gcd(b, r)$, it suffices to show that $X = Y$. If integer c is a divisor of a and b, it follows from the equation $a = bq + r$ and the divisibility properties that c is a divisor of r also. By the same argument, every common divisor of b and r is a divisor of a. ∎

Theorem 2.20 can be used to reduce the problem of finding $\gcd(a, b)$ to the simpler problem of finding $\gcd(b, r)$. The problem is simpler because the numbers are smaller, but it has the same answer as the original one. The process of finding $\gcd(a, b)$ by repeated application of Theorem 2.20 is called Euclid's algorithm which proceeds as follows:

$a = bq_0 + r_1,$	$0 \leq r_1 < b$	(dividing b into a),
$b = r_1q_1 + r_2,$	$0 \leq r_2 < r_1$	(dividing r_1 into b),
$r_1 = r_2q_2 + r_3,$	$0 \leq r_3 < r_2$	(dividing r_2 into r_1),
$r_2 = r_3q_3 + r_4,$	$0 \leq r_4 < r_3$	(dividing r_3 into r_2),
$\vdots$	$\vdots$	$\vdots$
$r_{n-2} = r_{n-1}q_{n-1} + r_n,$	$0 \leq r_n < r_{n-1}$	(dividing r_{n-1} into r_{n-2}),
$r_{n-1} = r_nq_n + 0,$	$r_{n+1} = 0$	(arriving at a zero-remainder)

or, diagrammatically,

$$\begin{array}{r||c||r}
a & & \\
-bq_0 & q_0 & b \\
\hline
r_1 & q_1 & -r_1q_1 \\
-r_2q_2 & q_2 & \hline r_2 \\
\hline
r_3 & q_3 & -r_3q_3 \\
\vdots & \vdots & \vdots \\
r_{n-1} & q_{n-1} & -r_{n-1}q_{n-1} \\
-r_nq_n & q_n & \hline \boxed{r_n} \\
\hline
r_{n+1}=0 & &
\end{array}$$

Then the greatest common divisor *gcd* of a and b is r_n. That is,

$$d = \gcd(a, b) = r_n. \tag{2.38}$$

We now restate it in a theorem form.

Theorem 2.21 (Euclid's algorithm) *Let a and b be positive integers with $a \geq b$. If $b \mid a$, then* $\gcd(a, b) = b$. *If $b \nmid a$, then apply the division algorithm repeatedly as follows:*

$$\left.\begin{array}{ll}
a = bq_0 + r_1, & 0 < r_1 < b, \\
b = r_1q_1 + r_2, & 0 < r_2 < r_1, \\
r_1 = r_2q_2 + r_3, & 0 < r_3 < r_2, \\
r_2 = r_3q_3 + r_4, & 0 < r_4 < r_3, \\
\quad\vdots & \quad\vdots \\
r_{n-2} = r_{n-1}q_{n-1} + r_n, & 0 < r_n < r_{n-1}, \\
r_{n-1} = r_nq_n + 0. &
\end{array}\right\} \tag{2.39}$$

Then r_n, the last nonzero remainder, is the greatest common divisor *of a and b. That is,*

$$\gcd(a, b) = r_n. \tag{2.40}$$

Values of x and y in

$$\gcd(a, b) = ax + by \tag{2.41}$$

can be obtained by writing each r_i as a linear combination of a and b.

Proof: The system of equations is obtained by the series divisions:

$$\frac{a}{b}, \quad \frac{b}{r_1}, \quad \frac{r_1}{r_2}, \quad \cdots$$

The process stops whenever $r_i = 0$ for $i = 1, 2, \ldots, n$.

We now prove that r_n is the greatest common divisor of a and b. By Theorem 2.20, we have

$$\begin{aligned}
\gcd(a, b) &= \gcd(a - bq_0, b) \\
&= \gcd(r_1, b) \\
&= \gcd(r_1, b - r_1 q_1) \\
&= \gcd(r_1, r_2) \\
&= \gcd(r_1 - r_2 q_2, r_2) \\
&= \gcd(r_3, r_2)
\end{aligned}$$

Continuing by mathematical induction, we have

$$\gcd(a, b) = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n.$$

To see that r_n is a linear combination of a and b, we argue by induction that each r_i is a linear combination of a and b. Clearly, r_1 is a linear combination of a and b, since $r_1 = a - bq_0$, so does r_2. In general, r_i is a linear combination of r_{i-1} and r_{i-2}. By the inductive hypothesis we may suppose that these latter two numbers are linear combinations of a and b, and it follows that r_i is also a linear combination of a and b. ∎

Algorithm 2.2 (Euclid's algorithm) Given integers a and b with $a > b > 0$, this algorithm will compute $\gcd(a, b)$.

[1] (Initialization) Set

$$\begin{aligned}
r_{-1} &\leftarrow a \\
r_0 &\leftarrow b \\
i &= 0.
\end{aligned}$$

[2] (Decision) If $r_i = 0$, Output $r_{i-1} = \gcd(a, b)$ and Exit.

[3] (Computation)

$$q_i \leftarrow \lfloor r_{i-1}/r_i \rfloor$$
$$r_{i+1} \leftarrow r_{i-1} - q_i \cdot r_i$$
$$i \leftarrow i + 1$$
go to step [2].

Remark 2.3 Euclid's algorithm is found in Book VII, Proposition 1 and 2 of his *Elements*, but it probably wasn't his own invention. Scholars believe that the method was known up to 200 years earlier. However, it first appeared in Euclid's work and, more importantly, it is the first nontrivial algorithm to have survived to this day.

Remark 2.4 It is evident that the algorithm cannot recur indefinitely, since the second argument strictly decreases in each recursive call. Therefore, the algorithm always terminates with the correct answer. More importantly, it can be performed in polynomial time. That is, if Euclid's algorithm is applied to two positive integers a and b with $a \geq b$, then the number of divisions required to find $\gcd(a, b)$ is $\mathcal{O}(\log b)$, a polynomial-time complexity.

Example 2.23 Use Euclid's algorithm to find the gcd of 1281 and 243. Since

1281		
−1215	5	243
66	3	−198
−45	1	45
21	2	−42
−21	7	$\boxed{3}$
0		

we have $\gcd(1281, 243) = 3$.

Theorem 2.22 *If a and b are any two integers, then*

$$Q_k a - P_k b = (-1)^{k-1} r_k, \quad k = 1, 2, \ldots, n \tag{2.42}$$

where

$$\left.\begin{aligned} P_0 &= 1,\ P_1 = q_0,\ P_k = q_{k-1}P_{k-1} + P_{k-2} \\ Q_0 &= 0,\ Q_1 = 1,\ Q_k = q_{k-1}Q_{k-1} + Q_{k-2} \end{aligned}\right\} \tag{2.43}$$

for $k = 2, 3, \ldots, n$.

Proof: When $k = 1$, 2.42 is clearly true, since $Q_1a - P_1b = (-1)^{1-1}r_1$ implies $a - q_0b = r_1$. When $k = 2$, $r_2 = -(aq_1 - b(1 + q_0q_1))$. But $1 + q_0q_1 = q_2P_1 + P_0$, $q_1 = q_1 \cdot 1 + 0 = q_1Q_1 + Q_0$, therefore, $Q_2a - P_2b = (-1)^{2-1}r_2$, $P_2 = q_1P_1 + P_0$, $Q_2 = q_1Q_1 + Q_0$. Assume (2.42) and (2.43) hold for all positive integers $\leq k$, then

$$\begin{aligned} (-1)^k r_{k+1} &= (-1)^k(r_{k-1} - q_k r_k) \\ &= (Q_{k-1}a - P_k b) + q_k(Q_k a - P_k b) \\ &= (q_k Q_k + Q_{k-1})a - (q_{k+1}P_k + P_{k+1})b. \end{aligned}$$

Thus, $Q_{k+1}a - P_{k+1}b = (-1)^k r_{k+1}$, where $P_{k+1} = q_kP_k + P_{k-1}$, $Q_{k+1} = q_{k+1}Q_k + Q_{k-1}$. By induction, the result is true for all positive integers. ∎

Euclid's algorithm for computing the greatest common divisor of two integers is intimately connected with continued fractions.

Definition 2.23 Let a and b be integers and let Euclid's algorithm run as

$$\begin{aligned} a &= bq_0 + r_1, \\ b &= r_1q_1 + r_2, \\ r_1 &= r_2q_2 + r_3, \\ r_2 &= r_3q_3 + r_4, \\ &\vdots \\ r_{n-2} &= r_{n-1}q_{n-1} + r_n, \\ r_{n-1} &= r_nq_n + 0. \end{aligned}$$

That is,

$$\begin{array}{r||c||r}
a & & \\
-bq_0 & q_0 & b \\
\hline
r_1 & q_1 & -r_1q_1 \\
-r_2q_2 & q_2 & \hline r_2 \\
\hline
r_3 & q_3 & -r_3q_3 \\
\vdots & \vdots & \vdots \\
r_{n-1} & q_{n-1} & -r_{n-1}q_{n-1} \\
-r_nq_n & q_n & \hline r_n \\
\hline
r_{n+1} = 0 & &
\end{array}$$

Then the fraction $\frac{a}{b}$ can be expressed as a simple continued fraction:

$$\frac{a}{b} = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots q_{n-1} + \cfrac{1}{q_n}}}} \tag{2.44}$$

where $q_0, q_1, \ldots, q_{n-1}, q_n$ are taken directly from Euclid's algorithm expressed in (2.39), and are called the *partial quotients* of the continued fraction. For simplicity, the continued fraction expansion (2.44) of $\frac{a}{b}$ is usually written as

$$\frac{a}{b} = q_0 + \frac{1}{q_1+}\frac{1}{q_2+}\cdots\frac{1}{q_{n-1}+}\frac{1}{q_n} \tag{2.45}$$

or even more briefly as

$$\frac{a}{b} = [q_0, q_1, q_2, \ldots q_{n-1}, q_n]. \tag{2.46}$$

If each q_i is an integer, the continued fraction is called *simple*; a simple continued fraction can either be *finite* or *infinite*. A continued fraction formed from $[q_0, q_1, q_2, \ldots q_{n-1}, q_n]$ by

neglecting all of the terms after a given term is called a *convergent* of the original continued fraction. If we denote the kth convergent by $C_k = \dfrac{P_k}{Q_k}$, then

$$(1)\quad \begin{cases} C_0 = \dfrac{P_0}{Q_0} = \dfrac{q_0}{1}; \\ C_1 = \dfrac{P_1}{Q_1} = \dfrac{q_0 q_1 + 1}{q_1}; \\ \quad \vdots \\ C_k = \dfrac{P_k}{Q_k} = \dfrac{q_k P_{k-1} + P_{k-2}}{q_k Q_{k-1} + Q_{k-2}}, \text{ for } k \geq 2. \end{cases}$$

(2) If $P_k = q_k Q_{k-1} + Q_{k-2}$ and $Q_k = q_k P_{k-1} + P_{k-2}$, then $\gcd(P_k, Q_k) = 1$.
(3) $P_k Q_{k-1} - P_{k-1} Q_k = (-1)^{k-1}$, for $k \geq 1$.

The following example shows how to use Euclid's algorithm to express a rational number as a finite simple continued fraction.

Example 2.24 Expand the rational number $\dfrac{1281}{243}$ as a simple continued fraction. First let $a = 1281$ and $b = 243$, and then let Euclid's algorithm run as follows:

$$\begin{array}{r||c||r}
1281 & & \\
-1215 & 5 & 243 \\
\hline
66 & 3 & -198 \\
\hline
-45 & 1 & 45 \\
\hline
21 & 2 & -42 \\
\hline
-21 & 7 & 3 \\
\hline
0 & &
\end{array}$$

So $\dfrac{1281}{243} = [5, 3, 1, 2, 7]$. Thus

$$\frac{1281}{243} = 5 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{7}}}}.$$

Of course, as a by-product, we also find that $\gcd(1281, 243) = 3$.

Theorem 2.23 *Any finite simple continued fraction represents a rational number. Conversely, any rational number can be expressed as a finite simple continued fraction, in exactly two ways, one with an odd number of terms and one with an even number of terms.*

Proof: The first assertion is proved by induction. When $n = 1$, we have

$$[q_0, q_1] = q_0 + \frac{1}{q_1} = \frac{q_0 q_1 + 1}{q_1}$$

which is rational. Now we assume for $n = k$ the simple continued fraction $[q_0, q_1, \ldots, q_k]$ is rational whenever $q_0, q_1, \ldots, q_k$ are integers with $q_1, \ldots, q_k$ positive. Let $q_0, q_1, \ldots, q_{k+1}$ be integers with $q_1, \ldots, q_{k+1}$ positive. Note that

$$[q_0, q_1, \ldots, q_k, q_{k+1}] = a_0 + \frac{1}{[q_1, \ldots, q_k, q_{k+1}]}.$$

By the induction hypothesis, $[q_1, q_2, \ldots, q_k, q_{k+1}]$ is rational. That is, there exist two integers r and s with $s \neq 0$ such that

$$[q_1, q_2, \ldots, q_k, q_{k+1}] = \frac{r}{s}.$$

Thus,

$$[q_0, q_1, \ldots, q_k, q_{k+1}] = a_0 + \frac{1}{r/s} = \frac{q_0 r + s}{r}$$

which is rational.

Now we use Euclid's algorithm to show that every rational number can be written as a finite simple continued fraction. Let a/b be a rational number with $b > 0$. Euclid's algorithm tells us that

$$\begin{aligned}
a &= bq_0 + r_1, && 0 < r_1 < b,\\
b &= r_1 q_1 + r_2, && 0 < r_2 < r_1,\\
r_1 &= r_2 q_2 + r_3, && 0 < r_3 < r_2,\\
r_2 &= r_3 q_3 + r_4, && 0 < r_4 < r_3,\\
&\vdots && \vdots\\
r_{n-2} &= r_{n-1} q_{n-1} + r_n, && 0 < r_n < r_{n-1},\\
r_{n-1} &= r_n q_n + 0.
\end{aligned}$$

In these equations, $q_1, q_2, \ldots, q_n$ are positive integers. Rewriting these equations, we obtain

$$\begin{aligned}
\frac{a}{b} &= q_0 + \frac{r_1}{b} \\
\frac{b}{r_1} &= q_1 + \frac{r_2}{r_1} \\
\frac{r_1}{r_2} &= q_2 + \frac{r_3}{r_2} \\
&\vdots \\
\frac{r_{n-1}}{r_n} &= q_n
\end{aligned}$$

By successive substitution

$$\begin{aligned}
\frac{a}{b} &= q_0 + \frac{1}{\frac{b}{r_1}} \\
&= q_0 + \frac{1}{q_1 + \frac{1}{\frac{r_1}{r_2}}} \\
&\vdots \\
&= q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots\, q_{n-1} + \cfrac{1}{q_n}}}}
\end{aligned}$$

This shows that every rational number can be written as a finite simple continued fraction.

Further, it can be shown that any rational number can be expressed as a finite simple continued fraction in exactly two ways, one with an odd number of terms and one with an even number of terms; we leave this as an exercise. ■

Definition 2.24 Let $q_0, q_1, q_2, \ldots$ be a sequence of integers, all positive except possibly q_0. Then the expression $[q_0, q_1, q_2, \ldots]$ is called an *infinite* simple continued fraction and is defined to be equal to the number $\lim_{n\to\infty} [q_0, q_1, q_2, \ldots, q_{n-1}, q_n]$.

Theorem 2.24 *Any* irrational *number can be written uniquely as an* infinite *simple continued fraction. Conversely, if α is an infinite simple continued fraction, then α is irrational.*

Proof: Let α be an irrational number. We write

$$\alpha = [\alpha] + \{\alpha\} = [\alpha] + \frac{1}{\frac{1}{\{\alpha\}}}$$

where $[\alpha]$ is the integral part and $\{\alpha\}$ the fractional part of α, respectively. Because α is irrational, $1/\{\alpha\}$ is irrational and greater than 1. Let

$$q_0 = [\alpha], \quad \text{and} \quad \alpha_1 = \frac{1}{\{\alpha\}}.$$

We now write

$$\alpha_1 = [\alpha_1] + \{\alpha_1\} = [\alpha_1] + \frac{1}{\dfrac{1}{\{\alpha_1\}}}$$

where $1/\{\alpha_1\}$ is irrational and greater than 1. Let

$$q_1 = [\alpha_1], \quad \text{and} \quad \alpha_2 = \frac{1}{\{\alpha_1\}}.$$

We continue inductively

$$\begin{aligned}
q_2 &= [\alpha_2], \quad \text{and} \quad \alpha_3 = \frac{1}{\{\alpha_2\}} > 1 \quad (\alpha_3 \text{ irrational}) \\
q_3 &= [\alpha_3], \quad \text{and} \quad \alpha_4 = \frac{1}{\{\alpha_3\}} > 1 \quad (\alpha_3 \text{ irrational}) \\
&\vdots \\
q_n &= [\alpha_n], \quad \text{and} \quad \alpha_n = \frac{1}{\{\alpha_{n-1}\}} > 1 \quad (\alpha_3 \text{ irrational}) \\
&\vdots
\end{aligned}$$

Since each α_n, $n = 2, 3 \cdots$ is greater than 1, then $q_{n-1} \geq 1$, $n = 2, 3, \ldots$. If we substitute successively, we obtain

$$\begin{aligned}
\alpha &= [q_0, \alpha_1] \\
&= [q_0, q_1, \alpha_2] \\
&= [q_0, q_1, q_2, \alpha_3] \\
&\vdots \\
&= [q_0, q_1, q_2, \ldots, q_n, \alpha_{n+1}] \\
&\vdots
\end{aligned}$$

Next we shall show that $\alpha = [q_0, q_1, q_2, \ldots]$. Note that C_n, the nth convergent to $[q_0, q_1, q_2, \ldots]$ is also the nth convergent to $[q_0, q_1, q_2, \ldots, q_n, \alpha_{n+1}]$. If we denote the $(n+1)$st convergent to this finite continued fraction by $P'_{n+1}/Q'_{n+1} = \alpha$, then

$$\alpha - C_n = \frac{P'_{n+1}}{Q'_{n+1}} - \frac{P_n}{Q_n} = \frac{(-1)^{n+1}}{Q'_{n+1}Q_n}.$$

Since Q_n and Q'_{n+1} become infinite as $n \to \infty$, then

$$\lim_{n\to\infty}(\alpha - C_n) = \lim_{n\to\infty}\frac{(-1)^{n+1}}{Q'_{n+1}Q_n} = 0$$

and

$$\alpha = \lim_{n\to\infty} C_n = [q_0, q_1, \ldots].$$

The uniqueness of the representation, as well as the second assertion are left as an exercise. ∎

Definition 2.25 A real irrational number which is the root of a quadratic equation $ax^2 + bx + c = 0$ with integer coefficients is called *quadratic irrational*.

For example, $\sqrt{3}, \sqrt{5}, \sqrt{7}$ are quadratic irrationals. For convenience, we shall denote $\sqrt{N}$, with N not a perfect square, as a quadratic irrational. Quadratic irrationals are the simplest possible irrationals.

Definition 2.26 An infinite simple continued fraction is said to be *periodic* if there exists integers k and m such that $q_{i+m} = q_i$ for all $i \geq k$. The periodic simple continued fraction is usually denoted by $[q_0, q_1, \ldots, q_k, \overline{q_{k+1}, q_{k+2}, \ldots, q_{k+m}}]$. If it is of the form $[\overline{q_0, q_1, \ldots, q_{m-1}}]$, then it is called *purely periodic*. The smallest positive integer m satisfying the above relationship is called the *period* of the expansion.

Theorem 2.25 *Any* periodic *simple continued fraction is a quadratic irrational. Conversely, any* quadratic irrational *has a periodic expansion as a simple continued fraction.*

Proof: The proof is rather lengthy and left as an exercise. ∎

We are now in a position to present an algorithm for finding the simple continued fraction expansion of a *real number*.

Theorem 2.26 (Continued fraction algorithm) *Suppose x is irrational, and let $x_0 = x$. Then x can be expressed as a simple continued fraction*

$$[q_0, q_1, q_2, \ldots, q_n, q_{n+1}, \ldots]$$

by the following process:

$$\left.\begin{array}{ll} x_0 = x & \\ q_0 = \lfloor x_0 \rfloor, & x_1 = \dfrac{1}{x_0 - q_0} \\ q_1 = \lfloor x_1 \rfloor, & x_2 = \dfrac{1}{x_1 - q_1} \\ \quad \vdots & \quad \vdots \\ q_n = \lfloor x_n \rfloor, & x_{n+1} = \dfrac{1}{x_n - q_n} \\ q_{n+1} = \lfloor x_{n+1} \rfloor, & x_{n+2} = \dfrac{1}{x_{n+1} - q_{n+1}} \\ \quad \vdots & \quad \vdots \end{array}\right\} \tag{2.47}$$

Proof: Follows from Theorem 2.24. ∎

Algorithm 2.3 (Continued fraction algorithm) Given a real number x, this algorithm will compute and output the partial quotients $q_0, q_1, q_2, \ldots, q_n$ of the continued fraction x.

[1] (Initialization) Set
$i \leftarrow 0$,
$x_i \leftarrow x$,
$q_i \leftarrow \lfloor x_i \rfloor$,
print(q_i).

[2] (Decision) If $x_i = q_i$, Exit.
[3] (Computation)
$x_{i+1} \leftarrow \dfrac{1}{x_i - q_i}$,
$i \leftarrow i + 1$,
$q_i \leftarrow \lfloor x_i \rfloor$,
print(q_i),
go to Step [2].

Example 2.25 Let $x = 160523347/60728973$. Then by applying Algorithm 2.3, we get $160523347/60728973 = [2, 1, 1, 1, 4, 12, 102, 1, 1, 2, 3, 2, 2, 36]$. That is,

$$\frac{160523347}{60728973} = 2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{12 + \cfrac{1}{102 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{2 + \frac{1}{36}}}}}}}}}}}}}$$

Theorem 2.27 *Each quadratic irrational number $\sqrt{N}$ has a periodic expansion as an infinite simple continued fraction of the form*

$$[q_0, q_1, q_2, \ldots, q_k, \overline{q_{k+1}, \ldots, q_{k+m}}].$$

Example 2.26 Expand $\sqrt{3}$ as a periodic simple continued fraction. Let $x_0 = \sqrt{3}$. Then we have

$$\begin{aligned}
q_0 &= \lfloor x_0 \rfloor = \lfloor \sqrt{3} \rfloor = 1 \\
x_1 &= \frac{1}{x_0 - q_0} = \frac{1}{\sqrt{3} - 1} = \frac{\sqrt{3} + 1}{2} \\
q_1 &= \lfloor x_1 \rfloor = \left\lfloor \frac{\sqrt{3} + 1}{2} \right\rfloor = \left\lfloor 1 + \frac{\sqrt{3} - 1}{2} \right\rfloor = 1 \\
x_2 &= \frac{1}{x_1 - q_1} = \frac{1}{\frac{\sqrt{3} + 1}{2} - 1} = \frac{1}{\frac{\sqrt{3} - 1}{2}} = \frac{2(\sqrt{3} + 1)}{(\sqrt{3} - 1)(\sqrt{3} + 1)} = \sqrt{3} + 1 \\
q_2 &= \lfloor x_2 \rfloor = \lfloor \sqrt{3} + 1 \rfloor = 2 \\
x_3 &= \frac{1}{x_2 - q_2} = \frac{1}{\sqrt{3} + 1 - 2} = \frac{1}{\sqrt{3} - 1} = \frac{\sqrt{3} + 1}{2} = x_1 \\
q_3 &= \lfloor x_3 \rfloor = \left\lfloor \frac{\sqrt{3} + 1}{2} \right\rfloor = \left\lfloor 1 + \frac{\sqrt{3} - 1}{2} \right\rfloor = 1 = q_1
\end{aligned}$$

$$
\begin{aligned}
x_4 &= \frac{1}{x_3 - q_3} = \frac{1}{\frac{\sqrt{3}+1}{2} - 1} = \frac{1}{\frac{\sqrt{3}-1}{2}} = \frac{2(\sqrt{3}+1)}{(\sqrt{3}-1)(\sqrt{3}+1)} = \sqrt{3}+1 = x_2 \\
q_4 &= \lfloor x_3 \rfloor = \lfloor \sqrt{3}+1 \rfloor = 2 = q_2 \\
x_5 &= \frac{1}{x_4 - q_4} = \frac{1}{\sqrt{3}+1-2} = \frac{1}{\sqrt{3}-1} = \frac{\sqrt{3}+1}{2} = x_3 = x_1 \\
q_5 &= \lfloor x_5 \rfloor = \lfloor x_3 \rfloor = 1 = q_3 = q_1 \\
&\vdots
\end{aligned}
$$

So, for $n = 1, 2, 3, \ldots$, we have $q_{2n-1} = 1$ and $q_{2n} = 2$. Thus, the *period* of the continued fraction expansion of $\sqrt{3}$ is 2. Therefore, we finally get

$$
\sqrt{3} = 1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{\ddots}}}}} = [1, \overline{1, 2}].
$$

Definition 2.27 The algebraic equation with two variables

$$
ax + by = c \tag{2.48}
$$

is called a *linear Diophantine equation*, for which we wish to find integer solutions in x and y.

Theorem 2.28 *Let a, b, c be integers with not both a and b equal to 0. If $d \nmid c$, then the linear Diophantine equation*

$$
ax + by = c
$$

has no integer solution. The equation has an integer solution in x and y if and only if $d \mid c$. Moreover, if (x_0, y_0) is a solution of the equation, then the general solution of the equation is

$$
(x, y) = \left(x_0 + \frac{b}{d} \cdot t,\ y_0 - \frac{a}{d} \cdot t \right), \quad t \in \mathbb{Z}. \tag{2.49}
$$

Proof: Assume that x and y are integers such that $ax + by = c$. Since $d \mid a$ and $d \mid b$, $d \mid c$. Hence, if $d \nmid c$, there is no integer solution of the equation.

Now suppose $d \mid c$. There is an integer k such that $c = kd$. Since d is a sum of multiples of a and b, we may write

$$am + bn = d.$$

Multiplying this equation by k, we get

$$a(mk) + b(nk) = dk = c$$

so that $x = mk$ and $y = nk$ is a solution.

For the "only if" part, suppose x_0 and y_0 is a solution of the equation. Then

$$ax_0 + by_0 = c.$$

Since $d \mid a$ and $d \mid b$, then $d \mid c$. ∎

Theorem 2.29 *Let the convergents of the finite continued fraction of a/b be as follows:*

$$\left[\frac{P_0}{Q_0}, \frac{P_1}{Q_1}, \ldots, \frac{P_{n-1}}{Q_{n-1}}, \frac{P_n}{Q_n}\right] = \frac{a}{b}. \tag{2.50}$$

Then the integer solution in x and y of the equation $ax - by = d$ is

$$\left.\begin{aligned} x &= (-1)^{n-1} Q_{n-1}, \\ y &= (-1)^{n-1} P_{n-1}. \end{aligned}\right\} \tag{2.51}$$

Remark 2.5 We have already seen a method to solve the linear Diophantine equations by applying Euclid's algorithm to a and b and working backwards through the resulting equations (the so-called extended Euclid's algorithm). Our new method here turns out to be equivalent to this since the continued fraction for a/b is derived from Euclid's algorithm. However, it is quicker to generate the convergents P_i/Q_i using the recurrence relations than to work backwards through the equations in Euclid's algorithm.

Example 2.27 Use the continued fraction method to solve the following linear Diophantine equation:

$$364x - 227y = 1.$$

Since $364/227$ can be expanded as a finite continued fraction with convergents

$$\left[1, 2, \frac{3}{2}, \frac{5}{3}, \frac{8}{5}, \frac{85}{53}, \frac{93}{58}, \frac{364}{227}\right]$$

we have

$$x = (-1)^{n-1} q_{n-1} = (-1)^{7-1} 58 = 58,$$
$$y = (-1)^{n-1} p_{n-1} = (-1)^{7-1} 93 = 93.$$

That is,

$$364 \cdot 58 - 227 \cdot 93 = 1.$$

Example 2.28 Use the continued fraction method to solve the following linear Diophantine equation:

$$20719x + 13871y = 1.$$

Note first that

$$20719x + 13871y = 1 \Longleftrightarrow 20719x - (-13871y) = 1.$$

Now since 20719/13871 can be expanded as a finite simple continued fraction with convergents

$$\left[1, \frac{3}{2}, \frac{118}{79}, \frac{829}{555}, \frac{947}{634}, \frac{1776}{1189}, \frac{2723}{1823}, \frac{4499}{3012}, \frac{20719}{13871}\right],$$

we have

$$x = (-1)^{n-1} q_{n-1} = (-1)^{8-1} 3012 = -3012,$$
$$y = (-1)^{n-1} p_{n-1} = (-1)^{8-1} 4499 = -4499.$$

That is,

$$20719 \cdot (-3012) - 13871 \cdot (-4499) = 1.$$

Remark 2.6 To find the integral solution to equation $ax + by = d$, the equation

$$(-1)^{n-1} a q_{n-1} - (-1)^{n-1} b p_{n-1} = d$$

for $ax - by = d$ must be changed to

$$(-1)^{n-1} a q_{n-1} + (-1)(-1)^{n-1} b p_{n-1} = d.$$

That is,

$$(-1)^{n-1}aq_{n-1} + (-1)^n bp_{n-1} = d \tag{2.52}$$

Thus a solution to equation $ax + by = d$ is given by

$$\begin{cases} x = (-1)^{n-1}q_{n-1}, \\ y = (-1)^n p_{n-1}. \end{cases} \tag{2.53}$$

Generally, we have the following four cases:

$$\begin{cases} x = (-1)^{n-1}q_{n-1}, \\ y = (-1)^{n-1} p_{n-1} \end{cases} \quad \text{for } ax - by = d. \tag{2.54}$$

$$\begin{cases} x = (-1)^{n-1}q_{n-1}, \\ y = (-1)^n p_{n-1} \end{cases} \quad \text{for } ax + by = d. \tag{2.55}$$

$$\begin{cases} x = (-1)^n q_{n-1}, \\ y = (-1)^{n-1} p_{n-1} \end{cases} \quad \text{for } -ax - by = d. \tag{2.56}$$

$$\begin{cases} x = (-1)^n q_{n-1}, \\ y = (-1)^n p_{n-1} \end{cases} \quad \text{for } -ax + by = d. \tag{2.57}$$

All the above four cases are, in fact, of the same type of linear Diophantine equations.

Example 2.29 Use the continued fraction method to solve the following bilinear Diophantine equation:

$$9x + 16y = 1.$$

Since $9/16$ can be expanded as a finite continued fraction with convergents

$$\left[0, 1, \frac{1}{2}, \frac{4}{7}, \frac{9}{16}\right]$$

then we have

$$\begin{cases} x = (-1)^{n-1}q_{n-1} = (-1)^{4-1}7 = -7, \\ y = (-1)^n p_{n-1} = (-1)^4 4 = 4. \end{cases}$$

That is,

$$9 \cdot (-7) + 16 \cdot 4 = 1.$$

Problems for Section 2.2

1. In bases $2 \le b \le 12$, the number 1010101 are always composite:

$$\begin{aligned}
1010101_2 &= 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 = 85 = 5 \cdot 17 \\
1010101_3 &= 1 \cdot 3^6 + 1 \cdot 3^4 + 1 \cdot 3^2 + 1 \cdot 3^0 = 820 = 2^2 \cdot 5 \cdot 41 \\
1010101_4 &= 1 \cdot 4^6 + 1 \cdot 4^4 + 1 \cdot 4^2 + 1 \cdot 4^0 = 4369 = 17 \cdot 257 \\
1010101_5 &= 1 \cdot 5^6 + 1 \cdot 5^4 + 1 \cdot 5^2 + 1 \cdot 5^0 = 16276 = 2^2 \cdot 13 \cdot 313 \\
&\vdots \\
1010101_{10} &= 1 \cdot 10^6 + 1 \cdot 10^4 + 1 \cdot 10^2 + 1 \cdot 10^0 = 1010101 = 73 \cdot 101 \cdot 137 \\
1010101_{11} &= 1 \cdot 11^6 + 1 \cdot 11^4 + 1 \cdot 11^2 + 1 \cdot 11^0 = 1786324 = 2^2 \cdot 61 \cdot 7321 \\
1010101_{12} &= 1 \cdot 12^6 + 1 \cdot 12^4 + 1 \cdot 12^2 + 1 \cdot 12^0 = 3006865 = 5 \cdot 29 \cdot 89 \cdot 233
\end{aligned}$$

   (1) Show that in any basis the number 1010101 cannot be prime.
   (2) How about the number 11010101? Can this number be always composite in any basis $b \ge 2$? For $2 \le b \le 100$, list the numbers 11010101_b which are not composite.
2. A number is a *perfect square*, sometimes also called a *square number*, if it is of the form n^2 for some integer n, e.g., 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 122, 144, 169, 196 are the first 15 perfect squares.
   (1) Show that the product of four consecutive positive integers $a, a+1, a+2, a+3$ cannot be a perfect square.
   (2) Are there infinitely many primes p such that $p-1$ is a perfect square? This is one of the four problems proposed by the German number theorist Edmund Landau (1877–1938) in 1921; it is unsolved to this day.
   (3) Show that there is a prime number between two consecutive perfect squares n^2 and $(n+1)^2$ for every positive integer n. This is the famous Legendre conjecture, unsolved to this day.
   (4) Show that there is a prime number between consecutive perfect squares n^2 and $(n+1)^2$ for every positive integer n. (This is the famous Legendre conjecture; it was unproven as of 2008. However, partial results have been obtained. For example, a result due to Ingham shows that there is a prime between n^3 and $(n+1)^3$ for every positive integer n, and the Chinese Mathematician J. R. Chen showed in 1975 that there always exists a number P which is either a prime or product of two primes between the consecutive perfect squares n^2 and $(n+1)^2$.)
   (5) Are there infinitely many primes p such that $p-1$ is a perfect square? In other words: Are there infinitely many primes (called generalized Fermat primes) of the form n^2+1? (This is one of the four problems about prime numbers proposed by the German mathematician Edmund Landau in the 1912 International Congress of Mathematicians. Although the problem has still not been settled, some progress has been made, for example, the famous The Bombieri–Friedlander–Iwaniec theorem shows that infinitely many primes are of the form $x^2 + y^4$.)
   (6) Show that a perfect square cannot be a perfect number.

3. A positive integer that has no perfect square divisors except 1 is called square-free, for example,10 is square-free but 18 is not, as it is divisible by $9 = 3^2$. The first 25 square-free numbers are as follows:

   1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34, 35, 37, 38.

   (1) Show that n is square-free if and only if in every factorization $n = ab$, $\gcd(a, b) = 1$.
   (2) The radical of an integer is always square-free. (The radical of a positive integer n is defined to be the product of the prime numbers dividing n:

$$\mathrm{Rad}(n) = \prod_{p|n} p$$

   e.g., $n = 600 = 2^3 \cdot 3 \cdot 5^2$, $\mathrm{Rad}(n) = 2 \cdot 3 \cdot 5 = 30$.)
   (3) Show that each odd prime p can be written as the difference of two perfect squares.
4. Show that $7 \mid \left(1^{47} + 2^{47} + 3^{47} + 4^{47} + 5^{47} + 6^{47}\right)$.
5. Let p_k be the kth prime. Prove that

$$p_k = 1 + \sum_{m=1}^{2^k} \left\lfloor \left\lfloor \frac{k}{1 + \sum_{j=2}^{m} \left\lfloor \frac{(j-1)! + 1}{j} - \left\lfloor \frac{(j-1)!}{j} \right\rfloor \right\rfloor} \right\rfloor^{1/k} \right\rfloor.$$

6. Use mathematical induction to prove that when $n \geq 1$,

$$F_0 F_1 F_2 \cdots F_{n-1} = F_{n-2} \tag{2.58}$$

   where $F_i = 2^{2^i} + 1,\ i = 0, 1, 2, 3, \ldots$ are the Fermat numbers. Use (2.58) to prove that if m and n are distinctive positive integers, then

$$\gcd(F_m, F_n) = 1. \tag{2.59}$$

   Furthermore, use (2.59) to prove that there are infinitely many primes.
7. Let n be a positive integer. Find

$$\gcd\left(\binom{2n}{1}, \binom{2n}{3}, \binom{2n}{5}, \ldots, \binom{2n}{2n-1}\right)$$

   where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

   is the binomial coefficient.

8. Find the inverse of the matrix

$$\begin{pmatrix} 1 & 1 \\ 6 & 1 \end{pmatrix} \pmod{26}.$$

Find also all the values of $b \bmod 26$ such that

$$\begin{pmatrix} 1 & 1 \\ b & 1 \end{pmatrix} \pmod{26}$$

is invertible.

9. Let p be prime and n a positive integer. An integer $n \geq 2$ is called a powerful number if $p \mid n$ implies $p^2 \mid n$. That is, n is of the form $n = a^2b^3$, where a and b are positive integers. Find all the powerful numbers up to 1000, and prove that every sufficiently large integer is a sum of at most three *powerful numbers*. (this result was proved by Heath-Brown of Oxford University in 1987).
10. Prove that none of the following numbers is prime:

$$12321,\ 1234321,\ 123454321,\ 12345654321,\ 1234567654321,$$

$$123456787654321,\ 12345678987654321$$

11. For any positive integers a and b, prove that

$$ab = \gcd(a, b)\mathrm{lcm}(a, b).$$

12. Prove that if Euclid's algorithm runs with $a = f_{k+2}$ and $b = f_{k+1}$, then exactly k divisions are needed for computing $\gcd(a, b)$, where $f_0 = 0$, $f_1 = 1$, and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$ are defined to be the Fibonacci numbers beginning with numbers 0, 1, 1, 2, 3, 5, 8, 13, ...
13. Use the continued fraction method to solve $377x - 120y = -3$ and $314x \equiv 271 \pmod{11111}$.
14. Prove that if α is an irrational number, then there exist infinitely many rational numbers $\dfrac{p}{q}$ such that

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{q^2}.$$

15. Prove that if α is an irrational number and $\dfrac{P_i}{Q_i}$ the ith convergent of the continued fraction of α, then

$$\left|\alpha - \frac{P_i}{Q_i}\right| < \frac{1}{Q_i Q_{i+1}}.$$

16. Prove that if α is an irrational number and $\dfrac{c}{d}$ is a rational number with $d > 1$ such that

$$\left|\alpha - \frac{c}{d}\right| < \frac{1}{2d^2},$$

then $\dfrac{c}{d}$ is one of the convergents of the infinite continued fraction of α.

17. Let $\pi = 3.14159926\cdots$. Prove that the first three convergents to π are $\dfrac{22}{7}$, $\dfrac{333}{106}$ and $\dfrac{355}{113}$. Verify that

$$\left|\pi - \frac{355}{113}\right| < 10^{-6}.$$

18. Prove that the denominators Q_n in the convergents to any real number θ satisfy that

$$Q_n \leq \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}.$$

19. Prove that if $m < n$, then

$$(2^{2^m}+1) \nmid (2^{2^n}+1), \quad \gcd((2^{2^m}+1),\ (2^{2^n}+1)) = 1.$$

20. Find the integer solution (x, y, z) to the Diophantine equation $35x + 55y + 77z = 1$.

2.3 Arithmetic Functions

This section discusses some of the most useful arithmetic functions such as $\sigma(n)$, $\tau(n)$, $\phi(n)$, $\lambda(n)$, and $\mu(n)$.

Definition 2.28 A function f is called an *arithmetic function* or a *number-theoretic function* if it assigns to each positive integer n a unique real or complex number $f(n)$. Typically, an arithmetic function is a real-valued function whose domain is the set of positive integers.

Example 2.30 The equation

$$f(n) = \sqrt{n}, \quad n \in \mathbb{Z}^+ \tag{2.60}$$

defines an arithmetic function f which assigns the real number $\sqrt{n}$ to each positive integer n.

Definition 2.29 A real function f defined on the positive integers is said to be *multiplicative* if

$$f(m)f(n) = f(mn), \quad \forall m, n \in \mathbb{Z}^+, \tag{2.61}$$

where $\gcd(m, n) = 1$. If

$$f(m)f(n) = f(mn), \quad \forall m, n \in \mathbb{Z}^+, \tag{2.62}$$

then f is *completely multiplicative*. Every completely multiplicative function is multiplicative.

Theorem 2.30 *Let*

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}$$

be the prime factorization of n and let f be a multiplicative function, then

$$f(n) = \prod_{i=1}^{k} f(p_i^{\alpha_i}).$$

Proof: Clearly, if $k = 1$, we have the identity, $f(p_i^{\alpha_i}) = f(p_i^{\alpha_i})$. Assume that the representation is valid whenever n has r or fewer distinct prime factors, and consider $n = \prod_{i=1}^{r+1} f(p_i^{\alpha_i})$. Since $\gcd\left(\prod_{i=1}^{r} p_i^{\alpha_i},\ p_{r+1}^{\alpha_{r+1}}\right) = 1$ and f is multiplicative, we have

$$\begin{aligned} f(n) &= f\left(\prod_{i=1}^{r+1} p_i^{\alpha_i}\right) \\ &= f\left(\prod_{i=1}^{r} p_i^{\alpha_i} \cdot p_{r+1}^{\alpha_{r+1}}\right) \\ &= f\left(\prod_{i=1}^{r} p_i^{\alpha_i}\right) \cdot f\left(p_{r+1}^{\alpha_{r+1}}\right) \\ &= \prod_{i=1}^{r} f\left(p_i^{\alpha_i}\right) \cdot f\left(p_{r+1}^{\alpha_{r+1}}\right) \\ &= \prod_{i=1}^{r+1} f\left(p_i^{\alpha_i}\right). \end{aligned}$$

■

Theorem 2.31 *If f is multiplicative and if g is given by*

$$g(n) = \sum_{d|n} f(d) \tag{2.63}$$

where the sum is over all divisors d of n, then g is also multiplicative.

Proof: Since f is multiplicative, if $\gcd(m, n) = 1$, then

$$\begin{aligned} g(mn) &= \sum_{d|mn} f(d) \\ &= \sum_{d_1|m\, d_2|n} f(d_1 d_2) \\ &= \sum_{d_1|m\, d_2|n} f(d_1) f(d_2) \\ &= \sum_{d_1|m} f(d_1) \sum_{d_2|n} f(d_2) \\ &= g(m)g(n). \end{aligned}$$

■

Theorem 2.32 *If f and g are multiplicative, then so is*

$$F(n) = \sum_{d|m} f(d) g\left(\frac{n}{d}\right).$$

Proof: If $\gcd(m, n) = 1$, then $d \mid mn$ if and only if $d = d_1 d_2$, where $d_1 \mid m$ and $d_2 \mid n$, $\gcd(d_1, d_2) = 1$ and $\gcd(m/d_1, n/d_2) = 1$. Thus,

$$\begin{aligned} F(mn) &= \sum_{d|mn} f(d) g\left(\frac{mn}{d}\right) \\ &= \sum_{d_1|m} \sum_{d_2|n} f(d_1 d_2) g\left(\frac{mn}{d_1 d_2}\right) \\ &= \sum_{d_1|m} \sum_{d_2|n} f(d_1) f(d_2) g\left(\frac{m}{d_1}\right) g\left(\frac{n}{d_2}\right) \\ &= \left[\sum_{d_1|m} f(d_1) g\left(\frac{m}{d_1}\right)\right] \left[\sum_{d_2|m} f(d_2) g\left(\frac{n}{d_2}\right)\right] \\ &= F(m)F(n). \end{aligned}$$

■

Definition 2.30 Let n be a positive integer. Then the arithmetic functions $\tau(n)$ and $\sigma(n)$ are defined as follows:

$$\tau(n) = \sum_{d|n} 1, \qquad \sigma(n) = \sum_{d|n} d. \tag{2.64}$$

That is, $\tau(n)$ designates the number of all positive divisors of n, and $\sigma(n)$ designates the sum of all positive divisors of n.

Example 2.30 By Definition 2.30, we have

n	1	2	3	4	5	6	7	8	9	10	100	101	220	284
$\tau(n)$	1	2	2	3	2	4	2	4	3	4	9	2	12	6
$\sigma(n)$	1	3	4	7	6	12	8	15	13	18	217	102	504	504

Lemma 2.1 If n is a positive integer greater than 1 and has the following standard prime factorization form

$$n = \prod_{i}^{k} p_i^{\alpha_i},$$

then the positive divisors of n are precisely those integers d of the form

$$d = \prod_{i}^{k} p_i^{\beta_i},$$

where $0 \le \beta_i \le \alpha_i$.

Proof: If $d \mid n$, then $n = dq$. By the Fundamental Theorem of Arithmetic, the prime factorization of n is unique, so the prime numbers in the prime factorization of d must occur in p_j, $(j = 1, 2, \ldots, k)$. Furthermore, the power β_j of p_j occurring in the prime factorization of d cannot be greater than α_j, that is, $\beta_j \le \alpha_j$. Conversely, when $\beta_j \le \alpha_j$, d clearly divides n. ■

Theorem 2.33 *Let n be a positive integer. Then*

(1) $\tau(n)$ *is multiplicative. That is,*

$$\tau(mn) = \tau(m)\tau(n) \tag{2.65}$$

where $\gcd(m, n) = 1$.

(2) *If n is a prime, say p, then $\tau(p) = 2$. More generally, if n is a prime power p^α, then*

$$\tau(p^\alpha) = \alpha + 1. \tag{2.66}$$

(3) *If n is a composite and has the standard prime factorization form, then*

$$\tau(n) = (\alpha_1 + 1)(\alpha_2 + 1)\cdots(\alpha_k + 1) = \prod_{i=1}^{k}(\alpha_i + 1). \tag{2.67}$$

Proof:

(1) Since the constant function $f(n) = 1$ is multiplicative and $\tau(n) = \sum_{d|n} 1$, the result follows immediately from Theorem 2.31.
(2) Clearly, if n is a prime, there are only two divisors, namely, 1 and n itself. If $n = p^\alpha$, then by Lemma 2.1, the positive divisors of n are precisely those integers $d = p^\beta$, with $0 \le \beta \le \alpha$. Since there are $\alpha + 1$ choices for the exponent β, there are $\alpha + 1$ possible positive divisors of n.
(3) By Lemma 2.1 and Part (2) of this theorem, there are $\alpha_1 + 1$ choices for the exponent β_1, $\alpha_2 + 1$ choices for the exponent β_2, $\cdots$, $\alpha_k + 1$ choices for the exponent β_k. From the multiplication principle it follows that there are $(\alpha_1 + 1)(\alpha_2 + 1)\cdots(\alpha_k + 1)$ different choices for the $\beta_1, \beta_2, \ldots, \beta_k$, thus that many divisors of n. Therefore, $\tau(n) = (\alpha_1 + 1)(\alpha_2 + 1)\cdots(\alpha_k + 1)$. ∎

Theorem 2.34 *The product of all divisors of a number n is*

$$\prod_{d|n} d = n^{\tau(n)/2}. \tag{2.68}$$

Proof: Let d denote an arbitrary positive divisor of n, so that

$$n = dd'$$

for some d'. As d ranges over all $\tau(n)$ positive divisors of n, there are $\tau(n)$ such equations. Multiplying these together, we get

$$n^{\tau(n)} = \prod_{d|n} d \prod_{d'|n} d'.$$

But as d runs through the divisors of n, so does d', hence

$$\prod_{d|n} d = \prod_{d'|n} d'.$$

So,

$$n^{\tau(n)} = \left(\prod_{d|n} d \right)^2,$$

or equivalently

$$n^{\tau(n)/2} = \prod_{d|n} d.$$

■

Example 2.32 Let $n = 1371$, then

$$\tau(1371) = 4.$$

Therefore

$$\prod d = 1371^{4/2} = 1879641.$$

It is of course true, since

$$d(1371) = \{1, 3, 457, 1371\}$$

implies that

$$\prod d = 1 \cdot 3 \cdot 457 \cdot 1371 = 1879641.$$

Theorem 2.35 *Let n be a positive integer. Then*

(1) $\sigma(n)$ *is multiplicative. That is,*

$$\sigma(mn) = \sigma(m)\sigma(n) \tag{2.69}$$

where $\gcd(m, n) = 1$.

(2) *If n is a prime, say p, then* $\sigma(p) = p + 1$. *More generally, if n is a prime power* p^α, *then*

$$\sigma(p^\alpha) = \frac{p^{\alpha+1} - 1}{p - 1}. \tag{2.70}$$

(3) *If n is a composite and has the standard prime factorization form, then*

$$\sigma(n) = \frac{p_1^{\alpha_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{\alpha_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{\alpha_k+1} - 1}{p_k - 1}$$
$$= \prod_{i=1}^{k} \frac{p_i^{\alpha_i+1} - 1}{p_i - 1}. \tag{2.71}$$

Proof:

(1) The results follows immediately from Theorem 2.31 since the identity function $f(n) = n$ and $\sigma(n)$ can be represented in the form $\sigma(n) = \sum_{d|n} d$.
(2) Left as an exercise; we prove the most general case in Part (3).
(3) The sum of the divisors of the positive integer

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$$

can be expressed by the product

$$\left(1 + p_1 + p_1^2 + \cdots + p_1^{\alpha_1}\right)\left(1 + p_2 + p_2^2 + \cdots + p_2^{\alpha_2}\right)$$
$$\cdots \left(1 + p_k + p_k^2 + \cdots + p_k^{\alpha_k}\right).$$

Using the finite geometric series

$$1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1,}$$

we simplify each of the k sums in the above product to find that the sum of the divisors can be expressed as

$$\sigma(n) = \frac{p_1^{\alpha_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{\alpha_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{\alpha_k+1} - 1}{p_k - 1}$$
$$= \prod_{i=1}^{k} \frac{p_i^{\alpha_i+1} - 1}{p_i - 1}. \tag{2.72}$$

■

Definition 2.31 Let n be a positive integer. *Euler's (totient) ϕ-function*, $\phi(n)$, is defined to be the number of positive integers k less than n which are relatively prime to n:

$$\phi(n) = \sum_{\substack{0 \le k < n \\ \gcd(k,n)=1}} 1. \tag{2.73}$$

Example 2.33 By Definition 2.31, we have

n	1	2	3	4	5	6	7	8	9	10	100	101	102	103
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	40	100	32	102

Lemma 2.2 For any positive integer n,

$$\sum_{d|n} \phi(d) = n. \tag{2.74}$$

Proof: Let n_d denote the number of elements in the set $\{1, 2, \ldots, n\}$ having a greatest common divisor of d with n. Then

$$n = \sum_{d|n} n_d = \sum_{d|n} \phi\left(\frac{n}{d}\right) = \sum_{d|n} \phi(d).$$

■

Theorem 2.36 *Let n be a positive integer and* $\gcd(m, n) = 1$. *Then*

(1) *Euler's ϕ-function is multiplicative. That is,*

$$\phi(mn) = \phi(m)\phi(n) \tag{2.75}$$

where $\gcd(m, n) = 1$.

(2) *If n is a prime, say p, then*

$$\phi(p) = p - 1. \tag{2.76}$$

(Conversely, if p is a positive integer with $\phi(p) = p - 1$, then p is prime.)

(3) *If n is a prime power p^α with $\alpha > 1$, then*

$$\phi(p^\alpha) = p^\alpha - p^{\alpha-1}. \tag{2.77}$$

(4) *If n is a composite and has the standard prime factorization form, then*

$$\begin{aligned} \phi(n) &= p_1^{\alpha_1}\left(1 - \frac{1}{p_1}\right) p_2^{\alpha_2}\left(1 - \frac{1}{p_2}\right) \cdots p_k^{\alpha_k}\left(1 - \frac{1}{p_k}\right) \\ &= n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right). \end{aligned} \tag{2.78}$$

Proof:

(1) Use Theorem 2.32 and Lemma 2.2. (A nicer way to prove this result is to use the Chinese Remainder theorem, which will be discussed in Section 2.4.)
(2) If n is prime, then $1, 2, \ldots, n-1$ are relatively prime to n, so it follows from the definition of Euler's ϕ-function that $\phi(n) = n - 1$. Conversely, if n is not prime, n has a divisor d such that $\gcd(d, n) \neq 1$. Thus, there is at least one positive integer less than n that is not relatively prime to n, and hence $\phi(n) \leq n - 2$.
(3) Note that $\gcd(n, p^\alpha) = 1$ if and only if $p \nmid n$. There are exactly $p^{\alpha-1}$ integers between 1 and p^α divisible by p, namely,

$$p, 2p, 3p, \ldots, (p^{\alpha-1})p.$$

Thus, the set $\{1, 2, \ldots, p^\alpha\}$ contains exactly $p^\alpha - p^{\alpha-1}$ integers that are relatively prime to p^α, and so by the definition of the ϕ-function, $\phi(p^\alpha) = p^\alpha - p^{\alpha-1}$.
(4) By Part (1) of this theorem, ϕ-function is multiplicative, thus

$$\phi(n) = \phi\left(p_1^{\alpha_1}\right)\phi\left(p_2^{\alpha_2}\right)\cdots\phi\left(p_k^{\alpha_k}\right).$$

In addition, by Part (3) of this theorem and Theorem 2.30, we have

$$\begin{aligned}
\phi(n) &= p_1^{\alpha_1}\left(1 - \frac{1}{p_1}\right)p_2^{\alpha_2}\left(1 - \frac{1}{p_2}\right)\cdots p_k^{\alpha_k}\left(1 - \frac{1}{p_k}\right) \\
&= p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_k^{\alpha_k}\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right) \\
&= n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right) \\
&= n\prod_{i=1}^{k}\left(1 - \frac{1}{p_i}\right).
\end{aligned}$$

■

Definition 2.32 Carmichael's λ-function, $\lambda(n)$, is defined as follows

$$\left.\begin{array}{ll}
\lambda(p) = \phi(p) = p - 1 & \text{for prime } p, \\
\lambda(p^\alpha) = \phi(p^\alpha) & \text{for } p = 2 \text{ and } \alpha \leq 2, \\
 & \text{and for } p \geq 3 \\
\lambda(2^\alpha) = \dfrac{1}{2}\phi(2^\alpha) & \text{for } \alpha \geq 3 \\
\lambda(n) = \operatorname{lcm}\left(\lambda(p_1^{\alpha_1}), \lambda(p_2^{\alpha_2}), \ldots, \lambda(p_k^{\alpha_k})\right) & \text{if } n = \prod_{i=1}^{k} p_i^{\alpha_i}.
\end{array}\right\} \tag{2.79}$$

Example 2.34 By Definition 2.32, we have

n	1	2	3	4	5	6	7	8	9	10	100	101	102	103
$\lambda(n)$	1	1	2	2	4	2	6	2	6	4	20	100	16	102

Example 2.35 Let $n = 65520 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13$, and $a = 11$. Then $\gcd(65520, 11) = 1$ and we have

$$\phi(65520) = 8 \cdot 6 \cdot 4 \cdot 6 \cdot 12 = 13824,$$
$$\lambda(65520) = \text{lcm}(4, 6, 4, 6, 12) = 12.$$

Definition 2.33 Let n be a positive integer. Then the *Möbius μ-function*, $\mu(n)$, is defined as follows:

$$\mu(n) = \begin{cases} 1, & \text{if } n = 1, \\ 0, & \text{if } n \text{ contains a squared factor,} \\ (-1)^k, & \text{if } n = p_1 p_2 \cdots p_k \text{ is the product of} \\ & \quad k \text{ distinct primes.} \end{cases} \tag{2.80}$$

Example 2.36 By Definition 2.80, we have

n	1	2	3	4	5	6	7	8	9	10	100	101	102
$\mu(n)$	1	-1	-1	0	-1	1	-1	0	0	1	0	-1	-1

Theorem 2.37 *Let $\mu(n)$ be the Möbius function. Then*

(1) *$\mu(n)$ is multiplicative, that is, for* $\gcd(m, n) = 1$,

$$\mu(mn) = \mu(m)\mu(n). \tag{2.81}$$

(2) *Let*

$$v(n) = \sum_{d|n} \mu(d). \tag{2.82}$$

Then

$$\nu(n) = \begin{cases} 1, & \text{if } n = 1, \\ 0, & \text{if } n > 1. \end{cases} \tag{2.83}$$

Proof:

(1) If either $p^2 \mid m$ or $p^2 \mid n$, p is a prime, then $p^2 \mid mn$. Hence, $\mu(mn) = 0 = \mu(m)\mu(n)$. If both m and n are square-free integers, say, $m = p_1 p_2 \cdots p_s$ and $n = q_1 q_2 \cdots q_t$. Then

$$\begin{aligned} \mu(mn) &= \mu(p_1 p_2 \cdots p_s q_1 q_2 \cdots q_t) \\ &= (-1)^{s+t} \\ &= (-1)^s (-1)^t \\ &= \mu(m)\mu(n). \end{aligned}$$

(2) If $n = 1$, then $\nu(1) = \sum_{d \mid n} \nu(d) = \mu(1) = 1$. If $n > 1$, since $\nu(n)$ is multiplicative, we need only to evaluate ν on prime to powers. In addition, if p is prime,

$$\begin{aligned} \nu(p^\alpha) &= \sum_{d \mid p^\alpha} \mu(d) \\ &= \mu(1) + \mu(p) + \mu(p^2) + \cdots + \mu(p^\alpha) \\ &= 1 + (-1) + 0 + \cdots + 0 \\ &= 0. \end{aligned}$$

Thus, $\nu(n) = 0$ for any positive integer n greater than 1. ∎

The importance of the Möbius function lies in the fact that it plays an important role in the inversion formula given in the following theorem. The formula involves a general arithmetic function f which is not necessarily multiplicative.

Theorem 2.38 (The Möbius inversion formula) *If f is any arithmetic function and if*

$$g(n) = \sum_{d \mid n} f(d), \tag{2.84}$$

then

$$f(n) = \sum_{d \mid n} \mu\left(\frac{n}{d}\right) g(d) = \sum_{d \mid n} \mu(d)\, g\left(\frac{n}{d}\right). \tag{2.85}$$

Proof: If f is an arithmetic function and $g(n) = \sum_{d|n} f(d)$. Then

$$\begin{aligned}\sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) &= \sum_{d|n} \mu(d) \sum_{a|(n/d)} f(a) \\ &= \sum_{d|n} \sum_{a|(n/d)} \mu(d) f(a) \\ &= \sum_{a|n} \sum_{d|(n/a)} f(a)\mu(d) \\ &= \sum_{a|n} f(a) \sum_{d|(n/a)} \mu(d) \\ &= f(n) \cdot 1 \\ &= f(n).\end{aligned}$$

■

The converse of Theorem 2.38 is also true and can be stated as follows:

Theorem 2.39 (The converse of the Möbius inversion formula) *If*

$$f(n) = \sum_{d|n} \mu\left(\frac{n}{d}\right) g(d), \tag{2.86}$$

then

$$g(n) = \sum_{d|n} f(d). \tag{2.87}$$

Note that the functions τ and σ

$$\tau(n) = \sum_{d|n} 1 \text{ and } \sigma(n) = \sum_{d|n} d$$

may be inverted to give

$$1 = \sum_{d|n} \mu\left(\frac{n}{d}\right) \tau(d) \text{ and } n = \sum_{d|n} \mu\left(\frac{n}{d}\right) \sigma(d)$$

for all $n \geq 1$. The relationship between Euler's ϕ-function and Möbius' μ-function is given by the following theorem.

Theorem 2.40 *For any positive integer n,*

$$\phi(n) = n \sum_{d|n} \frac{\mu(d)}{d}. \tag{2.88}$$

Proof: Apply Möbius inversion formula to

$$g(n) = n = \sum_{d|n} \phi(d)$$

we get

$$\begin{aligned} \phi(n) &= \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) \\ &= \sum_{d|n} \frac{\mu(d)}{d} n. \end{aligned}$$

■

Problems for Section 2.3

1. Let

$$\Lambda(n) = \begin{cases} \log p, & \text{if } n \text{ is a power of a prime } p \\ 0, & \text{otherwise} \end{cases}$$

   Evaluate

$$\sum_{d|n} \Lambda(d).$$

2. Evaluate

$$\sum_{d|n} \mu(d)\sigma(d)$$

   in terms of the distinctive prime factors of n.
3. Let $n > 1$ and a run over all integers with $1 \le a \le n$ and $\gcd(a, n) = 1$. Prove that

$$\frac{1}{n^3} \sum a^3 = \frac{1}{4}\phi(n)\left(1 + \frac{(-1)^k p_1 p_2 \cdots p_k}{n^2}\right),$$

   where p_1, $p_2 \cdots p_k$ are the distinct prime factors of n.

4. (Ramanujan sum) Let m, n be positive integers and d run over all divisors of $\gcd(m, n)$. Prove that

$$\sum d\mu\left(\frac{n}{d}\right) = \frac{\mu\left(\frac{n}{\gcd(m,n)}\right)\phi(n)}{\phi\left(\frac{n}{\gcd(m,n)}\right)}$$

5. (Lambert series) Prove that

$$\sum_{n=1}^{\infty} \frac{\phi(n)x^n}{1-x^n} = \frac{x}{(1-x)^2}.$$

6. Prove that

$$\sum_{n\le x} \frac{\phi(n)}{n} = \frac{6x}{\pi^2} + \mathcal{O}(\log x).$$

7. Let $p_1, p_2, \ldots, p_k$ be distinct primes. Show that

$$\frac{(p_1+1)(p_2+1)\cdots(p_k+1)}{p_1p_2\cdots p_k} \le 2 \le \frac{p_1p_2\cdots p_k}{(p_1-1)(p_2-1)\cdots(p_k-1)}$$

   is the necessary condition for

$$n = p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_k^{\alpha_k}$$

   to be a perfect number.
8. Show that $\tau(n)$ is odd if and only if n is a perfect square, and that $\sigma(n)$ is odd if and only if n is a square or two times a square.
9. Show that for $n > 2$,

$$\sum_{\substack{k=1\\ \gcd(k,n)=1}}^{\phi(n)} \frac{1}{k}$$

   cannot be an integer.
10. Prove that for each positive integer n,

$$\sum_{\substack{k=1\\ \gcd(k,n)=1}}^{n} k = \frac{n}{2}\phi(n) + \frac{n}{2}\sum_{d|n}\mu(d).$$

$$\sum_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} k^2 = \frac{n^2}{3}\phi(n) + \frac{n^2}{2}\sum_{d|n}\mu(d) + \frac{n}{6}\prod_{p|n}(1-p).$$

$$\sum_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} k^3 = \frac{n^3}{4}\phi(n) + \frac{n^3}{2}\sum_{d|n}\mu(d) + \frac{n^2}{4}\prod_{p|n}(1-p).$$

2.4 Congruence Theory

The notion of congruences was first introduced by Gauss, who gave their definition in his celebrated *Disquisitiones Arithmeticae* in 1801, though the ancient Greeks and Chinese had the idea first.

Definition 2.34 Let a be an integer and n a positive integer greater than 1. We define "$a \bmod n$" to be the remainder r when a is divided by n, that is

$$r = a \bmod n = a - \lfloor a/n \rfloor n. \tag{2.89}$$

We may also say that "r is equal to a reduced modulo n".

Remark 2.7 It follows from the above definition that $a \bmod n$ is the integer r such that $a = \lfloor a/n \rfloor n + r$ and $0 \leq r < n$, which was known to the ancient Greeks 2000 years ago.

Example 2.37 The following are some examples of $a \bmod n$:

$$\begin{aligned} &35 \bmod 12 = 11, \\ &-129 \bmod 7 = 4, \\ &3210 \bmod 101 = 79, \\ &1412^{13115} \bmod 12349 = 1275. \end{aligned}$$

Given the well-defined notion of the remainder of one integer when divided by another, it is convenient to provide a special notion to indicate equality of remainders.

Definition 2.35 Let a and b be integers and n a positive integer. We say that "a is *congruent* to b modulo n", denoted by

$$a \equiv b \,(\bmod n) \tag{2.90}$$

if n is a divisor of $a - b$, or equivalently, if $n \mid (a - b)$. Similarly, we write

$$a \not\equiv b \,(\bmod n) \tag{2.91}$$

if a is not congruent (or incongruent) to b modulo n, or equivalently, if $n \nmid (a-b)$. Clearly, for $a \equiv b \pmod{n}$ (resp. $a \not\equiv b \pmod{n}$), we can write $a = kn + b$ (resp. $a \neq kn + b$) for some integer k. The integer n is called the *modulus*.

Clearly,

$$a \equiv b \pmod{n} \Longleftrightarrow n \mid (a-b) \Longleftrightarrow a = kn + b, \quad k \in \mathbb{Z}$$

and

$$a \not\equiv b \pmod{n} \Longleftrightarrow n \nmid (a-b) \Longleftrightarrow a \neq kn + b, \quad k \in \mathbb{Z}$$

So, the above definition of congruences, introduced by Gauss in his *Disquisitiones Arithmeticae*, does not offer any new idea other than the divisibility relation, since "$a \equiv b \pmod{n}$" and "$n \mid (a-b)$" (resp. "$a \not\equiv b \pmod{n}$" and "$n \nmid (a-b)$") have the same meaning, although each of them has its own advantages. However, Gauss did present a *new* way (i.e., congruences) of looking at the old things (i.e., divisibility); this is exactly what we are interested in. It is interesting to note that the ancient Chinese mathematician Ch'in Chiu-Shao (1202–1261) had already noted the idea of congruences in his famous book *Mathematical Treatise in Nine Chapters* in 1247.

Definition 2.36 If $a \equiv b \pmod{n}$, then b is called a *residue* of a modulo n. If $0 \leq b \leq n-1$, b is called the *least non-negative residue* of a modulo n.

Remark 2.8 It is common, particularly in computer programs, to denote the least non-negative residue of a modulo n by $a \bmod n$. Thus, $a \equiv b \pmod{n}$ if and only if $a \bmod n = b \bmod n$, and, of course, $a \not\equiv b \pmod{n}$ if and only if $a \bmod n \neq b \bmod n$.

Example 2.38 The following are some examples of congruences or incongruences.

$$\begin{array}{rll} 35 \equiv 11 \pmod{12} & \text{since} & 12 \mid (35-11) \\ \not\equiv 12 \pmod{11} & \text{since} & 11 \nmid (35-12) \\ \equiv 2 \pmod{11} & \text{since} & 11 \mid (35-2). \end{array}$$

The congruence relation has many properties in common with the of equality relation. For example, we know from high-school mathematics that equality is

(1) reflexive: $a = a, \forall a \in \mathbb{Z}$;
(2) symmetric: if $a = b$, then $b = a, \forall a, b \in \mathbb{Z}$;
(3) transitive: if $a = b$ and $b = c$, then $a = c, \forall a, b, c \in \mathbb{Z}$.

We shall see that congruence modulo n has the same properties:

Theorem 2.41 *Let n be a positive integer. Then the congruence modulo n is*

(1) *reflexive: $a \equiv a \pmod{n}$, $\forall a \in \mathbb{Z}$;*
(2) *symmetric: if $a \equiv b \pmod{n}$, then $b \equiv a \pmod{n}$, $\forall a, b \in \mathbb{Z}$;*
(3) *transitive: if $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$, $\forall a, b, c \in \mathbb{Z}$.*

Proof:

(1) For any integer a, we have $a = 0 \cdot n + a$, hence $a \equiv a \pmod{n}$.
(2) For any integers a and b, if $a \equiv b \pmod{n}$, then $a = kn + b$ for some integer k. Hence $b = a - kn = (-k)n + a$, which implies $b \equiv a \pmod{n}$, since $-k$ is an integer.
(3) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a = k_1 n + b$ and $b = k_2 n + c$. Thus, we can get

$$a = k_1 n + k_2 n + c = (k_1 + k_2)n + c = k' n + c$$

which implies $a \equiv c \pmod{n}$, since k' is an integer.

■

Theorem 2.41 shows that congruence modulo n is an equivalence relation on the set of integers $\mathbb{Z}$. But note that the divisibility relation $a \mid b$ is reflexive, and transitive but not symmetric; in fact if $a \mid b$ and $b \mid a$ then $a = b$, so it is not an equivalence relation. The congruence relation modulo n partitions $\mathbb{Z}$ into n *equivalence classes*. In number theory, we call these classes *congruence classes*, or *residue classes*.

Definition 2.37 If $x \equiv a \pmod{n}$, then a is called a *residue* of x modulo n. The *residue class* of a modulo n, denoted by $[a]_n$ (or just $[a]$ if no confusion will be caused), is the set of all those integers that are congruent to a modulo n. That is,

$$[a]_n = \{x : x \in \mathbb{Z} \text{ and } x \equiv a \,(\text{mod } n)\} = \{a + kn : k \in \mathbb{Z}\}. \tag{2.92}$$

Note that writing $a \in [b]_n$ is the same as writing $a \equiv b \,(\text{mod } n)$.

Example 2.39 Let $n = 5$. Then there are five residue classes, modulo 5, namely the sets:

$$\begin{aligned}
[0]_5 &= \{\ldots, -15, -10, -5, 0, 5, 10, 15, 20, \ldots\},\\
[1]_5 &= \{\ldots, -14, -9, -4, 1, 6, 11, 16, 21, \ldots\},\\
[2]_5 &= \{\ldots, -13, -8, -3, 2, 7, 12, 17, 22, \ldots\},\\
[3]_5 &= \{\ldots, -12, -7, -2, 3, 8, 13, 18, 23, \ldots\},\\
[4]_5 &= \{\ldots, -11, -6, -1, 4, 9, 14, 19, 24, \ldots\}.
\end{aligned}$$

The first set contains all those integers congruent to 0 modulo 5, the second set contains all those congruent to 1 modulo 5, $\cdots$, and the fifth (i.e., the last) set contains all those congruent

to 4 modulo 5. So, for example, the residue class $[2]_5$ can be represented by any one of the elements in the set

$$\{\ldots, -13, -8, -3, 2, 7, 12, 17, 22, \ldots\}.$$

Clearly, there are infinitely many elements in the set $[2]_5$.

Example 2.40 In residue classes modulo 2, $[0]_2$ is the set of all even integers, and $[1]_2$ is the set of all odd integers:

$$[0]_2 = \{\ldots, -6, -4, -2, 0, 2, 4, 6, 8, \ldots\}$$
$$[1]_2 = \{\ldots, -5, -3, -1, 1, 3, 5, 7, 9, \ldots\}$$

Example 2.41 In congruence modulo 5, we have

$$[9]_5 = \{9 + 5k : k \in \mathbb{Z}\} = \{9, 9 \pm 5, 9 \pm 10, 9 \pm 15, \ldots\}$$
$$= \{\ldots, -11, -6, -1, 4, 9, 14, 19, 24, \ldots\}.$$

We also have

$$[4]_5 = \{4 + 5k : k \in \mathbb{Z}\} = \{4, 4 \pm 5, 4 \pm 10, 4 \pm 15, \ldots\}$$
$$= \{\ldots, -11, -6, -1, 4, 9, 14, 19, 24, \ldots\}.$$

So, clearly, $[4]_5 = [9]_5$.

Example 2.42 Let $n = 7$. There are seven residue classes, modulo 7. In each of these seven residue classes, there is exactly one least residue of x modulo 7. So the complete set of all least residues x modulo 7 is $\{0, 1, 2, 3, 4, 5, 6\}$.

Definition 2.38 The set of all residue classes modulo n, often denoted by $\mathbb{Z}/n\mathbb{Z}$ or $\mathbb{Z}/n\mathbb{Z}$, is

$$\mathbb{Z}/n\mathbb{Z} = \{[a]_n : 0 \leq a \leq n - 1\}. \tag{2.93}$$

Remark 2.9 One often sees the definition

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, \ldots, n - 1\}, \tag{2.94}$$

which should be read as equivalent to (2.93) with the understanding that 0 represents $[0]_n$, 1 represents $[1]_n$, 2 represents $[2]_n$, and so on; each class is represented by its least non-negative residue, but the underlying residue classes must be kept in mind. For example, a reference to $-a$ as a member of $\mathbb{Z}/n\mathbb{Z}$ is a reference to $[n-a]_n$, provided $n \geq a$, since $-a \equiv n - a \pmod{n}$.

The following theorem gives some elementary properties of residue classes:

Theorem 2.42 *Let n be a positive integer. Then we have*

(1) $[a]_n = [b]_n$ *if and only if* $a \equiv b \pmod{n}$*;*
(2) *Two residue classes modulo n are either disjoint or identical;*
(3) *There are exactly n distinct residue classes modulo n, namely,* $[0]_n, [1]_n, [2]_n, [3]_n, \ldots, [n-1]_n$*, and they contain all of the integers.*

Proof:

(1) If $a \equiv b \pmod{n}$, it follows from the transitive property of congruence that an integer is congruent to a modulo n if and only if it is congruent to b modulo n. Thus, $[a]_n = [b]_n$. To prove the converse, suppose $[a]_n = [b]_n$. Because $a \in [a]_n$ and $a \in [b]_n$, Thus, $a \equiv b \pmod{n}$.
(2) Suppose $[a]_n$ and $[b]_n$ have a common element c. Then $c \equiv a \pmod{n}$ and $c \equiv b \pmod{n}$. From the symmetric and transitive properties of congruence, it follows that $a \equiv b \pmod{n}$. From part (1) of this theorem, it follows that $[a]_n = [b]_n$. Thus, either $[a]_n$ and $[b]_n$ are disjoint or identical.
(3) If a is an integer, we can divide a by n to get

$$a = kn + r, \quad 0 \leq r < k.$$

Thus, $a \equiv r \pmod{n}$ and so $[a]_n = [r]_n$. This implies that a is in one of the residue classes $[0]_n, [1]_n, [2]_n, \ldots, [n-1]_n$, Because the integers $0, 1, 2, \ldots, n-1$ are incongruent modulo n, it follows that there are exactly n residue classes modulo n.

■

Definition 2.39 Let n be a positive integer. A set of integers $a_1, a_2, \ldots, a_n$ is called a *complete system of residues* modulo n, if the set contains exactly one element from each residue class modulo n.

Example 2.43 Let $n = 4$. Then $\{-12, 9, -6, -1\}$ is a complete system of residues modulo 4, since $-12 \in [0]$, $9 \in [1]$, $-6 \in [2]$, and $-1 \in [3]$. Of course, it can be easily verified that $\{12, -7, 18, -9\}$ is another complete system of residues modulo 4. It is clear that the simplest complete system of residues modulo 4 is $\{0, 1, 2, 3\}$, the set of all non-negative least residues modulo 4.

Example 2.44 Let $n = 7$. Then

$$\{x,\ x+3,\ x+3^2,\ x+3^3,\ x+3^4,\ x+3^5,\ x+3^6\}$$

is a complete system of residues modulo 7, for any $x \in \mathbb{Z}$. To see this let us first evaluate the powers of 3 modulo 7:

$$3 \qquad 3^2 \equiv 2 \pmod 7 \qquad 3^3 \equiv 6 \pmod 7$$
$$3^4 \equiv 4 \pmod 7 \qquad 3^5 \equiv 5 \pmod 7 \qquad 3^6 \equiv 1 \pmod 7$$

hence, the result follows from $x = 0$. Now the general result follows immediately, since $(x + 3^i) - (x + 3^j) = 3^i - 3^j$.

Theorem 2.43 *Let n be a positive integer and S a set of integers. S is a complete system of residues modulo n if and only if S contains n elements and no two elements of S are congruent, modulo n.*

Proof: If S is a complete system of residues, then the two conditions are satisfied. To prove the converse, we note that if no two elements of S are congruent, the elements of S are in different residue classes modulo n. Since S has n elements, all the residue classes must be represented among the elements of S. Thus, S is a complete system of residues modulo n. ∎

We now introduce one more type of system of residues, the *reduced* system of residues modulo n.

Definition 2.40 Let $[a]_n$ be a residue class modulo n. We say that $[a]_n$ is relatively prime to n if each element in $[a]_n$ is relatively prime to n.

Example 2.45 Let $n = 10$. Then the ten residue classes, modulo 10, are as follows:

$$[0]_{10} = \{\ldots, -30, -20, -10, 0, 10, 20, 30, \ldots\}$$
$$[1]_{10} = \{\ldots, -29, -19, -9, 1, 11, 21, 31, \ldots\}$$
$$[2]_{10} = \{\ldots, -28, -18, -8, 2, 12, 22, 32, \ldots\}$$
$$[3]_{10} = \{\ldots, -27, -17, -7, 3, 13, 23, 33, \ldots\}$$
$$[4]_{10} = \{\ldots, -26, -16, -6, 4, 14, 24, 34, \ldots\}$$
$$[5]_{10} = \{\ldots, -25, -15, -5, 5, 15, 25, 35, \ldots\}$$
$$[6]_{10} = \{\ldots, -24, -14, -4, 6, 16, 26, 36, \ldots\}$$
$$[7]_{10} = \{\ldots, -23, -13, -3, 7, 17, 27, 37, \ldots\}$$
$$[8]_{10} = \{\ldots, -22, -12, -2, 8, 18, 28, 38, \ldots\}$$
$$[9]_{10} = \{\ldots, -21, -11, -1, 9, 19, 29, 39, \ldots\}.$$

Clearly, $[1]_{10}$, $[3]_{10}$, $[7]_{10}$, and $[9]_{10}$ are residue classes that are relatively prime to 10.

Proposition 2.1 If a residue class modulo n has *one* element which is relatively prime to n, then every element in that residue class is relatively prime to n.

Proposition 2.2 If n is prime, then every residue class modulo n (except $[0]_n$) is relatively prime to n.

Definition 2.41 Let n be a positive integer, then $\phi(n)$ is the number of residue classes modulo n, which is relatively prime to n. A set of integers $\{a_1, a_2, \ldots, a_{\phi(n)}\}$ is called a *reduced system of residues*, if the set contains exactly one element from each residue class modulo n which is relatively prime to n.

Example 2.46 In Example 2.45, we know that $[1]_{10}, [3]_{10}, [7]_{10}$, and $[9]_{10}$ are residue classes that are relatively prime to 10, so by choosing -29 from $[1]_{10}$, -17 from $[3]_{10}$, 17 from $[7]_{10}$ and 39 from $[9]_{10}$, we get a reduced system of residues modulo 10: $\{-29, -17, 17, 39\}$. Similarly, $\{31, 3, -23, -1\}$ is another reduced system of residues modulo 10.

One method of obtaining a reduced system of residues is to start with a complete system of residues and delete those elements that are not relatively prime to the modulus n. Thus, the simplest reduced system of residues $(\text{mod } n)$ is just the collections of all integers in the set $\{0, 1, 2, \ldots, n-1\}$ that are relatively prime to n.

Theorem 2.44 *Let n be a positive integer, and S a set of integers. Then S is a reduced system of residues* $(\text{mod } n)$ *if and only if*

(1) *S contains exactly $\phi(n)$ elements;*
(2) *no two elements of S are congruent* $(\text{mod } n)$*;*
(3) *each element of S is relatively prime to n.*

Proof: It is obvious that a reduced system of residues satisfies the three conditions. To prove the converse, we suppose that S is a set of integers having the three properties. Because no two elements of S are congruent, the elements are in different residues modulo n. Since the elements of S are relatively prime n, there are in residue classes that are relatively prime n. Thus, the $\phi(n)$ elements of S are distributed among the $\phi(n)$ residue classes that are relatively prime n, one in each residue class. Therefore, S is a reduced system of residues modulo n. ∎

Corollary 2.3 *Let $\{a_1, a_2, \ldots, a_{\phi(n)}\}$ be a reduced system of residues modulo m, and suppose that* $\gcd(k, n) = 1$. *Then $\{ka_1, ka_2, \ldots, ka_{\phi(n)}\}$ is also a reduced system of residues modulo n.*

Proof: Left as an exercise. ∎

The finite set $\mathbb{Z}/n\mathbb{Z}$ is closely related to the infinite set $\mathbb{Z}$. So it is natural to ask if it is possible to define addition and multiplication in $\mathbb{Z}/n\mathbb{Z}$ and do some reasonable kind of

arithmetic there. Surprisingly, the addition, subtraction, and multiplication in $\mathbb{Z}/n\mathbb{Z}$ will be much the same as that in $\mathbb{Z}$.

Theorem 2.45 *For all $a, b, c, d \in \mathbb{Z}$ and $n \in \mathbb{Z}_{>1}$, if $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$. then*

(1) $a \pm b \equiv c \pm d \pmod{n}$;
(2) $a \cdot b \equiv c \cdot d \pmod{n}$;
(3) $a^m \equiv b^m \pmod{n}$, $\forall m \in \mathbb{Z}^+$.

Proof:

(1) Write $a = kn + b$ and $c = ln + d$ for some $k, l \in \mathbb{Z}$. Then $a + c = (k + l)n + b + d$. Therefore, $a + c = b + d + tn$, $t = k + l \in \mathbb{Z}$. Consequently, $a + c \equiv b + d \pmod{n}$, which is what we wished to show. The case for subtraction is left as an exercise.
(2) Similarly,

$$\begin{aligned} ac &= bd + bln + knd + kln^2 \\ &= bd + n(bl + k(d + ln)) \\ &= bd + n(bl + kc) \\ &= bd + sn \end{aligned}$$

where $s = bl + kc \in \mathbb{Z}$. Thus, $a \cdot b \equiv c \cdot d \pmod{n}$.
(3) We prove Part (3) by induction. We have $a \equiv b \pmod{n}$ (base step) and $a^m \equiv b^m \pmod{n}$ (inductive hypothesis). Then by Part (2) we have $a^{m+1} \equiv aa^m \equiv bb^m \equiv b^{m+1} \pmod{n}$. ∎

Theorem 2.45 is equivalent to the following theorem, since

$$a \equiv b \pmod{n} \iff a \bmod n = b \bmod n,$$

$$a \bmod n \iff [a]_n,$$

$$b \bmod n \iff [b]_n.$$

Theorem 2.46 *For all $a, b, c, d \in \mathbb{Z}$, if $[a]_n = [b]_n$, $[c]_n = [d]_n$, then*

(1) $[a \pm b]_n = [c \pm d]_n$,
(2) $[a \cdot b]_n = [c \cdot d]_n$,
(3) $[a^m]_n = [b^m]_n$, $\forall m \in \mathbb{Z}^+$.

The fact that the congruence relation modulo n is stable for addition (subtraction) and multiplication means that we can define binary operations, again called addition (subtraction)

and multiplication on the set of $\mathbb{Z}/n\mathbb{Z}$ of equivalence classes modulo n as follows (in case only one n is being discussed, we can simply write $[x]$ for the class $[x]_n$):

$$[a]_n + [b]_n = [a + b]_n \tag{2.95}$$

$$[a]_n - [b]_n = [a - b]_n \tag{2.96}$$

$$[a]_n \cdot [b]_n = [a \cdot b]_n \tag{2.97}$$

Example 2.47 Let $n = 12$, then

$$\begin{aligned}
[7]_{12} + [8]_{12} &= [7 + 8]_{12} = [15]_{12} = [3]_{12} \\
[7]_{12} - [8]_{12} &= [7 - 8]_{12} = [-1]_{12} = [11]_{12} \\
[7]_{12} \cdot [8]_{12} &= [7 \cdot 8]_{12} = [56]_{12} = [8]_{12}.
\end{aligned}$$

In many cases, we may still prefer to write the above operations as follows:

$$\begin{aligned}
7 + 8 &= 15 \equiv 3 \pmod{12} \\
7 - 8 &= -1 \equiv 11 \pmod{12} \\
7 \cdot 8 &= 56 \equiv 8 \pmod{12}.
\end{aligned}$$

We summarize the properties of addition and multiplication modulo n in the following two theorems.

Theorem 2.47 *The set $\mathbb{Z}/n\mathbb{Z}$ of integers modulo n has the following properties with respect to addition:*

(1) *Closure:* $[x] + [y] \in \mathbb{Z}/n\mathbb{Z}$, *for all* $[x], [y] \in \mathbb{Z}/n\mathbb{Z}$;
(2) *Associative:* $([x] + [y]) + [z] = [x] + ([y] + [z])$, *for all* $[x], [y], [z] \in \mathbb{Z}/n\mathbb{Z}$;
(3) *Commutative:* $[x] + [y] = [y] + [x]$, *for all* $[x], [y] \in \mathbb{Z}/n\mathbb{Z}$;
(4) *Identity, namely,* $[0]$;
(5) *Additive inverse:* $-[x] = [-x]$, *for all* $[x] \in \mathbb{Z}/n\mathbb{Z}$.

Proof: These properties follow directly from the stability and the definition of the operation in $\mathbb{Z}/n\mathbb{Z}$. ∎

Theorem 2.48 *The set $\mathbb{Z}/n\mathbb{Z}$ of integers modulo n has the following properties with respect to multiplication:*

(1) *Closure:* $[x] \cdot [y] \in \mathbb{Z}/n\mathbb{Z}$, *for all* $[x], [y] \in \mathbb{Z}/n\mathbb{Z}$;
(2) *Associative:* $([x] \cdot [y]) \cdot [z] = [x] \cdot ([y] \cdot [z])$, *for all* $[x], [y], [z] \in \mathbb{Z}/n\mathbb{Z}$;

(3) *Commutative:* $[x] \cdot [y] = [y] \cdot [x]$, *for all* $[x], [y] \in \mathbb{Z}/n\mathbb{Z}$;
(4) *Identity, namely,* $[1]$;
(5) *Distributivity of multiplication over addition:* $[x] \cdot ([y] + [z]) = ([x] \cdot [y]) + ([x] \cdot [z])$, *for all* $[x], [y], [z] \in \mathbb{Z}/n\mathbb{Z}$.

Proof: These properties follow directly from the stability of the operation in $\mathbb{Z}/n\mathbb{Z}$ and the corresponding properties of $\mathbb{Z}$. ∎

The division a/b (we assume a/b is in lowest terms and $b \not\equiv 0 \pmod{n}$) in $\mathbb{Z}/n\mathbb{Z}$, however, will be more of a problem; sometimes you can divide, sometimes you cannot. For example, let $n = 12$ again, then

$$\begin{aligned} 3/7 &\equiv 9 \pmod{12} \quad \text{(no problem)}, \\ 3/4 &\equiv \perp \pmod{12} \quad \text{(impossible)}. \end{aligned}$$

Why is division sometimes possible (e.g., $3/7 \equiv 9 \pmod{12}$) and sometimes impossible (e.g., $3/8 \equiv \perp \pmod{12}$)? The problem is with the modulus n; if n is a prime number, then the division $a/b \pmod{n}$ is always possible and unique, whilst if n is a composite then the division $a/b \pmod{n}$ may be not possible or the result may be not unique. Let us observe two more examples, one with $n = 13$ and the other with $n = 14$. First note that $a/b \equiv a \cdot 1/b \pmod{n}$ if and only if $1/b \pmod{n}$ is possible, since multiplication modulo n is always possible. We call $1/b \pmod{n}$ the *multiplicative inverse* (or the *modular inverse*) of b modulo n. Now let $n = 13$ be a prime, then the following table gives all the values of the multiplicative inverses $1/x \pmod{13}$ for $x = 1, 2, \ldots, 12$:

x		1	2	3	4	5	6	7	8	9	10	11	12
$1/x$	(mod 13)	1	7	9	10	8	11	2	5	3	4	6	12

This means that division in $\mathbb{Z}/13\mathbb{Z}$ is always possible and unique. On the other hand, if $n = 14$ (the n now is a composite), then

x		1	2	3	4	5	6	7	8	9	10	11	12	13
$1/x$	(mod 14)	1	$\perp$	5	$\perp$	3	$\perp$	$\perp$	$\perp$	11	$\perp$	9	$\perp$	13

This means that only the numbers 1, 3, 5, 9, 11 and 13 have multiplicative inverses modulo 14, or equivalently only those divisions by 1, 3, 5, 9, 11 and 13 modulo 14 are possible. This observation leads to the following important results:

Theorem 2.49 *The multiplicative inverse* $1/b$ *modulo n exists if and only if* $\gcd(b, n) = 1$.

But how many b's satisfy $\gcd(b, n) = 1$? The following result answers this question.

Corollary 2.4 *There are $\phi(n)$ numbers b for which $1/b \pmod{n}$ exists.*

Example 2.48 Let $n = 21$. Since $\phi(21) = 12$, there are twelve values of b for which $1/b \pmod{21}$ exists. In fact, the multiplicative inverse modulo 21 only exists for each of the following b:

b	1	2	4	5	8	10	11	13	16	17	19	20
$1/b \pmod{21}$	1	11	16	17	8	19	2	13	4	5	10	20

Corollary 2.5 *The division a/b modulo n (assume that a/b is in lowest terms) is possible if and only if $1/b \pmod{n}$ exists, that is, if and only if $\gcd(b, n) = 1$.*

Example 2.49 Compute $6/b \pmod{21}$ whenever it is possible. By the multiplicative inverses of $1/b \pmod{21}$ in the previous table, we just need to calculate $6 \cdot 1/b \pmod{21}$:

b	1	2	4	5	8	10	11	13	16	17	19	20
$6/b \pmod{21}$	6	3	12	18	6	9	12	15	3	9	18	15

As can be seen, addition (subtraction) and multiplication are always possible in $\mathbb{Z}/n\mathbb{Z}$, with $n > 1$, since $\mathbb{Z}/n\mathbb{Z}$ is a ring. Note also that $\mathbb{Z}/n\mathbb{Z}$ with n prime is an Abelian group with respect to addition, and all the nonzero elements in $\mathbb{Z}/n\mathbb{Z}$ form an Abelian group with respect to multiplication (i.e., a division is always possible for any two nonzero elements in $\mathbb{Z}/n\mathbb{Z}$ if n is prime); hence $\mathbb{Z}/n\mathbb{Z}$ with n prime is a field. That is:

Theorem 2.50 *$\mathbb{Z}/n\mathbb{Z}$ is a field if and only if n is prime.*

The above results only tell us when the multiplicative inverse $1/a$ modulo n is possible, without mentioning how to find the inverse. To actually find the multiplicative inverse, we let

$$1/a \,(\mathrm{mod}\, n) = x, \tag{2.98}$$

which is equivalent to

$$ax \equiv 1 \,(\mathrm{mod}\, n). \tag{2.99}$$

Since

$$ax \equiv 1 \,(\mathrm{mod}\, n) \Longleftrightarrow ax - ny = 1. \tag{2.100}$$

Thus, finding the multiplicative inverse $1/a \pmod{n}$ is the same as finding the solution of the linear Diophantine equation $ax - ny = 1$, which, as we know, can be solved by using the continued fraction expansion of a/n or by using Euclid's algorithm.

Example 2.50 Find

(1) $1/154 \pmod{801}$,
(2) $4/154 \pmod{801}$.

Solution

(1) Since

$$1/a \pmod{n} = x \iff ax \equiv 1 \pmod{n} \iff ax - ny = 1,$$

we only need to find x and y in

$$154x - 801y = 1.$$

To do so, we first use the Euclid's algorithm to find gcd(154, 801) as follows:

$$\begin{aligned} 801 &= 154 \cdot 5 + 31 \\ 154 &= 31 \cdot 4 + 30 \\ 31 &= 30 \cdot 1 + 1 \\ 3 &= 1 \cdot 3. \end{aligned}$$

Since $\gcd(154, 801) = 1$, by Theorem 2.49, the equation $154x - 801y = 1$ is soluble. We now rewrite the above resulting equations

$$\begin{aligned} 31 &= 801 - 154 \cdot 5 \\ 30 &= 154 - 31 \cdot 4 \\ 1 &= 31 - 30 \cdot 1 \end{aligned}$$

and work backwards on the above new equations

$$\begin{aligned} 1 &= 31 - 30 \cdot 1 \\ &= 31 - (154 - 31 \cdot 4) \cdot 1 \\ &= 31 - 154 + 4 \cdot 31 \\ &= 5 \cdot 31 - 154 \\ &= 5 \cdot (801 - 154 \cdot 5) - 154 \\ &= 5 \cdot 801 - 26 \cdot 154 \\ &= 801 \cdot 5 - 154 \cdot 26. \end{aligned}$$

So, $x \equiv -26 \equiv 775 \pmod{801}$. That is,

$$1/154 \bmod 801 = 775.$$

(2) By Part (1) above, we have

$$\begin{aligned} 4/154 &\equiv 4 \cdot 1/154 \\ &\equiv 4 \cdot 775 \\ &\equiv 697 \pmod{801}. \end{aligned}$$

The above procedure used to find the x and y in $ax + by = 1$ can be generalized to find the x and y in $ax + by = c$; this procedure is usually called the *extended Euclid's algorithm.*

Congruences have much in common with equations. In fact, the linear congruence $ax \equiv b \pmod{n}$ is equivalent to the linear Diophantine equation $ax - ny = b$. That is,

$$ax \equiv b \pmod{n} \iff ax - ny = b. \tag{2.101}$$

Thus, linear congruences can be solved by using the continued fraction method just as for linear Diophantine equations.

Theorem 2.51 *Let* $\gcd(a, n) = d$. *If* $d \nmid b$, *then the linear congruence*

$$ax \equiv b \pmod{n} \tag{2.102}$$

has no solution.

Proof: We will prove the contrapositive of the assertion: If $ax \equiv b \pmod{n}$ has a solution, then $\gcd(a, n) \mid b$. Suppose that s is a solution. Then $as \equiv b \pmod{n}$, and from the definition of the congruence, $n \mid (as - b)$, or from the definition of divisibility, $as - b = kn$ for some integer k. Since $\gcd(a, m) \mid a$ and $\gcd(a, n) \mid kn$, it follows that $\gcd(a, n) \mid b$. ∎

Theorem 2.52 *Let* $\gcd(a, n) = d$. *Then the linear congruence* $ax \equiv b \pmod{n}$ *has solutions if and only if* $d \mid b$.

Proof: Follows from Theorem 2.51. ∎

Theorem 2.53 *Let* $\gcd(a, n) = 1$. *Then the linear congruence* $ax \equiv b \pmod{n}$ *has exactly one solution.*

Proof: If $\gcd(a, n) = 1$, then there exist x and y such that $ax + ny = 1$. Multiplying by b gives

$$a(xb) + n(yb) = b.$$

As $a(xb) - b$ is a multiple of n, or $a(xb) \equiv b \pmod{n}$, the least residue of xb modulo n is then a solution of the linear congruence. The uniqueness of the solution is left as an exercise. ∎

Theorem 2.54 *Let* $\gcd(a, n) = d$ *and suppose that* $d \mid b$. *Then the linear congruence*

$$ax \equiv b \pmod{n}. \tag{2.103}$$

has exactly d *solutions modulo* n. *These are given by*

$$t,\ t + \frac{n}{d},\ t + \frac{2n}{d},\ \ldots,\ t + \frac{(d-1)n}{d} \tag{2.104}$$

where t *is the solution, unique modulo* n/d, *of the linear congruence*

$$\frac{a}{d}x \equiv \frac{b}{d} \left(\mod \frac{n}{d}\right). \tag{2.105}$$

Proof: By Theorem 2.52, the linear congruence has solutions since $d \mid b$. Now let t be be such a solution, then $t + k(n/d)$ for $k = 1, 2, \ldots, d - 1$ are also solutions, since $a(t + k(n/d)) \equiv at + kn(a/d) \equiv at \equiv b \pmod{n}$. ∎

Example 2.51 Solve the linear congruence $154x \equiv 22 \pmod{803}$. Notice first that

$$154x \equiv 22 \pmod{803} \Longleftrightarrow 154x - 803y = 22.$$

Now we use the Euclid's algorithm to find gcd(154, 803) as follows:

$$\begin{aligned} 803 &= 154 \cdot 5 + 33 \\ 154 &= 33 \cdot 4 + 22 \\ 33 &= 22 \cdot 1 + 11 \\ 22 &= 11 \cdot 2 + 0. \end{aligned}$$

Since gcd(154, 803) = 11 and $11 \mid 22$, by Theorem 2.31, the equation $154x - 801y = 22$ is soluble. Now we rewrite the above resulting equations

$$\begin{aligned} 33 &= 803 - 154 \cdot 5 \\ 22 &= 154 - 33 \cdot 4 \\ 11 &= 33 - 22 \cdot 1 \end{aligned}$$

and work backwards on the above new equations

$$\begin{aligned} 11 &= 33 - 22 \cdot 1 \\ &= 33 - (154 - 33 \cdot 4) \cdot 1 \\ &= 33 - 154 + 4 \cdot 33 \\ &= 5 \cdot 33 - 154 \\ &= 5 \cdot (803 - 154 \cdot 5) - 154 \\ &= 5 \cdot 803 - 26 \cdot 154 \\ &= 803 \cdot 5 - 154 \cdot 26. \end{aligned}$$

So, $x \equiv -26 \equiv 777 \pmod{803}$. By Theorems 2.53 and 2.54, $x \equiv -26 \equiv 47 \pmod{73}$ is the only solution to the simplified congruence:

$$154/11 \equiv 22/11 \pmod{803/11} \Longrightarrow 14x \equiv 2 \pmod{73},$$

since $\gcd(14, 73) = 1$. By Theorem 2.54, there are, in total, eleven solutions to the congruence $154x \equiv 11 \pmod{803}$, as follows:

$$x = \begin{pmatrix} 777 \\ 47 \\ 120 \\ 193 \\ 266 \\ 339 \\ 412 \\ 485 \\ 558 \\ 631 \\ 704 \end{pmatrix}.$$

Thus,

$$x = \begin{pmatrix} 751 \\ 94 \\ 240 \\ 386 \\ 532 \\ 678 \\ 21 \\ 167 \\ 313 \\ 459 \\ 605 \end{pmatrix}$$

are the eleven solutions to the original congruence $154x \equiv 22 \pmod{803}$.

Remark 2.10 To find the solution for the linear Diophantine equation

$$ax \equiv b \,(\mathrm{mod}\, n) \tag{2.106}$$

is equivalent to finding the quotient of the modular division

$$x \equiv \frac{b}{a} \,(\mathrm{mod}\, n) \tag{2.107}$$

which is, again, equivalent to finding the multiplicative inverse

$$x \equiv \frac{1}{a} \,(\mathrm{mod}\, n) \tag{2.108}$$

because if $\frac{1}{a}$ modulo n exists, the multiplication $b \cdot \frac{1}{a}$ is always possible.

Theorem 2.55 (Fermat's little theorem) *Let a be a positive integer and* $\gcd(a, p) = 1$. *If p is prime, then*

$$a^{p-1} \equiv 1 \,(\mathrm{mod}\, p). \tag{2.109}$$

Proof: First notice that the residues modulo p of $a,\ 2a,\ \ldots, (p-1)a$ are $1, 2, \ldots, (p-1)$ in some order, because no two of them can be equal. So, if we multiply them together, we get

$$\begin{aligned} a \cdot 2a \cdots (p-1)a &\equiv [(a \bmod p) \cdot (2a \bmod p) \cdots (p-1)a \bmod p)] \,(\mathrm{mod}\, p) \\ &\equiv (p-1)! \,(\mathrm{mod}\, p). \end{aligned}$$

This means that

$$(p-1)!a^{p-1} \equiv (p-1)! \,(\mathrm{mod}\, p).$$

Now we can cancel the $(p-1)!$ since $p \nmid (p-1)!$, and the result thus follows. ∎

There is a more convenient and more general form of Fermat's little theorem:

$$a^p \equiv a \,(\mathrm{mod}\, p), \tag{2.110}$$

for $a \in \mathbb{N}$. The proof is easy: If $\gcd(a, p) = 1$, we simply multiply (2.109) by a. If not, then $p \mid a$. So $a^p \equiv 0 \equiv a \,(\mathrm{mod}\, p)$.

Fermat's theorem has several important consequences which are very useful in compositeness; one of the these consequences is as follows:

Corollary 2.6 (Converse of the Fermat little theorem, 1640) *Let n be an odd positive integer. If* $\gcd(a, n) = 1$ *and*

$$a^{n-1} \not\equiv 1 \pmod{n}, \tag{2.111}$$

then n is composite.

Remark 2.11 In 1640, Fermat made a false conjecture that all the numbers of the form $F_n = 2^{2^n} + 1$ were prime. Fermat really should not have made such a "stupid" conjecture, since F_5 can be relatively easily verified to be composite, just by using his own recently discovered theorem – Fermat's little theorem:

$$\begin{aligned}
3^{2^2} &\equiv 81 && \pmod{4294967297}\\
3^{2^3} &\equiv 6561 && \pmod{4294967297}\\
3^{2^4} &\equiv 43046721 && \pmod{4294967297}\\
3^{2^5} &\equiv 3793201458 && \pmod{4294967297}\\
&\vdots\\
3^{2^{32}} &\equiv 3029026160 && \pmod{4294967297}\\
&\not\equiv 1 && \pmod{4294967297}.
\end{aligned}$$

Thus, by Fermat's little theorem, $2^{32} + 1$ is not prime!

Based on Fermat's little theorem, Euler established a more general result in 1760:

Theorem 2.56 (Euler's theorem) *Let a and n be positive integers with* $\gcd(a, n) = 1$. *Then*

$$a^{\phi(n)} \equiv 1 \pmod{n}. \tag{2.112}$$

Proof: Let $r_1, r_2, \ldots, r_{\phi(n)}$ be a reduced residue system modulo n. Then $ar_1, ar_2, \ldots, ar_{\phi(n)}$ is also a residue system modulo n. Thus we have

$$(ar_1)(ar_2)\cdots(ar_{\phi(n)}) \equiv r_1 r_2 \cdots r_{\phi(n)} \pmod{n},$$

since $ar_1, ar_2, \ldots, ar_{\phi(n)}$, being a reduced residue system, must be congruent in some order to $r_1, r_2, \ldots, r_{\phi(n)}$. Hence,

$$a^{\phi(n)} r_1 r_2 \cdots r_{\phi(n)} \equiv r_1 r_2 \cdots r_{\phi(n)} \pmod{n},$$

which implies that $a^{\phi(n)} \equiv 1 \pmod{n}$. ∎

It can be difficult to find the order[1] of an element a modulo n but sometimes it is possible to improve (2.112) by proving that every integer a modulo n must have an order smaller than the number $\phi(n)$ – this order is actually a number that is a factor of $\lambda(n)$.

Theorem 2.57 (Carmichael's theorem) *Let a and n be positive integers with* $\gcd(a, n) = 1$. *Then*

$$a^{\lambda(n)} \equiv 1 \,(\mathrm{mod}\, n), \tag{2.113}$$

where $\lambda(n)$ is Carmichael's function, given in Definition 2.32.

Proof: Let $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$. We shall show that

$$a^{\lambda(n)} \equiv 1 \,(\mathrm{mod}\, p_i^{\alpha_i})$$

for $1 \leq i \leq k$, since this implies that $a^{\lambda(n)} \equiv 1 \,(\mathrm{mod}\, n)$. If $p_k^{\alpha_k} = 2, 4$ or a power of an odd prime, then by Definition 2.32, $\lambda(\alpha_k) = \phi(\alpha_k)$, so $a^{\lambda(p_i^{\alpha_i})} \equiv 1 \,(\mathrm{mod}\, p_i^{\alpha_i})$. Since $\lambda(p_i^{\alpha_i}) \mid \lambda(n)$, $a^{\lambda(n)} \equiv 1 \,(\mathrm{mod}\, p_i^{\alpha_i})$. The case that $p_i^{\alpha_i}$ is a power of 2 greater than 4 is left as an exercise. ■

Note that $\lambda(n)$ will never exceed $\phi(n)$ and is often much smaller than $\phi(n)$; it is the value of the largest order it is possible to have.

Example 2.52 Let $a = 11$ and $n = 24$. Then $\phi(24) = 8$, $\lambda(24) = 2$. So,

$$11^{\phi(24)} = 11^8 \equiv 1 \,(\mathrm{mod}\, 24),$$

$$11^{\lambda(24)} = 11^2 \equiv 1 \,(\mathrm{mod}\, 24).$$

That is, $\mathrm{ord}_{24}(11) = 2$.

In 1770 Edward Waring (1734–1793) published the following result, which is attributed to John Wilson (1741–1793).

Theorem 2.58 (Wilson's theorem) *If p is a prime, then*

$$(p-1)! \equiv -1 \,(\mathrm{mod}\, p). \tag{2.114}$$

Proof: It suffices to assume that p is odd. Now to every integer a with $0 < a < p$ there is a unique integer a' with $0 < a' < p$ such that $aa' \equiv 1 \,(\mathrm{mod}\, p)$. Further if $a = a'$ then $a^2 \equiv 1 \,(\mathrm{mod}\, p)$ whence $a = 1$ or $a = p - 1$. Thus the set $2, 3, \ldots, p - 2$ can be divided into

[1]The order of an element a modulo n is the smallest integer r such that $a^r \equiv 1 \,(\mathrm{mod}\, n)$; we shall discuss this later in Section 2.5.

$(p-3)/2$ pairs a, a' with $aa' \equiv 1 \pmod{p}$. Hence we have $2 \cdot 3 \cdots (p-2) \equiv 1 \pmod{p}$, and so $(p-1)! \equiv -1 \pmod{p}$, as required. ■

Theorem 2.59 (Converse of Wilson's theorem) *If n is an odd positive integer greater than 1 and*

$$(n-1)! \equiv -1 \pmod{n}, \tag{2.115}$$

then n is a prime.

Remark 2.12 Prime p is called a *Wilson prime* if

$$W(p) \equiv 0 \pmod{p}, \tag{2.116}$$

where

$$W(p) = \frac{(p-1)! + 1}{p}$$

is an integer, or equivalently if

$$(p-1)! \equiv -1 \pmod{p^2}. \tag{2.117}$$

For example, $p = 5, 13, 563$ are Wilson primes, but 599 is not since

$$\frac{(599-1)! + 1}{599} \bmod 599 = 382 \neq 0.$$

It is not known whether there are infinitely many Wilson primes; to date, the only known Wilson primes for $p < 5 \cdot 10^8$ are $p = 5, 13, 563$. A prime p is called a *Wieferich prime*, named after A. Wieferich, if

$$2^{p-1} \equiv 1 \pmod{p^2}. \tag{2.118}$$

To date, the only known Wieferich primes for $p < 4 \cdot 10^{12}$ are $p = 1093$ and 3511.

In what follows, we shall show how to use Euler's theorem to calculate the multiplicative inverse modulo n, and hence the solutions of a linear congruence.

Theorem 2.60 *Let x be the multiplicative inverse $1/a$ modulo n. If* $\gcd(a, n) = 1$*, then*

$$x \equiv \frac{1}{a} \pmod{n} \tag{2.119}$$

is given by

$$x \equiv a^{\phi(n)-1} \pmod{n}. \tag{2.120}$$

Proof: By Euler's theorem, we have $a^{\phi(n)} \equiv 1 \pmod{n}$. Hence

$$aa^{\phi(n)-1} \equiv 1 \pmod{n},$$

and $a^{\phi(n)-1}$ is the multiplicative inverse of a modulo n, as desired. ■

Corollary 2.7 *Let x be the division b/a modulo n (b/a is assumed to be in lowest terms). If* $\gcd(a, n) = 1$*, then*

$$x \equiv \frac{b}{a} \pmod{n} \tag{2.121}$$

is given by

$$x \equiv b \cdot a^{\phi(n)-1} \pmod{n}. \tag{2.122}$$

Corollary 2.8 *If* $\gcd(a, n) = 1$*, then the solution of the linear congruence*

$$ax \equiv b \pmod{n} \tag{2.123}$$

is given by

$$x \equiv ba^{\phi(n)-1} \pmod{n}. \tag{2.124}$$

Example 2.53 Solve the congruence $5x \equiv 14 \pmod{24}$. First note that because $\gcd(5,24) = 1$, the congruence has exactly one solution. Using (2.124) we get

$$x \equiv 14 \cdot 5^{\phi(24)-1} \pmod{24} = 22.$$

Example 2.54 Solve the congruence $20x \equiv 15 \pmod{135}$. First note that as $d = \gcd(20,135) = 5$ and $d \mid 15$, the congruence has exactly five solutions modulo 135. To find these five solutions, we divide by 5 and get a new congruence

$$4x' \equiv 3 \pmod{27}.$$

To solve this new congruence, we get

$$x' \equiv 3 \cdot 4^{\phi(27)-1} \equiv 21 \pmod{27}.$$

Therefore, the five solutions are as follows:

$$\begin{aligned}(x_0, x_1, x_2, x_3, x_4) &\equiv \left(x', x' + \frac{n}{d}, x' + \frac{2n}{d}, x' + \frac{3n}{d}, x' + \frac{4n}{d}\right) \\ &\equiv (21, 21 + 27, 21 + 2 \cdot 27, 21 + 3 \cdot 27, 21 + 4 \cdot 27) \\ &\equiv (21, 48, 75, 102, 129) \pmod{135}.\end{aligned}$$

Next we shall introduce a method for solving systems of linear congruences. The method, widely known as the Chinese Remainder theorem (or just CRT, for short), was discovered by the ancient Chinese mathematician Sun Tsu (who lived sometime between 200 B.C. and 200 A.D.).

Theorem 2.61 (The Chinese Remainder theorem CRT) *If $m_1, m_2, \cdots, m_n$ are pairwise relatively prime and greater than 1, and $a_1, a_2, \cdots, a_n$ are any integers, then there is a solution x to the following simultaneous congruences:*

$$\left.\begin{aligned} x &\equiv a_1 \pmod{m_1}, \\ x &\equiv a_2 \pmod{m_2}, \\ &\vdots \\ x &\equiv a_n \pmod{m_n}. \end{aligned}\right\} \tag{2.125}$$

If x and x' are two solutions, then $x \equiv x' \pmod{M}$, where $M = m_1 m_2 \cdots m_n$.

Proof: Existence: Let us first solve a special case of the simultaneous congruences (2.125), where i is some fixed subscript,

$$a_i = 1,\ a_1 = a_2 = \cdots = a_{i-1} = a_{i+1} = \cdots = a_n = 0.$$

Let $k_i = m_1 m_2 \cdots m_{i-1} m_{i+1} \cdots m_n$. Then k_i and m_i are relatively prime, so we can find integers r and s such that $rk_i + sm_i = 1$. This gives the congruences:

$$\begin{aligned} rk_i &\equiv 0 \pmod{k_i}, \\ rk_i &\equiv 1 \pmod{m_i}. \end{aligned}$$

Since $m_1, m_2, \ldots, m_{i-1}, m_{i+1}, \ldots m_n$ all divide k_i, it follows that $x_i = rk_i$ satisfies the simultaneous congruences:

$$\begin{aligned} x_i &\equiv 0 \pmod{m_1}, \\ x_i &\equiv 0 \pmod{m_2}, \\ &\vdots \\ x_i &\equiv 0 \pmod{m_{i-1}}. \\ x_i &\equiv 1 \pmod{m_i}. \\ x_i &\equiv 0 \pmod{m_{i+1}}. \\ &\vdots \\ x_i &\equiv 0 \pmod{m_n}. \end{aligned}$$

For each subscript i, $1 \leq i \leq n$, we find such an x_i. Now to solve the system of the simultaneous congruences (2.125), set $x = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$. Then $x \equiv a_i x_i \equiv a_i \pmod{m_i}$ for each i, $1 \leq i \leq n$, such that x is a solution of the simultaneous congruences.

Uniqueness: Let x' be another solution to the simultaneous congruences (2.125), but different from the solution x, so that $x' \equiv x \pmod{m_i}$ for each x_i. Then $x - x' \equiv 0 \pmod{m_i}$ for each i. So m_i divides $x - x'$ for each i; hence the least common multiple of all the m_j's divides $x - x'$. But since the m_i are pairwise relatively prime, this least common multiple is the product M. So $x \equiv x' \pmod{M}$. ∎

Remark 2.13 If the system of the linear congruences (2.125) is soluble, then its solution can be conveniently described as follows:

$$x \equiv \sum_{i=1}^{n} a_i M_i M_i' \pmod{m} \tag{2.126}$$

where

$$\begin{aligned} m &= m_1 m_2 \cdots m_n, \\ M_i &= m/m_i, \\ M_i' &= M_i^{-1} \pmod{m_i}, \end{aligned}$$

for $i = 1, 2, \ldots, n$.

Example 2.55 Consider the Sun Zi problem:

$$\begin{aligned} x &\equiv 2 \,(\mathrm{mod}\, 3), \\ x &\equiv 3 \,(\mathrm{mod}\, 5), \\ x &\equiv 2 \,(\mathrm{mod}\, 7). \end{aligned}$$

By (2.126), we have

$$\begin{aligned} m &= m_1 m_2 m_3 = 3 \cdot 5 \cdot 7 = 105, \\ M_1 &= m/m_1 = 105/3 = 35, \\ M_1' &= M_1^{-1} \,(\mathrm{mod}\, m_1) = 35^{-1} \,(\mathrm{mod}\, 3) = 2, \\ M_2 &= m/m_2 = 105/5 = 21, \\ M_2' &= M_2^{-1} \,(\mathrm{mod}\, m_2) = 21^{-1} \,(\mathrm{mod}\, 5) = 1, \\ M_3 &= m/m_3 = 105/7 = 15, \\ M_3' &= M_3^{-1} \,(\mathrm{mod}\, m_3) = 15^{-1} \,(\mathrm{mod}\, 7) = 1. \end{aligned}$$

Hence,

$$\begin{aligned} x &= a_1 M_1 M_1' + a_2 M_2 M_2' + a_3 M_3 M_3' \,(\mathrm{mod}\, m) \\ &= 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 \,(\mathrm{mod}\, 105) \\ &= 23. \end{aligned}$$

The congruences $ax \equiv b \pmod{m}$ we have studied so far are a special type of congruence; they are all linear congruences. In this section, we shall study the higher degree congruences, particularly the quadratic congruences.

Definition 2.42 Let m be a positive integer, and let

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

be any polynomial with integer coefficients. Then a *high-order congruence* or a *polynomial congruence* is a congruence of the form

$$f(x) \equiv 0 \,(\mathrm{mod}\, n). \tag{2.127}$$

A polynomial congruence is also called a *polynomial congruential equation*.

Let us consider the polynomial congruence

$$f(x) = x^3 + 5x - 4 \equiv 0 \,(\mathrm{mod}\, 7).$$

This congruence holds when $x = 2$, since

$$f(2) = 2^3 + 5 \cdot 2 - 4 \equiv 0 \,(\mathrm{mod}\, 7).$$

Just as for algebraic equations, we say that $x = 2$ is a root or a solution of the congruence. In fact, any value of x which satisfies the following condition

$$x \equiv 2 \,(\mathrm{mod}\, 7)$$

is also a solution of the congruence. In general, as in linear congruence, when a solution x_0 has been found, all values x for which

$$x \equiv x_0 \,(\mathrm{mod}\, n)$$

are also solutions. But by convention, we still consider them as a *single* solution. Thus, our problem is to find all incongruent (different) solutions of $f(x) \equiv 0 \,(\mathrm{mod}\, n)$. In general, this problem is very difficult, and many techniques of solution depend partially on trial-and-error methods. For example, to find all solutions of the congruence $f(x) \equiv 0 \,(\mathrm{mod}\, n)$, we could certainly try all values $0, 1, 2, \ldots, n-1$ (or the numbers in the complete residue system modulo n), and determine which of them satisfy the congruence; this would give us the total number of *incongruent* solutions modulo n.

Theorem 2.62 *Let $M = m_1 m_2 \cdots m_n$, where $m_1, m_2, \ldots, m_n$ are pairwise relatively prime. Then the integer x_0 is a solution of*

$$f(x) \equiv 0 \,(\mathrm{mod}\, M) \tag{2.128}$$

if and only if x_0 is a solution of the system of polynomial congruences:

$$\left.\begin{aligned} f(x) &\equiv 0 \,(\mathrm{mod}\, m_1) \\ f(x) &\equiv 0 \,(\mathrm{mod}\, m_2) \\ &\vdots \\ f(x) &\equiv 0 \,(\mathrm{mod}\, m_n). \end{aligned}\right\} \tag{2.129}$$

If x and x' are two solutions, then $x \equiv x' \,(\mathrm{mod}\, M)$, where $M = m_1 m_2 \cdots m_n$.

Proof: If $f(a) \equiv 0 \,(\mathrm{mod}\, M)$, then obviously $f(a) \equiv 0 \,(\mathrm{mod}\, m_i)$, for $i = 1, 2, \ldots, n$. Conversely, suppose a is a solution of the system

$$f(x) \equiv 0 \,(\mathrm{mod}\, m_i), \quad \text{for } i = 1, 2, \ldots, n.$$

Then $f(a)$ is a solution of the system

$$\left.\begin{aligned} y &\equiv 0 \,(\mathrm{mod}\, m_1) \\ y &\equiv 0 \,(\mathrm{mod}\, m_2) \\ &\vdots \\ y &\equiv 0 \,(\mathrm{mod}\, m_n) \end{aligned}\right\}$$

and it follows from the Chinese Remainder theorem that $f(a) \equiv 0 \,(\mathrm{mod}\, m_1 m_2 \cdots m_n)$. Thus, a is a solution of $f(x) \equiv 0 \,(\mathrm{mod}\, M)$. ∎

We now restrict ourselves to quadratic congruences, the simplest possible nonlinear polynomial congruences.

Definition 2.43 A quadratic congruence is a congruence of the form:

$$x^2 \equiv a \,(\mathrm{mod}\, n) \tag{2.130}$$

where $\gcd(a, n) = 1$. To solve the congruence is to find an integral solution for x which satisfies the congruence.

In most cases, it is sufficient to study the above congruence rather than the following more general quadratic congruence

$$ax^2 + bx + c \equiv 0 \,(\mathrm{mod}\, n) \tag{2.131}$$

since if $\gcd(a, n) = 1$ and b is even or n is odd, then the congruence 2.131 can be reduced to a congruence of type (2.130). The problem can even be further reduced to solving a congruence of the type (if $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, where $p_1, p_2, \ldots p_k$ are distinct primes, and $\alpha_1, \alpha_2, \ldots, \alpha_k$ are positive integers):

$$x^2 \equiv a \,(\mathrm{mod}\, p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}) \tag{2.132}$$

because solving the congruence (2.132) is equivalent to solving the following system of congruences:

$$\left.\begin{aligned} x^2 &\equiv a \,(\mathrm{mod}\, p_1^{\alpha_1}) \\ x^2 &\equiv a \,(\mathrm{mod}\, p_2^{\alpha_2}) \\ &\vdots \\ x^2 &\equiv a \,(\mathrm{mod}\, p_k^{\alpha_k}). \end{aligned}\right\} \tag{2.133}$$

In what follows, we shall be only interested in quadratic congruences of the form

$$x^2 \equiv a \pmod{p} \tag{2.134}$$

where p is an odd prime and $a \not\equiv 0 \pmod{p}$.

Definition 2.44 Let a be any integer and n a natural number, and suppose that $\gcd(a, n) = 1$. Then a is called a *quadratic residue* modulo n if the congruence

$$x^2 \equiv a \pmod{n}$$

is soluble. Otherwise, it is called a *quadratic non-residue* modulo n.

Remark 2.14 Similarly, we can define the cubic residues, and fourth-power residues, etc. For example, a is a *kth power residue* modulo n if the congruence

$$x^k \equiv a \pmod{n} \tag{2.135}$$

is soluble. Otherwise, it is a *kth power non-residue* modulo n.

Theorem 2.63 *Let p be an odd prime and a an integer not divisible by p. Then the congruence*

$$x^2 \equiv a \pmod{p} \tag{2.136}$$

has either no solution or exactly two congruence solutions modulo p.

Proof: If x and y are solutions to $x^2 \equiv a \pmod{p}$, then $x^2 \equiv y^2 \pmod{p}$, that is, $p \mid (x^2 - y^2)$. Since $x^2 - y^2 = (x + y)(x - y)$, we must have $p \mid (x - y)$ or $p \mid (x + y)$, that is, $x \equiv \pm y \pmod{p}$. Hence, any two distinct solutions modulo p differ only by a factor of -1. ■

Example 2.56 Find the quadratic residues and quadratic nonresidues for moduli 5, 7, 11, 15, 23, respectively.

(1) Modulo 5, the integers 1, 4 are quadratic residues, whilst 2, 3 are quadratic nonresidues, since

$$1^2 \equiv 4^2 \equiv 1, \qquad 2^2 \equiv 3^2 \equiv 4.$$

(2) Modulo 7, the integers 1, 2, 4 are quadratic residues, whilst 3, 5, 6 are quadratic non-residues, since

$$1^2 \equiv 6^2 \equiv 1, \qquad 2^2 \equiv 5^2 \equiv 4, \qquad 3^2 \equiv 4^2 \equiv 2.$$

(3) Modulo 11, the integers $1, 3, 4, 5, 9$ are quadratic residues, whilst $2, 6, 7, 8, 10$ are quadratic nonresidues, since

$$1^2 \equiv 10^2 \equiv 1, \qquad 2^2 \equiv 9^2 \equiv 4, \qquad 3^2 \equiv 8^2 \equiv 9,$$
$$4^2 \equiv 7^2 \equiv 5, \qquad 5^2 \equiv 6^2 \equiv 3.$$

(4) Modulo 15, only the integers 1 and 4 are quadratic residues, whilst $2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14$ are all quadratic nonresidues, since

$$1^2 \equiv 4^2 \equiv 11^2 \equiv 14^2 \equiv 1, \qquad 2^2 \equiv 7^2 \equiv 8^2 \equiv 13^2 \equiv 4.$$

(5) Modulo 23, the integers $1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18$ are quadratic residues, whilst $5, 7, 10, 11, 14, 15, 17, 19, 20, 21, 22$ are quadratic nonresidues, since

$$\begin{array}{lll} 1^2 \equiv 22^2 \equiv 1, & 5^2 \equiv 18^2 \equiv 2, & 7^2 \equiv 16^2 \equiv 3, \\ 2^2 \equiv 21^2 \equiv 4, & 11^2 \equiv 12^2 \equiv 6, & 10^2 \equiv 13^2 \equiv 8, \\ 3^2 \equiv 20^2 \equiv 9, & 9^2 \equiv 14^2 \equiv 12, & 6^2 \equiv 17^2 \equiv 13, \\ 4^2 \equiv 19^2 \equiv 16, & 8^2 \equiv 15^2 \equiv 18. & \end{array}$$

The above example illustrates the following two theorems:

Theorem 2.64 *Let p be an odd prime and $N(p)$ the number of consecutive pairs of quadratic residues modulo p in the interval $[1, p-1]$. Then*

$$N(p) = \frac{1}{4}\left(p - 4 - (-1)^{(p-1)/2}\right). \tag{2.137}$$

Proof: (Sketch) The complete proof of this theorem can be found in [1] and [2]; here we only give a sketch of the proof. Let (**RR**), (**RN**), (**NR**) and (**NN**) denote the number of pairs of two quadratic residues, of a quadratic residue followed by a quadratic non-residue, of a quadratic non-residue followed by a quadratic residue, of two quadratic non-residues, among pairs of consecutive positive integers less than p, respectively. Then

$$(\mathbf{RR}) + (\mathbf{RN}) = \frac{1}{2}\left(p - 2 - (-1)^{(p-1)/2}\right)$$

$$(\mathbf{NR}) + (\mathbf{NN}) = \frac{1}{2}\left(p - 2 + (-1)^{(p-1)/2}\right)$$

$$(\mathbf{RR}) + (\mathbf{NR}) = \frac{1}{2}(p-1) - 1$$

$$(\mathbf{RN}) + (\mathbf{NN}) = \frac{1}{2}(p-1)$$

$$(\mathbf{RR}) + (\mathbf{NN}) - (\mathbf{RN}) - (\mathbf{NR}) = -1$$

$$(\mathbf{RR}) + (\mathbf{NN}) = \frac{1}{2}(p-3)$$

$$(\mathbf{RR}) - (\mathbf{NN}) = -\frac{1}{2}\left(1 + (-1)^{(p-1)/2}\right)$$

Hence $(\mathbf{RR}) = \frac{1}{4}\left(p - 4 - (-1)^{(p-1)/2}\right)$. ∎

Remark 2.15 Similarly, let $\nu(p)$ denote the number of consecutive triples of quadratic residues in the interval $[1, p-1]$, where p is odd prime. Then

$$\nu(p) = \frac{1}{8}p + E_p, \tag{2.138}$$

where $|E_p| < \frac{1}{8}\sqrt{p} + 2$.

Example 2.57 For $p = 23$, there are five consecutive pairs of quadratic residues, namely, (1, 2), (2, 3), (3, 4), (8, 9), and (12, 13), modulo 23; there is also one consecutive triple of quadratic residues, namely, (1, 2, 3), modulo 23.

Theorem 2.65 *Let p be an odd prime. Then there are exactly $(p-1)/2$ quadratic residues and exactly $(p-1)/2$ quadratic nonresidues modulo p.*

Proof: Consider the $p-1$ congruences:

$$x^2 \equiv 1 \pmod{p}$$
$$x^2 \equiv 2 \pmod{p}$$
$$\vdots$$
$$x^2 \equiv p-1 \pmod{p}.$$

Since each of the above congruences has either no solution or exactly two congruence solutions modulo p, there must be exactly $(p-1)/2$ quadratic residues modulo p among the integers $1, 2, \ldots, p-1$. The remaining

$$p - 1 - (p-1)/2 = (p-1)/2$$

positive integers less than $p-1$ are quadratic nonresidues modulo p. ∎

Example 2.58 Again for $p = 23$, there are eleven quadratic residues, and eleven quadratic nonresidues modulo 23.

Euler devised a simple criterion for deciding whether an integer a is a quadratic residue modulo a prime number p.

Theorem 2.66 (Euler's criterion) *Let p be an odd prime and* $\gcd(a, p) = 1$. *Then a is a quadratic residue modulo p if and only if*

$$a^{(p-1)/2} \equiv 1 \,(\mathrm{mod}\ p).$$

Proof: Using Fermat's little theorem, we find that

$$\left(a^{(p-1)/2} - 1\right)\left(a^{(p-1)/2} + 1\right) \equiv a^{p-1} - 1 \equiv 0 \,(\mathrm{mod}\ p)$$

and thus $a^{(p-1)/2} \equiv 1 \,(\mathrm{mod}\ p)$. If a is a quadratic residue modulo p, then there exists an integer x_0 such that $x_0^2 \equiv a (\mathrm{mod}\ p)$. By Fermat's little theorem, we have

$$a^{(p-1)/2} \equiv (x_0^2)^{(p-1)/2} \equiv x_0^{p-1} \equiv 1 \,(\mathrm{mod}\ p).$$

To prove the converse, we assume that $a^{(p-1)/2} \equiv 1 \,(\mathrm{mod}\ p)$. If g is a primitive root modulo p (g is a primitive root modulo p if $\mathrm{order}(g, p) = \phi(p)$; we shall formally define primitive roots in Section 2.5), then there exists a positive integer t such that $g^t \equiv a \,(\mathrm{mod}\ p)$. Then

$$g^{t(p-1)/2} \equiv a^{(p-1)/2} \equiv 1 \,(\mathrm{mod}\ p)$$

which implies that

$$t(p-1)/2 \equiv 0 \,(\mathrm{mod}\ p-1).$$

Thus, t is even, and so

$$(g^{t/2})^2 \equiv g^t \equiv a \,(\mathrm{mod}\ p)$$

which implies that a is a quadratic residue modulo p. ∎

Euler's criterion is not very useful as a practical test for deciding whether or not an integer is a quadratic residue, unless the modulus is small. Euler's studies on quadratic residues were further developed by Legendre, who introduced the Legendre symbol.

Definition 2.45 Let p be an odd prime and a an integer. Suppose that $\gcd(a, p) = 1$. Then the *Legendre symbol*, $\left(\frac{a}{p}\right)$, is defined by

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1, & \text{if } a \text{is a quadratic non-residue modulo } p. \end{cases} \tag{2.139}$$

We shall use the notation $a \in Q_p$ to denote that a is a quadratic residue modulo p; similarly, $a \in \overline{Q}_p$ will be used to denote that a is a quadratic nonresidue modulo p.

Example 2.59 Let $p = 7$ and

$$1^2 \equiv 1 \pmod 7, \qquad 2^2 \equiv 4 \pmod 7, \qquad 3^2 \equiv 2 \pmod 7,$$
$$4^2 \equiv 2 \pmod 7, \qquad 5^2 \equiv 4 \pmod 7, \qquad 6^2 \equiv 1 \pmod 7.$$

Then

$$\left(\frac{1}{7}\right) = \left(\frac{2}{7}\right) = \left(\frac{4}{7}\right) = 1, \quad \left(\frac{3}{7}\right) = \left(\frac{5}{7}\right) = \left(\frac{6}{7}\right) = -1.$$

Some elementary properties of the Legendre symbol, which can be used to evaluate it, are given in the following theorem.

Theorem 2.67 *Let p be an odd prime, and a and b integers that are relatively prime to p. Then*

(1) *If* $a \equiv b \pmod p$, *then* $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$;
(2) $\left(\frac{a^2}{p}\right) = 1$, *and so* $\left(\frac{1}{p}\right) = 1$;
(3) $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$;
(4) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$;
(5) $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$.

Proof: Assume p is an odd prime and $\gcd(p, a) = \gcd(p, b) = 1$.

(1) If $a \equiv b \pmod p$, then $x^2 \equiv a \pmod p$ has a solution if and only if $x^2 \equiv b \pmod p$ has a solution. Hence $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
(2) The quadratic congruence $x^2 \equiv a^2 \pmod p$ clearly has a solution, namely a, so $\left(\frac{a^2}{p}\right) = 1$.
(3) This is Euler's criterion in terms of Legendre's symbol.
(4) We have

$$\left(\frac{ab}{p}\right) \equiv (ab)^{(p-1)/2} \pmod p \quad \text{(by Euler's criterion)} \tag{2.140}$$

$$\equiv a^{(p-1)/2} b^{(p-1)/2} \pmod{p} \tag{2.141}$$

$$\equiv \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \tag{2.142}$$

(5) By Euler's criterion, we have

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}.$$

This completes the proof. ∎

Corollary 2.9 *Let p be an odd prime. Then*

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{if } p \equiv 1 \pmod{4} \\ -1 & \text{if } p \equiv 3 \pmod{4}. \end{cases} \tag{2.143}$$

Proof: If $p \equiv 1 \pmod{4}$, then $p = 4k + 1$ for some integer k. Thus,

$$(-1)^{(p-1)/2} = (-1)^{((4k+1)-1)/2} = (-1)^{2k} = 1,$$

so that $\left(\frac{-1}{p}\right) = 1$. The proof for $p \equiv 3 \pmod{4}$ is similar. ∎

Example 2.60 Does $x^2 \equiv 63 \pmod{11}$ have a solution? We first evaluate the Legendre symbol $\left(\frac{63}{11}\right)$ corresponding to the quadratic congruence as follows:

$$\begin{aligned} \left(\frac{63}{11}\right) &= \left(\frac{8}{11}\right) && \text{by (1) of Theorem 2.67} \\ &= \left(\frac{2}{11}\right)\left(\frac{2^2}{11}\right) && \text{by (2) of Theorem 2.67} \\ &= \left(\frac{2}{11}\right) \cdot 1 && \text{by (2) of Theorem 2.67} \\ &= -1 && \text{by "trial and error".} \end{aligned}$$

Therefore, the quadratic congruence $x^2 \equiv 63 \pmod{11}$ has no solution.

To avoid the "trial-and-error" in the above and similar examples, we introduce in the following the so-called Gauss's lemma for evaluating the Legendre symbol.

Definition 2.46 Let $a \in \mathbb{Z}$ and $n \in \mathbb{N}$. Then the *least residue* of a modulo n is the integer a' in the interval $(-n/2, n/2]$ such that $a \equiv a' \pmod{n}$. We denote the least residue of a modulo n by $\mathrm{LR}_n(a)$.

Example 2.61 The set $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$ is a complete set of of the least residues modulo 11. Thus, $\mathrm{LR}_{11}(21) = -1$ since $21 \equiv 10 \equiv -1 \pmod{11}$; similarly, $\mathrm{LR}_{11}(99) = 0$ and $\mathrm{LR}_{11}(70) = 4$.

Lemma 2.3 (Gauss's lemma) Let p be an odd prime number and suppose that $\gcd(a, p) = 1$. Further let ω be the number of integers in the set

$$\left\{1a,\ 2a,\ 3a,\ \ldots,\ \left(\frac{p-1}{2}\right)a\right\}$$

whose least residues modulo p are negative, then

$$\left(\frac{a}{p}\right) = (-1)^{\omega}. \tag{2.144}$$

Proof: When we reduce the following numbers (modulo p)

$$\left\{a,\ 2a,\ 3a,\ \ldots,\ \left(\frac{p-1}{2}\right)a\right\}$$

to lie in set

$$\left\{\pm 1, \pm 2, \ldots,\ \pm\left(\frac{p-1}{2}\right)\right\},$$

then no two different numbers ma and na can go to the same numbers. Further, it cannot happen that ma goes to k and na goes to $-k$, because then $ma + na \equiv k + (-k) \equiv 0 \pmod{p}$, and hence (multiplying by the inverse of a), $m + n \equiv 0 \pmod{p}$, which is impossible. Hence, when reducing the numbers

$$\left\{a,\ 2a,\ 3a,\ \ldots,\ \left(\frac{p-1}{2}\right)a\right\}$$

we get exactly one of -1 and 1, exactly one of -2 and 2, $\cdots$, exactly one of $-(p-1)/2$ and $(p-1)/2$. Hence, modulo p, we get

$$a \cdot 2a \cdots \left(\frac{p-1}{2}\right)a \equiv 1 \cdot 2 \cdots \left(\frac{p-1}{2}\right)(-1)^{\omega} \pmod{p}.$$

Cancelling the numbers $1, 2, \ldots, (p-1)/2$, we have

$$a^{(p-1)/2} \equiv (-1)^{\omega} \pmod{p}.$$

By Euler's criterion, we have $\left(\frac{a}{p}\right) \equiv (-1)^{\omega} \pmod{p}$. Since $\left(\frac{a}{p}\right) \equiv \pm 1$, we must have $\left(\frac{a}{p}\right) = (-1)^{\omega}$. ∎

Example 2.62 Use Gauss's lemma to evaluate the Legendre symbol $\left(\frac{6}{11}\right)$. By Gauss's lemma, $\left(\frac{6}{11}\right) = (-1)^{\omega}$, where ω is the number of integers in the set

$$\{1 \cdot 6,\ 2 \cdot 6,\ 3 \cdot 6,\ 4 \cdot 6,\ 5 \cdot 6\}$$

whose least residues modulo 11 are negative. Clearly,

$$(6, 12, 18, 24, 30) \bmod 11 \equiv (6, 1, 7, 2, 8) \equiv (-5, 1, -4, 2, -3) \pmod{11}$$

So there are 3 least residues that are negative. Thus, $\omega = 3$. Therefore, $\left(\frac{6}{11}\right) = (-1)^3 = -1$. Consequently, the quadratic congruence $x^2 \equiv 6 \pmod{11}$ is not solvable.

Remark 2.16 Gauss's lemma is similar to Euler's criterion in the following ways:

(1) Gauss's lemma provides a method for direct evaluation of the Legendre symbol;
(2) It has more significance as a theoretical tool than as a computational tool.

Gauss's lemma provides, among many other things, a means for deciding whether or not 2 is a quadratic residue modulo and odd prime p.

Theorem 2.68 *If p is an odd prime, then*

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} = \begin{cases} 1, & \text{if } p \equiv \pm 1 \pmod{8} \\ -1, & \text{if } p \equiv \pm 3 \pmod{8}. \end{cases} \tag{2.145}$$

Proof: By Gauss's lemma, we know that if ω is the number of least positive residues of the integers

$$1 \cdot 2, 2 \cdot 2, \ldots, \frac{p-1}{2} \cdot 2$$

that are greater than $p/2$, then $\left(\frac{2}{p}\right) = (-1)^{\omega}$. Let $k \in \mathbb{Z}$ with $1 \le k \le (p-1)/2$. Then $2k < p/2$ if and only if $k < p/4$; so $[p/4]$ of the integers $1 \cdot 2, 2 \cdot 2, \ldots, \frac{p-1}{2} \cdot 2$ are less than $p/2$. So there are $\omega = (p-1)/2 - [p/4]$ integers greater than $p/2$. Therefore, by Gauss's lemma, we have

$$\left(\frac{2}{p}\right) = (-1)^{\frac{p-1}{2} - \left[\frac{p}{4}\right]}.$$

For the first equality, it suffices to show that

$$\frac{p-1}{2} - \left[\frac{p}{4}\right] \equiv \frac{p^2-1}{8} \pmod 2.$$

If $p \equiv 1 \pmod 8$, then $p = 8k+1$ for some $k \in \mathbb{Z}$, from which

$$\frac{p-1}{2} - \left[\frac{p}{4}\right] = \frac{(8k+1)-1}{2} - \left[\frac{8k+1}{4}\right] = 4k - 2k = 2k \equiv 0 \pmod 2,$$

and

$$\frac{p^2-1}{8} = \frac{(8k+1)^2-1}{8} = \frac{64k^2+16k}{8} = 8k^2 + 2k \equiv 0 \pmod 2,$$

so the desired congruence holds for $p \equiv 1 \pmod 8$. The cases for $p \equiv -1, \pm 3 \pmod 8$ are similar. This completes the proof for the first equality of the theorem. Note that the cases above yield

$$\frac{p^2-1}{8} = \begin{cases} \text{even}, & \text{if } p \equiv \pm 1 \pmod 8 \\ \text{odd}, & \text{if } p \equiv \pm 3 \pmod 8 \end{cases}$$

which implies

$$(-1)^{(p^2-1)/8} = \begin{cases} 1, & \text{if } p \equiv \pm 1 \pmod 8 \\ -1, & \text{if } p \equiv \pm 3 \pmod 8 \end{cases}$$

This completes the second equality of the theorem. ∎

Example 2.63 Evaluate $\left(\frac{2}{7}\right)$ and $\left(\frac{2}{53}\right)$.

(1) By Theorem 2.68, we have $\left(\frac{2}{7}\right) = 1$, since $7 \equiv -1 \pmod 8$. Consequently, the quadratic congruence $x^2 \equiv 2 \pmod 7$ is solvable.

(2) By Theorem 2.68, we have $\left(\frac{2}{53}\right) = -1$, since $53 \equiv -3 \pmod 8$. Consequently, the quadratic congruence $x^2 \equiv 2 \pmod{53}$ is not solvable.

Using Lemma 2.3, Gauss proved the following theorem, which is one of the great results of mathematics:

Theorem 2.69 (Quadratic reciprocity law) *If p and q are distinct odd primes, then*

(1) $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$ *if one of* $p, q \equiv 1 \pmod 4$*;*
(2) $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$ *if both* $p, q \equiv 3 \pmod 4$.

Remark 2.17 This theorem may be stated equivalently in the form

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}. \tag{2.146}$$

Proof: We first observe that, by Gauss's lemma, $\left(\frac{p}{q}\right) = 1^{\omega}$, where ω is the number of lattice points (x, y) (that is, pairs of integers) satisfying $0 < x < q/2$ and $-q/2 < px - qy < 0$. These inequalities give $y < (px/q) + 1/2 < (p+1)/2$. Hence, since y is an integer, we see ω is the number of lattice points in the rectangle R defined by $0 < x < q/2$, $0 < y < p/2$, satisfying $-q/2 < px - qy < 0$ (see Figure 2.2). Similarly, $\left(\frac{q}{p}\right) = 1^{\mu}$, where μ is the number of lattice points in R satisfying $-p/2 < qx - py < 0$. Now it suffices to prove that $(p-1)(q-1)/4 - (\omega + \mu)$ is even. But $(p-1)(q-1)/4$ is just the number of lattice points in R satisfying that $px - qy \le q/2$ or $qy - px \le -p/2$. The regions in R defined by these inequalities are disjoint and they contain the same number of lattice points, since the substitution

$$x = (q+1)/2 - x',$$
$$y = (p+1)/2 - y'$$

furnishes a one-to-one correspondence between them. The theorem follows. ∎

Remark 2.18 The Quadratic Reciprocity Law was one of Gauss's major contributions to mathematics. For those who consider number theory "the Queen of Mathematics," this is one of the jewels in her crown. Since Gauss's time, over 150 proofs of it have been published; Gauss himself published no less than six different proofs. Among the eminent mathematicians who contributed to the proofs are Cauchy, Jacobi, Dirichlet, Eisenstein, Kronecker, and Dedekind.

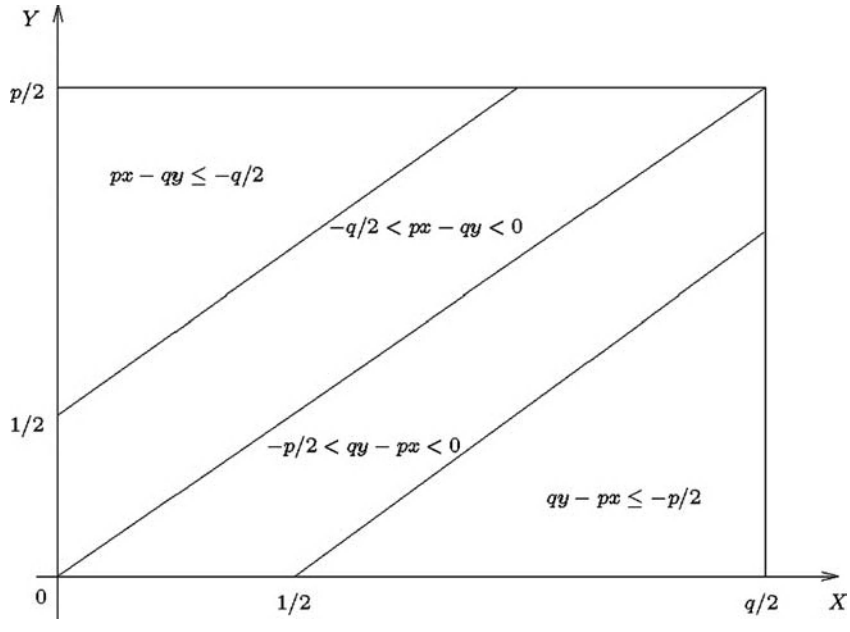


Figure 2.2 Proof of the quadratic reciprocity law

Combining all the above results for Legendre symbols, we get the following set of formulas for evaluating Legendre symbols:

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p} \tag{2.147}$$

$$\left(\frac{1}{p}\right) = 1 \tag{2.148}$$

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} \tag{2.149}$$

$$a \equiv b \pmod{p} \Longrightarrow \left(\frac{a}{p}\right) = \left(\frac{b}{p}\right) \tag{2.150}$$

$$\left(\frac{a_1 a_2 \cdots a_k}{p}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{p}\right)\cdots\left(\frac{a_k}{p}\right) \tag{2.151}$$

$$\left(\frac{ab^2}{p}\right) = \left(\frac{a}{p}\right), \text{for} p \nmid b \tag{2.152}$$

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} \tag{2.153}$$

$$\left(\frac{p}{q}\right) = (-1)^{(p-1)(q-1)/4}\left(\frac{q}{p}\right) \tag{2.154}$$

Example 2.64 Evaluate the Legendre symbol $\left(\frac{33}{83}\right)$.

$$\begin{aligned}
\left(\frac{33}{83}\right) &= \left(\frac{-50}{83}\right) && \text{by (2.150)}\\
&= \left(\frac{-2}{83}\right)\left(\frac{5^2}{83}\right) && \text{by (2.151)}\\
&= \left(\frac{-2}{83}\right) && \text{by (2.152)}\\
&= -\left(\frac{2}{83}\right) && \text{by (2.149)}\\
&= 1 && \text{by (2.153)}
\end{aligned}$$

It follows that the quadratic congruence $33 \equiv x^2 \pmod{83}$ is soluble.

Example 2.65 Evaluate the Legendre symbol $\left(\frac{46}{997}\right)$.

$$\begin{aligned}
\left(\frac{46}{997}\right) &= \left(\frac{2}{997}\right)\left(\frac{23}{997}\right) && \text{by (2.151)}\\
&= -\left(\frac{23}{997}\right) && \text{by (2.153)}\\
&= -\left(\frac{997}{23}\right) && \text{by (2.154)}\\
&= -\left(\frac{8}{23}\right) && \text{by (2.150)}\\
&= -\left(\frac{2^2 \cdot 2}{23}\right) && \text{by (2.151)}\\
&= -\left(\frac{2}{23}\right) && \text{by (2.152)}\\
&= -1 && \text{by (2.153)}
\end{aligned}$$

It follows that the quadratic congruence $46 \equiv x^2 \pmod{997}$ is not soluble.

Gauss's Quadratic Reciprocity Law enables us to evaluate the values of Legendre symbols $\left(\frac{a}{p}\right)$ very quickly provided a is a prime or a product of primes, and p is an odd prime. However, when a is a composite, we must factor it into its prime factorization form in order to use Gauss's quadratic reciprocity law. Unfortunately, there is no efficient algorithm so far for prime factorization (see Chapter 3 for more information). One way to overcome the difficulty of factoring a is to introduce the following Jacobi symbol (in honor of the German mathematician Carl Gustav Jacobi (1804–1851), which is a natural generalization of the Legendre symbol:

Definition 2.47 Let a be an integer and $n > 1$ an odd positive integer. If $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, then the *Jacobi symbol*, $\left(\frac{a}{n}\right)$, is defined by

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k}, \tag{2.155}$$

where $\left(\frac{a}{p_i}\right)$ for $i = 1, 2, \ldots, k$ is the Legendre symbol for the odd prime p_i. If n is an odd prime, the Jacobi symbol is *just* the Legendre symbol.

The Jacobi symbol has some similar properties to the Legendre symbol, as shown in the following theorem.

Theorem 2.70 *Let m and n be any positive odd composites, and* $\gcd(a, n) = \gcd(b, n) = 1$. *Then*

(1) *If* $a \equiv b \pmod{n}$, *then* $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$;
(2) $\left(\frac{a}{n}\right)\left(\frac{b}{n}\right) = \left(\frac{ab}{n}\right)$;
(3) *If* $\gcd(m, n) = 1$, *then* $\left(\frac{a}{mn}\right)\left(\frac{a}{m}\right) = \left(\frac{a}{n}\right)$;
(4) $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$;
(5) $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$;
(6) *If* $\gcd(m, n) = 1$, *then* $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4}$.

Remark 2.19 It should be noted that the Jacobi symbol $\left(\frac{a}{n}\right) = 1$ does not imply that a is a quadratic residue modulo n. Indeed a is a quadratic residue modulo n if and only if a is a quadratic residue modulo p for each prime divisor p of n. For example, the Jacobi symbol $\left(\frac{2}{3599}\right) = 1$, but the quadratic congruence $x^2 \equiv 2 \pmod{3599}$ is actually not soluble. This is the significant difference between the Legendre symbol and the Jacobi symbol. However, $\left(\frac{a}{n}\right) = -1$ does imply that a is a quadratic nonresidue modulo n. For example, the Jacobi symbol

$$\left(\frac{6}{35}\right) = \left(\frac{6}{5}\right)\left(\frac{6}{7}\right) = \left(\frac{1}{5}\right)\left(\frac{-1}{7}\right) = -1,$$

and so we can conclude that 6 is a quadratic nonresidue modulo 35. In short, we have

$$\left.\begin{aligned}\left(\frac{a}{p}\right) &= \begin{cases} 1, & a \equiv x^2 \pmod{p} \text{ is soluble} \\ -1, & a \equiv x^2 \pmod{p} \text{ is not soluble} \end{cases} \\ \left(\frac{a}{n}\right) &= \begin{cases} 1, & a \equiv x^2 \pmod{n} \text{ may or may not be soluble} \\ -1, & a \equiv x^2 \pmod{n} \text{ is not soluble} \end{cases}\end{aligned}\right\} \tag{2.156}$$

Combining all the above results for Jacobi symbols, we get the following set of formulas for evaluating Jacobi symbols:

$$\left(\frac{1}{n}\right) = 1 \tag{2.157}$$

$$\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2} \tag{2.158}$$

$$a \equiv b \pmod{p} \Longrightarrow \left(\frac{a}{n}\right) = \left(\frac{b}{n}\right) \tag{2.159}$$

$$\left(\frac{a_1 a_2 \cdots a_k}{n}\right) = \left(\frac{a_1}{n}\right)\left(\frac{a_2}{n}\right)\cdots\left(\frac{a_k}{n}\right) \tag{2.160}$$

$$\left(\frac{ab^2}{n}\right) = \left(\frac{a}{n}\right), \text{for } \gcd(b, n) = 1 \tag{2.161}$$

$$\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8} \tag{2.162}$$

$$\left(\frac{m}{n}\right) = (-1)^{(m-1)(m-1)/4}\left(\frac{n}{m}\right) \tag{2.163}$$

Example 2.66 Evaluate the Jacobi symbol $\left(\frac{286}{563}\right)$.

$$\begin{aligned}\left(\frac{286}{563}\right) &= \left(\frac{2}{563}\right)\left(\frac{143}{563}\right) && \text{by (2.160)} \\ &= -\left(\frac{143}{563}\right) && \text{by (2.162)} \\ &= \left(\frac{563}{143}\right) && \text{by (2.163)} \\ &= \left(\frac{-3^2}{143}\right) && \text{by (2.149)} \\ &= -\left(\frac{3^2}{143}\right) && \text{by (2.158)} \\ &= -1 && \text{by (2.161)}\end{aligned}$$

It follows that the quadratic congruence $286 \equiv x^2 \pmod{563}$ is not soluble.

Example 2.67 Evaluate the Jacobi symbol $\left(\frac{1009}{2307}\right)$.

$$\begin{aligned}\left(\frac{1009}{2307}\right) &= \left(\frac{2307}{1009}\right) && \text{by (2.163)}\\ &= \left(\frac{289}{1009}\right) && \text{by (2.159)}\\ &= \left(\frac{17^2}{1009}\right) && \text{by (2.160)}\\ &= 1 && \text{by (2.161)}\end{aligned}$$

Although the Jacobi symbol $\left(\frac{1009}{2307}\right) = 1$, we still cannot determine whether or not the quadratic congruence $1009 \equiv x^2 \pmod{2307}$ is soluble.

Remark 2.20 Jacobi symbols can be used to facilitate the calculation of Legendre symbols. In fact, Legendre symbols can be eventually calculated by Jacobi symbols. That is, the Legendre symbol can be calculated as if it were a Jacobi symbol. For example, consider the Legendre symbol $\left(\frac{335}{2999}\right)$, where $335 = 5 \cdot 67$ is not a prime (of course, 2999 is prime, otherwise, it would not be a Legendre symbol). To evaluate this Legendre symbol, we first regard it as a Jacobi symbol and evaluate it as if it were a Jacobi symbol (note that once it is regarded as a Jacobi symbol, it does not matter whether or not 335 is prime; it even does not matter whether or not 2999 is prime, but anyway, it is a Legendre symbol).

$$\left(\frac{335}{2999}\right) = -\left(\frac{2999}{335}\right) = -\left(\frac{-16}{335}\right) = -\left(\frac{-1 \cdot 4^2}{335}\right) = -\left(\frac{-1}{335}\right) = 1.$$

Since 2999 is prime, $\left(\frac{335}{2999}\right)$ is a Legendre symbol, and so 355 is a quadratic residue modulo 2999.

Example 2.68 In Table 2.4, we list the elements in $(\mathbb{Z}/21\mathbb{Z})^*$ and their Jacobi symbols. Incidentally, exactly half of the Legendre and Jacobi symbols $\left(\frac{a}{3}\right)$, $\left(\frac{a}{7}\right)$ and $\left(\frac{a}{21}\right)$ are equal to 1 and half equal to -1. Also for those Jacobi symbols $\left(\frac{a}{21}\right) = 1$, exactly half of the a's are

Table 2.4 Jacobi Symbols for $a \in (\mathbb{Z}/21\mathbb{Z})^*$

$a \in (\mathbb{Z}/21\mathbb{Z})^*$	1	2	4	5	8	10	11	13	16	17	19	20
$a^2 \bmod 21$	1	4	16	4	1	16	16	1	4	16	4	1
$\left(\frac{a}{3}\right)$	1	−1	1	−1	−1	1	−1	1	1	−1	1	−1
$\left(\frac{a}{7}\right)$	1	1	1	−1	1	−1	1	−1	1	−1	−1	−1
$\left(\frac{a}{21}\right)$	1	−1	1	1	−1	−1	−1	−1	1	1	−1	1

indeed quadratic residues, whereas the other half are not. (Note that a is a quadratic residue of 21 if and only if it is a quadratic residue of both 3 and 7.) That is,

$$\left(\frac{a}{3}\right) = \begin{cases} 1, & \text{for } a \in \{1, 4, 10, 13, 16, 19\} = Q_3 \\ -1, & \text{for } a \in \{2, 5, 8, 11, 17, 20\} = \overline{Q}_3 \end{cases}$$

$$\left(\frac{a}{7}\right) = \begin{cases} 1, & \text{for } a \in \{1, 2, 4, 8, 11, 16\} = Q_7 \\ -1, & \text{for } a \in \{5, 10, 13, 17, 19, 20\} = \overline{Q}_7 \end{cases}$$

$$\left(\frac{a}{21}\right) = \begin{cases} 1, & \text{for } a \in \{1, 4, 5, 16, 17, 20\} \quad \begin{cases} a \in \{1, 4, 16\} = Q_{21} \\ a \in \{5, 17, 20\} \subset \overline{Q}_{21} \end{cases} \\ -1, & \text{for } a \in \{2, 8, 10, 11, 13, 19\} \subset \overline{Q}_{21}. \end{cases}$$

Problems for Section 2.4

1. Solve the following system of linear congruences:

$$\begin{cases} 2x \equiv 1 \pmod 3 \\ 3x \equiv 1 \pmod 5 \\ 5x \equiv 1 \pmod 7. \end{cases}$$

2. Prove that n is prime if $\gcd(a, n) = 1$ and

$$a^{n-1} \equiv 1 \pmod n$$

but

$$a^m \not\equiv 1 \pmod n$$

for each divisor m of $n - 1$.

3. Show that the congruence

$$x^{p-1} \equiv 1 \pmod{p^k}$$

has just $p - 1$ solutions modulo p^k for every prime power p^k.

4. Show that for any positive integer n, either there is no primitive root modulo n or there are $\phi(\phi(n))$ primitive roots modulo n. (Note: Primitive roots are defined in Definition 2.49.)

5. Let D be the sum of all the distinct primitive roots modulo a prime p. Show that

$$D \equiv \mu(p-1) \pmod{n}.$$

6. Let n be a positive integer such that $n \equiv 3 \pmod 4$. Show that there are no integer solutions in x for

$$x^2 \equiv -1 \pmod{n}.$$

7. Show that for any prime p,

$$\sum_{j=1}^{p-1} \equiv -1 \pmod 4.$$

8. Suppose $p \equiv 3 \pmod 4$ is prime. Show that

$$\left(\frac{p-1}{2}\right) \equiv \pm 1 \pmod{n}.$$

9. Let p be a prime. Show that for all positive integer $j \leq p-1$, we have

$$\binom{p}{j} \equiv 0 \pmod{p}.$$

10. Prove if $\gcd(n_i, n_j) = 1$, $i, j = 1, 2, 3, \ldots, k$, $i \neq j$, then

$$\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/n_1\mathbb{Z} \oplus \mathbb{Z}/n_2\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/n_k\mathbb{Z}.$$

11. Find the x in $2x^2 + 3x + 1 \equiv 0 \pmod 7$ and $2x^2 + 3x + 1 \equiv 0 \pmod{101}$.
12. Compute the values for the Legendre symbol:

$$\left(\frac{1234}{4567}\right), \quad \left(\frac{1356}{2467}\right).$$

13. Which of the following congruences have solutions? If they have, then how many do they have?

$$x^2 \equiv 2 \pmod{61}, \ x^2 \equiv -2 \pmod{61},$$

$$x^2 \equiv 2 \pmod{59}, \ x^2 \equiv -2 \pmod{59},$$

$$x^2 \equiv -1 \pmod{61}, \ x^2 \equiv -1 \pmod{59},$$

$$x^2 \equiv 5 \pmod{229}, \ x^2 \equiv -5 \pmod{229},$$

$$x^2 \equiv 10 \pmod{127}, \ x^2 \equiv 11 \pmod{61}.$$

14. Let p be a prime and $\gcd(a, p) = \gcd(b, p) = 1$. Prove that if $x^2 \equiv a \pmod{p}$, and $x^2 \equiv b \pmod{p}$ are not soluble, then $x^2 \equiv ab \pmod{p}$ is soluble.
15. Prove that if p is a prime of the form $4k + 1$ then the sum of the quadratic residue modulo p in the interval $[1, p)$ is $p(p - 1)/4$.
16. Prove that if r is the quadratic residue modulo $n > 2$, then

$$r^{\phi(n)/2} \equiv 1 \pmod{n}.$$

17. Let p, q be twin primes. Prove that there are infinitely many a such that $p \mid (a^2 - q)$ if and only if there are infinitely many b such that $q \mid (b^2 - p)$.
18. Prove that if $\gcd(a, p) = 1$ and p an odd prime, then

$$\sum_{n=1}^{p} \left(\frac{n^2 + a}{p} \right) = -1.$$

19. Prove that if $\gcd(a, p) = \gcd(b, p) = 1$ and p an odd prime, then

$$\sum_{n=1}^{p} \left(\frac{an + b}{p} \right) = -1.$$

20. Let p be an odd prime, and let $N_{++}(p)$ be the number of n, $1 \leq n < p - 2$ such that

$$\left(\frac{n}{p} \right) = \left(\frac{n + 1}{p} \right) = 1.$$

Prove that

$$N_{++}(p) = \left(\frac{p - \left(\frac{-1}{p} \right) - 4}{4} \right).$$

2.5 Primitive Roots

Definition 2.48 Let n be a positive integer and a an integer such that $\gcd(a, n) = 1$. Then the *order* of a modulo n, denoted by $\mathrm{ord}_n(a)$ or by $\mathrm{ord}(a, n)$, is the smallest integer r such that $a^r \equiv 1 \pmod{n}$.

Remark 2.21 The terminology "the order of a modulo n" is the modern algebraic term from group theory (the theory of groups, rings, and fields will be formally introduced in Section 2.1). The older terminology "a belongs to the exponent r" is the classical term from number theory as used by Gauss.

Table 2.5 Values of $a^i \bmod 11$, for $1 \leq i < 11$

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
1	1	1	1	1	1	1	1	1	1
2	4	8	5	10	9	7	3	6	[1]
3	9	5	4	[1]	3	9	5	4	1
4	5	9	3	[1]	4	5	9	3	1
5	3	4	9	[1]	5	3	4	9	1
6	3	7	9	10	5	8	4	2	[1]
7	5	2	3	10	4	6	9	8	[1]
8	9	6	4	10	3	2	5	7	[1]
9	4	3	5	[1]	9	4	3	5	1
10	[1]	10	1	10	1	10	1	10	1

Example 2.69 In Table 2.5, values of $a^i \bmod 11$ for $i = 1, 2, \ldots, 10$ are given. From Table 2.5, we get

$$\begin{aligned} \text{ord}_{11}(1) &= 1 \\ \text{ord}_{11}(2) &= \text{ord}_{11}(6) = \text{ord}_{11}(7) = \text{ord}_{11}(8) = 10 \\ \text{ord}_{11}(3) &= \text{ord}_{11}(4) = \text{ord}_{11}(5) = \text{ord}_{11}(9) = 5 \\ \text{ord}_{11}(10) &= 2 \end{aligned}$$

We list in the following theorem some useful properties of the order of an integer a modulo n.

Theorem 2.71 *Let n be a positive integer, $\gcd(a, n) = 1$, and $r = \text{ord}_n(a)$. Then*

(1) *If $a^m \equiv 1 \pmod{n}$, where m is a positive integer, then $r \mid m$;*
(2) *$r \mid \phi(n)$;*
(3) *For integers s and t, $a^s \equiv a^t \pmod{n}$ if and only if $s \equiv t \pmod{n}$;*
(4) *No two of the integers $a, a^2, a^3, \ldots, a^r$ are congruent modulo r;*
(5) *If m is a positive integer, then the order of a^m modulo n is $\dfrac{r}{\gcd(r, m)}$;*
(6) *The order of a^m modulo n is r if and only if $\gcd(m, r) = 1$.*

Definition 2.49 Let n be a positive integer and a an integer such that $\gcd(a, n) = 1$. If the order of an integer a modulo n is $\phi(n)$, that is, $\text{order}(a, n) = \phi(n)$, then a is called a *primitive root* of n.

Example 2.70 Determine whether or not 7 is a primitive root of 45. First note that $\gcd(7, 45) = 1$. Now observe that

$$\begin{array}{llll} 7^1 \equiv 7 & (\text{mod } 45) & 7^2 \equiv 4 & (\text{mod } 45) \\ 7^3 \equiv 28 & (\text{mod } 45) & 7^4 \equiv 16 & (\text{mod } 45) \\ 7^5 \equiv 22 & (\text{mod } 45) & 7^6 \equiv 19 & (\text{mod } 45) \\ 7^7 \equiv 43 & (\text{mod } 45) & 7^8 \equiv 31 & (\text{mod } 45) \\ 7^9 \equiv 37 & (\text{mod } 45) & 7^{10} \equiv 34 & (\text{mod } 45) \\ 7^{11} \equiv 13 & (\text{mod } 47) & 7^{12} \equiv 1 & (\text{mod } 45). \end{array}$$

Thus, $\text{ord}_{45}(7) = 12$. However, $\phi(45) = 24$. That is, $\text{ord}_{45}(7) \neq \phi(45)$. Therefore, 7 is not a primitive root of 45.

Example 2.71 Determine whether or not 7 is a primitive root of 46. First note that $\gcd(7, 46) = 1$. Now observe that

$$\begin{array}{llll} 7^1 \equiv 7 & (\text{mod } 46) & 7^2 \equiv 3 & (\text{mod } 46) \\ 7^3 \equiv 21 & (\text{mod } 46) & 7^4 \equiv 9 & (\text{mod } 46) \\ 7^5 \equiv 17 & (\text{mod } 46) & 7^6 \equiv 27 & (\text{mod } 46) \\ 7^7 \equiv 5 & (\text{mod } 46) & 7^8 \equiv 35 & (\text{mod } 46) \\ 7^9 \equiv 15 & (\text{mod } 46) & 7^{10} \equiv 13 & (\text{mod } 46) \\ 7^{11} \equiv 45 & (\text{mod } 46) & 7^{12} \equiv 39 & (\text{mod } 46) \\ 7^{13} \equiv 43 & (\text{mod } 46) & 7^{14} \equiv 25 & (\text{mod } 46) \\ 7^{15} \equiv 37 & (\text{mod } 46) & 7^{16} \equiv 29 & (\text{mod } 46) \\ 7^{17} \equiv 19 & (\text{mod } 46) & 7^{18} \equiv 41 & (\text{mod } 46) \\ 7^{19} \equiv 11 & (\text{mod } 46) & 7^{20} \equiv 31 & (\text{mod } 46) \\ 7^{21} \equiv 33 & (\text{mod } 46) & 7^{22} \equiv 1 & (\text{mod } 46). \end{array}$$

Thus, $\text{ord}_{46}(7) = 22$. Note also that $\phi(46) = 22$. That is, $\text{ord}_{46}(7) = \phi(46) = 22$. Therefore 7 is a primitive root of 46.

Theorem 2.72 (Primitive roots as residue system) *Suppose* $\gcd(g, n) = 1$. *If* g *is a primitive root modulo* n, *then the set of integers* $\{g, g^2, g^3, \ldots, g^{\phi(n)}\}$ *is a reduced system of residues modulo* n.

Example 2.72 Let $n = 34$. Then there are $\phi(\phi(34)) = 8$ primitive roots of 34, namely, 3, 5, 7, 11, 23, 27, 29, 31. Now let $g = 5$ such that $\gcd(g, n) = \gcd(5, 34) = 1$. Then

$$\begin{aligned} \{g, g^2, \ldots, g^{\phi(n)}\} &= \{5, 5^2, 5^3, 5^4, 5^5, 5^6, 5^7, 5^8, 5^9, 5^{10}, 5^{11}, 5^{12}, 5^{13}, 5^{14}, 5^{15}, 5^{16}\} \bmod 34 \\ &= \{5, 25, 23, 13, 31, 19, 27, 33, 29, 9, 11, 21, 3, 15, 7, 1\} \\ &= \{1, 3, 5, 7, 9, 11, 13, 15, 19, 21, 23, 25, 27, 29, 33, 31\} \end{aligned}$$

which forms a reduced system of residues modulo 34. We can of course choose $g = 23$ such that $\gcd(g, n) = \gcd(23, 34) = 1$. Then we have

$$\begin{aligned}
&\{g, g^2, \ldots, g^{\phi(n)}\} \\
&\quad = \{23,\ 23^2,\ 23^3,\ 23^4,\ 23^5,\ 23^6,\ 23^7,\ 23^8,\ 23^9,\ 23^{10},\ 23^{11},\ 23^{12},\ 23^{13},\ 23^{14}, \\
&\quad 23^{15},\ 23^{16}\} \bmod 34 \\
&\quad = \{23, 19, 29, 21, 7, 25, 31, 33, 11, 15, 5, 13, 27, 9, 3, 1\} \\
&\quad = \{1, 3, 5, 7, 9, 11, 13, 15, 19, 21, 23, 25, 27, 29, 33, 31\}
\end{aligned}$$

which again forms a reduced system of residues modulo 34.

Theorem 2.73 *If p is a prime number, then there exist $\phi(p-1)$ (incongruent) primitive roots modulo p.*

Example 2.73 Let $p = 47$, then there are $\phi(47-1) = 22$ primitive roots modulo 47, namely,

5	10	11	13	15	19	20	22	23	26	29
30	31	33	35	38	39	40	41	43	44	45

Note that no method is known for predicting what will be the smallest primitive root of a given prime p, nor is there much known about the distribution of the $\phi(p-1)$ primitive roots among the least residues modulo p.

Corollary 2.10 *If n has a primitive root, then there are $\phi(\phi(n))$ (incongruent) primitive roots modulo n.*

Example 2.74 Let $n = 46$, then there are $\phi(\phi(46)) = 10$ primitive roots modulo 46, namely,

5	7	11	15	17	19	21	33	37	43

Note that not all moduli n have primitive roots; in Table 2.6 we give the smallest primitive root g for $2 \leq n \leq 911$ that has primitive roots.

The following theorem establishes conditions for moduli to have primitive roots:

Theorem 2.74 *An integer $n > 1$ has a primitive root modulo n if and only if*

$$n = 2,\ 4,\ p^{\alpha},\ \text{or}\ 2p^{\alpha}, \tag{2.164}$$

where p is an odd prime and α is a positive integer.

Table 2.6 Primitive roots g modulo n (if any) for $1 \leq n \leq 911$

n	g	n	g	n	g	n	g	n	g	n	g	n	g	n	g	n	g	n	g
2	1	3	2	4	3	5	2	6	5	7	3	9	2	10	3	11	2	13	2
14	3	17	3	18	5	19	2	22	7	23	5	25	2	26	7	27	2	29	2
31	3	34	3	37	2	38	3	41	6	43	3	46	5	47	5	49	3	50	3
53	2	54	5	58	3	59	2	61	2	62	3	67	2	71	7	73	5	74	5
79	3	81	2	82	7	83	2	86	3	89	3	94	5	97	5	98	3	101	2
103	5	106	3	107	2	109	6	113	3	118	11	121	2	122	7	125	2	127	3
131	2	134	7	137	3	139	2	142	7	146	5	149	2	151	6	157	5	158	3
162	5	163	2	166	5	167	5	169	2	173	2	178	3	179	2	181	2	193	5
194	5	197	2	199	3	202	3	206	5	211	2	214	5	218	11	223	3	226	3
227	2	229	6	233	3	239	7	241	7	242	7	243	2	250	3	251	6	254	3
257	3	262	17	263	5	269	2	271	6	274	3	277	5	278	3	281	3	283	3
289	3	293	2	298	3	302	7	307	5	311	17	313	10	314	5	317	2	326	3
331	3	334	5	337	10	338	7	343	3	346	3	347	2	349	2	353	3	358	7
359	7	361	2	362	21	367	6	373	2	379	2	382	19	383	5	386	5	389	2
394	3	397	5	398	3	401	3	409	21	419	2	421	2	422	3	431	7	433	5
439	15	443	2	446	3	449	3	454	5	457	13	458	7	461	2	463	3	466	3
467	2	478	7	479	13	482	7	486	5	487	3	491	2	499	7	502	11	503	5
509	2	514	3	521	3	523	2	526	5	529	5	538	3	541	2	542	15	547	2
554	5	557	2	562	3	563	2	566	3	569	3	571	3	577	5	578	3	586	3
587	2	593	3	599	7	601	7	607	3	613	2	614	5	617	3	619	2	622	17
625	2	626	15	631	3	634	3	641	3	643	11	647	5	653	2	659	2	661	2
662	3	673	5	674	15	677	2	683	5	686	3	691	3	694	5	698	7	701	2
706	3	709	2	718	7	719	11	722	3	727	5	729	2	733	6	734	11	739	3
743	5	746	5	751	3	757	2	758	3	761	6	766	5	769	11	773	2	778	3
787	2	794	5	797	2	802	3	809	3	811	3	818	21	821	2	823	3	827	2
829	2	838	11	839	11	841	2	842	23	853	2	857	3	859	2	862	7	863	5
866	5	877	2	878	15	881	3	883	2	886	5	887	5	898	3	907	2	911	17

Corollary 2.11 *If $n = 2^\alpha$ with $\alpha \geq 3$, or $n = 2^\alpha p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ with $\alpha \geq 2$ or $k \geq 2$, then there are no primitive roots modulo n.*

Example 2.75 For $n = 16 = 2^4$, since it is of the form $n = 2^\alpha$ with $\alpha \geq 3$, there are no primitive roots modulo 16.

Although we know which numbers possess primitive roots, it is not a simple matter to find these roots. Except for trial and error methods, very few general techniques are known. Artin in 1927 made the following conjecture (Rose [5]):

Conjecture 2.1 Let $N_a(x)$ be the number of primes less than x of which a is a primitive root, and suppose a is not a square and is not equal to -1, 0 or 1. Then

$$N_a(x) \sim A\frac{x}{\ln x}, \tag{2.165}$$

where A depends only on a.

Hooley in 1967 showed that if the extended Riemann Hypothesis is true then so is Artin's conjecture. It is also interesting to note that before the age of computers Jacobi in 1839 listed all solutions $\{a, b\}$ of the congruences $g^a \equiv b(\text{mod } p)$ where $1 \leq a < p$, $1 \leq b < p$, g is the least positive primitive root of p and $p < 1000$.

Another very important problem concerning the primitive roots of p is the estimate of the lower bound of the least positive primitive root of p. Let p be a prime and $g(p)$ the least positive primitive root of p. The Chinese mathematician Yuan Wang [3] showed in 1959 that

(1) $g(p) = \mathcal{O}(p^{1/4+\epsilon})$;
(2) $g(p) = \mathcal{O}((\log p)^8)$, if the Generalized Riemann Hypothesis (GRH) is true.

Wang's second result was improved to $g(p) = \mathcal{O}((\log p)^6)$ by Victor Shoup [4] in 1992.

The concept of *index* of an integer modulo n was first introduced by Gauss in his *Disquisitiones Arithmeticae*. Given an integer n, if n has primitive root g, then the set

$$\{g, g^2, g^3, \ldots, g^{\phi(n)}\} \tag{2.166}$$

forms a reduced system of residues modulo n; g is a generator of the cyclic group of the reduced residues modulo n. (Clearly, the group $(\mathbb{Z}/n\mathbb{Z})^*$ is cyclic if $n = 2, 4, p^\alpha$, or $2p^\alpha$, for p odd prime and α positive integer.) Hence, if $\gcd(a, n) = 1$, then a can be expressed in the form:

$$a \equiv g^k \,(\text{mod } n) \tag{2.167}$$

for a suitable k with $1 \leq k \leq \phi(n)$. This motivates our following definition, which is an analog of the real base logarithm function.

Definition 2.50 Let g be a primitive root of n. If $\gcd(a, n) = 1$, then the smallest positive integer k such that $a \equiv g^k \pmod{n}$ is called the *index* of a to the base g modulo n and is denoted by $\text{ind}_{g,n}(a)$, or simply by $\text{ind}_g a$.

Clearly, by definition, we have

$$a \equiv g^{\text{ind}_g a} \pmod{n}. \tag{2.168}$$

The function $\text{ind}_g a$ is sometimes called the *discrete logarithm* and is denoted by $\log_g a$ so that

$$a \equiv g^{\log_g a} \pmod{n}. \tag{2.169}$$

Generally, the discrete logarithm is a computationally intractable problem; no efficient algorithm has been found for computing discrete logarithms and hence it has important applications in public key cryptography.

Theorem 2.75 (Index theorem) *If g is a primitive root modulo n, then $g^x \equiv g^y \pmod{n}$ if and only if $x \equiv y \pmod{\phi(n)}$.*

Proof: Suppose that $x \equiv y \pmod{\phi(n)}$. Then, $x = y + k\phi(n)$ for some integer k. Therefore,

$$\begin{aligned} g^x &\equiv g^{y+k\phi(n)} \pmod{n} \\ &\equiv g^y \cdot (g^{\phi(n)})^k \pmod{n} \\ &\equiv g^y \cdot 1^k \pmod{n} \\ &\equiv g^y \pmod{n}. \end{aligned}$$

The proof of the "only if" part of the theorem is left as an exercise. ∎

The properties of the function $\text{ind}_g a$ are very similar to those of the conventional real base logarithm function, as the following theorems indicate:

Theorem 2.76 *Let g be a primitive root modulo the prime p, and* $\gcd(a, p) = 1$. *Then $g^k \equiv a \pmod{p}$ if and only if*

$$k \equiv \text{ind}_g a \pmod{p - 1}. \tag{2.170}$$

Theorem 2.77 *Let n be a positive integer with primitive root g, and* $\gcd(a, n) = \gcd(b, n) = 1$. *Then*

(1) $\text{ind}_g 1 \equiv 0 \pmod{\phi(n)}$;
(2) $\text{ind}_g(ab) \equiv \text{ind}_g a + \text{ind}_g b \pmod{\phi(n)}$;
(3) $\text{ind}_g a^k \equiv k \cdot \text{ind}_g a \pmod{\phi(n)}$, *if k is a positive integer.*

Example 2.76 Compute the index of 15 base 6 modulo 109, that is, $6^{\text{ind}_6 15} \bmod 109 = 15$. To find the index, we just successively perform the computation $6^k \pmod{109}$ for $k = 1, 2, 3, \ldots$ until we find a suitable k such that $6^k \pmod{109} = 15$:

$$\begin{aligned}
6^1 &\equiv 6 \pmod{109} & 6^2 &\equiv 36 \pmod{109}\\
6^3 &\equiv 107 \pmod{109} & 6^4 &\equiv 97 \pmod{109}\\
6^5 &\equiv 37 \pmod{109} & 6^6 &\equiv 4 \pmod{109}\\
6^7 &\equiv 24 \pmod{109} & 6^8 &\equiv 35 \pmod{109}\\
6^9 &\equiv 101 \pmod{109} & 6^{10} &\equiv 61 \pmod{109}\\
6^{11} &\equiv 39 \pmod{109} & 6^{12} &\equiv 16 \pmod{109}\\
6^{13} &\equiv 96 \pmod{109} & 6^{14} &\equiv 31 \pmod{109}\\
6^{15} &\equiv 77 \pmod{109} & 6^{16} &\equiv 26 \pmod{109}\\
6^{17} &\equiv 47 \pmod{109} & 6^{18} &\equiv 64 \pmod{109}\\
6^{19} &\equiv 57 \pmod{109} & 6^{20} &\equiv 15 \pmod{109}.
\end{aligned}$$

Since $k = 20$ is the smallest positive integer such that $6^{20} \equiv 15 \pmod{109}$, $\text{ind}_6 15 \pmod{109} = 20$.

In what follows, we shall study the congruences of the form $x^k \equiv a \pmod{n}$, where n is an integer with primitive roots and $\gcd(a, n) = 1$. First of all, we present a definition, which is the generalization of quadratic residues.

Definition 2.51 Let a, n, and k be positive integers with $k \geq 2$. Suppose $\gcd(a, n) = 1$, then a is called a kth (higher) power residue of n if there is an x such that

$$x^k \equiv a \pmod{n}. \tag{2.171}$$

The set of all kth (higher) power residues is denoted by $K(k)_n$. If the congruence has no solution, then a is called a *kth (higher) power nonresidue* of n. The set of such a is denoted by $\overline{K(k)}_n$. For example, $K(9)_{126}$ would denote the set of the 9th power residues of 126, whereas $\overline{K(5)}_{31}$ the set of the 5th power nonresidue of 31.

Theorem 2.78 (kth power theorem) *Let n be a positive integer having a primitive root, and suppose* $\gcd(a, n) = 1$. *Then the congruence (2.171) has a solution if and only if*

$$a^{\phi(n)/\gcd(k,\phi(n))} \equiv 1 \pmod{n}. \tag{2.172}$$

If 2.171 is soluble, then it has exactly $\gcd(k, \phi(n))$ *incongruent solutions.*

Proof: Let x be a solution of $x^k \equiv a \pmod{n}$. Since $\gcd(a, n) = 1$, $\gcd(x, n) = 1$. Then

$$\begin{aligned}
a^{\phi(n)/\gcd(k,\phi(n))} &\equiv (x^k)^{\phi(n)/\gcd(k,\phi(n))}\\
&\equiv (x^{\phi(n)})^{k/\gcd(k,\phi(n))}\\
&\equiv 1^{k/\gcd(k,\phi(n))}\\
&\equiv 1 \pmod{n}.
\end{aligned}$$

Conversely, if $a^{\phi(n)/\gcd(k,\phi(n))} \equiv 1 \pmod{n}$, then $r^{(\mathrm{ind}_r a)\phi(n)/\gcd(k,\phi(n))} \equiv 1 \pmod{n}$. Since $\mathrm{ord}_n r = \phi(n)$, $\phi(n) \mid (\mathrm{ind}_r a)\phi(n)/\gcd(k, \phi(n))$, and hence $\gcd(k, \phi(n)) \mid \mathrm{ind}_r a$ because $(\mathrm{ind}_r a)/\gcd(k, \phi(n))$ must be an integer. Therefore, there are $\gcd(k, \phi(n))$ incongruent solutions to $k(\mathrm{ind}_r x) \equiv (\mathrm{ind}_r a) \pmod{\phi(n)}$ and hence $\gcd(k, \phi(n))$ incongruent solutions to $x^k \equiv a \pmod{n}$. ∎

If n is a prime number, say, p, then we have

Corollary 2.12 *Suppose p is prime and $\gcd(a, p) = 1$. Then a is a kth power residue of p if and only if*

$$a^{(p-1)/\gcd(k,(p-1))} \equiv 1 \pmod{p}. \tag{2.173}$$

Example 2.77 Determine whether or not 5 is a sixth power of 31, that is, decide whether or not the congruence

$$x^6 \equiv 5 \pmod{31}$$

has a solution. First of all, we compute

$$5^{(31-1)/\gcd(6,31-1)} \equiv 25 \not\equiv 1 \pmod{31}$$

since 31 is prime. By Corollary 2.12, 5 is not a sixth power of 31. That is, $5 \notin K(6)_{31}$. However,

$$5^{(31-1)/\gcd(7,31-1)} \equiv 1 \pmod{31}.$$

So, 5 is a seventh power of 31. That is, $5 \in K(7)_{31}$.

Now let us introduce a new symbol $\left(\frac{a}{p}\right)_k$, the kth power residue symbol, analogous to the Legendre symbol for quadratic residues.

Definition 2.52 Let p be an odd prime, $k > 1$, $k \mid p - 1$ and $q = \dfrac{p-1}{k}$. Then the symbol

$$\left(\frac{\alpha}{p}\right)_k = \alpha^q \bmod p \tag{2.174}$$

is called the kth power residue symbol modulo p, where $\alpha^q \bmod p$ represents the absolute

smallest residue of α^q modulo p. (The complete set of the absolute smallest residues modulo p are: $-(p-1)/2, \ldots, -1, 0, 1, \ldots, (p-1/2)$).

Theorem 2.79 *Let* $\left(\frac{\alpha}{p}\right)_k$ *be the kth power residue symbol. Then*

(1) $p \mid a \Longrightarrow \left(\dfrac{a}{p}\right)_k = 0$;

(2) $a \equiv a_1 \pmod{p} \Longrightarrow \left(\dfrac{a}{p}\right)_k = \left(\dfrac{a_1}{p}\right)_k$;

(3) *For* $a_1, a_2 \in \mathbb{Z} \Longrightarrow \left(\dfrac{a_1 a_2}{p}\right)_k \equiv \left(\dfrac{a_1}{p}\right)_k \left(\dfrac{a_2}{p}\right)_k$;

(4) $\text{ind}_g a \equiv b \pmod{k}, 0 \leq b < k \Longrightarrow \left(\dfrac{a}{p}\right)_k \equiv g^{aq} \pmod{p}$;

(5) *a is the kth power residue of* $p \Longleftrightarrow \left(\dfrac{a}{p}\right)_k = 1$;

(6) $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_l^{\alpha_l} \Longrightarrow \left(\dfrac{n}{p}\right)_k = \left(\dfrac{p_1}{p}\right)_k^{\alpha_1} \left(\dfrac{p_2}{p}\right)_k^{\alpha_2} \cdots \left(\dfrac{p_l}{p}\right)_k^{\alpha_l}$.

Example 2.78 Let $p = 19$, $k = 3$ and $q = 6$. Then

$$\left(\frac{-1}{19}\right)_3 = \left(\frac{1}{19}\right)_3 = 1$$

$$\left(\frac{2}{19}\right)_3 = 7$$

$$\left(\frac{3}{19}\right)_3 = \left(\frac{-16}{19}\right)_3 \equiv \left(\frac{-1}{19}\right)_3 \left(\frac{16}{19}\right)_3 \equiv \left(\frac{-1}{19}\right)_3 \left(\frac{2}{19}\right)_3^4 = \left(\frac{2}{19}\right)_3 = 7$$

$$\left(\frac{5}{19}\right)_3 = \left(\frac{24}{19}\right)_3 \equiv \left(\frac{2}{19}\right)_3^3 \left(\frac{3}{19}\right)_3 = \left(\frac{3}{19}\right)_3 = 7$$

$$\left(\frac{7}{19}\right)_3 = \left(\frac{45}{19}\right)_3 \equiv \left(\frac{3}{19}\right)_3^2 \left(\frac{5}{19}\right)_3 = 7^3 \equiv 1$$

$$\left(\frac{11}{19}\right)_3 = \left(\frac{30}{19}\right)_3 \equiv \left(\frac{2}{19}\right)_3 \left(\frac{3}{19}\right)_3 \left(\frac{5}{19}\right)_3 = 7^3 \equiv 1$$

$$\left(\frac{13}{19}\right)_3 = \left(\frac{32}{19}\right)_3 \equiv \left(\frac{2}{19}\right)_3 = -8$$

$$\left(\frac{17}{19}\right)_3 = \left(\frac{-2}{19}\right)_3 \equiv \left(\frac{-1}{19}\right)_3 \left(\frac{2}{19}\right)_3 = 7$$

All the above congruences are modular 19.

Problems for Section 2.5

1. Find the primitive roots for primes 3, 5, 7, 11, 13, 17, 23.
2. Prove $a^2 \equiv 1 \pmod{p}$ if and only if $a \equiv -1 \pmod{p}$.
3. Show that the numbers $1^k, 2^k, 3^k, \ldots, (p-1)^k$ form a reduced residue system modulo p if and only if $\gcd(k, p-1) = 1$.
4. Prove that if g and g' are primitive roots modulo an odd prime p, then gg' is not a primitive root modulo p.
5. Let g be a primitive root modulo a prime p. Show that

$$(p-1)! \equiv g \cdot g^2 \cdot g^3 \cdots g^{p-1} \equiv g^{p(p-1)/2} \pmod{p}.$$

   Use this to prove the Wilson theorem:

$$(p-1)! \equiv -1 \pmod{p}.$$

6. Prove that if a and $n > 1$ be any integers such that $a^{n-1} \equiv 1 \pmod{n}$, but $a^d \not\equiv 1 \pmod{n}$ for every proper divisor d of $n-1$, then n is a prime.
7. For any positive integer n, prove that the arithmetic progression

$$n+1, 2n+1, 3n+1, \ldots$$

   contains infinitely many primes.
8. Show that if $n > 1$, then $n \nmid (2^n - 1)$.
9. Determine how many solutions each of the following congruence have.

$$x^{12} \equiv 16 \pmod{17}, \; x^{48} \equiv 9 \pmod{17},$$

$$x^{20} \equiv 13 \pmod{17}, \; x^{11} \equiv 9 \pmod{17}.$$

10. (Victor Shoup) Let $g(p)$ be the least positive primitive root modulo a prime p. Show that $g(p) = \mathcal{O}((\log p)^6)$.

2.6 Elliptic Curves

The study of elliptic curves is intimately connected with the the study of Diophantine equations. The theory of Diophantine equations is a branch of number theory which deals with the solution of polynomial equations in either integers or rational numbers. As a solvable polynomial equation always has a corresponding geometrical diagram (e.g., curves or even surfaces). thus to find the integer or rational solution to a polynomial equation is equivalent to find the integer or rational points on the corresponding geometrical diagram, this leads naturally to *Diophantine geometry*, a subject dealing with the integer or rational points on

curves or surfaces represented by polynomial equations. For example, in analytic geometry, the linear equation

$$f(x, y) = ax + by + c \tag{2.175}$$

represents a straight line. The points (x, y) in the plane whose coordinates x and y are integers are called *lattice points*. Solving the linear equation in integers is therefore equivalent to determining those lattice points that lie on the line; The integer points on this line give the solutions to the linear Diophantine equation $ax + by + c = 0$. The general form of the integral solutions for the equation shows that if (x_0, y_0) is a solution, then there are lattice points on the line:

$$x_0, x_0 \pm b, x_0 \pm 2b, \ldots. \tag{2.176}$$

If the polynomial equation is

$$f(x, y) = x^2 + y^2 - 1 \tag{2.177}$$

then its associate algebraic curve is the unit circle. The solution (x, y) for which x and y are rational correspond to the Pythagorean triples $x^2 + y^2 = 1$. In general, a polynomial $f(x, y)$ of degree 2

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{2.178}$$

gives either an ellipse, a parabola, or a hyperbola, depending on the values of the coefficients. If $f(x, y)$ is a cubic polynomial in (x, y), then the locus of points satisfying $f(x, y) = 0$ is a cubic curve. A general cubic equation in two variables is of the form

$$ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j = 0. \tag{2.179}$$

Again, we are only interested in the integer solutions of the Diophantine equations, or equivalently, the integer points on the curves of the equations.

The above discussions lead us very naturally to *Diophantine geometry*, a subject dealing with the integer or rational points on algebraic curves or even surfaces of Diophantine equations (a straight line is a special case of algebraic curves).

Definition 2.53 A *rational number*, as everybody knows, is a quotient of two integers. A point in the (x, y)-plane is called a *rational point* if both its coordinates are rational numbers. A line is a *rational line* if the equation of the line can be written with rational numbers; that is, the equation is of the form

$$ax + by + c = 0 \tag{2.180}$$

where a, b, c are rational numbers.

Definition 2.54 Let

$$ax^2 + bxy + cy^2 + dx + ey + f = 0. \tag{2.181}$$

be a *conic*. Then the conic is rational if we can write its equation with rational numbers.

We have already noted that the point of intersection of two rational lines is rational point. But what about the intersection of a rational line with a rational conic? Will it be true that the points of intersection are rational? In general, they are not. In fact, the two points of intersection are rational if and only if the roots of the quadratic equation are rational. However, if one of the points is rational, then so is the other.

There is a very general method to test, in a finite number of steps, whether or not a given rational conic has a rational point, due to Legendre. The method consists of determining whether a certain congruence can be satisfied.

Theorem 2.80 (Legendre) *For the Diophantine equation*

$$ax^2 + by^2 = cz^2, \tag{2.182}$$

there is an integer n, depending on a, b, c, such that the equation has a solution in integers, not all zero, if and only if the congruence

$$ax^2 + by^2 \equiv cz^2 \,(\mathrm{mod}\, n) \tag{2.183}$$

has a solution in integers relatively prime to n.

An elliptic curve is an algebraic curve given by a *cubic Diophantine equation*

$$y^2 = x^3 + ax + b. \tag{2.184}$$

More general cubics in x and y can be reduced to this form, known as Weierstrass normal form, by rational transformations.

Example 2.79 Two examples of elliptic curves are shown in Figure 2.3 (from left to right). The graph on the left is the graph of a *single* equation, namely $E_1 : y^2 = x^3 - 4x + 2$; even though it breaks apart into two pieces, we refer to it as a *single* curve. The graph on the right is given by the equation $E_2 : y^2 = x^3 - 3x + 3$. Note that an elliptic curve is not an *ellipse*; a more accurate name for an elliptic curve, in terms of *algebraic geometry*, is an *Abelian variety of dimension one*. It should also be noted that *quadratic* polynomial equations are fairly well understood by mathematicians today, but cubic equations still pose enough difficulties to be topics of current research.

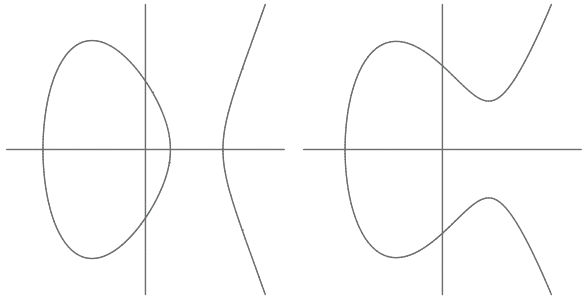


Figure 2.3 Two examples of elliptic curves

Definition 2.55 An elliptic curve $E : y^2 = x^3 + ax + b$ is called nonsingular if its discriminant

$$\Delta(E) = -16(4a^3 + 27b^2) \neq 0. \tag{2.185}$$

Remark 2.22 By elliptic curve, we always mean that the cubic curve is nonsingular. A cubic curve, such as $y^2 = x^3 - 3x + 2$ for which $\Delta = -16(4(-3)^3 + 27 \cdot 2^2) = 0$, is actually not an elliptic curve; such a cubic curve with $\Delta(E) = 0$ is called a *singular curve*. It can be shown that a cubic curve $E : y^2 = x^3 + ax + b$ is singular if and only if $\Delta(E) = 0$.

Definition 2.56 Let $\mathcal{K}$ be a field. Then the *characteristic* of the field $\mathcal{K}$ is 0 if

$$\underbrace{1 \oplus 1 \oplus \cdots \oplus 1}_{n \text{ summands}}$$

is never equal to 0 for any $n > 1$. Otherwise, the *characteristic* of the field $\mathcal{K}$ is the least positive integer n such that

$$\sum_{i=1}^{n} 1 = 0.$$

Example 2.80 The fields $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ all have characteristic 0, whereas the field $\mathbb{Z}/p\mathbb{Z}$ is of characteristic p, where p is prime.

Definition 2.57 Let $\mathcal{K}$ be a field (either the field $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, or the finite field $\mathbb{F}_q$ with $q = p^\alpha$ elements), and $x^3 + ax + b$ with $a, b \in \mathcal{K}$ be a cubic polynomial. Then

(1) If $\mathcal{K}$ is a field of characteristic $\neq 2, 3$, then an *elliptic curve* over $\mathcal{K}$ is the set of points (x, y) with $x, y \in \mathcal{K}$ that satisfy the following cubic Diophantine equation:

$$E : y^2 = x^3 + ax + b, \tag{2.186}$$

(where the cubic on the right-hand side has no multiple roots) together with a single element, denoted by $\mathcal{O}_E = (\infty, \infty)$, called the *point at infinity*.

(2) If $\mathcal{K}$ is a field of characteristic 2, then an *elliptic curve* over $\mathcal{K}$ is the set of points (x, y) with $x, y \in \mathcal{K}$ that satisfy one of the following cubic Diophantine equations:

$$\left.\begin{array}{l} E : y^2 + cy = x^3 + ax + b, \\ E : y^2 + xy = x^3 + ax^2 + b, \end{array}\right\} \tag{2.187}$$

(here we do not care whether or not the cubic on the right-hand side has multiple roots) together with a *point at infinity* $\mathcal{O}_E$.

(3) If $\mathcal{K}$ is a field of characteristic 3, then an *elliptic curve* over $\mathcal{K}$ is the set of points (x, y) with $x, y \in \mathcal{K}$ that satisfy the cubic Diophantine equation:

$$E : y^2 = x^3 + ax^2 + bx + c, \tag{2.188}$$

(where the cubic on the right-hand side has no multiple roots) together with a *point at infinity* $\mathcal{O}_E$.

In practice, we are actually more interested in the elliptic curves modulo a positive integer N.

Definition 2.58 Let N be a positive integer with $\gcd(n, 6) = 1$. An *elliptic curve* over $\mathbb{Z}/n\mathbb{Z}$ is given by the following cubic Diophantine equation:

$$E : y^2 = x^3 + ax + b, \tag{2.189}$$

where $a, b \in \mathbb{Z}$ and $\gcd(N,\ 4a^3 + 27b^2) = 1$. The set of points on E is the set of solutions in $(\mathbb{Z}/n\mathbb{Z})^2$ to equation 2.189, together with a *point at infinity* $\mathcal{O}_E$.

Remark 2.23 The subject of elliptic curves is one of the jewels of 19th-century mathematics, originated by Abel, Gauss, Jacobi, and Legendre. Contrary to popular opinion, an elliptic curve (i.e., a nonsingular cubic curve) is not an ellipse; as Niven, Zuckerman, and Montgomery remarked, it is natural to express the arc length of an ellipse as an integral involving the square root of a quadratic polynomial. By making a rational change of variables, this may be reduced to an integral involving the square root of a cubic polynomial. In general, an integral involving the square root of a quadratic or cubic polynomial is called an *elliptic integral*. So, the word *elliptic* actually came from the theory of elliptic integrals of the form

$$\int R(x, y)dx \tag{2.190}$$

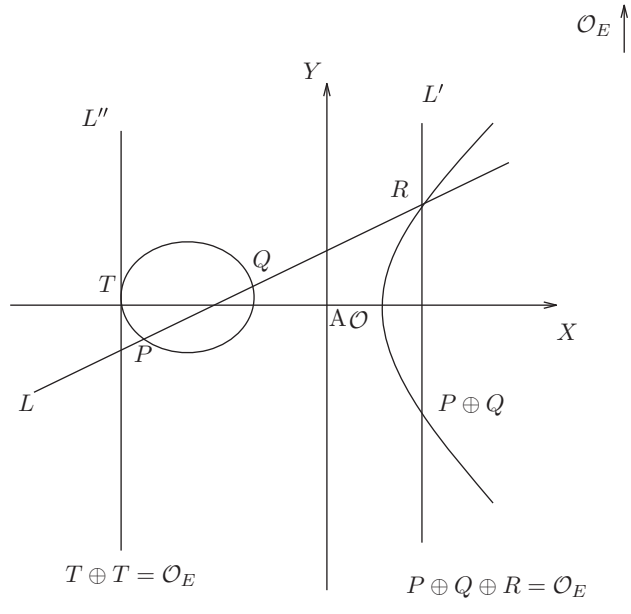


Figure 2.4 Geometric composition laws of an elliptic curve

where $R(x, y)$ is a rational function in x and y, and y^2 is a polynomial in x of degree 3 or 4 having no repeated roots. Such integrals were intensively studied in the 18th and 19th centuries. It is interesting to note that elliptic integrals serve as a motivation for the theory of elliptic functions, whilst elliptic functions parametrize elliptic curves. It is not our intention here to explain fully the theory of elliptic integrals and elliptic functions; interested readers are recommended to consult some more advanced texts.

The geometric interpretation of addition of points on an elliptic curve is quite straightforward. Suppose E is an elliptic curve as shown in Figure 2.4. A straight line L connecting points P and Q intersects the elliptic curve at a third point R, and the point $P \oplus Q$ is the reflection of R in the X-axis.

As can be seen from Figure 2.4, an elliptic curve can have many rational points; any straight line connecting two of them intersects a third. The point at infinity $\mathcal{O}_E$ is the third point of intersection of any two points (not necessarily distinct) of a vertical line with the elliptic curve E. This makes it possible to generate all rational points out of just a few.

The above observations lead naturally to the following geometric composition law of elliptic curves.

Proposition 2.3 (Geometric composition law (See 2.4)) Let $P, Q \in E$, L be the line connecting P and Q (tangent line to E if $P = Q$), and R be the third point of intersection of L with E. Let L' be the line connecting R and $\mathcal{O}_E$ (the point at infinity). Then $P \oplus Q$ is the point such that L' intersects E at R, $\mathcal{O}_E$, and $P \oplus Q$.

The points on an elliptic curve form an Abelian group with the addition of points as the binary operation on the group.

Theorem 2.81 (Group laws on elliptic curves) *The geometric composition laws of elliptic curves have the following group-theoretic properties:*

(1) *If a line L intersects E at the (not necessary distinct) points P, Q, R, then*

$$(P \oplus Q) \oplus R = \mathcal{O}_E.$$

(2) $P \oplus \mathcal{O}_E = P, \quad \forall P \in E.$
(3) $P \oplus Q = Q \oplus P, \quad \forall P, Q \in E.$
(4) *Let $P \in E$, then there is a point of E, denoted $\ominus P$, such that*

$$P \oplus (\ominus P) = \mathcal{O}_E.$$

(5) *Let $P, Q, R \in E$, then*

$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R).$$

In other words, the composition law makes E into an Abelian group with identity element $\mathcal{O}_E$. We further have
(6) *Let E be defined over a field $\mathcal{K}$, then*

$$E(\mathcal{K}) = \{(x, y) \in \mathcal{K}^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}_E\}.$$

is a subgroup of E.

Example 2.81 Let $E(\mathbb{Q})$ be the set of rational points on E. Then $E(\mathbb{Q})$ with the addition operation defined on it forms an Abelian group.

We shall now introduce the important concept of the order of a point on E.

Definition 2.59 Let P be an element of the set $E(\mathbb{Q})$. Then P is said to have *order k* if

$$kP = \underbrace{P \oplus P \oplus \cdots \oplus P}_{k \text{ summands}} = \mathcal{O}_E \tag{2.191}$$

with $k'P \neq \mathcal{O}_E$ for all $1 < k' < k$ (that is, k is the smallest integer such that $kP = \mathcal{O}_E$). If such a k exists, then P is said to have *finite order*, otherwise, it has *infinite order*.

Example 2.82 Let $P = (3, 2)$ be a point on the elliptic curve $E : y^2 = x^3 - 2x - 3$ over $\mathbb{Z}/7\mathbb{Z}$ (see Example 2.87). Since $10P = \mathcal{O}_E$ and $kP \neq \mathcal{O}_E$ for $k < 10$, P has order 10.

Example 2.83 Let $P = (-2, 3)$ be a point on the elliptic curve $E : y^2 = x^3 + 17$ over $\mathbb{Q}$ (see Example 2.31). Then P apparently has infinite order.

Now let us move on to the problem as to *how many points (rational or integral) are there on an elliptic curve*? First let us look at an example:

Example 2.84 Let E be the elliptic curve $y^2 = x^3 + 3x$ over $\mathbb{F}_5$, then

$$\mathcal{O}_E, (0, 0), (1, 2), (1, 3), (2, 2), (2, 3), (3, 1), (3, 4), (4, 1), (4, 4)$$

are the 10 points on E. However, the elliptic curve $y^2 = 3x^3 + 2x$ over $\mathbb{F}_5$ has only two points:

$$\mathcal{O}_E,\ (0, 0).$$

How many points are there on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$? The following theorem answers this question:

Theorem 2.82 *Let $|E(\mathbb{F}_p)|$ with p prime be the number of points on $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$. Then*

$$|E(\mathbb{F}_p)| = 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{x^3 + ax + b}{p} \right) = 1 + p + \epsilon \tag{2.192}$$

points on E over $\mathbb{F}_p$, including the point at infinity $\mathcal{O}_E$, where $\left(\frac{x^3+ax+b}{p}\right)$ is the Legendre symbol.

The quantity ϵ in (2.192) is constrained in the following theorem, due to Hasse (1898–1979) in 1933:

Theorem 2.83 (Hasse)

$$|\epsilon| \leq 2\sqrt{p}. \tag{2.193}$$

That is,

$$1 + p - 2\sqrt{p} \leq |E(\mathbb{F}_p)| \leq 1 + p + 2\sqrt{p}. \tag{2.194}$$

Example 2.84 Let $p = 5$, then $|\epsilon| \leq 4$. Hence, $1 + 5 - 4 \leq |E(\mathbb{F}_5)| \leq 1 + 5 + 4$, that is, we have between 2 and 10 points on an elliptic curve over $\mathbb{F}_5$. In fact, all the possibilities occur in the elliptic curves given in Table 2.7.

Table 2.7 Number of points on elliptic curves over $\mathbb{F}_5$

Elliptic curve	Number of points	Elliptic curve	Number of points
$y^2 = x^3 + 2x$	2	$y^2 = x^3 + 4x + 2$	3
$y^2 = x^3 + x$	4	$y^2 = x^3 + 3x + 2$	5
$y^2 = x^3 + 1$	6	$y^2 = x^3 + 2x + 1$	7
$y^2 = x^3 + 4x$	8	$y^2 = x^3 + x + 1$	9
$y^2 = x^3 + 3x$	10		

A more general question is: How many rational points are there on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{Q}$? Louis Joel Mordell (1888–1972) solved this problem in 1922:

Theorem 2.84 (Mordell's finite basis theorem) *Suppose that the cubic polynomial $f(x, y)$ has rational coefficients, and that the equation $f(x, y) = 0$ defines an elliptic curve E. Then the group $E(\mathbb{Q})$ of rational points on E is a finitely generated Abelian group.*

In elementary language, this says that on any elliptic curve that contains a rational point, there exists a finite collection of rational points such that all other rational points can be generated by using the chord-and-tangent method. From a group-theoretic point of view, Mordell's theorem tells us that we can produce all of the rational points on E by starting from some finite set and using the group laws. It should be noted that for some cubic curves, we have tools to find this generating set, but unfortunately, there is no general method (i.e., algorithm) guaranteed to work for all cubic curves.

The fact that the Abelian group is finitely generated means that it consists of a finite "torsion subgroup" $\mathrm{E}_{\mathrm{tors}}$, consisting of the rational points of finite order, plus the subgroup generated by a finite number of points of infinite order:

$$E(\mathbb{Q}) \simeq \mathrm{E}_{\mathrm{tors}} \oplus \mathbb{Z}^r .$$

The number r of generators needed for the infinite part is called the *rank* of $E(\mathbb{Q})$; it is zero if and only if the entire group of rational points is finite. The study of the rank r and other features of the group of points on an elliptic curve over $\mathbb{Q}$ are related to many interesting problems in number theory and arithmetic algebraic geometry. One of such problems is the Birch and Swinerton-Dyer conjecture (BSD conjecture), which shall be dicussed later.

The most important and fundamental operation on an elliptic curve is the addition of points on the curve. To perform the addition of points on elliptic curves systematically, we need an algebraic formula. The following gives us a convenient computation formula.

Theorem 2.85 (Algebraic computation law) *Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ be points on the elliptic curve:*

$$E : y^2 = x^3 + ax + b, \tag{2.195}$$

then $P_3 = (x_3, y_3) = P_1 \oplus P_2$ *on* E *may be computed by*

$$P_1 \oplus P_2 = \begin{cases} \mathcal{O}_E, & \text{if } x_1 = x_2 \,\&\, y_1 = -y_2 \\ (x_3, y_3), & \text{otherwise.} \end{cases} \tag{2.196}$$

where

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2, \;\; \lambda(x_1 - x_3) - y_1) \tag{2.197}$$

and

$$\lambda = \begin{cases} \dfrac{(3x_1^2 + a)}{2y_1}, & \text{if } P_1 = P_2, \\ \dfrac{(y_2 - y_1)}{(x_2 - x_1)}, & \text{otherwise.} \end{cases} \tag{2.198}$$

Example 2.86 Let E be the elliptic curve $y^2 = x^3 + 17$ over $\mathbb{Q}$, and let $P_1 = (x_1, y_1) = (-2, 3)$ and $P_2 = (x_2, y_2) = (1/4, \;\; 33/8)$ be two points on E. To find the third point P_3 on E, we perform the following computation:

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{1}{2}, \\ x_3 &= \lambda^2 - x_1 - x_2 = 2, \\ y_3 &= \lambda(x_1 - x_3) - y_1 = -5. \end{aligned}$$

So $P_3 = P_1 \oplus P_2 = (x_3, \;\; y_3) = (2, -5)$.

Example 2.87 Let $P = (3, 2)$ be a point on the elliptic curve $E : y^2 = x^3 - 2x - 3$ over $\mathbb{Z}/7\mathbb{Z}$. Compute

$$10P = \underbrace{P \oplus P \oplus \cdots \oplus P}_{10 \text{ summands}} \pmod 7.$$

According to (2.197), we have:

$$\begin{aligned} 2P &= P \oplus P = (3, 2) \oplus (3, 2) = (2, 6), \\ 3P &= P \oplus 2P = (3, 2) \oplus (2, 6) = (4, 2), \\ 4P &= P \oplus 3P = (3, 2) \oplus (4, 2) = (0, 5) \\ 5P &= P \oplus 4P = (3, 2) \oplus (0, 5) = (5, 0) \\ 6P &= P \oplus 5P = (3, 2) \oplus (5, 0) = (0, 2) \\ 7P &= P \oplus 6P = (3, 2) \oplus (0, 2) = (4, 5) \\ 8P &= P \oplus 7P = (3, 2) \oplus (4, 5) = (2, 1) \\ 9P &= P \oplus 8P = (3, 2) \oplus (2, 1) = (3, 5) \\ 10P &= P \oplus 9P = (3, 2) \oplus (3, 5) = \mathcal{O}_E \end{aligned}$$

Example 2.88 Let $E : y^2 = x^3 + 17$ be the elliptic curve over $\mathbb{Q}$ and $P = (-2, 3)$ a point on E. Then

$$
\begin{aligned}
P &= (-2, 3)\\
2P &= (8, -23)\\
3P &= \left(\frac{19}{25}, \frac{522}{125}\right)\\
4P &= \left(\frac{752}{529}, \frac{-54239}{12167}\right)\\
5P &= \left(\frac{174598}{32761}, \frac{76943337}{5929741}\right)\\
6P &= \left(\frac{-4471631}{3027600}, \frac{-19554357097}{5268024000}\right)\\
7P &= \left(\frac{12870778678}{76545001}, \frac{1460185427995887}{669692213749}\right)\\
8P &= \left(\frac{-3705032916448}{1556248765009}, \frac{3635193007425360001}{1941415665602432473}\right)\\
9P &= \left(\frac{1508016107720305}{1146705139411225}, \frac{-185877155243117444053750 2}{3883091627056219156787 5}\right)\\
10P &= \left(\frac{2621479238320017368}{21550466484219504001}, \frac{4125080845025235054098132572 57}{1000426099138845575254147439 99}\right)\\
11P &= \left(\frac{9838648912910878733824 78}{4557708224535761198560 81}, \frac{-1600581839303565170139037888610254293}{3076945320470535093503259055179432 71}\right)\\
12P &= \left(\frac{1727701779459733569579962592 1}{4630688543838991376029953600}, \frac{2616325792251321558429704062367454696426719}{3151144781214267267043920536423376332160 00}\right)
\end{aligned}
$$

Suppose now we are interested in measuring the *size* (or the *height of point on elliptic curve*) of points on an elliptic curve E. One way to do this is to look at the numerator and denominator of the x-coordinates. If we write the coordinates of kP as

$$kP = \left(\frac{A_k}{B_k}, \frac{C_k}{D_k}\right), \tag{2.199}$$

we may define the height of these points as follows

$$H(kP) = \max(|A_k|, |B_k|). \tag{2.200}$$

It is interesting to note that for large k, the height of kP looks like:

$$D(H(kP)) \approx 0.1974k^2, \tag{2.201}$$

$$
\begin{aligned}
H(kP) &\approx 10^{0.1974k^2}\\
&\approx (1.574)^{k^2}
\end{aligned} \tag{2.202}
$$

where $D(H(kP))$ denotes the number of digits in $H(kP)$.

Remark 2.24 To provide greater flexibility, we may also consider the following form of elliptic curves:

$$E : y^2 = x^3 + ax^2 + bx + c. \tag{2.203}$$

In order for E to be an elliptic curve, it is necessary and sufficient that

$$\Delta(E) = a^2b^2 - 4a^3c - 4b^3 + 18abc - 27c^2 \neq 0. \tag{2.204}$$

Thus,

$$P_3(x_3, y_3) = P_1(x_1, y_1) \oplus P_2(x_2, y_2),$$

on E may be computed by

$$(x_3, y_3) = (\lambda^2 - a - x_1 - x_2, \ \ \lambda(x_1 - x_3) - y_1) \tag{2.205}$$

where

$$\lambda = \begin{cases} (3x_1^2 + 2a + b)/2y_1, & \text{if } P_1 = P_2 \\ (y_2 - y_1)/(x_2 - x_1), & \text{otherwise.} \end{cases} \tag{2.206}$$

The problem of determining the group of rational points on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{Q}$, denoted by $E(\mathbb{Q})$, is one of the oldest and most intractable in mathematics, and it remains unsolved to this day, although vast numerical evidences exist. In 1922, Louis Joel Mordell (1888–1972) showed that $E(\mathbb{Q})$ is a finitely generated (Abelian) group. That is, $E(\mathbb{Q}) \approx E(\mathbb{Q})_{\text{tors}} \oplus \mathbb{Z}^r$, where $r \geq 0$, $E(\mathbb{Q})_{\text{tors}}$ is a finite Abelian group (called torsion group). The integer r is called the *rank* of the elliptic curve E over $\mathbb{Q}$, denoted by $\text{rank}(E(\mathbb{Q}))$. Is there an algorithm to compute $E(\mathbb{Q})$ given an arbitrary elliptic curve E? The answer is not known, although $E(\mathbb{Q})_{\text{tors}}$ can be found easily, due to a theorem of Mazur in 1978: $\#(E(\mathbb{Q})_{\text{tors}}) \leq 16$. The famous Birch and Swinnerton-Dyer conjecture [6], or BSD conjecture for short, asserts that the size of the group of rational points on E over $\mathbb{Q}$, denoted by $\#(E(\mathbb{Q}))$, is related to the behavior of an associated zeta function $\zeta(s)$, called the Hasse–Weil L-function $L(E, s)$, near the point $s = 1$. That is, if we define the *incomplete* L-function $L(E, s)$ (we called it *incomplete* because we omit the set of "bad" primes $p \mid 2\triangle$) as follows:

$$L(E, s) := \prod_{p \nmid 2\triangle} (1 - a_p p^{-s} + p^{1-2s})^{-1},$$

where $\triangle = -16(4a^3 + 27b^2)$ is the discriminant of E, $N_p := \#\{\text{rationalsolutionsof} y^2 \equiv x^3 + ax + b \,(\text{mod } p)\}$ with p prime and $a_p = p - N_p$. This L-function converges for $\text{Re}(s) > \frac{3}{2}$, and can be analytically continued to an entire function. It was conjectured by Birch and Swinnerton-Dyer in the 1960s that the *rank* of the Abelian group of points over a number field of an elliptic curve E is related to the order of the zero of the associated L-function $L(E, s)$ at $s = 1$:

BSD Conjecture (Version 1): $\mathrm{ord}_{s=1} L(E, s) = \mathrm{rank}(E(\mathbb{Q}))$.

This amazing conjecture asserts particularly that

$$L(E, 1) = 0 \Longleftrightarrow E(\mathbb{Q}) \text{ is infinite.}$$

Conversely, if $L(E, 1) \neq 0$, then the set $E(\mathbb{Q})$ is finite. An alternative version of BSD, in term of the Taylor expansion of $L(E, s)$ at $s = 1$, is as follows:

BSD Conjecture (Version 2): $L(E, s) \sim c(s-1)^r$, where $c \neq 0$ and $r = \mathrm{rank}(E(\mathbb{Q}))$.

There is also a refined version of BSD for the *complete* L-function $L^*(E, s)$:

$$L^*(E, s) := \prod_{p|2\triangle} (1 - a_p p^{-s})^{-1} \cdot \prod_{p\nmid 2\triangle} (1 - a_p p^{-s} + p^{1-2s})^{-1}.$$

In this case, we have:

BSD Conjecture (Version 3): $L^*(E, s) \sim c^*(s-1)^r$, with

$$c^* = |\text{Ш}_E| R_\infty w_\infty \prod_{p|\triangle} w_p / |E(\mathbb{Q})_{\text{tors}}|^2,$$

where $|\text{Ш}_E|$ is the order of the Tate–Shafarevich group of elliptic curve E, the term R_∞ is an $r \times r$ determinant whose matrix entries are given by a height pairing applied to a system of generators of $E(\mathbb{Q})/E(\mathbb{Q})_{\text{tors}}$, the w_p are elementary local factors and w_∞ is a simple multiple of the real period of E.

The eminent American mathematician, John Tate commented on BSD in 1974 that "$\cdots$ this remarkable conjecture relates the behaviour of a function L at a point where it is at present not known to be defined to the order of a group Ш which is not known to be finite." So it was hoped that a proof of the conjecture would yield a proof of the finiteness of Ш_E. Using the idea of Kurt Heegner (1893–1965), Birch and his former PhD student Stephens established for the first time the existence of rational points of infinite order on certain elliptic curves over $\mathbb{Q}$, without actually writing down the coordinates of these points, and naively verifying that they had satisfied the equation of the curves. These points are now called *Heegner points* on elliptic curves (a Heegner point is a point on modular elliptic curves that is the image of a quadratic imaginary point of the upper half-plane). Based on Birch and Stephens' work, particular on their massive computation of the Heegner points on modular elliptic curves, Gross at Harvard and Zagier at Maryland/Bonn obtained a deep result in 1986, now widely known as the Gross–Zagier theorem, which describes the height of Heegner points in terms of a derivative of the L-function of the elliptic curve at the point $s = 1$. That is, if $L(E, 1) = 0$, then there is a closed formula to relate $L'(E, 1)$ and the height of the Heegner points on E. More generally, together with Kohnen, Gross and Zagier showed in 1987 that Heegner points could be used to construct rational points on the curve for each positive integer n, and the heights of these points were the coefficients of a modular form of weight $3/2$. Later, in

1989, the Russian mathematician Kolyvagin further used Heegner points to construct Euler systems, and used this to prove much of the Birch–Swinnerton-Dyer conjecture for rank 1 elliptic curves. More specifically, he showed that if the Heegner points are of infinite order, then rank$(E(\mathbb{Q})) = 1$. Other notable results in BSD also include S. W. Zhang's generalization of Gross–Zagier theorem for elliptic curves to Abelian varieties, and M. L. Brown's proof of BSD for most rank 1 elliptic curves over global fields of positive characteristic. Of course, all these resolutions are far away from the complete settlement of BSD. Just as Riemann's Hypothesis, the BSD conjecture was also chosen to be one of the seven Millennium Prize Problems [7].

Problems for Section 2.6

1. Describe an algorithm to find a point on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{Q}$. Use your algorithm to find a point on the $E : y^2 = x^3 - 13x + 21$ over $\mathbb{Q}$.
2. Find all the rational points on the elliptic curve $y^2 = x^3 - x$.
3. Find all the rational points on the elliptic curve $y^2 = x^3 + 4x$.
4. Describe an algorithm to find the order of a point on an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{Q}$. Let $P = (2, 4)$ be a point on $E : y^2 = x^3 - 13x + 21$ over $\mathbb{Q}$. Use your algorithm to find the order of the point on E.
5. Find all the torsion points of the elliptic curve $E : y^2 = x^3 - 13x + 21$ over $\mathbb{Q}$.
6. Find the point of infinite order on the elliptic curve $E : y^2 = x^3 - 2x$ over $\mathbb{Q}$.
7. Determine the number of points of the elliptic curve $E : y^2 = x^3 - 1$ for all odd primes up to 100.
8. Let $P = (0, 0)$ be a point on the elliptic curve $E : y^2 = x^3 + x^2 + 2x$. Compute $100P$ and $200P$.
9. Derive an addition formula for rational points on the elliptic curve

$$E : y^2 = x^3 + ax^2 + bx + c.$$

10. Show that $P = (9/4, 29/8)$ is a point on the elliptic curve $E : y^2 = x^3 - x + 4$.
11. Let $P = (1, 1)$ be a point on the elliptic curve $E : y^2 = x^3 - 6x + 6$ over $\mathbb{Z}_{4247}$. Compute $100P$ on $E(\mathbb{Z}_{4727})$.
12. Let n be a positive integers greater than 1, and P a point on an elliptic curve $E(\mathbb{Z}_{4727})$. Prove that there are some integers s and t such that $sP = tP$.
13. Prove or disprove the BSD conjecture.

2.7 Bibliographic Notes and Further Reading

This chapter was mainly concerned with the elementary theory of numbers, including Euclid's algorithm, continued fractions, the Chinese Remainder theorem, Diophantine equations, arithmetic functions, quadratic and power residues, primitive roots, and the arithmetic of elliptic curves. It also includes some algebraic topics such as groups, rings, fields, polynomials, and algebraic numbers. The theory of numbers is one of the oldest and most beautiful parts of pure mathematics, and there are many good books and papers in this field.

It is suggested that readers consult some of the following references for more information: [8–28].

Elliptic curves are used throughout the book for primality testing, integer factorization, and cryptography. We have only given an brief introduction to the theory and arithmetic of elliptic curves. Readers who are interested in elliptic curves and their applications should consult the following references for more information: [29–33]. Also, the new version of Hardy and E. M. Wright's famous book [14] also contains a new chapter on elliptic curves at the end of the book.

Abstract algebra is intimately connected to number theory and, in fact, many of the concepts and results of number theory can be described in algebraic language. Readers who wish to study number theory from the algebraic perspective are specifically advised to consult the following references: [34–48].

References

1. G. E. Andrews, *Number Theory*, W. B. Sayders Company, 1971. Also Dover Publications, 1994.
2. H. Davenport, *The Higher Arithmetic*, 7th Edition, Cambridge University Press, 1999.
3. Y. Wang, "On the Least Positive Primitive Root", *The Chinese Journal of Mathematics*, **9**, 4, 1959, pp. 432–441.
4. V. Shoup, "Searching for Primitive Roots in Finite Fields", *Mathematics of Computation*, **58**, 197, 1992, pp. 369–380.
5. H. E. Rose, *A Course in Number Theory*, 2nd Edition, Oxford University Press, 1994.
6. A. Wiles, "The Birch and Swinnerton-Dyer Conjecture", In [7]: *The Millennium Prize Problems*, American Mathematical Society, 2006, pp. 31–44.
7. J. Carlson, A. Jaffe, and A. Wiles, *The Millennium Prize Problems*, Clay Mathematics Institute and American Mathematical Society, 2006.
8. A. Baker, *A Concise Introduction to the Theory of Numbers*, Cambridge University Press, 1984.
9. E. D. Bolker, *Elementray Number Theory: An Algebraic Approach*, Dover, 2007.
10. D. M. Burton, *Elementary Number Theory*, 7th Edition, McGraw-Hill, 2011.
11. H. Davenport, *The Higher Arithmetic*, 7th Edition, Cambridge University Press, 1999.
12. U. Dudley, *A Guide to Elementary Number Theory*, Mathematical Association of America, 2010.
13. H. M. Edwards, *Higher Arithmetic: Al Algorithmic Introduction to Number Theory*, American Mathematical Society, 2008.
14. G. H. Hardy and E. M. Wright, *An Introduction to Theory of Numbers*, 6th Edition, Oxford University Press, 2008.
15. L.K. Hua, *Introduction to Number Theory*, Springer, 1980.
16. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd Edition, Graduate Texts in Mathematics **84**, Springer, 1990.
17. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer, 1994.
18. W. J. LeVeque, *Fundamentals of Number Theory*, Dover, 1977.
19. S. J. Miller and R. Takloo-Bighash, *An Invitation to Modern Number Theory*, Princeton University Press, 2006.
20. R. A. Mollin, *Fundamental Number Theory with Applications*, 2nd Edition, CRC Press, 2008.
21. R. A. Mollin, *Advanced Number Theory with Applications*, CRC Press, 2010.
22. R. A. Mollin, *Algebraic Number Theory*, 2nd Edition, CRC Press, 2011.
23. I. Niven, H. S. Zuckerman, and H. L. Montgomery, *An Introduction to the Theory of Numbers*, 5th Edition, John Wiley & Sons, 1991.
24. J. E. Pommersheim, T. K. Marks, and E. L. Flapan, *Number Theory*, Wiley, 2010.
25. J. E. Shockley, *Introduction to Number Theory*, Holt, Rinehart and Winston, 1967.
26. J. H. Silverman, *A Friendly Introduction to Number Theory*, 4th Edition, Prentice-Hall, 2012.
27. J. Stillwell, *Elements of Number Theory*, Springer, 2000.
28. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer, 2002.

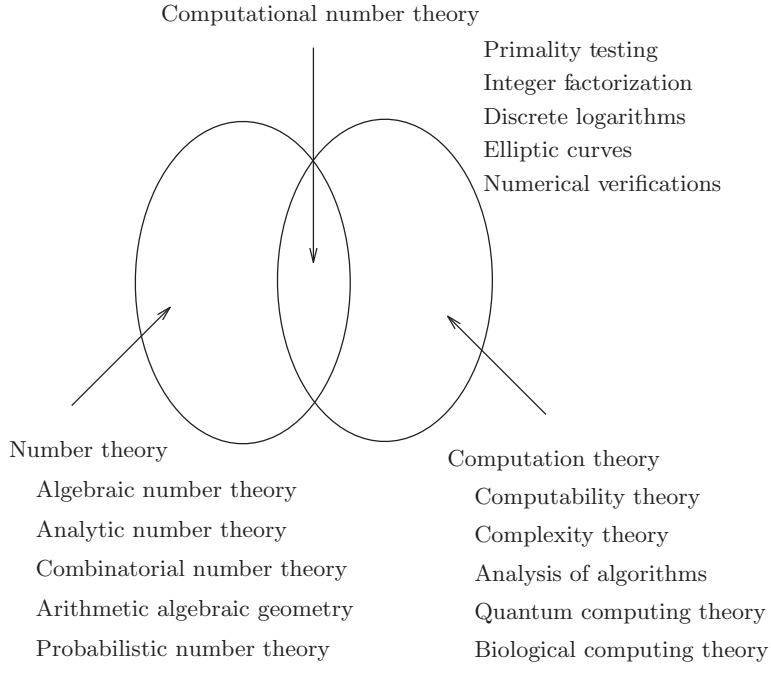
29. A. Ash and R. Gross, *Elliptic Tales: Curves, Counting, and Number Theory*, Princeton University Press, 2012.
30. D. Husemöller, *Elliptic Curves*, Graduate Texts in Mathematics **111**, Springer, 1987.
31. J. H. Silverman, *The Arithmetic of Elliptic Curves*, 2nd Edition, Graduate Texts in Mathematics **106**, Springer, 2009.
32. J. H. Silverman and J. Tate, *Rational Points on Elliptic Curves*, Undergraduate Texts in Mathematics, Springer, 1992.
33. L. C. Washinton, *Elliptic Curve: Number Theory and Cryptography*, 2nd Edition, CRC Press, 2008.
34. M. Artin, *Algebra*, 2nd Edition, Prentice-Hall, 2011.
35. P. E. Bland, *The Basics of Abstract Algebra*, W. H. Freeman and Company, 2002.
36. L. N. Childs, *A Concrete Introduction to Higher Algebra*, 3rd Edition, Springer, 2009.
37. J. B. Fraleigh, *First Course in Abstract Algebra*, 7th Edition, Addison-Wesley, 2003.
38. J. A. Gallian, *Contemporary Abstract Algebra*, 5th Edition, Houghton Mifflin Company, 2002.
39. D. W. Hardy, F. Richman, and C. L. Walker, *Applied Algebra*, 2nd Edition, Addison-Wesley, 2009.
40. I. N. Herstein, *Abstract Algebra*, 3rd Edition, Wiley, 1999.
41. T. W. Hungerford, *Abstract Algebra: An Introduction*, 2nd Edition, Brooks/Cole, 1997.
42. S. Lang, *Algebra*, 3rd Edition, Springer, 2002.
43. R. Lidl and G. Pilz, *Applied Abstract Algebra*, Springer, 1984.
44. S. MacLane and G. Birkhoff, *Algebra*, 3rd Edition, AMS Chelsea, 1992.
45. G. L. Mullen and C. Mummert, *Finite Fields and Applications*, Mathematical Association of America, 2007.
46. J. J. Rotman, *A First Course in Abstract Algebra*, 3rd Edition, Wiley, 2006.
47. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
48. J. Stillwell, *Elements of Algebra*, Springer, 1994.

Part II

Computational Number Theory

Computational number theory is a new branch of mathematics. Informally, it can be regarded as a combined and disciplinary subject of number theory and computer science, especially the theory of computation:

$$\text{Computational Number Theory} := \text{Number Theory} \oplus \text{Computation Theory}$$



The purposes of computational number theory are two-fold: (1) using computing techniques to solve number theoretic problems, and (2) using number theoretic techniques to solve computer science problems. In this part of the book, we shall concentrate on using computing techniques to solve number theoretic problems that have connections and applications in

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

modern cryptography. More specifically, we shall study computational methods (algorithms) for solving the:

1. Primality Testing Problem (PTP)
2. Integer Factorization Problem (IFP)
3. Discrete Logarithm Problem (DLP)
4. Elliptic Curve Discrete Logarithm Problem (ECDLP)

with a special emphasis on the last three infeasible (intractable) number theoretic problems, as they play a central role in the security of the cryptographic schemes and protocols in the next part of the book.

3

Primality Testing

Primality testing, possibly first studied by Euclid 2500 years ago but first identified as an important problem by Gauss in 1801, is one of the two important problems related to the computation of prime numbers. In this chapter we shall study various modern algorithms for primality testing, including

- Some simple and basic tests of primality, usually run in exponential-time $\mathcal{EXP}$.
- The Miller–Rabin test, runs in random polynomial-time $\mathcal{RP}$.
- The elliptic curve test, runs in zero-error probabilistic polynomial-time $\mathcal{ZPP}$.
- The AKS test, runs in deterministic polynomial-time $\mathcal{P}$.

3.1 Basic Tests

The Primality Test Problem (PTP) may be described as follows:

$$\text{PTP} := \begin{cases} \text{Input :} & n \in \mathbb{Z}_{>1} \\ \text{Output :} & \begin{cases} \text{Yes :} & n \in \text{Primes} \\ \text{No :} & \text{Otherwise} \end{cases} \end{cases} \tag{3.1}$$

The following theorem is well-known and fundamental to primality testing.

Theorem 3.1 *Let $n > 1$. If n has no prime factor less than or equal to $\lfloor\sqrt{n}\rfloor$, then n is prime.*

Thus the simplest possible primality test of n is by trial divisions of all possible prime factors of n up to $\lfloor\sqrt{n}\rfloor$ as follows (the Sieve of Eratosthenes for finding all prime numbers up to $\sqrt{n}$ is used in this test).

Primality test by trial divisions:

$$\begin{gathered} \text{Test}(p_i) \stackrel{\text{def}}{=} p_1, p_2, \ldots, p_k \le \lfloor\sqrt{n}\rfloor, p_i \nmid n \\ \Uparrow \\ \text{Eratosthenes Sieve} \end{gathered} \tag{3.2}$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

Thus, if n passes Test(p_i), then n is prime:

$$n \text{ passes Test}(p_i) \Longrightarrow n \in \text{Primes}. \tag{3.3}$$

Example 3.1 To test whether or not 3271 is prime, we only need to test the primes up to $\lfloor\sqrt{3271}\rfloor = 57$. That is, we will only need to do at most 16 trial divisions as follows:

$$\frac{3271}{2}, \frac{3271}{3}, \frac{3271}{5}, \frac{3271}{7}, \ldots, \frac{3271}{47}, \frac{3271}{53}.$$

As none of these division gives a zero remainder, so 3271 is a prime number. However, for $n = 3273$, we would normally expect to do the following trial divisions:

$$\frac{3273}{2}, \frac{3273}{3}, \frac{3273}{5}, \frac{3273}{7}, \ldots, \frac{3273}{47}, \frac{3273}{53}.$$

but fortunately we do not need to do all these trial divisions as 3273 is a composite, in fact, when we do the trial division 3273/3, it gives a zero remainder, so we conclude immediately that 3273 is a composite number.

This test, although easy to implement, is not practically useful for large numbers since it needs $\mathcal{O}\left(2^{(\log n)/2}\right)$ bit operations. In other words, it runs in exponential-time, $\mathcal{EXP}$. In the next sections, we shall introduce some modern and fast primality testing methods in current use. From a computational complexity point of view, the PTP has been completely solved since we have various algorithms for PTP, with the fastest runs in deterministic polynomial-time (see Figure 3.1).

In 1876 (although it was published in 1891), Lucas discovered a type of converse of the Fermat little theorem, based on the use of primitive roots.

Theorem 3.2 (Lucas' converse of Fermat's little theorem, 1876) *Let $n > 1$. Assume that there exists a primitive root of n, i.e., an integer a such that*

(1) $a^{n-1} \equiv 1 \pmod{n}$,
(2) $a^{(n-1)/p} \not\equiv 1 \pmod{n}$, *for each prime divisor p of $n-1$.*

Then n is prime.

Proof: Since $a^{n-1} \equiv 1 \pmod{n}$, Part (1) of Theorem 2.71 (see Chapter 1) tells us that $\text{ord}_n(a) \mid (n-1)$. We will show that $\text{ord}_n(a) = n-1$. Suppose $\text{ord}_n(a) \neq n-1$. Since $\text{ord}_n(a) \mid (n-1)$, there is an integer k satisfying $n - 1 = k \cdot \text{ord}_n(a)$. Since $\text{ord}_n(a) \neq n-1$, we know that $k > 1$. Let p be a prime factor of k. Then

$$x^{n-1}/q = x^{k/q \cdot \text{ord}_n(a)} = \left(x^{\text{ord}_n(a)}\right)^{k/q} \equiv 1 \pmod{n}.$$

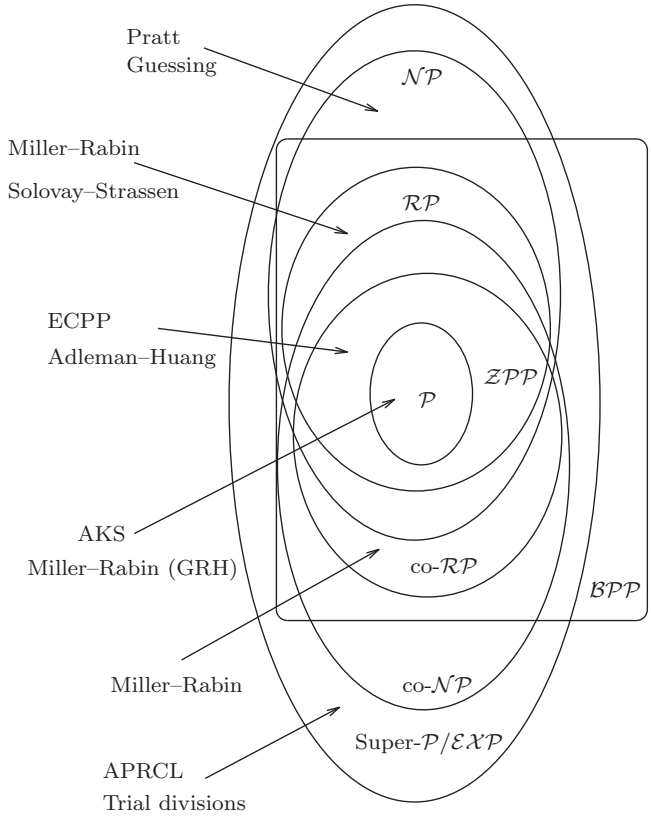


Figure 3.1 Algorithms/Methods for PTP

However, this contradicts the hypothesis of the theorem, so we must have $\text{ord}_n(a) = n - 1$. Now, since $\text{ord}_n(a) \leq \phi(n)$ and $\phi(n) \leq n - 1$, it follows that $\phi(n) = n - 1$. So finally by Part (2) of Theorem 2.36, n must be prime. ∎

Lucas' theorem can be converted to rigorous primality test as follows:

Primality test based on primitive roots:

$$\begin{aligned} \text{Test}(a) \stackrel{\text{def}}{=} & \; a^{n-1} \equiv 1 \pmod{n}, \\ & a^{(n-1)/p} \not\equiv 1 \pmod{n}, \forall p \mid (n-1). \end{aligned} \tag{3.4}$$

If n passes the test, then n is prime:

$$n \text{ passes Test}(a) \Longrightarrow n \in \text{Primes}. \tag{3.5}$$

Primality test based on primitive roots is also called $n-1$ primality test, as it is based on the prime factorization of $n-1$.

Example 3.2 Let $n = 2011$, then $2011 - 1 = 2 \cdot 3 \cdot 5 \cdot 67$. Note first 3 is a primitive root (in fact, the smallest) primitive root of 2011, since order$(3, 2011) = \phi(2011) = 2010$. So, we have

$$\begin{aligned} 3^{2011-1} &\equiv 1 \pmod{2011}, \\ 3^{(2011-1)/2} &\equiv -1 \not\equiv 1 \pmod{2011}, \\ 3^{(2011-1)/3} &\equiv 205 \not\equiv 1 \pmod{2011}, \\ 3^{(2011-1)/5} &\equiv 1328 \not\equiv 1 \pmod{2011}, \\ 3^{((2011-1)/67} &\equiv 1116 \not\equiv 1 \pmod{2011}. \end{aligned}$$

Thus, by Theorem 3.2, 2011 must be prime.

Remark 3.1 In practice, primitive roots tend to be small integers and can be quickly found (although there are some primes with arbitrary large smallest primitive roots), and the computation for $a^{n-1} \equiv 1 \pmod{n}$ and $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ can also be performed very efficiently. However, to determine if n is prime, the above test requires the prime factorization of $n-1$, a problem of almost the same size as that of factoring n, and a problem that is much harder than the primality testing of n.

Note that Theorem 3.2 is actually equivalent to the following theorem:

Theorem 3.3 *If there is an integer a for which the order of a modulo n is equal to $\phi(n)$ and $\phi(n) = n - 1$, then n is prime. That is, if*

$$\text{ord}_n(a) = \phi(n) = n - 1, \tag{3.6}$$

or

$$\mathbb{Z}_n^+ = \mathbb{Z}_n^*, \tag{3.7}$$

then n is prime.

Thus, we have the following primality test.

Primality test based on $\text{ord}_n(a)$

$$\text{Test}(a) \stackrel{\text{def}}{=} \text{ord}_n(a) = \phi(n) = n - 1 \tag{3.8}$$

Thus, if n passes the test, then n is prime:

$$n \text{ passes Test}(a) \Longrightarrow n \in \text{Primes}. \tag{3.9}$$

Example 3.3 Let $n = 3779$. We find, for example, that the integer $a = 19$ with $\gcd(19, 3779) = 1$ satisfies

(1) $\text{ord}_{3779}(19) = 3778$,
(2) $\phi(3779) = 3778$.

That is, $\text{ord}_{3779}(19) = \phi(3779) = 3778$. Thus by Theorem 3.3, 3779 is prime.

Remark 3.2 It is not a simple matter to find the order of an element a modulo n, $\text{ord}_n(a)$, if n is large. In fact, if $\text{ord}_n(a)$ can be calculated efficiently, the primality and prime factorization of n can be easily determined. At present, the best known method for computing $\text{ord}_n(a)$ requires one to factor n.

Remark 3.3 If we know the value of $\phi(n)$, we can immediately determine whether or not n is prime, since by Part (2) of Theorem 2.36 we know that n is prime if and only if $\phi(n) = n - 1$. Of course, this method is not practically useful, since to determine the primality of n, we need to find $\phi(n)$, but to find $\phi(n)$, we need to factor n, a problem even harder than the primality testing of n.

Remark 3.4 The difficulty in applying Theorem 3.3 for primality testing lies in finding the order of an integer a modulo n, which is computationally intractable. As we will show later, the finding of the order of an integer a modulo n can be efficiently done on a quantum computer.

It is possible to use different bases a_i (rather than a single base a) for different prime factors p_i of $n - 1$ in Theorem 3.2:

Theorem 3.4 *If for each prime p_i of $n - 1$ there exists an integer a_i such that*

(1) $a_i^{n-1} \equiv 1 \pmod{n}$,
(2) $a_i^{(n-1)/p_i} \not\equiv 1 \pmod{n}$.

Then n is prime.

Proof: Suppose that $n - 1 = \prod_{i=1}^{k} p_i^{\alpha_i}$, with $\alpha_i > 0$, for $i = 1, 2, \ldots, k$. Let also $r_i = \text{ord}_n(a_i)$. Then $r_i \mid (n-1)$ and $r_i \nmid (n-1)/p_i$ gives that $p_i^{\alpha_i} \mid r_i$. But for each i, we have $r_i \mid \phi(n)$ and hence $p_i^{\alpha_i} \mid \phi(n)$. This gives $(n-1) \mid \phi(n)$, so n must be prime. ∎

Example 3.4 Let $n = 997$, then $n - 1 = 2^2 \cdot 3 \cdot 83$. We choose three different bases 5, 7, 11 for the prime factors 2, 3, 83, respectively, and get

$$\begin{aligned} 5^{997-1} &\equiv 1 \pmod{997}, \\ 5^{(997-1)/2} &\equiv -1 \not\equiv 1 \pmod{997}, \\ 7^{(997-1)/3} &\equiv 304 \not\equiv 1 \pmod{997}, \\ 11^{(997-1)/83} &\equiv 697 \not\equiv 1 \pmod{997}. \end{aligned}$$

Thus, we can conclude that 997 is prime.

The above tests require the factorization of $n - 1$, a problem even harder than the primality test of n. In 1914, Henry C. Pocklington (1870–1952) showed that it is not necessary to know all the prime factors of $n - 1$; part of them will be sufficient, as indicated in the following theorem.

Theorem 3.5 (Pocklington, 1914) *Let $n - 1 = mj$, with $m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, $m \geq \sqrt{n}$, and $\gcd(m, j) = 1$. If for each prime p_i, $i = 1, 2, \ldots, k$, there exists an integer a such that*

(1) $a^{n-1} \equiv 1 \pmod{n}$,
(2) $\gcd(a^{(n-1)/p_i} - 1, n) = 1$.

Then n is prime.

Proof: Let q be any one of the prime factors of n, and $\mathrm{ord}_n(a)$ the order of a modulo n. We have $\mathrm{ord}_n(a) \mid (q - 1)$ and also $\mathrm{ord}_n(a) \mid (n - 1)$, but $\mathrm{ord}_n(a) \nmid (n - 1)/p_i$. Hence, $p_i^{\alpha_i} \mid \mathrm{ord}_n(a)$. Since $\mathrm{ord}_n(a) \mid (q - 1)$, the result thus follows. ∎

As already pointed out by Pocklington, the above theorem can lead to a primality test:

Pocklington's Test

$$\begin{aligned} \mathrm{Test}(a, p_i) \stackrel{\mathrm{def}}{=} & \; n - 1 = mj, m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}, m \geq \sqrt{n}, \gcd(m, j) = 1 \\ & a^{n-1} \equiv 1 \pmod{n}, \gcd(a^{(n-1)/p_i} - 1, n) = 1. \end{aligned} \tag{3.10}$$

Thus, if n passes the test, then n is prime:

$$n \text{ passes } \mathrm{Test}(a, p_i) \Longrightarrow n \in \mathrm{Primes}. \tag{3.11}$$

Example 3.5 Let also $n = 997$, and $n - 1 = 12 \cdot 83$, $83 > \sqrt{997}$. Choose $a = 3$ for $m = 83$. Then we have

$$\begin{aligned} 3^{997-1} &\equiv 1 \pmod{997}, \\ \gcd(3^{(997-1)/83} - 1, 997) &= 1. \end{aligned}$$

Thus, we can conclude that 997 is prime.

There are some other rigorous, although inefficient, primality tests, for example one of them follows directly from the converse of Wilson's theorem (Theorem 2.59)

Wilson's test

$$\text{Test}(n) \stackrel{\text{def}}{=} n > 1\text{odd}, (n-1)! \equiv -1 \pmod{n} \tag{3.12}$$

Thus, if n passes the test, then n is prime:

$$n \text{ passes Test}(n) \Longrightarrow n \in \text{Primes}. \tag{3.13}$$

Remark 3.5 Unfortunately, very few primes satisfy the condition, in fact, for $p \leq 5 \cdot 10^8$, there are only three primes, namely, $p = 5, 13, 563$. So the Wilson test is essentially of no use in primality testing, not just because of its inefficiency.

Pratt's primality proving

It is interesting to note that, although primality testing is difficult, the verification (proving) of primality is easy, since the primality (as well as the compositeness) of an integer n can be verified very quickly in polynomial-time:

Theorem 3.6 *If n is composite, it can be proved to be composite in $\mathcal{O}((\log n)^2)$ bit operations.*

Proof: If n is composite, there are integers a and b with $1 < a < n$, $1 < b < n$ and $n = ab$. Hence, given the two integers a and b, we multiply a and b, and verify that $n = ab$. This takes $\mathcal{O}((\log n)^2)$ bit operations and proves that n is composite. ∎

Theorem 3.7 *If n is prime, then it can be proved to be prime in $\mathcal{O}((\log n)^4)$ bit operations.*

Theorem 3.7 was discovered by Pratt [3] in 1975; he interpreted the result as showing that every prime has a *succinct primality certification*. The proof can be written as a finite tree whose vertices are labeled by pairs (p, g_p) where p is a prime number and g_p is primitive root modulo p; we illustrate the primality proving of prime number 2557 in Figure 3.2. In the top level of the tree, we write (2557, 2) with 2 the primitive root modulo 2557. As $2557 - 1 = 2^2 \cdot 3^2 \cdot 71$, we have in the second level three vertices (2, 1), (3, 2), (71, 7). Since $3, 71 > 2$, we have in the third level the child vertices (2, 1) for (3, 2), (2, 1), (5, 2) ,and (7, 3) for (71, 7). In the fourth level of the tree, we have (2, 1) for (5, 2), (2, 1) and (3, 2) for (7, 3). Finally, in the fifth level we have (2, 1) for (3, 2). The leaves of the tree now are all labeled (2, 1), completing the certification of the primality of 2557.

Remark 3.6 It should be noted that Theorem 3.7 cannot be used for finding the short proof of primality, since the factorization of $n - 1$ and the primitive root a of n are required.

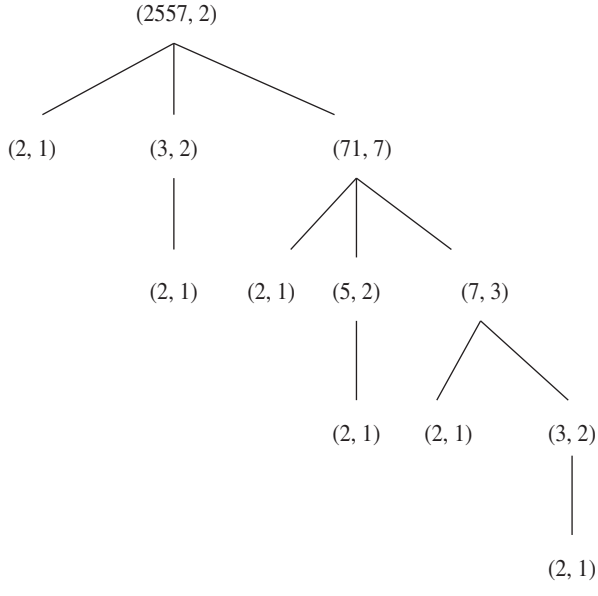


Figure 3.2 Certificate of primality for $n = 2557$

Note that for some primes, Pratt's certificate is considerably shorter. For example, if $p = 2^{2^k} + 1$ is a Fermat number with $k \geq 1$, the p is prime if and only if

$$3^{(p-1)/2} \equiv -1 \pmod{p}. \tag{3.14}$$

This result, known as Papin's test, gives a Pratt certificate for Fermat primes. The work in verifying (3.14) is just $\mathcal{O}(p)$, since $2^k - 1 = \lfloor \log_2 p \rfloor - 1$. In fact, it can be shown that every prime p has an $\mathcal{O}(p)$ certificate. More precisely, we can have:

Theorem 3.8 *For every prime p there is a proof that it is prime which requires for its certification $(5/2 + o(1)) \log_2 p$ multiplications modulo p.*

Problems for Section 3.1

1. Use the Primality test based on primitive roots defined in (3.4) to prove that 1299709 is prime (choose $a = 6$).
2. Use the primality test based on Pocklington's theorem defined in (3.10) to prove that 179424673 is prime.
3. Prove that if $p > 1$ is an odd integer and

$$\begin{cases} a^{(p-1)/2} \equiv -1 \pmod{p} \\ a^{(p-1)/2q} \not\equiv -1 \pmod{p}, \text{ for every odd prime } q \mid (p-1) \end{cases} \tag{3.15}$$

Then p is prime. Conversely, if p is an odd prime, then every primitive root a of p satisfies conditions (3.15).

4. Prove or disprove that there are infinitely many Wilson primes (i.e., those primes p satisfying the condition $(p-1)! \equiv -1 \pmod{p}$).
5. Prove that if

$$\begin{cases} F > \sqrt{n} \\ n-1 = FR, \text{ the complete prime factorization of } F \text{ is known} \\ a^{n-1} \equiv 1 \pmod{n} \\ \gcd(a^{(n-1)/q} - 1,\ n) = 1 \text{ for every prime } q \mid F. \end{cases}$$

Then p is prime.

6. (Brillhart, Lehmer, and Selfridge). Suppose

$$\begin{cases} \sqrt[3]{n} \le F \le \sqrt{n} \\ n-1 = FR, \text{ the complete prime factorization of } F \text{ is known} \\ a^{n-1} \equiv 1 \pmod{n} \\ \gcd(a^{(n-1)/q} - 1,\ n) = 1 \text{ for every prime } q \mid F. \end{cases}$$

Represent a as a base F form $n = c_2F^2 + c_1F + 1$, where $c_1, c_2 \in [0, F-1]$. Show that p is prime if and only if $c_1^2 - 4c_2$ is not a square.

7. (Proth, 1878). Let $n = 2^k m + 1$, with $2 \nmid m$, $2^k > m$, and suppose that $\left(\frac{a}{n}\right) = -1$. Then n is prime if and only if

$$a^{(n-1)/2} \equiv -1 \pmod{n}.$$

8. Use Proth's result to prove that the following three numbers are prime:

$$\begin{aligned} 45334718235548594051481 61 &= 60 \cdot 2^{76} + 1, \\ 62864142619960717084721153 &= 52 \cdot 2^{80} + 1, \\ 188785855991020491922116444 17 &= 61 \cdot 2^{88} + 1. \end{aligned}$$

9. (Pocklington, 1914). Let $n = p^k m + 1$, with p prime and $p \nmid m$, Show that if for some a we have

$$\begin{cases} a^{n-1} \equiv -1 \pmod{n}, \\ \gcd(a^{(n-1)/p} - 1, n) = 1, \end{cases} \tag{3.16}$$

then each prime factor of n is of the form $sp^k + 1$ for some s.

10. Use the Pratt tree to prove the primality of 123456791.
11. Show that for any positive integer $n > 4$, the following three statements are equivalent:
(1) n is composite,
(2) $(n-1)! \equiv 0 \pmod{n}$,
(3) $(n-1)! \not\equiv -1 \pmod{n}$.

12. Let

$$\begin{cases} p \text{ be an odd prime,} \\ p \nmid b(b^2-1), \\ n = \dfrac{b^{2p}-1}{b^2-1}. \end{cases}$$

Then n is a based b pseudoprime.

3.2 Miller–Rabin Test

The Miller–Rabin test[1, 2], also known as the strong pseudoprimality test, or more precisely, the Miller–Selfridge–Rabin test, is a fast and practical probabilistic primality test; its probabilistic error can be reduced to as small as we desire, but not to zero. In terms of computational complexity, it runs in $\mathcal{RP}$. The test is also some times called strong pseudoprimality test. The test was first studied by Miller in 1976 and Rabin in 1980, but was used by Selfridge in 1974, even before Miller and Rabin published their result.

Theorem 3.9 *Let p be a prime. Then*

$$x^2 \equiv 1 \ (mod\ p) \tag{3.17}$$

if and only if $x \equiv \pm 1 \ (mod\ p)$.

Proof: First notice that

$$\begin{aligned} x^2 \equiv \pm 1 \pmod{p} &\Longleftrightarrow (x+1)(x-1) \equiv 0 \pmod{p} \\ &\Longleftrightarrow p \mid (x+1)(x-1) \\ &\Longleftrightarrow p \mid (x+1) \text{ or } p \mid (x-1) \\ &\Longleftrightarrow x+1 \equiv 0 \pmod{p} \text{ or } x-1 \equiv 0 \pmod{p} \\ &\Longleftrightarrow x \equiv -1 \pmod{p} \text{ or } x \equiv 1 \pmod{p}. \end{aligned}$$

Conversely, if either $x \equiv -1 \pmod{p}$ or $x \equiv 1 \pmod{p}$ holds, then $x^2 \equiv 1 \pmod{p}$. ∎

Definition 3.1 The number x is called a *nontrivial square root* of 1 modulo n if it satisfies (3.17) but $x \not\equiv \pm 1 \pmod{n}$.

Example 3.6 The number 6 is a nontrivial square root of 1 modulo 35. Since $x^2 = 6^2 \equiv 1 \pmod{35}$, $x = 6 \not\equiv \pm 1 \pmod{35}$.

Corollary 3.1 *If there exists a nontrivial square root of* 1 *modulo n, then n is composite.*

Example 3.7 Show that 1387 is composite. Let $x = 2^{693}$. We have $x^2 = (2^{693})^2 \equiv 1 \pmod{1387}$, but $x = 2^{693} \equiv 512 \not\equiv \pm 1 \pmod{1387}$. So, 2^{693} is a nontrivial square root of 1 modulo 1387. Then by Corollary 3.4, 1387 is composite.

Theorem 3.10 (Miller–Rabin Test) *Let n be an odd prime number: $n = 1 + 2^j d$, with d odd. Then the b-sequence defined by*

$$\{b^d, b^{2d}, b^{4d}, b^{8d}, \ldots, b^{2^{j-1}d}, b^{2^j d}\} \bmod n \tag{3.18}$$

has one of the following two forms:

$$(1, 1, \ldots, 1, 1, 1, \ldots, 1), \tag{3.19}$$

$$(?, ?, \ldots, ?, -1, 1, \ldots, 1), \tag{3.20}$$

reduced to modulo n, for any $1 < b < n$. (The question mark "?" denotes a number different from $\pm$ 1.)

The correctness of the above theorem relies on Theorem 3.9: If n is prime, then the only solutions to $x^2 \equiv 1 \pmod{n}$ are $x \equiv \pm 1$. To use the strong pseudoprimality test on n, we first choose a base b, usually a small prime. Then compute the b-sequence of n; write $n - 1$ as $2^j d$ where d is odd, compute $b^d \bmod n$, the first term of the b-sequence, and then square repeatedly to obtain the b-sequence of $j + 1$ numbers defined in (3.18), all reduced to modulo n. If n is prime, then the b-sequence of n will be of the form of either (3.19) or (3.20). If the b-sequence of n has any one of the following three forms

$$(?, \ldots, ?, 1, 1, \ldots, 1), \tag{3.21}$$

$$(?, \ldots, ?, ?, ?, \ldots, -1), \tag{3.22}$$

$$(?, \ldots, ?, ?, ?, \ldots, ?), \tag{3.23}$$

then n is *certainly* composite. However, a composite can masquerade as a prime for a few choices of base b, but should not be "too many." The above idea leads naturally to a very efficient and also a practically useful algorithm for (pseudo) primality testing:

Algorithm 3.1 (Miller–Rabin test) This algorithm will test n for primality with high probability:

[1] Let n be an odd number, and the base b a random number in the range $1 < b < n$. Find j and d with d odd, so that $n - 1 = 2^j d$.

[2] Set $i \leftarrow 0$ and $y \leftarrow b^d \pmod{n}$.
If $i = 0$ and $y = 1$, or $y = n - 1$, then terminate the algorithm and output "n is probably prime." If $i > 0$ and $y = 1$ go to [5].

[4] $i \leftarrow i + 1$. If $i < j$, set $y \equiv y^2 \pmod{n}$ and return to [3].
[5] Terminate the algorithm and output "n is definitely not prime."

Theorem 3.11 *The strong pseudoprimality test above runs in time* $\mathcal{O}((\log n)^3)$.

Definition 3.2 A positive integer n with $n - 1 = d \cdot 2^j$ and d odd, is called a base-b *strong probable prime* if it passes the strong pseudoprimality test described above (i.e., the last term in sequence (3.18) is 1, and the first occurrence of 1 is either the first term or is preceded by -1). A base-b strong probable prime is called a base-b *strong pseudoprime* if it is a composite.

If n is prime and $1 < b < n$, then n passes the test. The converse is usually true, as shown by the following theorem.

Theorem 3.12 *Let* $n > 1$ *be an odd composite integer. Then* n *passes the strong test for at most* $(n - 1)/4$ *bases* b *with* $1 \leq b < n$.

Proof: The proof is rather lengthy, we thus only give a sketch of the proof. A more detailed proof can be found either in Section 8.4 of Rosen [44], or in Chapter V of Koblitz [37]. First note that if p is an odd prime, and α and q are positive integers, then the number of incongruent solutions of the congruence

$$x^{q-1} \equiv 1 \pmod{p^\alpha}$$

is $\gcd(q, p^{\alpha-1}(p-1))$. Let $n - 1 = d \cdot 2^j$, where d is an odd positive integer and j is a positive integer. For n to be a strong pseudoprime to the base b, either

$$b^d \equiv 1 \pmod{n}$$

or

$$b^{2^i d} \equiv -1 \pmod{n}$$

for some integer i with $0 < i < j - 1$. In either case, we have

$$b^{n-1} \equiv 1 \pmod{n}.$$

Let the standard prime factorization of n be

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}.$$

By the assertion made at the beginning of the proof, we know that there are

$$\gcd\left(n-1,\, p_i^{\alpha_i}(p_i-1)\right) = \gcd(n-1,\, p_i-1)$$

incongruent solutions to the congruence

$$x^{n-1} \equiv 1 \pmod{p_i^{\alpha_i}},\ i = 1, 2, \ldots, k.$$

Further, by the Chinese Remainder theorem, we know that there are exactly

$$\prod_{i=1}^{k} \gcd(n-1,\, p_i-1)$$

incongruent solutions to the congruence

$$x^{n-1} \equiv 1 \pmod{n}.$$

To prove the theorem, there are three cases to consider:

(1) the standard prime factorization of n contains a prime power $p_r^{\alpha_r}$ with exponent $\alpha_r \geq 2$;
(2) $n = pq$, with p and q distinct odd primes;
(3) $n = p_1 p_2 \cdots p_k$, with $p_1, p_2, \ldots, p_k$ distinct odd primes.

The second case can actually be included in the third case. We consider here only the first case. Since

$$\begin{aligned}\frac{p_r-1}{p_r^{\alpha_r}} &= \frac{1}{p_r^{\alpha_r-1}} - \frac{1}{p_r^{\alpha_r}} \\ &\leq \frac{2}{9},\end{aligned}$$

we have

$$\begin{aligned}\prod_{i=1}^{k} \gcd(n-1,\, p_i-1) &\leq \prod_{i=1}^{k}(p_i-1) \\ &\leq \left(\prod_{\substack{i=1 \\ i\neq r}}^{k} p_i\right)\left(\frac{2}{9}p_r^{\alpha_r}\right) \\ &\leq \frac{2}{9}n \\ &\leq \frac{n-1}{4} \qquad \text{for } n \geq 9.\end{aligned}$$

Thus, there are at most $(n-1)/4$ integers b, $1 < b < n-1$, for which n is a base-b strong pseudoprime and n can pass the strong test. ■

A probabilistic interpretation of Theorem 3.12 is as follows:

Corollary 3.2 *Let $n > 1$ be an odd composite integer and b be chosen randomly from $\{2, 3, \ldots, n-1\}$. Then the probability that n passes the strong test is less than $1/4$.*

From Corollary 3.2, we can construct a simple, general purpose, polynomial time primality test which has a positive (but arbitrarily small) probability of giving the wrong answer. Suppose an error probability of ϵ is acceptable. Choose k such that $4^{-k} < \epsilon$, and select $b_1, b_2, \ldots, b_k$ randomly and independently from $\{2, 3, \ldots, n-1\}$. If n fails the strong test on b_i, $i = 1, 2, \ldots, k$, then n is a strong probable prime.

Theorem 3.13 *The strong test (i.e., Algorithm 3.1) requires, for $n - 1 = 2^j d$ with d odd and for k randomly selected bases, at most $k(2 + j) \log n$ steps. If n is prime, then the result is always correct. If n is composite, then the probability that n passes all k tests is at most $1/4^k$.*

Proof: The first two statements are obvious, only the last statement requires proof. An error will occur only when the n to be tested is composite and the bases $b_1, b_2, \ldots, b_k$ chosen in this particular run of the algorithm are all nonwitnesses. (An integer a is a *witness* to the compositeness of n if it is possible using a to prove that n is composite, otherwise it is a *nonwitness*). Since the probability of randomly selecting a nonwitness is smaller than $1/4$ (by Corollary 3.2), then the probability of independently selecting k nonwitnesses is smaller than $1/4^k$. Thus the probability that with any given number n, a particular run of the algorithm will produce an erroneous answer is smaller than $1/4^k$ [2]. ■

Problems for Section 3.2

1. Show that there are infinitely many base 2 strong pseudoprime numbers.
2. Show that there are infinitely many base b strong pseudoprime numbers.
3. Show that there are infinitely many base 2 Euler pseudoprime numbers.
4. Show that there are infinitely many base b Euler pseudoprime numbers.
5. Show that if n is an odd composite number, then there exists an integer b such that $\gcd(b, n) = 1$ and n is not a base b Euler pseudoprime.
6. Show that if n is a base b strong pseudoprime, then n is also a base b Euler pseudoprime.
7. Let $n > 1$ be an odd integer. Show that n is prime if and only if
$$\forall a \in \mathbb{Z}_n^*, a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}.$$
8. Let $n > 1$ be an odd integer. Show that n is prime if and only if
$$\forall a \in \mathbb{Z}_n^*, a^{(n-1)/2} \equiv \pm 1 \pmod{n},$$
$$\exists a \in \mathbb{Z}_n^*, a^{(n-1)/2} \equiv -1 \pmod{n}.$$

9. Let R_n be the set

$$\{b \in \mathbb{Z}^* : b^{n-1} \not\equiv 1 \pmod{n} \text{ or } \exists t \geq 0, 1 < \gcd(b^{(n-1)/2^t} - 1, n) < n\}$$

where $n - 1 = 2^k u$, u odd, and $\exists t$ is restricted to $t \leq k$. Prove that for all odd composite integer $n > 9$,

$$\frac{|\mathbb{Z}_n^* - R_n|}{\phi(n)} \leq \frac{1}{4}.$$

10. Prove the following theorem: Let $n = FR + 1$, where $0 < R < F$. If for some a we have

$$\begin{cases} a^{n-1} \equiv 1 \pmod{n}, \\ \gcd\left(a^{(n-1)/q} - 1,\ n\right) = 1, \text{ for each distinct prime } q \mid F. \end{cases}$$

Then n is a prime.

11. Prove Selfridge's theorem: Let $n > 1$ be an odd integer. If

$$n - 1 = \prod_{i-1}^{k} p_i^{\alpha_i}$$

where $p_i,\ i = 1, 2, \ldots, k$ are primes and each p_i there exists an a_i such that

$$\begin{cases} a_i^{n-1} \equiv 1 \pmod{n}, \\ a_i^{(n-1)/p_i} \not\equiv 1 \pmod{n}. \end{cases}$$

Then n is a prime.

3.3 Elliptic Curve Tests

In the last 20 years or so, there have been some surprising applications of elliptic curves to problems in primality testing, integer factorization and public-key cryptography. In 1985, Lenstra [4] announced an elliptic curve factoring method (the formal publication was in 1987), just one year later, Goldwasser and Kilian in 1986 [5] adapted Lenstra's factoring algorithm to obtain an elliptic curve primality test, and Miller in 1986 [6] and Koblitz in 1987 [7] independently arrived at the idea of elliptic curve cryptography. In this section, we discuss fast primality tests based elliptic curves. These tests, though, are still probabilistic, with zero error. In terms of computational complexity, they run in $\mathcal{ZPP}$.

First we introduce a test based on Cox in 1989 [8].

Theorem 3.14 (Cox) *Let $n \in \mathbb{N}$ with $n > 13$ and $\gcd(n, 6) = 1$, and let $E : y^2 \equiv x^3 + ax + b \pmod{n}$ be an elliptic curve over $\mathbb{Z}/n\mathbb{Z}$. Suppose that*

(1) $n + 1 - 2\sqrt{n} \leq |E(\mathbb{Z}_n)| \leq n + 1 + 2\sqrt{n}$.
(2) $|E(\mathbb{Z}_n)| = 2q$, *with q an odd prime.*

If $P \neq \mathcal{O}_E$ is a point on E and $qP = \mathcal{O}_E$, then n is prime.

Proof: See pp. 324–325 of Cox in 1989 [8]. ∎

Example 3.8 Suppose we wish to prove that $n = 9343$ is a prime using the above elliptic curve test. First notice that $n > 13$ and $\gcd(n, 6) = 1$. Next, we choose an elliptic curve $y^2 = x^3 + 4x + 4$ over $\mathbb{Z}/n\mathbb{Z}$ with $P = (0, 2)$, and calculate $|E(\mathbb{Z}/n\mathbb{Z})|$, the number of points on the elliptic curve $E(\mathbb{Z}/n\mathbb{Z})$, Suppose we use the numerical exhaustive method, we then know that there are 9442 points on this curve:

No.	Points on the curve	No.	Points on the curve
1	$\mathcal{O}_E$	2	(0,2)
3	(0,9341)	4	(1,3)
5	(1,9340)	6	(3,1264)
7	(3,8079)	8	(7,3541)
9	(7,5802)	10	(10,196)
⋮	⋮	⋮	⋮
9439	(9340,4588)	9440	(9340,4755)
9441	(9341,3579)	9442	(9341,5764)

Thus, $|E(\mathbb{Z}_{9343})| = 9442$. Now we are ready to verify the two conditions in Theorem 3.14. For the first condition, we have:

$$\lfloor n + 1 - 2\sqrt{n} \rfloor = 9150 < 9442 < 9537 = \lfloor n + 1 + 2\sqrt{n} \rfloor.$$

For the second condition, we have:

$$|E(\mathbb{Z}_{9343})| = 9442 = 2q, \quad q = 4721 \in \text{Primes}.$$

So, both conditions are satisfied. Finally and most importantly, we calculate qP over the elliptic curve $E(\mathbb{Z}_{9343})$ by tabling its values as follows:

2P = (1,9340)	4P = (1297,1515)
9P = (6583,436)	18P = (3816,7562)
36P = (2128,1972)	147P = (6736,3225)
295P = (3799,4250)	590P = (7581,7757)
1180P = (5279,3262)	2360P = (3039,4727)
4721P = $\mathcal{O}_E$	

Since $P \in E(\mathbb{Z}_{9343})$ and $P \neq \mathcal{O}_E$, but $qP \in E(\mathbb{Z}_{9343})$ and $qP = \mathcal{O}_E$, we conclude that $n = 9343$ is a prime number!

The main problem with the above test is the calculation of $|E(\mathbb{Z}_n)|$; when n becomes large, finding the value of $|E(\mathbb{Z}_n)|$ is as difficult as proving the primality of n [9]. Fortunately, Goldwasser and Kilian found a way to overcome this difficulty [5]. To introduce the

Goldwasse-Kilian method, let us first introduce a useful converse of Fermat's little theorem, which is essentially Pocklington's theorem:

Theorem 3.15 (Pocklington's theorem) *Let s be a divisor of $n-1$. Let a be an integer prime to n such that*

$$\left.\begin{aligned} a^{n-1} &\equiv 1 \pmod{n} \\ \gcd\left(a^{(n-1)/q}, n\right) &= 1 \end{aligned}\right\} \tag{3.24}$$

for each prime divisor q of s. Then each prime divisor p of n satisfies

$$p \equiv 1 \pmod{s}. \tag{3.25}$$

Corollary 3.3 *If $s > \sqrt{N} - 1$, then n is prime.*

The Goldwasser–Kilian test can be regarded as an elliptic curve analog of Pocklington's theorem:

Theorem 3.16 (Goldwasser–Kilian) *Let n be an integer greater than 1 and prime to 6, E an elliptic curve over $\mathbb{Z}/n\mathbb{Z}$, P a point on E, m and s two integers with $s \mid m$. Suppose we have found a point P on E that satisfies $mP = \mathcal{O}_E$, and that for each prime factor q of s, we have verified that $(m/q)P \neq \mathcal{O}_E$. Then if p is a prime divisor of n, $|E(\mathbb{Z}_p)| \equiv 0 \pmod{s}$.*

Corollary 3.4 *If $s > (\sqrt[4]{n} + 1)^2$, then n is prime.*

Combining the above theorem with Schoof's algorithm [11] which computes $|E(\mathbb{Z}_p)|$ in time $\mathcal{O}\left((\log p)^{8+\epsilon}\right)$, we obtain the following Goldwasser–Kilian algorithm ([5, 10]).

Algorithm 3.2 (Goldwasser–Kilian Algorithm) Given a probable prime n, this algorithm will show whether or not n is indeed prime.

[1] Choose a non-singular elliptic curve E over $\mathbb{Z}/n\mathbb{Z}$, for which the number of points m satisfies $m = 2q$, with q a probable prime;
[2] If (E, m) satisfies the conditions of Theorem 3.16 with $s = m$, then n is prime, otherwise it is composite;
[3] Perform the same primality proving procedure for q;
[4] Exit.

The running time of the Goldwasser–Kilian algorithm is given in the following two theorems [12]:

Theorem 3.17 *Suppose that there exist two positive constants c_1 and c_2 such that the number of primes in the interval $[x, x + \sqrt{2x}]$, where $(x \geq 2)$, is greater than $c_1\sqrt{x}(\log x)^{-c_2}$, then the Goldwasser–Kilian algorithm proves the primality of n in expected time $\mathcal{O}\left((\log n)^{10+c_2}\right)$.*

Theorem 3.18 *There exist two positive constants c_3 and c_4 such that, for all $k \geq 2$, the proportion of prime numbers n of k bits for which the expected time of Goldwasser–Kilian is bounded by $c_3((\log n)^{11})$ is at least*

$$1 - c_4 2^{-k \frac{1}{\log \log k}}.$$

A serious problem with the Goldwasser–Kilian test is that Schoof's algorithm seems almost impossible to implement. In order to avoid the use of Schoof's algorithm, Atkin and Morain [12] in 1991 developed a new implementation method called ECPP (elliptic curve primality proving), which uses the properties of elliptic curves over finite fields related to complex multiplication. We summarize the principal properties of ECPP as follows:

Theorem 3.19 (Atkin–Morain) *Let p be a rational prime number that splits as the product of two principal ideals in a field $\mathcal{K}$: $p = \pi\pi'$ with π, π' integers of $\mathcal{K}$. Then there exists an elliptic curve E defined over $\mathbb{Z}_p$ having complex multiplication by the ring of integers of $\mathcal{K}$, whose cardinality is*

$$m = N_{\mathcal{K}}(\pi - 1) = (\pi - 1)(\pi' - 1) = p + 1 - t \tag{3.26}$$

with $|t| \leq 2\sqrt{p}$ (Hasse's theorem) and whose invariant is a root of a fixed polynomial $H_D(X)$ (depending only upon D) modulo p.

For more information on the computation of the polynomials H_D, readers are referred to Cox [8] and Morain ([14, 16]). Note that there are also some other important improvements on the Goldwasser–Kilian test, notably the Adleman–Huang's primality proving algorithm [17] using hyperelliptic curves.

The Goldwasser–Kilian algorithm begins by searching for a curve and computes its number of points, but the Atkin–Morain ECPP algorithm does exactly the opposite. The following is a brief description of the ECPP algorithm.

Algorithm 3.3 (Atkin–Morain ECPP) Given a probable prime n, this algorithm will show whether or not n is indeed prime.

[1] (Initialization) Set $i \leftarrow 0$ and $N_0 \leftarrow n$.
[2] (Building the sequence)
  While $N_i > N_{small}$
  [2.1] Find a D_i such that $N_i = \pi_i \pi_i' in \mathcal{K} = \mathcal{Q}(\sqrt{-D_i})$;
  [2.2] If one of the $w(-D_i)$ numbers $m_1, ..., m_w (m_r = N_K(\omega_r - 1)$ where ω_r is a conjugate of π) is probably factored go to step [2.3] else go to [2.1];
  [2.3] Store $\{i, N_i, D_i, \omega_r, m_r, F_i\}$ where $m_r = F_i N_{i+1}$. Here F_i is a completely factored integer and N_{i+1} a probable prime; set $i \leftarrow i + 1$ and go to step [2.1].
[3] (Proving)
  For i from k down to 0
  [3.1] Compute a root j of $H_{D_i}(X) \equiv 0 \pmod{N_i}$;

[3.2] Compute the equation of the curve E_i of the invariant j and whose cardinality modulo N_i is m_i;
[3.3] Find a point P_i on the curve E_i;
[3.4] Check the conditions of Theorem 3.19 with $s = N_{i+1}$ and $m = m_i$.
[4] (Exit) Terminate the execution of the algorithm.

For ECPP, only the following heuristic analysis is known [14].

Theorem 3.20 *The expected running time of the ECPP algorithm is roughly proportional to* $\mathcal{O}\left((\log n)^{6+\epsilon}\right)$ *for some* $\epsilon > 0$.

Corollary 3.5 *The ECPP algorithm is in* $\mathcal{ZPP}$.

Thus, for all practical purposes, we could just simply use a combined test of a probabilistic test and an elliptic curve test as follows:

Algorithm 3.4 (Practical primality testing) Given a random odd positive integer n, this algorithm will make a combined use of probabilistic tests and elliptic curve tests to determine whether or not n is prime:

[1] (Primality Testing – Probabilistic Testing) Use a combination of the strong pseudoprimality test and the Lucas pseudoprimality test to determine if n is a probable prime. (This has been implemented in Maple function *isprime*.) If it is, go to [2], else report that n is composite and go to [3].
[2] (Primality Proving – Elliptic Curve Proving) Use an elliptic curve test (e.g., the ECPP test) to verify whether or not n is indeed a prime. If it is, then report that n is prime, otherwise, report that n is composite.
[3] (Exit) Terminate the algorithm.

Problems for Section 3.3

1. Prove Cox's Theorem, (i.e., Theorem 3.14) for elliptic curve primality test.
2. Prove the following theorem: Let $n = FR + 1$, where $0 < R < F$. If for some a we have

$$\begin{cases} a^{n-1} \equiv 1 \pmod{n}, \\ \gcd\left(a^{(n-1)/q} - 1,\ n\right) = 1, \text{ for each distinct prime } q \mid F. \end{cases}$$

   Then n is a prime.
3. (Selfridge) Let $n > 1$ be an odd integer. Show that if

$$n - 1 = \prod_{i-1}^{k} p_i^{\alpha_i}$$

where $p_i,\ i = 1, 2, \ldots, k$ are primes and each p_i there exists an a_i such that

$$\begin{cases} a_i^{n-1} \equiv 1 \pmod{n}, \\ a_i^{(n-1)/p_i} \not\equiv 1 \pmod{n}, \end{cases}$$

then n is a prime.

4. Prove the Goldwasser–Kilian theorem (Theorem 3.16) and its corollary (Corollary 3.4).
5. Show that the Goldwasser–Kilian elliptic curve test algorithm is a $\mathcal{ZPP}$ algorithm.
6. Show that the Atkin–Morain ECPP algorithm is a $\mathcal{ZPP}$ algorithm.
7. Use Cox's test (Theorem 3.14) to prove that 26869 is a prime.
8. Use the Goldwasser–Kilian elliptic curve test to show that 907 is a prime.
9. Use some variant of ECPP to prove the following four numbers are prime:
   (1) $\dfrac{2^{3539}+1}{3}$,
   (2) $2177^{580} + 580^{2177}$,
   (3) $4405^{2638} + 2638^{4405}$,
   (4) $10^{9999} + 33603$.

3.4 AKS Test

On 6 August 2002, Agrawal, Kayal, and Saxena in the Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, proposed a deterministic polynomial-time test (AKS test for short) for primality [18], relying on no unproved assumptions. That is, AKS runs in $\mathcal{P}$. It was not a great surprise that such a test existed,[1] but the relatively easy algorithm and proof was indeed a big surprise. The key to the AKS test is in fact a very simple version of Fermat's little theorem:

Theorem 3.21 *Let x be an indeterminate and* $\gcd(a, n) = 1$ *with* $n > 1$. *Then*

$$n \in \text{Primes} \Longleftrightarrow (x-a)^n \equiv (x^n - a) \pmod{n}. \tag{3.27}$$

Proof: By the binomial theorem, we have

$$(x-a)^n = \sum_{r=0}^{n} \binom{n}{r} x^r (-a)^{n-r}.$$

If n is prime, then $n \mid \binom{n}{r}$ (i.e., $\binom{n}{r} \equiv 0 \pmod{n}$), for $r = 1, 2, \ldots, n-1$. Thus, $(x-a)^n \equiv (x^n - a^n) \pmod{n}$ and (3.27) follows from Fermat's little theorem. On the other hand, if n is composite, then it has a prime divisor q. Let q^k be the greatest power of q that divides n.

[1] Dixon [19] predicated in 1984 that "the prospect for a polynomial-time algorithm for proving primality seems fairly good, but it may turn out that, on the contrary, factoring is NP-hard."

Then q^k does not divide $\binom{n}{q}$ and is relatively prime to a^{n-q}, so the coefficient of the term x^q on the left of $(x-a)^n \equiv (x^n - a)$ is not zero, but it is on the right. ∎

Remark 3.7 In about 1670 Leibniz (1646–1716) used the fact that if n is prime then n divides the binomial coefficient $\binom{n}{r}$, $r = 1, 2, \ldots, n-1$ to show that if n is prime then n divides $(a_1 + a_2 + \cdots + a_m)^n - (a_1^n + a_2^n + \cdots + a_m^n)$. Letting $a_i = 1$, for $i = 1, 2, \ldots, m$, Leibniz proved that n divides $m^n - m$ for any positive integer m.

Example 3.9 Let $a = 5$ and $n = 11$. Then we have:

$$\begin{aligned}(x-5)^{11} &= x^{11} - 55x^{10} + 1375x^9 - 20625x^8 + 206250x^7 - 1443750x^6 \\ &\quad + 7218750x^5 - 25781250x^4 + 64453125x^3 - 107421875x^2 \\ &\quad + 107421875x - 48828125 \\ &\equiv x^{11} - 5 \pmod{11}.\end{aligned}$$

However, if we let $n = 12$, which is not a prime, then

$$\begin{aligned}(x-5)^{12} &= x^{12} - 60x^{11} + 1650x^{10} - 27500x^9 + 309375x^8 - 2475000x^7 \\ &\quad + 14437500x^6 - 61875000x^5 + 193359375x^4 - 429687500x^3 \\ &\quad + 644531250x^2 - 585937500x + 244140625 \\ &\not\equiv x^{12} - 5 \pmod{12}.\end{aligned}$$

Theorem 3.21 provides a deterministic test for primality. However, the test cannot be done in polynomial-time because of the intractability of $(x-a)^n$; we need to evaluate n coefficients in the left-hand side of (3.27) in the worst case. A simple way to avoid this computationally intractable problem is to evaluate both sides of (3.27) modulo a polynomial[2] of the form $x^r - 1$: For an *appropriately chosen* small r. Thus, we get a new characterization of primes

$$n \in \text{Primes} \Longrightarrow (x-a)^n \equiv (x^n - a) \pmod{x^r - 1,\ n}, \tag{3.28}$$

for all r and n relatively prime to a. A problem with this characterization is that for particular a and r, some composites can satisfy (3.28), too. However, no composite n satisfies (3.28) for all a and r, that is,

$$\begin{aligned}&n \in \text{Composites} \Longrightarrow \exists a, r \text{ such that} \\ &(x-a)^n \not\equiv (x^n - a) \pmod{x^r - 1,\ n}.\end{aligned} \tag{3.29}$$

[2]By analog with congruences in $\mathbb{Z}$, we say that polynomials $f(x)$ and $g(x)$ are congruent modulo $h(x)$ and write $f(x) \equiv g(x) \pmod{h(x)}$, whenever $f(x) - g(x)$ is divisible by $h(x)$. The set (ring) of polynomials modulo $h(x)$ is denoted by $\mathbb{Z}[x]/h(x)$. If all the coefficients in the polynomials are also reduced to n, then we write $f(x) \equiv g(x) \pmod{h(x)}$ as $f(x) \equiv g(x) \pmod{h(x), n}$, and $\mathbb{Z}[x]/h(x)$ as $\mathbb{Z}_n[x]/h(x)$ (or $\mathbb{F}_p[x]/h(x)$ if n is a prime p).

Example 3.10 Let $n = 6$ and $a = r = 5$. Then

$$(x-5)^6 \equiv 3x^4 - 2x^3 + 3x^2 + x + 1 \pmod{x^5 - 1, 6},$$
$$x^6 - 5 \equiv x + 1 \pmod{x^5 - 1, 6},$$
$$(x-5)^6 \not\equiv x^6 - 5 \pmod{x^5 - 1, 6}.$$

The main idea of the AKS test is to restrict the range of a and r enough to keep the complexity of the computation polynomial, while ensuring that no composite n can pass (3.28).

Theorem 3.22 (Agrawal–Kayal–Saxena) *Let $n \in \mathbb{Z}^+$. Let q and r be prime numbers. Let S be a finite set of integers. Assume that*

(1) $q \mid (r-1)$,
(2) $n^{(r-1)/q} \not\equiv 0, 1 \pmod{r}$,
(3) $\gcd(a - a', n) = 1$, *for all distinct* $a, a' \in S$,
(4) $\text{binom}|\ S\ | + q - 1|\ S\ | \geq n^{2\lfloor\sqrt{r}\rfloor}$,
(5) $(x-a)^n \equiv (x^n - a) \pmod{x^r - 1, n}$, *for all* $a \in S$.

Then n is a prime power.

Proof: We follow the streamlined presentation of Bernstein [20]. First find a prime factor p of n, with $p^{(r-1)/q} \not\equiv 0, 1 \pmod{r}$ and $q \mid (r-1)$. If every prime factor p of n has $p^{(r-1)/q} \equiv 0, 1 \pmod{r}$, then $n^{(r-1)/q} \equiv 0, 1 \pmod{r}$. By assumption, we have $(x-a)^n \equiv (x^n - a) \pmod{x^r - 1, p}$ for all $a \in S$. Substituting x^{n^i} for x, we get $(x^{n^i} - a)^n \equiv (x^{n^{i+1}} - a) \pmod{x^{n^i r} - 1, p}$, and also $(x^{n^i} - a)^n \equiv (x^{n^{i+1}} - a) \pmod{x^r - 1, p}$. By induction, $(x-a)^{n^i} \equiv (x^{n^i} - a) \pmod{x^r - 1, p}$ for all $i \geq 0$. By Fermat's little theorem, $(x-a)^{n^i p^j} \equiv (x^{n^i} - a)^{p^j} \equiv (x^{n^i p^j} - a) \pmod{x^r - 1, p}$ for all $j \geq 0$. Now consider the products $n^i p^j$, with $0 \leq i, j \leq \lfloor\sqrt{r}\rfloor$ and $1 \leq n^i p^j \leq n^{2\lfloor\sqrt{r}\rfloor}$. There are $(1 + \lfloor\sqrt{r}\rfloor)^2 > r$ such (i, j) pairs, so there are (by the pigeon-hole principle) distinct pairs (i_1, j_1) and (i_2, j_2) for which $t_1 \equiv t_2 \pmod{r}$ where $t_1 = n^{i_1} p^{j_1}$, $t_2 = n^{i_2} p^{j_2}$. So, $x^{t_1} \equiv x^{t_2} \pmod{x^r - 1, p}$, therefore, $(x-a)^{t_1} \equiv (x^{t_1} - a) \equiv x^{t_2} - a \equiv (x-a)^{t_2} \pmod{x^r - 1, p}$ for all $a \in S$. Next find an irreducible polynomial $h(x)$ in $\mathbb{F}_p[x]$ dividing $(x^r - 1)/(x - 1)$ such that $(x-a)^{t_1} \equiv (x-a)^{t_2} \pmod{h(x), p}$ for all $a \in S$. Define G as a subgroup of $(\mathbb{F}_p[x]/h(x))^*$ generated by $\{x - a : a \in S\}$, then $g^{t_1} = g^{t_2}$ for all $g \in G$. Since G has at least $\binom{|S|+q-1}{|S|}$ elements (by some combinatorics and the elementary theory of cyclotomic polynomials), we have

$$|t_1 - t_2| < n^{\lfloor\sqrt{r}\rfloor} p^{\lfloor\sqrt{r}\rfloor} \leq n^{2\lfloor\sqrt{r}\rfloor} \leq \binom{|S| + q - 1}{|S|} \leq |G|,$$

so, $t_1 = t_2$, as desired. ∎

Remark 3.8 There are some new interesting developments and refinements over the above result. The American Institute of Mathematics had a workshop on 24–28 March 2003 on "Future Directions in Algorithmic Number Theory"; the institute has made the lecture notes and a set of problems of the workshop available through `http://www.aimath.org`.

To turn the above theorem into a deterministic polynomial-time test for primality, we first find a small odd prime r such that $n^{(r-1)/q} \equiv 0, 1 \pmod{r}$ and $\binom{q+s-1}{s} > n^{2\lfloor\sqrt{r}\rfloor}$; here q is the largest prime factor of $r-1$, and s is any moderately large integer. A theorem of Fouvry [21] from analytic number theory implies that a suitable r exists on the order $\mathcal{O}((\log n)^6)$ and s on the order $\mathcal{O}((\log n)^4)$. Given such a triple (q, r, s), we can easily test that n have no prime factors $< s$ and that $(x-a)^n = x^n - b \pmod{x^r - 1, n}$ for all $a \in \{0, 1, 2, \ldots, s-1\}$. Any failure of the first test reveals a prime factor of n and any failure of the second test proves that n is composite. If n passes both tests then n is a prime power. Here is the algorithm.

Algorithm 3.5 (The AKS algorithm) Give a positive integer $n > 1$, this algorithm will decide whether or not n is prime in deterministic polynomial-time.

[1] If $n = a^b$ with $a \in \mathbb{N}$ and $b > 1$, then output COMPOSITE.
[2] Find the smallest r such that $\text{ord}_r(n) > 4(\log n)^2$.
[3] If $1 < \gcd(a, n) < n$ for some $a \leq r$, then output COMPOSITE.
[4] If $n \leq r$, then output PRIME.
[5] For $a = 1$ to $\lfloor 2\sqrt{\phi(r)} \log n \rfloor$ do
    if $(x-a)^n \not\equiv (x^n - a) \pmod{x^r - 1, n}$,
    then output COMPOSITE.
[6] Output PRIME.

The algorithm is indeed very simple and short (with only 6 statements), possibly the shortest algorithm for a (big) unsolved problem ever!

Theorem 3.23 (Correctness) *The above algorithm returns* `PRIME` *if and only if n is prime.*

Proof: If n is prime, then the steps 1 and 3 will never return `COMPOSITE`. By Theorem 3.21, the `for` loop in step 5 will also never return `COMPOSITE`. Thus, n can only be identified to be `PRIME` in either step 4 or step 6. The "only if" part of the theorem left as an exercise. ■

Theorem 3.24 *The AKS algorithm runs in time* $\mathcal{O}((\log n)^{12+\epsilon})$.

*Proof (*Sketch*):* The algorithm has two main steps, 2 and 5. Step 2 finds a suitable r (such an r exists by Fouvry [21] and Baker and Harman [22]), and can be carried out in time $\mathcal{O}((\log n)^{9+\epsilon})$. Step 5 verifies (3.28), and can be performed in time $\mathcal{O}((\log n)^{12+\epsilon})$. So, the overall runtime of the algorithm is $\mathcal{O}((\log n)^{12+\epsilon})$. ■

Remark 3.9 Under the assumption of a conjecture on the density of the Sophie Germain primes, the AKS algorithm runs in time $\mathcal{O}((\log n)^{6+\epsilon})$. If a conjecture of Bhatacharjee and Pandey[3] is true, then this can be further reduced to $\mathcal{O}((\log n)^{3+\epsilon})$. Of course, we do not know if these conjectures are true.

[3] Bhatacharjee and Pandey conjectured in 2001 [23] that if $r \in \text{Primes}$ and $\gcd(r, n) = 1$, and $(x-1)^n \equiv x^n - 1 \pmod{x^r - 1, n}$, then either n is prime or $n^2 \equiv 1 \pmod{r}$.

Remark 3.10 The AKS algorithm is a major breakthrough in computational number theory. However, it can only be of theoretical interest, since its current runtime is in $\mathcal{O}((\log n)^{12+\epsilon})$, which is much higher than $\mathcal{O}((\log n)^{6+\epsilon})$ for ECPP and $\mathcal{O}((\log n)^3)$ for Miller–Rabin's test. For all practical purposes, we would still prefer to use Miller–Rabin's probabilistic test [2] in the first instance and the zero-error probabilistic test ECPP in the last step of a primality testing process.

Remark 3.11 The efficiency of the AKS algorithm for test primality does not have (at least at present) any obvious connections to that of integer factorization, although the two problems are related to each other. The fastest factoring algorithm, namely the Number Field Sieve [25], has an expected running time $\mathcal{O}(\exp(c\sqrt[3]{\log n}\sqrt[3]{(\log\log n)^2}))$. We do not know if the simple mathematics used in the AKS algorithm for primality testing can be used for other important mathematical problems, such as the Integer Factorization Problem.

Remark 3.12 The efficiency of the AKS algorithm has not yet become a threat to the security of the factoring base (such as the RSA) cryptographic systems, since the security of RSA depends on the computational intractability of the Integer Factorization Problem.

At the end of this section on AKS test, and also the end of this chapter on primality testing, we have given a comparison of some general purpose primality tests in terms of computational complexity (running time).

Recall that multiplying two $\log n$ bit integers has a running time of

$$\mathcal{O}((\log n)^2)$$

and the fastest known algorithm, the Schönhage–Strassen algorithm, has a running time of

$$\begin{aligned}\mathcal{O}(\log n \log\log n \log\log\log n) &= \mathcal{O}((\log n)^{1+\epsilon}) \\ &= \tilde{\mathcal{O}}(\log n).\end{aligned}$$

Thus, if we let $\mathcal{O}((\log n)^{\mu})$ be the running time of integer multiplication, then

$$\begin{aligned}\mathcal{O}((\log n)^{\mu}) &\overset{u=2}{\Longrightarrow} \mathcal{O}((\log n)^2) \quad \text{use practical multiplication algorithm} \\ &\overset{u=1+\epsilon}{\Longrightarrow} \mathcal{O}((\log n)^{1+\epsilon}) \quad \text{use fast multiplication algorithm.}\end{aligned}$$

The Miller–Rabin test runs in time

$$\begin{aligned}\mathcal{O}((\log n)^{1+\mu}) &\Rightarrow \mathcal{O}((\log n)^3) \\ &\Rightarrow \mathcal{O}((\log n)^{2+\epsilon}) \\ &= \tilde{\mathcal{O}}((\log n)^2)\end{aligned}$$

so it is a very fast (polynomial-time) primality test, as its degree of complexity is just one more than integer multiplications. A drawback of the Miller–Rabin test is that it is probabilistic,

not deterministic, that is, there will be a small error of probability when it declares an integer to be prime. However, if we assume the Generalized Riemann Hypothesis (GRH), then the Miller–Rabin test can be made deterministic with running time in

$$\begin{aligned}\mathcal{O}((\log n)^{3+\mu}) &\Rightarrow \mathcal{O}((\log n)^5)\\ &\Rightarrow \mathcal{O}((\log n)^{4+\epsilon})\\ &= \tilde{\mathcal{O}}((\log n)^4)\end{aligned}$$

It still has polynomial-time complexity, just two degrees higher than its probabilistic version.

The AKS (Agrawal, Kayal, and Saxena) test takes time

$$\begin{aligned}\mathcal{O}((\log n)^{6(1+\mu)}) &\Rightarrow \mathcal{O}((\log n)^{18})\\ &\Rightarrow \mathcal{O}((\log n)^{12+\epsilon})\\ &= \tilde{\mathcal{O}}((\log n)^{12}).\end{aligned}$$

That is, by practical multiplication algorithm, AKS runs in $\mathcal{O}((\log n)^{18})$ whereas by SchönhageStrassen algorithm in $\tilde{\mathcal{O}}((\log n)^{12})$. It can be show that [13] the AKS algorithm cannot be expected to be proved to run faster than

$$\mathcal{O}\left((\log n)^{6(1+\mu)}\right).$$

However, in practice, it is easy to find a suitable prime of the smallest possible size $\mathcal{O}((\log n)^2)$, thus, the practical running time of the AKS algorithm is

$$\mathcal{O}\left((\log n)^{3(1+\mu)}\right).$$

It also can be shown that one cannot find a deterministic test that runs faster than $\mathcal{O}((\log n)^{3(1+\mu)})$. That is, $\mathcal{O}((\log n)^{3(1+\mu)})$ is the fastest possible running time for a deterministic primality test. Recently, H. Lenstra and Pomerance showed that a test having running time in

$$\mathcal{O}\left((\log n)^{(3+\epsilon)(1+\mu)}\right)$$

is possible, but which is essentially the same as the practical running time $\mathcal{O}((\log n)^{3(1+\mu)})$. Of course, if one is willing to accept a small error of probability, a randomized version of AKS is possible and can be faster, but this is not the point, as if one is willing to use a probabilistic test, one would prefer to use the Miller–Rabin test.

The APR (Adleman, Pomerance, and Rumely) cyclotomic (or Jacobi sum) test is a deterministic and nearly polynomial-time test, and it runs in time

$$\mathcal{O}((\log n)^{c \log\log\log n}),\ c > 0.$$

Table 3.1 Running time comparison of three general primality tests

Test	Practical running time	Fast running time
Miller–Rabin	$\mathcal{O}((\log n)^3)$	$\tilde{\mathcal{O}}((\log n)^2)$
AKS	$\mathcal{O}((\log n)^{18})$	$\tilde{\mathcal{O}}((\log n)^{12})$
ECPP	$\mathcal{O}((\log n)^6)$	$\tilde{\mathcal{O}}((\log n)^4)$

In fact, Odlyzko and Pomerance have shown that for all large n, the running time is in the interval

$$\left[\mathcal{O}((\log n)^{c_1 \log\log\log n}), \mathcal{O}((\log n)^{c_2 \log\log\log n})\right],$$

where $c_1, c_2 > 0$. This test was further improved by Lenstra and Cohen, hence the name APRCL test. It can be used to test numbers of several thousand digits in less than, say, ten minutes.

The elliptic curve test ECPP of Atkin and Morain, based on earlier work Goldwasser and Kilian, runs in time

$$\begin{aligned}\mathcal{O}((\log n)^{2+2\mu}) &\Rightarrow \mathcal{O}((\log n)^6)\\ &\Rightarrow \mathcal{O}((\log n)^{4+\epsilon})\\ &= \tilde{\mathcal{O}}((\log n)^4).\end{aligned}$$

That is, it runs in $\mathcal{O}((\log n)^6)$ if a practical multiplication algorithm is used and $\tilde{\mathcal{O}}((\log n)^4)$ if a fast multiplication algorithm is employed. ECPP is a probabilistic algorithm but with zero error; other names for this type of probabilistic algorithms are the ZPP algorithm or Las Vegas algorithm.

In what follows, Table 3.1 summarizes the running times for all the different tests mentioned in this section.

Brent [24] at Oxford University (now at the Australian National University) did some numerical experiments for the comparison of the Miller–Rabin, ECPP, and AKS tests on a 1GHz machine for the number $10^{100} + 267$. Table 3.2 gives times for Magama (a computer

Table 3.2 Times for various tests for $10^{100} + 267$

Test	Trials	Time
Miller–Rabin	1	0.003 second
Miller–Rabin	10	0.03 second
Miller–Rabin	100	0.3 second
ECPP		2.0 seconds
Maple Test (Miller–Rabin + Lucas)		2.751 seconds
AKS		37 weeks (Estimated)

algebra system) implementation of of the Miller–Rabin, ECPP, and AKS tests, plus our experiment on a Fujitsu P7230 Laptop computer, all for the number $10^{100} + 267$.

Finally, we wish to mention that there is a practical primality test specifically for Mersenne primes numbers:

Algorithm 3.6 (Lucas–Lehmer test for Mersenne primes)

Initialize the value for $p \in$ Primes
$L \leftarrow 4$
for i from 1 to $p - 2$ do
  $L \leftarrow L^2 - 2 \pmod{(2^p - 1)}$
if $L = 0$ then $2^p - 1$ is prime
  else $2^p - 1$ is composite

Remark 3.13 The above Lucas–Lehmer test for Mersenne primes is very efficient, since the major step in the algorithm is to compute

$$L = L^2 - 2 \pmod{(2^p - 1)}$$

which can be performed in polynomial-time. But still, the computation required to test a single Mersenne prime M_p increases with p to the order of $\mathcal{O}(p^3)$. Thus, to test M_{2r+1} would take approximately eight times as long as to test M_r with the same algorithm (Slowinski [26]). Historically, it has required about four times as much computation to discover the next Mersenne prime as to re-discover all previously known Mersenne primes. The search for Mersenne primes has been an accurate measure of computing power for the past 200 years and, even in the modern era, it has been an accurate measure of computing power for new supercomputers.

Problems for Section 3.4

1. (Binomial Theorem) Prove that

$$(x + y)^n = \sum_{i=0}^{n} \binom{n}{r} x^i y^{n-i}$$

where

$$\binom{n}{r} = \frac{n!}{i!(n-i)!} \quad \text{and} \quad m! = m(m-1)(m-2)\cdots 3 \cdot 2 \cdot 1.$$

2. Prove that

$$(x + y)^n \equiv x^n + y^n \pmod{n}$$

for all variables x and y and primes n.

3. Prove that n is prime if and only if

$$(x+1)^n \equiv x^n + 1 \pmod{n}$$

in $\mathbb{Z}[x]$.

4. Show that if a square root x of 1 modulo n which is neither 1 nor -1, then n is composite.
5. It is easy to find a square modulo an odd prime p, however, it is hard to find a nonsquare modulo an odd prime p. Show that, by the Extended Riemann Hypothesis, there is a nonsquare less than or equal to $2(\log p)^2$.
6. (AKS) Let $n \geq 2$ be an integer, r a positive integer $< p$, for which n has order greater than $(\log n)^2$ modulo r. Then n is prime if and only if
   (1) n is not a perfect power,
   (2) n does not have any prime divisor less than or equal to r,
   (3) $(x+a)^n = x^n + a \pmod{n, x^r - 1}$ for each integer a, $1 \leq a \leq \sqrt{r}\log n$.
7. (Bernstein) Let $f(x)$ be a monic polynomial $f(x) \in \mathbb{Z}[x]$ of degree $d \geq 1$ and n positive integer. $\mathbb{Z}[x]/(n, f(x)$ is called an almostfield with parameters $(e, v(x))$ if
   (1) $e \mid n^d - 1$, with e positive integer,
   (2) $v(x)^{n^d-1} \equiv 1 \pmod{n, f(x)}$,
   (3) $v(x)^{(n^d-1)/q} - 1$ is a unit in $\mathbb{Z}[x]/(n, f(x))$ for all prime $q \mid e$.

   If n is a prime and $f(x)$ modulo n is irreducible, then $\mathbb{Z}[x]/(n, f(x)$ is a field. Moreover, any generator $v(x)$ of the multiplicative group of elements of this field satisfies (2) and (3) for each e satisfying (1).

   Prove that for $n > 2$ and let $\mathbb{Z}[x]/(n, f(x))$ be an almostfield with parameters $(e, v(x))$ where $e > (2d\log n)^2$, then n is prime if and only if
   (a) n is not a perfect power,
   (b) $(t-1)^{n^d} \equiv t^{b^d} \pmod{n, f(x), t^e - v(x)}\text{in}\mathbb{Z}[x,t]$.
8. (Lenstra, 1985) Add one more condition (4) in Problem 7:
   (4) $g(T) := \prod_{i=0}^{d-1} (T - v(x)^{n^i}) \in (\mathbb{Z}[x]/(n, f(x)))[T]$.

   Show that $p \equiv n^j \pmod{e}$ for some j, $0 \leq j \leq d-1$, and for each prime $p \mid n$.
9. (Lenstra–Pomerance) Let $f(x)$ be a monic polynomial $f(x) \in \mathbb{Z}[x]$ of degree $d \geq 1$ and n positive integer. $\mathbb{Z}[x]/(n, f(x)$ is called a pseudofield if
   (1) $f(x^n) \equiv 0 \pmod{n, f(x)}$,
   (2) $x^{n^d} - x \equiv 0 \pmod{n, f(x)}$,
   (3) $x^{n^{d/q}} - x$ is a unit in $\mathbb{Z}[x]/(n, f(x))$ for all prime $q \mid d$.

   Prove that for $n \geq 2$ and let $\mathbb{Z}[x]/(n, f(x))$ be a pseudofield with $f(x)$ a monic polynomial of degree d in $((\log n)^2, n)$, then n is prime if and only if
   (a) n is not a perfect power,
   (b) n does not have any prime divisor less than or equal to d,
   (c) $(x+a)^n \equiv x^n + q \pmod{n, f(x)}$, for each integer a, $1 \leq a \leq \sqrt{d}\log n$.
10. (AKS) Let $n \geq 2$ be an integer, r a positive integer $< p$, for which n has order greater than $(\log n)^2$ modulo r. Then n is prime if and only if
    (1) n is not a perfect power,
    (2) n does not have any prime divisor less than or equal to r,
    (3) $(x+a)^n = x^n + a \pmod{n, x^r - 1}$ for each integer a, $1 \leq a \leq \sqrt{r}\log n$.

11. (Pomerance–Odlyzko) Let $f(n)$ be the least positive square-free integer such that the product of all primes $q, q-1 \mid f(n)$ exceeds $\sqrt{n}$. Show that there exists positive absolute constants c_1, c_2 such that for all integers $n \geq 2$,

$$(\log n)^{c_1 \log\log\log n} < f(n) < (\log n)^{c_2 \log\log\log n}.$$

12. (Adleman–Pomerance–Rumely) Let $T(n)$ be the number of bit operations needed by the APR test. Show that there exists positive absolute constants c_3, c_4 such that

$$(\log n)^{c_3 \log\log\log n} < f(n) < (\log n)^{c_4 \log\log\log n}.$$

13. Prove the following theorem. Let p and q be primes with $p \mid (q-1)$, ζ_{pq} the primitive pqth roots of unity in $\mathbb{C}$, g the generator of $(\mathbb{Q}_q)^*$. Let also χ be a character of $(\mathbb{Q}_q)^*$ with order p. Define the Gauss sum as follows

$$\tau(\chi) = \sum_{a=1}^{q-1} \chi(a)\zeta_q^a \in \mathbb{Z}[\zeta_{pq}].$$

If n is a prime, then

$$\tau(\chi)^{n^{p-1}} \equiv \chi(n) \pmod{n\mathbb{Z}[\zeta_{pq}]}.$$

14. Prove that if

$$\tau(\chi)^{n^{p-1}} \equiv \chi(n) \pmod{n\mathbb{Z}[\zeta_{pq}]}.$$

is true, then for any prime divisor r of n, there is a number b modulo p, such that

$$\chi(n)^b = \chi(r).$$

3.5 Bibliographic Notes and Further Reading

Primality testing is a large topic in computational number theory. Although primality testing problems are now proved to be solvable in polynomial-time, there is still a room for improvement to make it practically efficient. For general references on primality testing, including the historical development of efficient algorithms for primality, see, [8, 13, 17, 19, 27–49].

The original sources for the Miller–Rabin test and many other types of probabilistic primality tests (pseudoprimality tests) can be found in, for example, [1–3, 24, 50–56].

The elliptic curve primality test was first proposed by Goldwasser and Kilian in [5, 10, 57], but the most practical and fastest version of elliptic curve primality test, ECPP, was proposed and improved in [12, 14–16]. Gross [58] discussed the application of elliptic curve test for Mersenne primes.

The AKS deterministic polynomial-time primality test was proposed by Agrawal, Kayal, and N. Saxena in [18]. More discussions and improvements of AKS test may be found in [20, 23, 59, 60]. Prior to AKS, the super-polynomial-time primality test APR was proposed by Adleman, Pomerance, and Rumely in 1983 [61], and subsequently improved by Cohen and Lenstra in 1984 [62].

References

1. G. Miller, "Riemann's Hypothesis and Tests for Primality", *Journal of Systems and Computer Science*, **13**, 1976, pp. 300–317.
2. M. O. Rabin, "Probabilistic Algorithms for Testing Primality", *Journal of Number Theory*, **12**, 1980, pp. 128–138.
3. V. R. Pratt, "Every Prime Has a Succinct Certificate", *SIAM Journal on Computing*, **4**, 1975, pp. 214–220.
4. H. W. Lenstra, Jr., "Factoring Integers with Elliptic Curves", *Annals of Mathematics*, **126**, 1987, pp. 649–673.
5. S. Goldwasser and J. Kilian, "Almost All Primes Can be Quickly Certified", *Proceedings of the 18th ACM Symposium on Theory of Computing*, Berkeley, 1986, pp. 316–329.
6. V. Miller, "Uses of Elliptic Curves in Cryptography", *Lecture Notes in Computer Science*, **218**, Springer, 1986, pp. 417–426.
7. N. Koblitz, "Elliptic Curve Cryptography", *Mathematics of Computation*, **48** 1987, pp. 203–209.
8. D. A. Cox, *Primes of the Form* $x^2 + ny^2$, Wiley, 1989.
9. R. A. Mollin, *Introduction to Cryptography*, 2nd Edition, CRC Press, 2006.
10. S. Goldwasser and J. Kilian, "Primality Testing Using Elliptic Curves", *Journal of ACM*, **46**, 4, 1999, pp. 450–472.
11. R. Schoof, "Elliptic Curves over Finite Fields and the Computation of Square Roots mod p", *Mathematics of Computation*, **44**, 1985, pp. 483–494.
12. A. O. L. Atkin and F. Morain, "Elliptic Curves and Primality Proving", *Mathematics of Computation*, **61**, 1993, pp. 29–68.
13. R. Schoof, "Four Primality Testing Algorithms", *Algorithmic Number Theory*. Edited by J. P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 69–81.
14. F. Morain, *Courbes Elliptiques et Tests de Primalité*, Université Claude Bernard, Lyon I, 1990.
15. F. Morain, "Primality Proving Using Elliptic Curves: An Update", *Algorithmic Number Theory*, Lecture Notes in Computer Science **1423**, Springer, 1998, pp. 111–127.
16. F. Morain, "Implementing the Asymptotically Fast Version of the Elliptic Curve Primality Proving Algorithm", *Mathematics of Computation*, **76**, 2007, pp. 493–505.
17. L. M. Adleman and M. D. A. Huang, *Primality Testing and Abelian Varieties over Finite Fields*, Lecture Notes in Mathematics **1512**, Springer-Verlag, 1992.
18. M. Agrawal, N. Kayal, and N. Saxena, "Primes is in P", *Annals of Mathematics*, **160**, 2, 2004, pp. 781–793.
19. J. D. Dixon, "Factorization and Primality tests", *The American Mathematical Monthly*, June–July 1984, pp. 333–352.
20. D. J. Bernstein, *Proving Primality After Agrawal-Kayal-Saxena*, Dept of Mathematics, Statistics and Computer Science, The University of Illinois at Chicago, 25 Jan 2003.
21. E. Fouvry, "Theorème de Brun-Titchmarsh: Application au Theorème de Fermat", *ventiones Mathematicae*, **79**, 1985, pp. 383–407.
22. R. C. Baker and G. Harman, "The Brun-Tichmarsh Theorem on Average", in *Proceedings of a Conference in Honor of Heini Halberstam*, Volume 1, 1996, pp. 39–103.
23. R. Bhattacharjee and P. Pandey, *Primality Testing*, Dept of Computer Science & Engineering, Indian Institute of Technology Kanpur, India, 2001.
24. R. P. Brent, "Uncertainty Can Be Better Than certainty: Some Algorithms for Primality Testing", Mathematical Institute, Australian National University, 2006.
25. A. K. Lenstra and H. W. Lenstra, Jr. (eds), *The Development of the Number Field Sieve*, Lecture Notes in Mathematics **1554**, Springer-Verlag, 1993.

26. D. Slowinski, "Searching for the 27th Mersenne Prime", *Journal of Recreational Mathematics*, **11**, 4, 1978–79, pp. 258–261.
27. L. M. Adleman, "Algorithmic Number Theory – The Complexity Contribution", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1994, pp. 88–113.
28. E. Bach and J. Shallit, *Algorithmic Number Theory I – Efficient Algorithms*, MIT Press, 1996.
29. R. P. Brent, "Primality Testing and Integer Factorization", in *Proceedings of Australian Academy of Science Annual General Meeting Symposium on the Role of Mathematics in Science*, Canberra, 1991, pp. 14–26.
30. D. M. Bressound, *Factorization and Primality Testing*, Springer, 1989.
31. H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer-Verlag, 1993.
32. T. H. Cormen, C. E. Ceiserson, and R. L. Rivest, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009.
33. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer-Verlag, 2005.
34. J. H. Davenport, *Primality Testing Revisited*, School of Mathematical Sciences, University of Bath, UK, 1992.
35. C. F. Gauss, *Disquisitiones Arithmeticae*, G. Fleischer, Leipzig, 1801. English translation by A. A. Clarke, Yale University Press, 1966. Revised English translation by W. C. Waterhouse, Springer-Verlag, 1975.
36. D. E. Knuth, *The Art of Computer Programming II – Seminumerical Algorithms*, 3rd Edition, Addison-Wesley, 1998.
37. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer-Verlag, 1994.
38. E. Kranakis, *Primality and Cryptography*, Wiley, 1986.
39. H. W. Lenstra, Jr., *Primality Testing Algorithms*, Séminaire N. Bourbaki 1980–1981, Number 561, pp. 243–257. Lecture Notes in Mathematics **901**, Springer, 1981.
40. J. M. Pollard, "Theorems on Factorization and Primality Testing", *Proceedings of Cambridge Philosophy Society*, **76**, 1974, pp. 521–528.
41. C. Pomerance, "Very Short Primality Proofs", *Mathematics of Computation*, **48**, 1987, pp. 315–322.
42. C. Pomerance, " Computational Number Theory", In: *Princeton Companion to Mathematics*, W. T. Gowers, (ed.) Princeton University Press, 2008, pp. 348–362.
43. H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, Boston, 1990.
44. K. Rosen, *Elementary Number Theory and its Applications*, 5th Edition, Addison-Wesley, 2005.
45. R. Rumely, "Recent Advances in Primality Testing", *Notices of the AMS*, **30**, 8, 1983, pp. 475–477.
46. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
47. S. Wagon, "Primality Testing", *The Mathematical Intelligencer*, **8**, 3, 1986, pp. 58–61.
48. H. S. Wilf, *Algorithms and Complexity*, 2nd Edition, A. K. Peters, 2002.
49. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.
50. W. Alford, G. Granville, and C. Pomerance, "There Are Infinitely Many Carmichael Numbers", *Annals of Mathematics*, **140**, 1994, pp. 703–722.
51. A. Granville, "Primality Testing and Carmichael Numbers", *Notice of the American Mathematical Society*, **Vol 39**, 1992, pp. 696–700.
52. C. Pomerance, "Primality Testing: Variations on a Theme of Lucas", in *Proceedings of the 13th Meeting of the Fibonacci Association, Congressus Numerantium*, **201**, 2010, pp. 301–312.
53. C. Pomerance, J. L. Selfridge and S. S. Wagstaff, Jr., "The Pseudoprimes to $25 \cdot 10^9$", *Mathematics of Computation*, **35**, 1980, pp. 1003–1026.
54. P. Ribenboim, "Selling Primes", *Mathematics Magazine*, **68**, 3, 1995, pp. 175–182.
55. R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality", *SIAM Journal on Computing*, **6**, 1, 1977, pp. 84–85. "Erratum: A Fast Monte-Carlo Test for Primality", *SIAM Journal on Computing*, **7**, 1, 1978, p. 118.
56. H. C. Williams, *Édouard Lucas and Primality Testing*, John Wiley & Sons, 1998.
57. J. Kilian, *Uses of Randomness in Algorithms and Protocols*, MIT Press, 1990.
58. B. H. Gross, "An Elliptic Curve Test for Mersenne Primes", *Journal of Number Theory*, **110**, 2005, pp. 114–119.
59. A. Granville, "It is Easy to Determine Whether a Given Integer is Prime", *Bulletin (New Series) of the American Mathematical Society*, **Vol 42**, 1, 2004, pp. 3–38.

60. H. W. Lenstra, Jr., and C. Pomerance, "Primality Testing with Gaussian Periods", Department of Mathematics, Dartmouth College, Hanover, NH 03874, USA, 12 April 2011.
61. L. M. Adleman, C. Pomerance, and R. S. Rumely, "On Distinguishing Prime Numbers from Composite Numbers", *Annals of Mathematics*, **117**, 1, 1983, pp. 173–206.
62. H. Cohen and H. W. Lenstra, Jr., "Primality Testing and Jacobi Sums, *Mathematics of Computation*, **42**, 165, 1984, pp. 297–330.

4

Integer Factorization

The Integer Factorization Problem (IFP) is one of the most important infeasible problems in computational number theory, for which there is still no polynomial-time algorithm for solving it. In this chapter we discuss several important modern algorithms for factoring, including

- Pollard's ρ and $p-1$ Methods;
- Lenstra's Elliptic Curve Method (ECM);
- Pomerance's Quadratic Sieve (QS), and
- The currently fastest method, Number Field Sieve (NFS).

4.1 Basic Concepts

The Integer Factorization Problem (IFP) may be described as follows:

$$\text{IFP} := \begin{cases} \text{Input}: & n \in \text{Composites} \\ \text{Output}: & f \text{ such that } f \mid n \ \& \ 1 < f < n \end{cases} \tag{4.1}$$

Clearly, if there is an algorithm to test whether or not an integer n is a prime, and an algorithm to find a nontrivial factor f of a composite integer n, then by recursively calling the primality testing algorithm and the integer factorization algorithm, it should be easy to find the prime factorization of

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}.$$

Generally speaking, the most useful factoring algorithms fall into one of the following two main classes:

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

(A) General purpose factoring algorithms, which has running time that depends mainly on the size of N, the number to be factored, and is not strongly dependent on the size of the factor p found. Examples are:
   (1) *Lehman's method*, which has a rigorous worst-case running time bound $\mathcal{O}\left(n^{1/3+\epsilon}\right)$.
   (2) *Shanks' SQUare FOrm Factorization method*SQUFOF, which has expected running time $\mathcal{O}\left(n^{1/4}\right)$.
   (3) *Shanks' class group method*, which has running time $\mathcal{O}\left(n^{1/5+\epsilon}\right)$.
   (4) *Continued FRACtion (*CFRAC*) method*, which under plausible assumptions has expected running time

$$\mathcal{O}\left(\exp\left(c\sqrt{\log n \log\log n}\right)\right) = \mathcal{O}\left(n^{c\sqrt{\log\log n/\log n}}\right),$$

   where c is a constant (depending on the details of the algorithm); usually $c = \sqrt{2} \approx 1.414213562$.
   (5) *Multiple Polynomial Quadratic Sieve (*MPQS*)*, which under plausible assumptions has expected running time

$$\mathcal{O}\left(\exp\left(c\sqrt{\log n \log\log n}\right)\right) = \mathcal{O}\left(n^{c\sqrt{\log\log n/\log n}}\right),$$

   where c is a constant (depending on the details of the algorithm); usually $c = \dfrac{3}{2\sqrt{2}} \approx 1.060660172$.
   (6) *Number Field Sieve (*NFS*)*, which under plausible assumptions has the expected running time

$$\mathcal{O}\left(\exp\left(c\sqrt[3]{\log n}\sqrt[3]{(\log\log n)^2}\right)\right),$$

   where $c = (64/9)^{1/3} \approx 1.922999427$ if GNFS (a general version of NFS) is used to factor an arbitrary integer n, whereas $c = (32/9)^{1/3} \approx 1.526285657$ if SNFS (a special version of NFS) is used to factor a special integer n such as $n = r^e \pm s$, where r and s are small, $r > 1$ and e is large. This is substantially and asymptotically faster than any other currently known factoring method.

(B) Special purpose factoring algorithms: The running time depends mainly on the size of p (the factor found) of n. (We can assume that $p \leq \sqrt{n}$.) Examples are:
   (1) *Trial division*, which has running time $\mathcal{O}\left(p(\log n)^2\right)$.
   (2) *Pollard's ρ Method* (also known as Pollard's "*rho*" algorithm), which under plausible assumptions has expected running time $\mathcal{O}\left(p^{1/2}(\log n)^2\right)$.
   (3) *Lenstra's Elliptic Curve Method (*ECM*)*, which under plausible assumptions has expected running time

$$\mathcal{O}\left(\exp\left(c\sqrt{\log p \log\log p}\right)\cdot(\log n)^2\right),$$

   where $c \approx 2$ is a constant (depending on the details of the algorithm).

The term $\mathcal{O}((\log n)^2)$ is for the cost of performing arithmetic operations on numbers which are $\mathcal{O}(\log n)$ or $\mathcal{O}\left((\log n)^2\right)$ bits long; the second can be theoretically replaced by $\mathcal{O}((\log n)^{1+\epsilon})$ for any $\epsilon > 0$.

Note that there is a quantum factoring algorithm, first proposed by Shor, which can run in polynomial-time

$$\mathcal{O}((\log n)^{2+\epsilon}).$$

However, this quantum algorithm requires running on a practical quantum computer with several thousand quantum bits, which is not available at present.

In practice, algorithms in both categories are important. It is sometimes very difficult to say whether one method is better than another, but it is generally worth attempting to find small factors with algorithms in the second class before using the algorithms in the first class. That is, we could first try the *trial division algorithm*, then use some other method such as NFS. This fact shows that the trial division method is still useful for integer factorization, even though it is simple. In this chapter we shall introduce some most the useful and widely used factoring algorithms.

From a computational complexity point of view, the IFP is an infeasible (intractable) problem, since there is no polynomial-time algorithm for solving it; all the existing algorithms for IFP run in subexponential-time or above (see Figure 4.1).

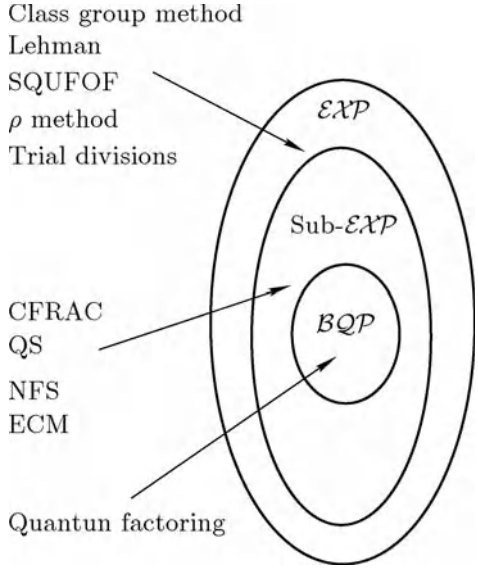


Figure 4.1 Algorithms/Methods for IFP

Problems for Section 4.1

1. (Sierpinski) Prove that there are infinitely many integers k such that

$$n = k \cdot 2^n + 1, \quad n = 1, 2, 3, \ldots$$

   are all composite. Find three such composites.
2. Show that the sequence $2^{2^n} + 3$, $n = 1, 2, 3, \ldots$ contains infinitely many composites.
3. Suppose $n = pq$ with p, q prime and $p < q < 2p$. Let δ be the number defined by $q/p = 1 + \delta$ such that $0 < \delta < 1$. Show that the number of steps used in the Fermat factoring method is approximately $p\delta^2/8$.
4. Let $n = pq = 18886013$ such that

$$\left| \frac{p}{q} - 3 \right| < \frac{1}{100}.$$

   Find the factorization of n.
5. Explain why general purpose factoring algorithms are slower than special purpose factoring algorithms, or why the special numbers are easy to factor than general numbers.
6. Show that addition of two $\log n$ bit integers can be performed in $\mathcal{O}(\log n)$ bit operations.
7. Show that multiplication of two $\log n$ bit integers can be performed in $\mathcal{O}((\log n)^2)$ bit operations.
8. Show that there is an algorithm which can multiply two $\log n$ bit integers in

$$\mathcal{O}(\log n \log\log n \log\log\log n) = \mathcal{O}((\log n)^{1+\epsilon})$$

   bit operations.
9. Show that if $\mathcal{P} = \mathcal{NP}$, then IFP $\in \mathcal{P}$.
10. Prove or disprove that IFP $\in \mathcal{NP}$-complete.

4.2 Trial Divisions Factoring

The simplest factoring algorithm is perhaps the trial division method, which tries all the possible divisors of n to obtain its complete prime factorization:

$$n = p_1 p_2 \cdots p_t, \qquad p_1 \le p_2 \le \cdots \le p_t. \tag{4.2}$$

The following is the algorithm:

Algorithm 4.1 (Factoring by trial divisions) This algorithm tries to factor an integer $n > 1$ using trial divisions by all the possible divisors of n.

[1] [Initialization] Input n and set $t \leftarrow 0$, $k \leftarrow 2$.
[2] [$n = 1$?] If $n = 1$, then go to [5].

[3] [Compute Remainder]

$q \leftarrow n/k$ and $r \leftarrow n \pmod{k}$.
If $r \neq 0$, go to [4].
$t \leftarrow t + 1$, $p_t \leftarrow k$, $n \leftarrow q$, go to [2].

[4] [Factor Found?]

If $q > k$, then $k \leftarrow k + 1$, go to [3].
$t \leftarrow t + 1$; $p_t \leftarrow n$.

[5] [Exit] Terminate the algorithm.

Exercise 4.1 Use Algorithm 4.1 to factor $n = 2759$.

An immediate improvement of Algorithm 4.1 is to make use of an auxiliary sequence of *trial divisors*:

$$2 = d_0 < d_1 < d_2 < d_3 < \cdots \tag{4.3}$$

which includes all primes $\leq \sqrt{n}$ (possibly some composites as well if it is convenient to do so) and at least one value $d_k \geq \sqrt{n}$. The algorithm can be described as follows:

Algorithm 4.2 (Factoring by Trial Division) This algorithm tries to factor an integer $n > 1$ using trial divisions by an auxiliary sequence of trial divisors.

[1] [Initialization] Input n and set $t \leftarrow 0$, $k \leftarrow 0$.
[2] [$n = 1$?] If $n = 1$, then go to [5].
[3] [Compute Remainder]

$q \leftarrow n/d_k$ and $r \leftarrow n \pmod{d_k}$.
If $r \neq 0$, go to [4].
$t \leftarrow t + 1$, $p_t \leftarrow d_k$, $n \leftarrow q$, go to [2].

[4] [Factor Found?]

If $q > d_k$, then $k \leftarrow k + 1$, and go to [3].
$t \leftarrow t + 1$; $p_t \leftarrow n$.

[5] Exit: terminate the algorithm.

Exercise 4.2 Use Algorithm 4.2 to factor $n = 2759$; assume that we have the list L of all primes $\leq \lfloor\sqrt{2759}\rfloor = 52$ and at least one $\geq \sqrt{n}$, that is, $L = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53\}$.

Theorem 4.1 Algorithm 4.2 requires a running time in

$$\mathcal{O}\left(\max\left(p_{t-1},\ \sqrt{p_t}\right)\right).$$

If a primality test between steps [2] and [3] were inserted, the running time would then be in $\mathcal{O}(p_{t-1})$, or $\mathcal{O}(\frac{p_{t-1}}{\ln p_{t-1}})$ if one does trial division only by primes, where p_{t-1} is the second largest prime factor of n.

The trial division test is very useful for removing small factors, but it should not be used for factoring completely, except when n is very small, say, for example, $n < 10^8$.

Algorithm 4.2 seems to be very simple, but its analysis is rather complicated. For example, in order to analyse the average behavior of the algorithm, we will need to know how many trial divisions are necessary and how large the largest prime factor p_t will tend to be.

Let $\pi(x)$ be the primes $\leq x$. Then, if the Riemann Hypothesis is true, it would imply that

$$\pi(x) = L(x) + \mathcal{O}(\sqrt{x}\log x) \tag{4.4}$$

where $L(x) = \int_2^x dt/\ln t$. In 1930, Karl Dickman investigated the probability that a random integer between 1 and x will have its largest prime factor $\leq x^\alpha$. He gave a heuristic argument to show that this probability approaches the limiting value $F(\alpha)$ as $x \to \infty$, where F can be calculated from the functional equation

$$F(\alpha) = \int_0^\alpha F\left(\frac{t}{1-t}\right)\frac{\mathrm{d}t}{t}, \quad \text{for } 0 \leq \alpha \leq 1; \quad F(\alpha) = 1 \quad \text{for } \alpha \geq 1. \tag{4.5}$$

His argument was essentially this: Given $0 < t < 1$, the number of integers less than x whose largest prime factor is between x^t and $x^{t+\mathrm{d}t}$ is $xF'(t)\mathrm{d}t$. The number of prime p in that range is

$$\pi(x^{t+\mathrm{d}t}) - \pi(x^t) = \pi(x^t + (\ln x)x^t\mathrm{d}t) - \pi(x^t) = x^t\mathrm{d}t/t. \tag{4.6}$$

For every such p, the number of integers n such that "$np \leq x$ and the largest prime factor of n is $\leq p$" is the number of $n \leq x^{1-t}$ whose largest prime factor is $\leq (x^{1-t})^{t/(1-t)}$, namely $x^{1-t}F(t/(1-t))$. Hence $xF'(t)\mathrm{d}t = (x^t\mathrm{d}t/t)(x^{1-t}F(t/(1-t)))$ and (4.5) follows by integration.

If $\frac{1}{2} \leq \alpha \leq 1$, formula (4.5) simplifies to

$$F(\alpha) = 1 - \int_\alpha^1 F\left(\frac{t}{1-t}\right)\frac{\mathrm{d}t}{t} = 1 - \int_\alpha^1 \frac{\mathrm{d}t}{t} = 1 + \ln\alpha. \tag{4.7}$$

Thus, for example, the probability that a random positive integer $\leq x$ has a prime factor $> \sqrt{x}$ is $1 - F(\frac{1}{2}) = \ln 2 \approx 0.6931$, In all such cases, Algorithm 4.2 must work hard. So, practically, Algorithm 4.2 will give the answer rather quickly if we want to factor a 6-digit integer; but for large n, the amount of computer time for factoring by trial division will

rapidly exceed practical limits, unless we are unusually lucky. The algorithm usually finds a few small factors and then begins a long-drawn-out search for the large ones that are left. Interested readers are referred to [4] for more detailed discussion.

By an argument analogous to Dickman's, it has been shown that the second largest prime factor of a random integer x will be $\le x^\beta$ with approximate probability $G(\beta)$, where

$$G(\beta) = \int_0^\beta \left(G\left(\frac{t}{1-t}\right) - F\left(\frac{t}{1-t}\right)\right)\frac{\mathrm{d}t}{t}, \quad \text{for } 0 \le \beta \le \frac{1}{2}. \tag{4.8}$$

Clearly, $G(\beta) = 1$ for $\beta \ge \frac{1}{2}$.

The total number of prime factors, t is obviously $1 \le t \le \log N$, but these lower and upper bounds are seldom achieved. It is possible to prove that if n is chosen at random between 1 and x, the probability that $t \le \ln\ln x + c\sqrt{\ln\ln x}$ approaches

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c} e^{-u^2/2}\mathrm{d}u \tag{4.9}$$

as $x \to \infty$, for any fixed c. In other words, the distribution of t is essentially normal, with mean and variance $\ln\ln x$; about 99.73 % of all the large integers $\le x$ have $|t - \ln\ln x| \le 3\sqrt{\ln\ln x}$. Furthermore, the average value of $t - \ln\ln x$ for $1 \le n \le x$ is known to approach

$$\gamma + \sum_{p \text{ prime}} (\ln(1 - 1/p) + (1/(p-1)) = 1.034653881897438. \tag{4.10}$$

It follows from the above analysis that Algorithm 4.2 will be useful for factoring a 6-digit integer, or a large integer with small prime factors. Fortunately, many more efficient factorization algorithms have been developed since 1980, requiring fewer bit operations than the trial division method; with the fastest being the Number Field Sieve (NFS), which will be discussed later.

Problems for Subsection 4.2

1. (Hardy and Littlewood, 1918) Show that there is a positive real constant c such that

$$\pi(x) > \int_2^x \frac{\mathrm{d}t}{\log t} + c\sqrt{x}\frac{\log\log\log x}{\log x}$$

   for infinitely many x.
2. Let p_t be the largest prime factor of a random integer n between 1 and x. What is the probability that n has its largest prime factor $\le \sqrt{x}$?
3. Let p_{t-1} be the second largest prime factor of a random integer n between 1 and x. What is the probability that n has its second largest prime factor $\le x^{0.217}$?
4. Let t be the total number of prime factors of n between 1 and x. What is the probability that $t \le \log\log x + c\sqrt{\log\log x}$ for any fixed c, as $x \to \infty$?

5. Suppose $n = pq$ with p, q prime and $p < q < 2p$. Let δ be the number defined by $q/p = 1 + \delta$ such that $0 < \delta < 1$. Show that the number of steps used in the Fermat factoring method is approximately $p\delta^2/8$.
6. Let p be an od prime and $n \neq kp$ for $k = 1, 2, 3, \ldots$. Show that the number of solutions of (x, y) to $x^2 - n \equiv y^2 \pmod{p}$ with $0 \leq x < p$ is equal to $(p \pm 1)/2$.
7. Let n be an odd integer. Show that if n can be factored as

$$n = ab, \quad 1 \leq a \leq b,$$

then n can be written as a difference of the two squares x^2 and y^2, where

$$x = \frac{b+a}{2}, \quad y = \frac{b-a}{2}.$$

8. The Fermat factoring method is based on the fact that if n is a difference of the two squares such as x^2 and y^2, then n can be factored as $n = (x - y)(x + y)$. Use the Fermat factoring method to factor the numbers 1254713 and 2027651281.
9. Show that the Fermat factoring method is of the complexity $\mathcal{O}(n^{1/2})$.

4.3 ρ and $p - 1$ Methods

In 1975 John M. Pollard proposed a very efficient Monte Carlo method [2], now widely known as Pollard's "*rho*" or ρ Method, for finding a small nontrivial factor d of a large integer n. Trial division by all integers up to $\sqrt{n}$ is guaranteed to factor completely any number up to N. For the same amount of work, Pollard's "*rho*" Method will factor any number up to n^2 (unless we are unlucky).

The method uses an iteration of the form

$$\left.\begin{aligned} x_0 &= \text{random}(0,\ n-1), \\ x_i &\equiv f(x_{i-1}) \pmod{n}, \qquad i = 1, 2, 3, \ldots \end{aligned}\right\} \tag{4.11}$$

where x_0 is a random starting value, n is the number to be factored, and $f \in \mathbb{Z}[x]$ is a polynomial with integer coefficients; usually, we just simply choose $f(x) = x^2 \pm a$ with $a \neq -2, 0$. Then starting with some initial value x_0, a "random" sequence $x_1, x_2, x_3, \ldots$ is computed modulo n in the following way:

$$\left.\begin{aligned} x_1 &= f(x_0), \\ x_2 &= f(f(x_0)) = f(x_1), \\ x_3 &= f(f(f(x_0))) = f(f(x_1)) = f(x_2), \\ &\vdots \\ x_i &= f(x_{i-1}). \end{aligned}\right\} \tag{4.12}$$

Let d be a nontrivial divisor of n, where d is small compared with n. Since there are relatively few congruence classes modulo d (namely, d of them), there will probably exist integers

x_i and x_j which lie in the same congruence class modulo d, but belong to different classes modulo N; in short, we will have

$$\left.\begin{aligned} x_i &\equiv x_j \pmod{d}, \\ x_i &\not\equiv x_j \pmod{n}. \end{aligned}\right\} \tag{4.13}$$

Since $d \mid (x_i - x_j)$ and $n \nmid (x_i - x_j)$, it follows that $\gcd(x_i - x_j,\ n)$ is a nontrivial factor of n. In practice, a divisor d of n is not known in advance, but it can most likely be detected by keeping track of the integers x_i, which we do know; we simply compare x_i with the earlier x_j, calculating $\gcd(x_i - x_j,\ n)$ until a nontrivial gcd occurs. The divisor obtained in this way is not necessarily the smallest factor of n and indeed it may not be prime. The possibility exists that when a gcd greater that 1 is found, it may also turn out to be equal to n itself, though this happens very rarely.

Example 4.1 For example, let $n = 1387 = 19 \cdot 73$, $f(x) = x^2 - 1$ and $x_1 = 2$. Then the "random" sequence $x_1, x_2, x_3, \ldots$ is as follows:

$$2, 3, 8, 63, 1194, \overline{1186, 177, 814, 996, 310, 396, 84, 120, 529, 1053, 595, 339}$$

where the repeated values are overlined. Now we find that

$$\begin{aligned} x_3 &\equiv 6 \pmod{19} \\ x_3 &\equiv 63 \pmod{1387} \\ x_4 &\equiv 16 \pmod{19} \\ x_4 &\equiv 1194 \pmod{1387} \\ x_5 &\equiv 8 \pmod{19} \\ x_5 &\equiv 1186 \pmod{1387} \\ &\vdots \end{aligned}$$

So we have

$$\gcd(63 - 6, 1387) = \gcd(1194 - 16, 1387) = \gcd(1186 - 8, 1387) = \cdots = 19.$$

Of course, as mentioned earlier, d is not known in advance, but we can keep track of the integers x_i which we do know, and simply compare x_i with all the previous x_j with $j < i$, calculating $\gcd(x_i - x_j,\ n)$ until a nontrivial gcd occurs:

$$\begin{aligned} \gcd(x_1 - x_0,\ n) &= \gcd(3 - 2,\ 1387) = 1, \\ \gcd(x_2 - x_1,\ n) &= \gcd(8 - 3,\ 1387) = 1, \\ \gcd(x_2 - x_0,\ n) &= \gcd(8 - 2,\ 1387) = 1, \\ \gcd(x_3 - x_2,\ n) &= \gcd(63 - 8,\ 1387) = 1, \\ \gcd(x_3 - x_1,\ n) &= \gcd(63 - 3,\ 1387) = 1, \\ \gcd(x_3 - x_0,\ n) &= \gcd(63 - 2,\ 1387) = 1, \\ \gcd(x_4 - x_3,\ n) &= \gcd(1194 - 63,\ 1387) = 1, \end{aligned}$$

$$\begin{aligned}
&\gcd(x_4 - x_2,\ n) = \gcd(1194 - 8,\ 1387) = 1,\\
&\gcd(x_4 - x_1,\ n) = \gcd(1194 - 3,\ 1387) = 1,\\
&\gcd(x_4 - x_0,\ n) = \gcd(1194 - 2,\ 1387) = 1,\\
&\gcd(x_5 - x_4,\ n) = \gcd(1186 - 1194,\ 1387) = 1,\\
&\gcd(x_5 - x_3,\ n) = \gcd(1186 - 63,\ 1387) = 1,\\
&\gcd(x_5 - x_2,\ n) = \gcd(1186 - 8,\ 1387) = 19.
\end{aligned}$$

So after 13 comparisons and calculations, we eventually find the divisor 19.

As k increases, the task of computing $\gcd(x_i - x_j,\ n)$ for all $j < i$ becomes very time-consuming; for $n = 10^{50}$, the computation of $\gcd(x_i - x_j,\ n)$ would require about $1.5 \cdot 10^6$ bit operations, as the complexity for computing one gcd is $\mathcal{O}((\log n)^3)$. Pollard actually used Floyd's method to detect a cycle in a long sequence $\langle x_i \rangle$, which just looks at cases in which $x_i = x_{2i}$. To see how it works, suppose that $x_i \equiv x_j \pmod{n}$, then

$$\left.\begin{aligned}
x_{i+1} &\equiv f(x_i) \equiv f(x_j) \equiv x_{j+1} \pmod{d},\\
x_{i+2} &\equiv f(x_{i+1}) \equiv f(x_{j+1}) \equiv x_{j+2} \pmod{d},\\
&\vdots\\
x_{i+k} &\equiv f(x_{i+k-1}) \equiv f(x_{j+k-1}) \equiv x_{j+k} \pmod{d}.
\end{aligned}\right\} \tag{4.14}$$

If $k = j - i$, then $x_{2i} \equiv x_i \pmod{d}$. Hence, we only need look at $x_{2i} - x_i$ (or $x_i - x_{2i}$) for $i = 1, 2, \ldots$. That is, we only need to check one gcd for each i. Note that the sequence $x_0, x_1, x_2, \ldots$ modulo a prime number p, say, looks like a circle with a tail; it is from this behavior that the method gets its name (see Figure 4.2 for a graphical sketch; it looks like the Greek letter ρ).

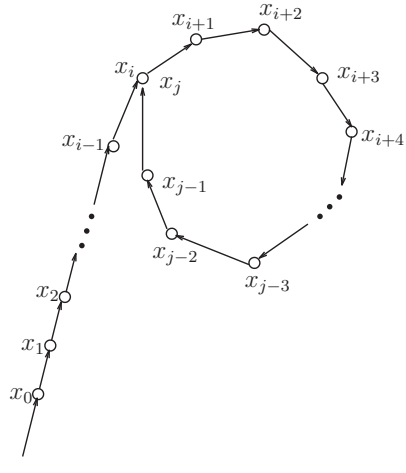


Figure 4.2 Illustration of the ρ Method

Example 4.2 Again, let $n = 1387 = 19 \cdot 73$, $f(x) = x^2 - 1$, and $x_1 = 2$. By comparing pairs x_i and x_{2i}, for $i = 1, 2, \ldots$, we have:

$$\begin{aligned}
&\gcd(x_1 - x_2,\ n) = \gcd(3 - 8,\ 1387) = 1,\\
&\gcd(x_2 - x_4,\ n) = \gcd(8 - 1194,\ 1387) = 1,\\
&\gcd(x_3 - x_6,\ n) = \gcd(63 - 177,\ 1387) = 19.
\end{aligned}$$

So after only three comparisons and gcd calculations, the divisor 19 of 1387 is found.

In what follows, we shall show that to compute $y_i = x_{2i}$, we do not need to compute $x_{i+1}, x_{i+2}, \ldots, x_{2i-1}$ until we get x_{2i}. Observe that

$$\left.\begin{aligned}
y_1 &= x_2 = f(x_1) = f(f(x_0)) = f(f(y_0)),\\
y_2 &= x_4 = f(x_3) = f(f(x_2)) = f(f(y_1)),\\
y_3 &= x_6 = f(x_5) = f(f(x_4)) = f(f(y_2)),\\
&\vdots\\
y_i &= x_{2i} = f(f(y_{i-1})).
\end{aligned}\right\} \tag{4.15}$$

So at each step, we compute

$$\left.\begin{aligned}
x_i &= f(x_{i-1}) \pmod{n},\\
y_i &= f(f(y_{i-1}) \pmod{n}.
\end{aligned}\right\} \tag{4.16}$$

Therefore, only three evaluations of f will be required.

Example 4.3 Let once again $n = 1387 = 19 \cdot 73$, $f(x) = x^2 - 1$, and $x_0 = y_0 = 2$. By comparing pairs x_i and x_{2i}, for $i = 1, 2, \ldots$, we get:

$$\begin{aligned}
&f(y_0) = 2^2 - 1 = 3,\\
&f(f(y_0)) = 3^2 - 1 = 8 = y_1\\
&\quad \Longrightarrow \gcd(y_1 - x_1,\ N) = \gcd(3 - 8,\ 1387) = 1\\
&f(y_1) = 8^2 - 1 = 63,\\
&f(f(y_1)) = 63^2 - 1 = 1194 = y_2\\
&\quad \Longrightarrow \gcd(y_2 - x_2,\ N) = \gcd(8 - 1194,\ 1387) = 1\\
&f(y_2) = 1194^2 - 1 \bmod 1387 = 1186,\\
&f(f(y_2)) = 1186^2 - 1 \bmod 1387 = 177 = y_3\\
&\quad \Longrightarrow \gcd(y_3 - x_3,\ N) = \gcd(63 - 177,\ 1387) = 19.
\end{aligned}$$

The divisor 19 of 1387 is then found.

Remark 4.1 There is an even more efficient algorithm, due to Richard Brent [3], that looks only at the following differences and the corresponding gcd results:

$$\begin{aligned}
x_1 - x_3 &\Longrightarrow \gcd(x_1 - x_3,\ n),\\
x_3 - x_6 &\Longrightarrow \gcd(x_3 - x_6,\ n),\\
x_3 - x_7 &\Longrightarrow \gcd(x_3 - x_7,\ n),\\
x_7 - x_{12} &\Longrightarrow \gcd(x_7 - x_{12},\ n),\\
x_7 - x_{13} &\Longrightarrow \gcd(x_7 - x_{13},\ n),\\
x_7 - x_{14} &\Longrightarrow \gcd(x_7 - x_{14},\ n),\\
x_7 - x_{15} &\Longrightarrow \gcd(x_7 - x_{15},\ n),\\
&\vdots
\end{aligned}$$

and in general:

$$x_{2^n-1} - x_j, \quad 2^{n+1} - 2^{n-1} \le j \le 2^{n+1} - 1. \tag{4.17}$$

Brent's algorithm is about 24% faster than Pollard's original version.

Now we are in a position to present an algorithm for the ρ Method.

Algorithm 4.3 (Brent–Pollard's ρ Method) Let n be a composite integer greater than 1. This algorithm tries to find a nontrivial factor d of n, which is small compared to $\sqrt{n}$. Suppose the polynomial to use is $f(x) = x^2 + 1$.

[1] (Initialization) Choose a seed, say $x_0 = 2$, a generating function, say $f(x) = x^2 + 1 \pmod{n}$. Choose also a value for t not much bigger than $\sqrt{d}$, perhaps $t < 100\sqrt{d}$.
[2] (Iteration and Computation) Compute x_i and y_i in the following way:

$$\begin{aligned}
x_1 &= f(x_0),\\
x_2 &= f(f(x_0)) = f(x_1),\\
x_3 &= f(f(f(x_0))) = f(f(x_1)) = f(x_2),\\
&\vdots\\
x_i &= f(x_{i-1}).
\end{aligned}$$

$$\begin{aligned}
y_1 &= x_2 = f(x_1) = f(f(x_0)) = f(f(y_0)),\\
y_2 &= x_4 = f(x_3) = f(f(x_2)) = f(f(y_1)),\\
y_3 &= x_6 = f(x_5) = f(f(x_4)) = f(f(y_2)),\\
&\vdots\\
y_i &= x_{2i} = f(f(y_{i-1})).
\end{aligned}$$

and simultaneously compare x_i and y_i by computing $d = \gcd(x_i - y_i,\ n)$.

[3] (Factor Found?) If $1 < d < n$, then d is a nontrivial factor of n, print d, and go to Step [5].
[4] (Another Search?) If $x_i = y_i \pmod{n}$ for some i or $i \geq \sqrt{t}$, then go to Step [1] to choose a new seed and a new generator and repeat.
[5] (Exit) Terminate the algorithm.

Example 4.4 The 8th Fermat number $F_8 = 2^{2^8} + 1$ was factored by Brent and Pollard in 1980 by using Brent–Pollard's "rho" Method:

$$2^{2^8} + 1 = 2^{256} + 1 = 1238926361552897 \cdot p_{63}.$$

Now let us move to consider the complexity of the ρ Method. Let p be the smallest prime factor of N, and j the smallest positive index such that $x_{2j} \equiv x_j \pmod{p}$. Making some plausible assumptions, it is easy to show that the expected value of j is $\mathcal{O}(\sqrt{p})$. The argument is related to the well-known "birthday" paradox: Suppose that $1 \leq k \leq n$ and that the numbers $x_1, x_2, \ldots, x_k$ are independently chosen from the set $\{1, 2, \ldots, n\}$. Then the probability that the numbers x_k are distinct is

$$\left(1 - \frac{1}{n}\right) \cdot \left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right) \sim \exp\left(\frac{-k^2}{2n}\right). \tag{4.18}$$

Note that the x_i's are likely to be distinct if k is small compared with $\sqrt{n}$, but unlikely to be distinct if k is large compared with $\sqrt{n}$. Of course, we cannot work out $x_i \bmod p$, since we do not know p in advance, but we can detect x_j by taking greatest common divisors. We simply compute $d = \gcd(x_{2i} - x_i, n)$ for $i = 1, 2, \ldots$ and stop when a $d > 1$ is found.

Conjecture 4.1 (Complexity of the ρ Method) Let p be a prime dividing n and $p = \mathcal{O}(\sqrt{p})$, then the ρ algorithm has expected running time

$$\mathcal{O}(\sqrt{p}) = \mathcal{O}(\sqrt{p}(\log n)^2) = \mathcal{O}(n^{1/4}(\log n)^2) \tag{4.19}$$

to find the prime factor p of n.

Remark 4.2 The ρ Method is an improvement over trial division, because in trial division, $\mathcal{O}(p) = \mathcal{O}(n^{1/4})$ divisions are needed to find a small factor p of n. But of course, one disadvantage of the ρ algorithm is that its running time is only a conjectured expected value, not a rigorous bound.

In 1974, Pollard [40] also invented another simple but effective factoring algorithm, now widely known as Pollard's "$p - 1$" Method, which can be described as follows:

Algorithm 4.4 (Pollard's "$p - 1$" Method) Let $n > 1$ be a composite number. This algorithm attempts to find a nontrivial factor of n.

[1] (Initialization) Pick out $a \in \mathbb{Z}/n\mathbb{Z}$ at random. Select a positive integer k that is divisible by many prime powers, for example, $k = \text{lcm}(1, 2, \ldots, B)$ for a suitable bound B (the larger B is the more likely the method will be to succeed in producing a factor, but the longer the method will take to work).
[2] (Exponentiation) Compute $a_k = a^k \bmod n$.
[3] (Compute GCD) Computing $d = \gcd(a_k - 1, n)$.
[4] (Factor Found?) If $1 < d < N$, then d is a nontrivial factor of n, output d and go to Step [6].
[5] (Start Over?) If d is not a nontrivial factor of n and if you still want to try more experiments, then go to Step [1] to start all over again with a new choice of a and/or a new choice of k, else go to Step [6].
[6] (Exit) Terminate the algorithm.

The "$p - 1$" algorithm is usually successful in the fortunate case where n has a prime divisor p for which $p - 1$ has no large prime factors. Suppose that $(p - 1) \mid k$ and that $p \nmid a$. Since $|(\mathbb{Z}/p\mathbb{Z})^*| = p - 1$, we have $a^k \equiv 1 \pmod p$, thus $p \mid \gcd(a_k - 1, n)$. In many cases, we have $p = \gcd(a_k - 1, n)$, so the method finds a nontrivial factor of n.

Example 4.5 Use the "$p - 1$" Method to factor the number $n = 540143$. Choose $B = 8$ and hence $k = 840$. Choose also $a = 2$. Then we have

$$\gcd(2^{840} - 1 \bmod 540143, 540143) = \gcd(53046, 540143) = 421.$$

Thus 421 is a (prime) factor of 540143. In fact, $421 \cdot 1283$ is the complete prime factorization of 540143. It is interesting to note that by using the "$p - 1$" method Baillie in 1980 found the prime factor

$$p_{25} = 1155685395246619182673033$$

of the Mersenne number $M_{257} = 2^{257} - 1$. In this case

$$p_{25} - 1 = 2^3 \cdot 3^2 \cdot 19^2 \cdot 47 \cdot 67 \cdot 257 \cdot 439 \cdot 119173 \cdot 1050151.$$

In the worst case, where $(p - 1)/2$ is prime, the "$p - 1$" algorithm is no better than trial division. Since the group has fixed order $p - 1$ there is nothing to be done except try a different algorithm. Note that there is a similar method to "$p - 1$," called "$p + 1$," that was proposed by H. C. Williams in 1982. It is suitable for the case where N has a prime factor p for which $p + 1$ has no large prime factors.

Problems for Section 4.3

1. Let f be a random function in a set of p elements, and x_0 a random element. Define iteratively that $x_{i+1} = f(x_i)$, $i = 1, 2, \ldots, t$, $t = 1 + \lfloor (2\lambda p)^{1/2} \rfloor$ for a given real number λ. Show that the probability $x_0, x_1, \ldots, x_t$ are pairwise different $\leq e^{\lambda}$.
2. Show that the complexity of the ρ Method for factoring a general number n is $\mathcal{O}(n^{1/4})$.

3. Let $n = 3161$ and x_0 is the seed with $f(x) = x^2 + 1$. Use the ρ Method to factor n.
4. Use $p - 1$ method to find the smallest prime factor of the 9th Fermat number $F_9 = 2^{2^9} + 1$.
5. Use $p - 1$ method to find three prime factors of $2^{71} - 1$.
6. Use the ρ Factoring Method to factor 4087 using $x_0 = 2$ and $f(x) = x^2 - 1$.
7. Let $x_i = f(x_{i-1}),\ i = 1, 2, 3, \ldots$. Let also $t, u > 0$ be the smallest numbers in the sequence $x_{t+i} = x_{t+u+i},\ i = 0, 1, 2, \ldots$, where t and u are called the lengths of the ρ tail and cycle, respectively. Give an efficient algorithm to determine t and u exactly, and analyze the running time of your algorithm.

4.4 Elliptic Curve Method

In this section, we shall introduce a factoring method depending on the use of elliptic curves. The method is actually obtained from Pollard's "$p - 1$" algorithm: If we can choose a *random* group G with order g close to p, we may be able to perform a computation similar to that involved in Pollard's "$p - 1$" algorithm, working in G rather than in F_p. If all prime factors of g are less than the bound B then we find a factor of n. Otherwise, we repeat this procedure with a different group G (and hence, usually, a different g) until a factor is found. This is the motivation of the ECM method, invented by H. W. Lenstra in 1987 [42].

Algorithm 4.5 (Lenstra's Elliptic Curve Method) Let $n > 1$ be a composite number, with $\gcd(n, 6) = 1$. This algorithm attempts to find a nontrivial factor of n. The method depends on the use of elliptic curves and is the analog to Pollard's "$p - 1$" Method.

[1] (Choose an Elliptic Curve) Choose a random pair (E, P), where E is an elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{Z}/n\mathbb{Z}$, and $P(x, y) \in E(\mathbb{Z}/n\mathbb{Z})$ is a point on E. That is, choose $a, x, y \in \mathbb{Z}/n\mathbb{Z}$ at random, and set $b \leftarrow y^2 - x^3 - ax$. If $\gcd(4a^3 + 27b^2, n) \neq 1$, then E is not an elliptic curve, start all over and choose another pair (E, P).

[2] (Choose an Integer k) Just as in the "$p - 1$" Method, select a positive integer k that is divisible by many prime powers, for example, $k = \mathrm{lcm}(1, 2, \ldots, B)$ or $k = B!$ for a suitable bound B; the larger B is the more likely the method will succeed in producing a factor, but the longer the method will take to work.

[3] (Calculate kP) Calculate the point $kP \in E(\mathbb{Z}/n\mathbb{Z})$. We use the following formula to compute $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$ modulo n:

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2 \bmod n,\quad \lambda(x_1 - x_3) - y_1 \bmod n)$$

where

$$\lambda = \begin{cases} \dfrac{m_1}{m_2} \equiv \dfrac{3x_1^2 + a}{2y_1} \pmod{n}, & \text{if } P_1 = P_2 \\ \dfrac{m_1}{m_2} \equiv \dfrac{y_1 - y_2}{x_1 - x_2} \pmod{n}, & \text{otherwise.} \end{cases}$$

The computation of $kP \bmod n$ can be done in $\mathcal{O}(\log k)$ doublings and additions.

[4] (Computing GCD) If $kP \equiv \mathcal{O}_E \pmod{n}$, then compute $d = \gcd(m_2, n)$, else go to Step [1] to make a new choice for "a" or even for a new pair (E, P).
[5] (Factor Found?) If $1 < d < n$, then d is a nontrivial factor of n, output d and go to Step [7].
[6] (Start Over?) If d is not a nontrivial factor of n and if you still wish to try more elliptic curves, then go to Step [1] to start all over again, else go to Step [7].
[7] (Exit) Terminate the algorithm.

As for the "$p-1$" method, one can show that a given pair (E, P) is likely to be successful in the above algorithm if n has a prime factor p for which $\mathbb{Z}/p\mathbb{Z}$ is composed of small primes only. The probability for this to happen increases with the number of pairs (E, P) that one tries.

Example 4.6 Use the ECM method to factor the number $n = 187$.

[1] Choose $B = 3$, and hence $k = 1\text{cm}(1, 2, 3) = 6$. Let $P = (0, 5)$ be a point on the elliptic curve $E : y^2 = x^3 + x + 25$ which satisfies $\gcd(N, 4a^3 + 27b^2) = \gcd(187, 16879) = 1$ (note that here $a = 1$ and $b = 25$).
[2] Since $k = 6 = 110_2$, we compute $6P = 2(P + 2P)$ in the following way:
[a] Compute $2P = P + P = (0, 5) + (0, 5)$:

$$\begin{cases} \lambda = \dfrac{m_1}{m_2} = \dfrac{1}{10} \equiv 131 \pmod{187} \\ x_3 = 144 \pmod{187} \\ y_3 = 18 \pmod{187} \end{cases}$$

So $2P = (144, 18)$ with $m_2 = 10$ and $\lambda = 131$.
[b] Compute $3P = P + 2P = (0, 5) + (144, 18)$:

$$\begin{cases} \lambda = \dfrac{m_1}{m_2} = \dfrac{13}{144} \equiv 178 \pmod{187} \\ x_3 = 124 \pmod{187} \\ y_3 = 176 \pmod{187} \end{cases}$$

So $3P = (124, 176)$ with $m_2 = 144$ and $\lambda = 178$.
[c] Compute $6P = 2(3P) = 3P + 3P = (124, 176) + (124, 176)$:

$$\lambda = \frac{m_1}{m_2} = \frac{46129}{352} \equiv \frac{127}{165} \equiv \mathcal{O}_E \pmod{187}.$$

This time $m_1 = 127$ and $m_2 = 165$, so the modular inverse for $127/165$ modulo 187 does not exist; but this is exactly what we want! – this type of failure is called a "pretend failure."
[3] Compute $d = \gcd(n, m_2) = \gcd(187, 165) = 11$. Since $1 < 11 < 187$, 11 is a (prime) factor of 187. In fact, $187 = 11 \cdot 17$.

Example 4.7 To factor $n = 7560636089$, we calculate $kP = k(1, 3)$ with $k = \text{lcm}(1, 2, 3, \ldots, 19)$ on $y^2 \equiv x^3 - x + 7 \pmod{n}$:

$3P = (1329185554, 395213649)$
$6P = (646076693, 5714212282)$
$12P = (5471830359, 5103472059)$
$24P = (04270711, 3729197625)$
$49P = (326178740, 3033431040)$
$99P = (5140727517, 2482333384)$
$199P = (1075608203, 3158750830)$
$398P = (4900089049, 2668152272)$
$797P = (243200145, 2284975169)$
$1595P = (3858922333, 4843162438)$
$3191P = (7550557590, 1472275078)$
$6382P = (4680335599, 1331171175)$
$12765P = (6687327444, 7233749859)$
$25530P = (6652513841, 6306817073)$
$51061P = (6578825631, 5517394034)$
$102123P = (1383310127, 2036899446)$
$204247P = (3138092894, 2918615751)$
$408495P = (6052513220, 1280964400)$
$816990P = (2660742654, 3418862519)$
$1633980P = (7023086430, 1556397347)$
$3267961P = (5398595429, 795490222)$
$6535923P = (4999132, 4591063762)$
$13071847P = (3972919246, 7322445069)$
$26143695P = (3597132904, 3966259569)$
$52287391P = (2477960886, 862860073)$
$104574782P = (658268732, 3654016834)$
$209149565P = (6484065460, 287965264)$
$418299131P = (1622459893, 4833264668)$
$836598262P = (7162984288, 487850179)$
$1673196525P = \mathcal{O}_E$.

Now, $1673196525P$ is the point at infinity since $5398907681/1016070716 \bmod n$ is impossible. Hence, $\gcd(1016070716, 7560636089) = 15121$ gives a factor of n.

Example 4.8 The following are some ECM factoring examples. In 1995 Richard Brent at the Australian National University completed the factorization of the 10th Fermat number using ECM:

$$2^{2^{10}} + 1 = 2^{1024} + 1 = 45592577 \cdot 6487031809 \cdot p_{40} \cdot p_{252}$$

where the 40-digit prime p_{40} was found using ECM, and p_{252} was proved to be a 252-digit prime. Brent also completed the factorization of the 11th Fermat number (with 617-digit) $F_{11} = 2^{2^{11}} + 1$ using ECM:

$$F_{11} = 319489 \cdot 974849 \cdot 167988556341760475137 \cdot 3560841906445833920513 \cdot p_{564}$$

where the 21-digit and 22-digit prime factors were found using ECM, and p_{564} is a 564-digit prime. Other recent ECM-records include a 38-digit prime factor (found by A. K. Lenstra and M. S. Manasse) in the 112-digit composite $(11^{118} + 1)/(2 \cdot 61 \cdot 193121673)$, a 40-digit prime factor of $26^{126} + 1$, a 43-digit prime factor of the partition number $p(19997)$, and a 44-digit prime factor of the partition number $p(19069)$ in the RSA Factoring Challenge List, and a 47-digit prime in c_{135} of $5^{2^8} + 1 = 2 \cdot 1655809 \cdot p_{38} \cdot c_{135}$.

Both Lenstra's ECM algorithm and Pollard's "$p - 1$" algorithm can be sped up by the addition of a second phase. The idea of the second phase in ECM is to find a factor in the case that the first phase terminates with a group element $P \neq I$, such that $|\langle P \rangle|$ is reasonably small, say $\mathcal{O}(n^2)$), here $\langle P \rangle$ is the cyclic group generated by P. There are several possible implementations of the second phase. One of the simplest uses a pseudorandom walk in $\langle P \rangle$. By the birthday paradox argument, there is a good chance that two points in the random

walk will coincide after $\mathcal{O}(\sqrt{|\langle P\rangle|})$ steps, and when this occurs a nontrivial factor of n can usually be found (see [6] and [5] for more detailed information on the implementation issues of the ECM.

Conjecture 4.2 (Complexity of the ECM method) Let p be the smallest prime dividing n. Then the ECM method will find p of n, under some plausible assumptions, in expected running time

$$\mathcal{O}\left(\exp\left(\sqrt{(2+o(1))\log p \log\log p}\right)\cdot(\log n)^2\right) \tag{4.20}$$

In the worst case, when n is the product of two prime factors of the same order of magnitude, we have

$$\begin{aligned}\mathcal{O}\left(\exp\left(\sqrt{(2+o(1))\log n \log\log n}\right)\right)\\ = \mathcal{O}\left(n^{\sqrt{(2+o(1))\log\log n/\log n}}\right).\end{aligned} \tag{4.21}$$

Remark 4.3 The most interesting feature of ECM is that its running time depends very much on p (the factor found) of n, rather than N itself. So one advantage of the ECM is that one may use it, in a manner similar to trial divisions, to locate the smaller prime factors p of a number n which is much too large to factor completely.

Problems for Section 4.4

1. Show that $\#(a,b) = p^2 - p$, where $\#(a,b)$ denotes the number of integer pairs (a,b) such that $0 \le a, b < p$, for which $4a^3 \not\equiv 27b^2 \pmod{p}$ is exactly $p^2 - p$.
2. Show that if $p > 3$, $4a^3 + 27b^2 \equiv 0 \pmod{p}$, $p \nmid a$, then the root r of the congruence $-2ar \equiv 3b \pmod{p}$ is a repeated rood modulo p of the polynomial $x^3 - ax - b$.
3. Let $p > 2$ be prime. Suppose that x and y are integers such that

$$x^2 + y^2 \equiv 1 \pmod{p}, \quad x \not\equiv 1 \pmod{p}.$$

   Let u be determined by the congruence

$$(1+x)u \equiv y \pmod{p}.$$

   Show that

$$u^2 + 1 \equiv 0 \pmod{p}.$$

4. Canfield–Erdös–Pomerance theorem: Let n be a positive integer which is not a prime power and not divisible by 2 or 3. if α is a real number, then the probability that a random

positive integer $s \le x$ has all its prime factors $\le L(x)^{\alpha}$ is $L(x)^{-1/(2\alpha)+o(1)}$ for $x \to \infty$, where $L(x) = e^{\sqrt{\log x \log\log x}}$ with x a real number $> e$.

We also need the following conjecture: Let $x = p$, the probability that a random integer has all its prime factors $\le L(x)^{\alpha}$ in the small interval $(x + 1 - \sqrt{x}, x + 1 + \sqrt{x})$ is $L(p)^{-1/(2\alpha)+o(1)}$ for $p \to \infty$.

By the Canfield–Erdös–Pomerance theorem and the conjecture, show that
(1) the probabilistic time estimate for ECM to find the smallest prime factor p of n is

$$e^{\sqrt{2+o(1)\log p \log\log p}}$$

(2) the probabilistic time estimate for ECM to find n is

$$e^{\sqrt{1+o(1)\log n \log\log n}}.$$

5. Use Algorithm 4.5 to factor the three integers 17531, 218548425731 and 190387615311371.
6. Modify and improve Algorithm 4.5 to a practical factoring algorithm for large integer n.
7. Modify and improve Algorithm 4.5 to a parallel practical factoring algorithm for large integer n.

4.5 Continued Fraction Method

The Continued Fraction Method is perhaps the first *modern*, *general* purpose integer factorization method, although its original idea may go back to M. Kraitchik in the 1920s, or even earlier to A. M. Legendre. It was used by D. H. Lehmer and R. E. Powers to devise a new technique in the 1930s, however the method was not very useful and applicable at the time because it was unsuitable for desk calculators. About 40 years later, it was first implemented on a computer by M. A. Morrison and J. Brillhart [7], who used it to successfully factor the seventh Fermat number

$$F_7 = 2^{2^7} + 1 = 59649589127497217 \cdot 5704689200685129054721$$

on the morning of 13 September 1970.

The Continued FRACtion (CFRAC) method looks for small values of $|W|$ such that $x^2 \equiv W \pmod{n}$ has a solution. Since W is small (specifically $W = \mathcal{O}(\sqrt{n})$), it has a reasonably good chance of being a product of primes in our factor base FB. Now if W is small and $x^2 \equiv W \pmod{n}$, then we can write $x^2 = W + knd^2$ for some k and d, hence $(x/d)^2 - kn = W/d^2$ will be small. In other words, the rational number x/d is an approximation of $\sqrt{kn}$. This suggests looking at the continued fraction expansion of $\sqrt{kn}$, since continued fraction expansions of real numbers give good rational approximations. This is exactly the idea behind

the CFRAC method! We first obtain a sequence of approximations (i.e., convergents) P_i/Q_i to $\sqrt{kn}$ for a number of values of k, such that

$$\left|\sqrt{kn} - \frac{P_i}{Q_i}\right| \leq \frac{1}{Q_i^2}. \tag{4.22}$$

Putting $W_i = P_i^2 - Q_i^2 kn$, then we have

$$W_i = (P_i + Q_i\sqrt{kn})(P_i - Q_i\sqrt{kn}) \sim 2Q_i\sqrt{kn}\frac{1}{Q_i} \sim 2\sqrt{kn}. \tag{4.23}$$

Hence, the $P_i^2 \bmod n$ are small and more likely to be smooth, as desired. Then, we try to factor the corresponding integers $W_i = P_i^2 - Q_i^2 kn$ over our factor base FB; with each success, we obtain a new congruence of the form

$$P_i^2 \equiv W_i \Longleftrightarrow x^2 \equiv (-1)^{e_0} p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m} \pmod{n}. \tag{4.24}$$

Once we have obtained at least $m+2$ such congruences, by Gaussian elimination over $\mathbb{Z}/2\mathbb{Z}$ we have obtained a congruence $x^2 \equiv y^2 \pmod{n}$. That is, if $(x_1, e_{01}, e_{11}, \cdots, e_{m1}), \cdots, (x_r, e_{0r}, e_{1r}, \cdots, e_{mr})$ are solutions of (4.24) such that the vector sum

$$(e_{01}, e_{11}, \cdots, e_{m1}) + \cdots + (e_{0r}, e_{1r}, \cdots, e_{mr}) = (2e_0', 2e_1', \cdots, 2e_m') \tag{4.25}$$

is even in each component, then

$$x \equiv x_1 x_2 \cdots x_r \pmod{n} \tag{4.26}$$

$$y \equiv (-1)^{e_0'} p_1^{e_1'} \cdots p_m^{e_m'} \pmod{n} \tag{4.27}$$

is a solution to

$$x^2 \equiv y^2 \pmod{n},\ 0 < x < y < n,\ x \neq y,\ x + y \neq n, \tag{4.28}$$

except for the possibility that $x \equiv \pm y \pmod{n}$, and hence (usually) a nontrivial factoring of n, by computing $\gcd(x \pm y, n)$.

Example 4.9 We now illustrate, by an example, the idea of CFRAC factoring. Let $n = 1037$. Then $\sqrt{1037} = [32, \overline{4, 1, 15, 3, 3, 15, 1, 4, 64}]$. The first ten continued fraction approximations of $\sqrt{1037}$ are:

Convergent P/Q	$P^2 - n \cdot Q^2 := W$
$32/1$	$-13 = -13$
$129/4$	$49 = 7^2$
$161/5$	$-4 = -2^2$
$2544/79 \equiv 470/79$	$19 = 19$
$7793/242 \equiv 534/242$	$-19 = -19$
$25923/805 \equiv 1035/805$	$4 = 2^2$
$396638/12317 \equiv 504/910$	$-49 = -7^2$
$422561/13122 \equiv 502/678$	$13 = 13$
$2086882/64805 \equiv 438/511$	$-1 = -1$
$133983009/4160642 \equiv 535/198$	$13 = 13$

Now we search for squares on both sides, either just by a single congruence, or by a combination (i.e., multiplying together) of several congruences and find that

$$\begin{aligned}
129^2 \equiv 7^2 &\Longleftrightarrow \gcd(1037,\ 129 \pm 7) = (17, 61)\\
1035^2 \equiv 2^2 &\Longleftrightarrow \gcd(1037,\ 1035 \pm 2) = (1037, 1)\\
129^2 \cdot 1035^2 \equiv 7^2 \cdot 2^2 &\Longleftrightarrow \gcd(1037,\ 129 \cdot 1035 \pm 7 \cdot 2) = (61, 17)\\
161^2 \cdot 504^2 \equiv (-1)^2 \cdot 2^2 \cdot 7^2 &\Longleftrightarrow \gcd(1037,\ 161 \cdot 504 \pm 2 \cdot 7) = (17, 61)\\
502^2 \cdot 535^2 \equiv 13^2 &\Longleftrightarrow \gcd(1037\ 502 \cdot 535 \pm 13) = (1037, 1).
\end{aligned}$$

Three of them yield a factorization of $1037 = 17 \cdot 61$.

It is clear that the CFRAC factoring algorithm is essentially just a continued fraction algorithm for finding the continued fraction expansion $[q_0, q_1, \cdots, q_k, \cdots]$ of $\sqrt{kn}$, or the P_k and Q_k of such an expansion. In what follows, we shall briefly summarize the CFRAC method just discussed above in the following algorithmic form:

Algorithm 4.6 (CFRAC factoring) Given a positive integer n and a positive integer k such that kn is not a perfect square, this algorithm tries to find a factor of n by computing the continued fraction expansion of $\sqrt{kN}$.

[1] Let n be the integer to be factored and k any small integer (usually 1), and let the factor base, FB, be a set of small primes $\{p_1, p_2, \cdots, p_r\}$ chosen such that it is possible to find some integer x_i such that $x_i^2 \equiv kn \pmod{p_i}$. Usually, FB contains all such primes less than or equal to some limit. Note that the multiplier $k > 1$ is needed only when the period is short. For example, Morrison and Brillhart used $k = 257$ in factoring F_7.

[2] Compute the continued fraction expansion $[q_0, \overline{q_1, q_2, \cdots, q_r}]$ of $\sqrt{kn}$ for a number of values of k. This gives us good rational approximations P/Q. The recursion formulas to

use for computing P/Q are as follows:

$$\begin{aligned}
\frac{P_0}{Q_0} &= \frac{q_0}{1}, \\
\frac{P_1}{Q_1} &= \frac{q_0 q_1 + 1}{q_1}, \\
&\vdots \\
\frac{P_i}{Q_i} &= \frac{q_i P_{i-1} + P_{i-2}}{q_i Q_{i-1} + Q_{i-2}}, \quad i \geq 2.
\end{aligned}$$

This can be done by a continued fraction algorithm such as Theorem 2.26 introduced earlier.

[3] Try to factor the corresponding integer $W = P^2 - Q^2 kn$ in our factor base FB. Since $W < 2\sqrt{kn}$, each of these W is only about half the length of kn. If we succeed, we get a new congruence. For each success, we obtain a congruence

$$x^2 \equiv (-1)^{e_0} p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m} \pmod{n},$$

since, if P_i/Q_i is the i^{th} continued fraction convergent to $\sqrt{kn}$ and $W_i = P_i^2 - N \cdot Q_i^2$, then

$$P_i^2 \equiv W_i \pmod{n}. \tag{4.29}$$

[4] Once we have obtained at least $m + 2$ such congruences, then by Gaussian elimination over $\mathbb{Z}/2\mathbb{Z}$ we obtain a congruence $x^2 \equiv y^2 \pmod{n}$. That is, if $(x_1, e_{01}, e_{11}, \cdots, e_{m1}), \cdots, (x_r, e_{0r}, e_{1r}, \cdots, e_{mr})$ are solutions of (4.24) such that the vector sum defined in (4.25) is even in each component, then

$$\begin{cases} x \equiv x_1 x_2 \cdots x_r \pmod{n} \\ y \equiv (-1)^{e_0'} p_1^{e_1'} \cdots p_m^{e_m'} \pmod{n} \end{cases}$$

is a solution to $x^2 \equiv y^2 \pmod{n}$, except for the possibility that $x \equiv \pm y \pmod{N}$, and hence we have

$$(d_1, d_2) = (\gcd(x + y,\ n),\ \ \gcd(x - y,\ N)),$$

which are then possibly nontrivial factors of N.

Conjecture 4.3 (The complexity of the CFRAC Method) If n is the integer to be factored, then under certain reasonable heuristic assumptions, the CFRAC method will factor n in time

$$\begin{aligned}
&\mathcal{O}\left(\exp\left((\sqrt{2} + o(1))\sqrt{\log N \log\log N}\right)\right) \\
&= \mathcal{O}\left(N^{\sqrt{(2+o(1))\log\log N/\log N}}\right).
\end{aligned} \tag{4.30}$$

Remark 4.4 This is a conjecture, not a theorem, because it is supported by some heuristic assumptions which have not been proven.

Problems for Section 4.5

1. Let

$$x = 1 + \cfrac{1}{4 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{8 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{14 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{22}}}}}}}}}}}}$$

   be the continued fraction expansion of x. Find the successive convergents P_i/Q_i of this continued fraction.
2. Let the successive convergents P_i/Q_i of the continued fraction of x be as follows:

$$\left[2, 3, \frac{14}{5}, \frac{17}{6}, \frac{65}{23}, \frac{82}{29}, \frac{967}{342}, \frac{1049}{371}, \frac{7261}{2568}, \frac{15571}{5507}, \frac{925950}{327481}, \frac{3719371}{1315431}\right].$$

   Find the continued fraction expansion of x.
3. Let n be a positive integer that is not a perfect square. Let P_k/Q_k the kth convergent of the simple continued fraction expansion of $\sqrt{n}$. Then

$$P_k^2 - nQ_k^2 = (-1)^k W_{k+1}, \quad 0 < W_i < 2\sqrt{n}.$$

4. Let

$$x^2 \equiv (-1)^{e_0} p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}. \tag{4.31}$$

   If

$$(x_1, e_{01}, e_{11}, \ldots, e_{m1}), \ldots, (x_r, e_{0r}, e_{1r}, \ldots, e_{mr})$$

   are solutions to (4.31) such that

$$(e_{01}, e_{11}, \ldots, e_{m1}) + \cdots + (e_{0r}, e_{1r}, \ldots, e_{mr}) = 2(e_1', e_2', \ldots, e_m'),$$

then

$$\begin{cases} x \equiv (x_1, x_2, \ldots, x_r) \pmod{n} \\ y \equiv (-1)^{e'_1} p_1^{e'_2} \cdots p_m^{e'_m} \pmod{n} \end{cases}$$

is a solution to

$$x^2 \equiv y^2 \pmod{n}, x \not\equiv \pm y \pmod{n}.$$

5. Give a heuristic analysis of the running time of the CFRAC method

$$\mathcal{O}(\exp(\sqrt{2 \log n \log \log n})).$$

6. Implement the CFRAC algorithm on a computer.
7. Use your CFRAC program above to factor the integers 1037, 193541963, and 19354196373153173137.

4.6 Quadratic Sieve

The idea of the quadratic sieve (QS) was first introduced by Carl Pomerance in 1982 [8]. QS is somewhat similar to CFRAC except that instead of using continued fractions to produce the values for $W_k = P_k^2 - n \cdot Q_k^2$, it uses expressions of the form

$$W_k = (k + \lfloor \sqrt{n} \rfloor)^2 - n \equiv (k + \lfloor \sqrt{n} \rfloor)^2 \pmod{n}. \tag{4.32}$$

Here, if $0 < k < L$, then

$$0 < W_k < (2L + 1)\sqrt{n} + L^2. \tag{4.33}$$

If we get

$$\prod_{i=1}^{t} W_{n_i} = y^2, \tag{4.34}$$

then we have $x^2 \equiv y^2 \pmod{n}$ with

$$x \equiv \prod_{i=1}^{t} (\lfloor \sqrt{n} \rfloor + n_i) \pmod{n}. \tag{4.35}$$

Once such x and y are found, there is a good chance that $\gcd(x - y, n)$ is a nontrivial factor of n.

Example 4.10 Use the Quadratic Sieve Method (QS) to factor $n = 2041$. Let $W(x) = x^2 - n$, with $x = 43, 44, 45, 46$. Then we have:

$W(43)$	=	$-2^6 \cdot 3$
$W(44)$	=	$-3 \cdot 5 \cdot 7$
$W(45)$	=	-2^4
$W(46)$	=	$3 \cdot 5^2$

p	$W(43)$	$W(44)$	$W(45)$	$W(46)$
-1	1		1	0
2	0		0	0
3	1		0	1
5	0		0	0

which leads to the following congruence:

$$(43 \cdot 45 \cdot 46)^2 \equiv (-1)^2 \cdot 2^{10} \cdot 3^2 \cdot 5^2 = (2^5 \cdot 3 \cdot 5)^2.$$

This congruence gives the factorization of $2041 = 13 \cdot 157$, since

$$\gcd(2041,\ 43 \cdot 45 \cdot 46 + 2^5 \cdot 3 \cdot 5) = 157, \quad \gcd(2041,\ 43 \cdot 45 \cdot 46 - 2^5 \cdot 3 \cdot 5) = 13.$$

For the purpose of implementation, we can use the same set FB as that used in CFRAC and the same idea as that described above to arrange (4.34) to hold.

The most widely used variation of the quadratic sieve is perhaps the Multiple Polynomial Quadratic Sieve (MPQS), first proposed by Peter Montgomery in 1986. The idea of the MPQS is as follows: To find the (x, y) pair in

$$x^2 \equiv y^2 \pmod{n} \tag{4.36}$$

we try to find triples (U_i, V_i, W_i), for $i = 1, 2, \ldots$, such that

$$U_i^2 \equiv V_i^2 W_i \pmod{n} \tag{4.37}$$

where W is easy to factor (at least easier than N). If sufficiently many congruences (4.37) are found, they can be combined, by multiplying together a subset of them, in order to get a relation of the form (4.36). The version of the MPQS algorithm described here is based on [9].

Algorithm 4.7 (Multiple Polynomial Quadratic Sieve) Given a positive integer $n > 1$, this algorithm will try to find a factor N using the multiple polynomial quadratic sieve.

[1] Choose B and M, and compute the factor base FB.

Note: M is some fixed integer so that we can define: $U(x) = a^2x + b$, $V = a$ and $W(x) = a^2x^2 + 2bx + c$, $x \in [-M, M)$, such that a, b, c satisfy the following relations:

$$a^2 \approx \sqrt{2n/M}, \quad b^2 - n = a^2c, \quad |b| < (a^2)/2. \tag{4.38}$$

Note: The potential prime divisors p of a given quadratic polynomial $W(x)$ may be characterized as: If $p \mid W(x)$, then

$$a^2W(x) = (a^2x + b)^2 - n \equiv 0 \pmod{p}. \tag{4.39}$$

That is, the congruence $t^2 - n \equiv 0 \pmod{p}$ should be solvable. So, the factor base FB (consisting of all primes less than a bound B) should be chosen in such a way that $t^2 \equiv n \pmod{p}$ is solvable. There are L primes p_j, $j = 1, 2, \ldots, L$ in FB; this set of primes is fixed in the whole factoring process.

[2] Generate a new quadratic polynomial $W(x)$.

Note: The quadratic polynomial $W(x)$ in

$$(U(x))^2 \equiv V(x)W(x) \pmod{n} \tag{4.40}$$

assumes extreme values in $x = 0, \pm M$ such that $|W(0)| \approx |W(\pm M)| \approx M\sqrt{n/2}$. If $M \ll n$, then $W(x) \ll n$, thus $W(x)$ is easier to factor than n.

[3] Solve $W(x) \equiv 0 \pmod{q}$ for all $q = p^e < B$, for all primes $p \in$ FB, and save the solutions for each q.

[4] Initialize the sieving array SI$[-M, M)$ to zero.

[5] Add $\log p$ to all elements SI(j), $j \in [-M, M]$, for which $W(j) \equiv 0 \pmod{q}$, for all $q = p^e < B$, and for all primes $p \in P$FB.

Note: Now we can collect those $x \in [-M, M)$ for which $W(x)$ is only composed of prime factors $< B$.

[6] Select those $j \in [-M, M)$ for which SI(j) is closed to $\log(n/2\sqrt{n/2})$.

[7] If the number of $W(x)$-values collected in Step 6 is $< L + 2$, then go to Step 2 to construct a new polynomial $W(x)$.

Note: If at least $L + 2$ completely factorized W-values have been collected, then the (x, y)-pairs satisfying (4.36) may be found as follows: For x_i, $i = 1, 2, \ldots, L + 2$,

$$W(x_i) = (-1)^{\alpha_{i0}} \prod_{j=1}^{L} p_j^{\alpha_{ij}}, \quad i = 1, 2, \ldots, L + 2. \tag{4.41}$$

[8] Perform Gaussian elimination on the matrix of exponents (mod 2) of $W(x)$.

Note: Associated with each $W(x_i)$, we define the vector α_i as follows

$$\alpha_i^T = (\alpha_{i0}, \alpha_{i1}, \ldots, \alpha_{iL}) \pmod{2}. \tag{4.42}$$

Since we have more vectors α_i (at least $L + 2$) than components $(L + 1)$, there exists at least one subset S of the set $\{1, 2, \ldots, L + 2\}$ such that

$$\sum_{i \in S} \alpha_i \equiv 0 \pmod{2}, \tag{4.43}$$

so that

$$\prod_{i \in S} W(x) = Z^2. \tag{4.44}$$

Hence, from (4.40) it follows that

$$\left[\prod_{i\in S}(a^2x_i+b)\right] \equiv Z^2\prod_{i\in S}a^2 \pmod{n} \tag{4.45}$$

which is of the required form $x^2 \equiv y^2 \pmod{n}$.

[9] Compute gcd.

Note: Now we can calculate gcd $(x \pm y, n)$ to find the prime factors of n.

Example 4.11 MPQS has been used to obtain many spectacular factorizations. One such factorization is the 103-digit composite number

$$\frac{2^{361}+1}{3\cdot 174763} = 68743016175348275093505757684543562450254 03 \cdot p_{61}.$$

The other record of the MPQS is the factorization of the RSA-129 in April 1994, a 129 digit composite number:

$$\begin{aligned}\text{RSA-129} &= 114381625757888867669235779976146612010218296721242 3\_\\ &\quad 625625618429357069352457338978305971235639587050589 8\_\\ &\quad 907514759929002687954354 1\\ &= p_{64}\cdot q_{65}\\ &= 349052951084765094914784961990389813341776463849338 7\_\\ &\quad 843990820577\cdot 327691329932667095499619881908344614 13\_\\ &\quad 177642967992942539798288533.\end{aligned}$$

It was estimated in Gardner [23] in 1977 that the running time required to factor numbers with about the same size as RSA-129 would be about 40 quadrillion years using the best algorithm and fastest computer at that time. It was factorized by Derek Atkins, Michael Graff, Arjen Lenstra, Paul Leyland, and more than 600 volunteers from more than 20 countries, on all continents except Antarctica. To factor this number, they used the double large prime variation of the Multiple Polynomial Quadratic Sieve Factoring Method. The sieving step took approximately 5000 mips years.

Conjecture 4.4 (The complexity of the QS/MPQS Method) If n is the integer to be factored, then under certain reasonable heuristic assumptions, the QS/MPQS method will factor n in time

$$\begin{aligned}&\mathcal{O}\left(\exp\left((1+o(1))\sqrt{\log n\log\log n}\right)\right)\\ &= \mathcal{O}\left(n^{(1+o(1))\sqrt{\log\log n/\log n}}\right).\end{aligned} \tag{4.46}$$

Problems for Section 4.6

1. Show that
$$n^{c(\log\log n/\log n)^{1/2}} = \exp(c(\log n \log\log n)^{1/2}).$$
2. Show that
$$(\log n)^{c\log\log\log n} = \exp(c\log\log n\log\log\log n).$$
3. Show that if $n = p^k$ with p prime and $k \geq 1$, then $(p-1) \mid (n-1)$.
4. Let $n = p_1 p_2 \cdots p_k$, where $p_1, p_2, \ldots, p_k$ are distinct odd primes. Let also $y \in (\mathbb{Z}/n\mathbb{Z})^*$. Then the congruence
$$x^2 \equiv y^2 \pmod{n}$$
has exact 2^k solutions modulo n, two of them are
$$x = \begin{cases} y, \\ -y. \end{cases}$$
5. Let x and y be randomly chosen so that
$$x^2 \equiv y^2 \pmod{n}.$$
Show that the chance of
$$x \not\equiv \pm y \pmod{n}$$
is greater than $1/2$. That is, the chance for
$$1 < \gcd(x-y, n) < n, \quad \text{and} \quad \gcd(x-y, n) \mid n$$
is greater than $1/2$.
6. Find a suitable pair of integers (x, y) such that
$$x^2 \equiv y^2 \pmod{139511931371319137}$$
and then factor 139511931371319137.
7. A number is smooth if all of its prime factors are small; a number is B-smooth if all of its prime factors are $\leq B$. Let $\pi(B)$ the numbers of primes in the interval $[1, B]$ and $u_1, u_2, \ldots, u_k$ be positive B-smooth integers with $k > \pi(B)$. Show that some nonempty subset of $\{u_i\}$ has product which is a square.
8. Let ϵ be an arbitrary small positive integer, and
$$L(x) = \exp(\sqrt{\log x \log\log x}).$$

Show that if $L(x)^{\sqrt{2}+\epsilon}$ is chosen from $[1, x]$ independently and uniformly, then the probability that some nonempty subset product is a square tends to 1 as $x \to \infty$, whereas the probability that some nonempty subset product is a square tends to 0 as $x \to \infty$ if $L(x)^{\sqrt{2}-\epsilon}$ is chosen.

9. Let $\psi(x, y)$ be y-smooth numbers up to x. Show that the expected number of choices of random integers in $[1, x]$ to find one y-smooth number is

$$\frac{x}{\psi(x, y)}$$

and to find $\pi(y) + 1$ such y-smooth numbers is

$$\frac{x(\pi(y) + 1)}{\psi(x, y)}.$$

10. Let $u_1, u_2, \ldots$ be y-smooth number, and let each be factored as follows

$$u_i = 2^{\alpha_{i,1}} 3^{\alpha_{i,2}} \cdots p_k^{\alpha_{i,k}}.$$

Show that

$$\prod_{i \in I} u_i = \beta^2, \text{ for some positive integer } \beta$$

if and only if

$$\sum_{i \in I} (\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,k}) = 0$$

as a vector in $(\mathbb{Z}/2\mathbb{Z})^k$. Moreover, such a nontrivial subset is guaranteed amongst $u_1, u_2, \ldots, u_{k+1}$.

11. Let ϵ be any fixed, positive real number. Show that

$$\psi(x, L(x)^{\epsilon}) = xL(x)^{-1/(2\epsilon)+o(1)}, \text{ as } x \to \infty.$$

12. Deduce that the Quadratic Sieve is a deterministic factoring algorithm with the following conjectured complexity

$$\exp((1 + o(1))(\log n \log \log n)^{1/2}).$$

4.7 Number Field Sieve

Before introducing the Number Field Sieve (NFS), it is interesting to briefly review some important milestones in the development of the factoring methods. In 1970, it was barely possible to factor "hard" 20-digit numbers. In 1980, by using the CFRAC method, factoring

of 50-digit numbers was becoming commonplace. In 1990, the QS method had doubled the length of the numbers that could be factored by CFRAC, with a record having 116 digits. In the spring of 1996, the NFS method had successfully split a 130-digit RSA challenge number in about 15% of the time the QS would have taken. At present, the Number Field Sieve (NFS) is the champion of all known factoring methods. NFS was first proposed by John Pollard in a letter to A. M. Odlyzko, dated August 31 1988, with copies to R. P. Brent, J. Brillhart, H. W. Lenstra, C. P. Schnorr, and H. Suyama, outlining an idea of factoring certain big numbers via *algebraic number fields*. His original idea was not for any large composite, but for certain "pretty" composites that had the property that they were close to powers. He illustrated the idea with a factorization of the seventh Fermat number $F_7 = 2^{2^7} + 1$ which was first factored by CFRAC in 1970. He also speculated in the letter that "if F_9 is still unfactored, then it might be a candidate for this kind of method eventually?" The answer now is of course "yes," since F_9 was factored by using NFS in 1990. It is worthwhile pointing out that NFS is not only a method suitable for factoring numbers in a special form like F_9, but also a general purpose factoring method for any integer of a given size. There are, in fact, two forms of NFS (Huizing [10], and Lenstra and Lenstra [11]): the *special* NFS (SNFS), tailored specifically for integers of the form $N = c_1 r^t + c_2 s^u$, and the *general* NFS (GNFS), applicable to any arbitrary numbers. Since NFS uses some ideas from algebraic number theory, a brief introduction to some basic concepts of algebraic number theory is in order.

Definition 4.1 A complex number α is an *algebraic number* if it is a root of a polynomial

$$f(x) = a_0 x^k + a_1 x^{k-1} + a_2 x^{k-2} \cdots + a_k = 0 \tag{4.47}$$

where $a_0, a_1, a_2, \ldots, a_k \in \mathbb{Q}$ and $a_0 \neq 0$. If $f(x)$ is irreducible over $\mathbb{Q}$ and $a_0 \neq 0$, then k is the degree of x.

Example 4.12 Two examples of algebraic numbers are as follows:

(1) rational numbers, which are the algebraic numbers of degree 1.
(2) $\sqrt{2}$, which is of degree 2 because we can take $f(x) = x^2 - 2 = 0$ ($\sqrt{2}$ is irrational).

Any complex number that is not algebraic is said to be *transcendental* such as π and e.

Definition 4.2 A complex number β is an *algebraic integer* if it is a root of a monic polynomial

$$x^k + b_1 x^{k-1} + b_2 x^{k-2} \cdots + b_k = 0 \tag{4.48}$$

where $b_0, b_1, b_2, \ldots, b_k \in \mathbb{Z}$.

Remark 4.5 A quadratic integer is an algebraic integer satisfying a monic quadratic equation with integer coefficients. A cubic integer is an algebraic integer satisfying a monic cubic equation with integer coefficients.

Example 4.13 Some examples of algebraic integers are as follows:

(1) ordinary (rational) integers, which are the algebraic integers of degree 1. That is they satisfy the monic equations $x - a = 0$ for $a \in \mathbb{Z}$.
(2) $\sqrt[3]{2}$ and $\sqrt[5]{3}$, because they satisfy the monic equations $x^3 - 2 = 0$ and $x^3 - 5 = 0$, respectively.
(3) $(-1 + \sqrt{-3})/2$, because it satisfies $x^2 + x + 1 = 0$.
(4) Gaussian integer $a + b\sqrt{-1}$, with $a, b \in \mathbb{Z}$.

Clearly, every algebraic integer is an algebraic number, but the converse is not true.

Proposition 4.1 A rational number $r \in \mathbb{Q}$ is an algebraic integer if and only if $r \in \mathbb{Z}$.

Proof: If $r \in \mathbb{Z}$, then r is a root of $x - r = 0$. Thus r is an algebraic integer Now suppose that $r \in \mathbb{Q}$ and r is an algebraic integer (i.e., $r = c/d$ is a root of (4.48), where $c, d \in \mathbb{Z}$; we may assume gcd $(c, d) = 1$). Substituting c/d into (4.48) and multiplying both sides by d^n, we get

$$c^k + b_1 c^{k-1} d + b_2 c^{k-2} d^2 \cdots + b_k d^k = 0.$$

It follows that $d \mid c^k$ and $d \mid c$ (since gcd $(c, d) = 1$). Again since gcd $(c, d) = 1$, it follows that $d = \pm 1$. Hence $r = c/d \in \mathbb{Z}$. It follows, for example, that $2/5$ is an algebraic number but not an algebraic integer. ∎

Remark 4.6 The elements of $\mathbb{Z}$ are the only rational numbers that are algebraic integers. We shall refer to the elements of $\mathbb{Z}$ as *rational integers* when we need to distinguish them from other algebraic integers that are not rational. For example, $\sqrt{2}$ is an algebraic integer but not a rational integer.

The most interesting results concerned with the algebraic numbers and algebraic integers are contained in the following theorem.

Theorem 4.2 *The set of algebraic numbers forms a field, and the set of algebraic integers forms a ring.*

Proof: See pp. 67–68 of Ireland and Rosen [12]. ∎

Lemma 4.1 Let $f(x)$ be an irreducible monic polynomial of degree d over integers and m an integer such that $f(m) \equiv 0 \pmod{n}$. Let α be a complex root of $f(x)$ and $\mathbb{Z}[\alpha]$ the set of all polynomials in α with integer coefficients. Then there exists a unique mapping $\Phi : \mathbb{Z}[\alpha] \mapsto \mathbb{Z}_n$ satisfying

(1) $\Phi(ab) = \Phi(a)\Phi(b), \quad \forall a, b \in \mathbb{Z}[\alpha]$;
(2) $\Phi(a + b) = \Phi(a) + \Phi(b), \quad \forall a, b \in \mathbb{Z}[\alpha]$;
(3) $\Phi(za) = z\Phi(a), \quad \forall a \in \mathbb{Z}[\alpha], z \in \mathbb{Z}$;
(4) $\Phi(1) = 1$;
(5) $\Phi(\alpha) = m \pmod{n}$.

Now we are in a position to introduce the Number Field Sieve (NFS). Note that there are two main types of NFS: NFS (general NFS) for general numbers and SNFS (special NFS) for numbers with special forms. The idea, however, behind the GNFS and SNFS is the same:

[1] Find a monic irreducible polynomial $f(x)$ of degree d in $\mathbb{Z}[x]$, and an integer m such that $f(m) \equiv 0 \pmod{n}$.
[2] Let $\alpha \in \mathbb{C}$ be an algebraic number that is the root of $f(x)$, and denote the set of polynomials in α with integer coefficients as $\mathbb{Z}[\alpha]$.
[3] Define the mapping (ring homomorphism): $\Phi : \mathbb{Z}[\alpha] \mapsto \mathbb{Z}_n$ via $\Phi(\alpha) = m$ which ensures that for any $f(x) \in \mathbb{Z}[x]$, we have $\Phi(f(\alpha)) \equiv f(m) \pmod{n}$.
[4] Find a finite set U of coprime integers (a, b) such that

$$\prod_{(a,b)\in U} (a - b\alpha) = \beta^2, \quad \prod_{(a,b)\in U} (a - bm) = y^2 \tag{4.49}$$

for $\beta \in \mathbb{Z}[\alpha]$ and $y \in \mathbb{Z}$. Let $x = \Phi(\beta)$. Then

$$\begin{aligned} x^2 &\equiv \Phi(\beta)\Phi(\beta) \\ &\equiv \Phi(\beta^2) \\ &\equiv \Phi\left(\prod_{(a,b)\in U} (a - b\alpha)\right) \\ &\equiv \prod_{(a,b)\in U} \Phi(a - b\alpha) \\ &\equiv \prod_{(a,b)\in U} (a - bm) \\ &\equiv y^2 \pmod{n} \end{aligned}$$

which is of the required form of the factoring congruence, and hopefully a factor of n can be found by calculating $\gcd(x \pm y, n)$.

There are many ways to implement the above idea, all of which follow the same pattern as we discussed previously in CFRAC and QS/MPQS: By a sieving process one first tries to find congruences modulo n by working over a factor base, and then does a Gaussian elimination over $\mathbb{Z}/2\mathbb{Z}$ to obtain a congruence of squares $x^2 \equiv y^2 \pmod{n}$. We give in the following a brief description of the NFS algorithm [13].

Algorithm 4.8 Given an odd positive integer n, the NFS algorithm has the following four main steps in factoring n:

[1] (Polynomials Selection) Select two irreducible polynomials $f(x)$ and $g(x)$ with small integer coefficients for which there exists an integer m such that

$$f(m) \equiv g(m) \equiv 0 \pmod{n} \tag{4.50}$$

The polynomials should not have a common factor over $\mathbb{Q}$.

[2] (Sieving) Let α be a complex root of f and β a complex root of g. Find pairs (a, b) with $\gcd(a, b) = 1$ such that the integral norms of $a - b\alpha$ and $a - b\beta$:

$$N(a - b\alpha) = b^{\deg(f)} f(a/b), \qquad N(a - b\beta) = b^{\deg(g)} g(a/b) \tag{4.51}$$

are smooth with respect to a chosen factor base. (The principal ideals $a - b\alpha$ and $a - b\beta$ factor into products of prime ideals in the number field $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$, respectively.)

[3] (Linear Algebra) Use techniques of linear algebra to find a set $U = \{a_i, b_i\}$ of indices such that the two products

$$\prod_U (a_i - b_i\alpha), \qquad \prod_U (a_i - b_i\beta) \tag{4.52}$$

are both squares of products of prime ideals.

[4] (Square Root) Use the set S in (4.52) to find algebraic numbers $\alpha' \in \mathbb{Q}(\alpha)$ and $\beta' \in \mathbb{Q}(\beta)$ such that

$$(\alpha')^2 = \prod_U (a_i - b_i\alpha), \qquad (\beta')^2 = \prod_U (a_i - b_i\beta) \tag{4.53}$$

Define $\Phi_\alpha : \mathbb{Q}(\alpha) \to \mathbb{Z}_n$ and $\Phi_\beta : \mathbb{Q}(\beta) \to \mathbb{Z}_n$ via $\Phi_\alpha(\alpha) = \Phi_\beta(\beta) = m$, where m is the common root of both f and g. Then

$$\begin{aligned} x^2 &\equiv \Phi_\alpha(\alpha')\Phi_\alpha(\alpha') \\ &\equiv \Phi_\alpha((\alpha')^2) \\ &\equiv \Phi_\alpha\left(\prod_{i \in U} (a_i - b_i\alpha)\right) \\ &\equiv \prod_U \Phi_\alpha(a_i - b_i\alpha) \end{aligned}$$

$$\equiv \prod_{U}(a_i - b_i m)$$

$$\equiv \Phi_\beta(\beta')^2$$

$$\equiv y^2 \pmod{n}$$

which is of the required form of the factoring congruence, and hopefully, a factor of N can be found by calculating gcd $(x \pm y, n)$.

Example 4.14 We first give a rather simple NFS factoring example. Let $n = 14885 = 5 \cdot 13 \cdot 229 = 122^2 + 1$. So we put $f(x) = x^2 + 1$ and $m = 122$, such that

$$f(x) \equiv f(m) \equiv 0 \pmod{n}.$$

If we choose $|a|, |b| \leq 50$, then we can easily find (by sieving) that

(a, b)	Norm$(a + bi)$	$a + bm$
$\vdots$	$\vdots$	$\vdots$
$(-49, 49)$	$4802 = 2 \cdot 7^4$	$5929 = 7^2 \cdot 11^2$
$\vdots$	$\vdots$	$\vdots$
$(-41, 1)$	$1682 = 2 \cdot 29^2$	$81 = 3^4$
$\vdots$	$\vdots$	$\vdots$

(Readers should be able to find many such pairs of (a_i, b_i) in the interval, that are smooth up to e.g., 29.) So, we have

$$\begin{aligned}
(49 + 49i)(-41 + i) &= (49 - 21i)^2, \\
f(49 - 21i) &= 49 - 21m \\
&= 49 - 21 \cdot 122 \\
&= -2513, \\
&\updownarrow \\
&x \\
5929 \cdot 81 &= (2^2 \cdot 7 \cdot 11)^2 \\
&= 693^2 \\
&\Rightarrow 693. \\
&\updownarrow \\
&y
\end{aligned}$$

Thus,

$$\begin{aligned}
\gcd(x \pm y, n) &= \gcd(-2513 \pm 693, 14\,885) \\
&= (65, 229).
\end{aligned}$$

In the same way, if we wish to factor $n = 84101 = 290^2 + 1$, then we let $m = 290$, and $f(x) = x^2 + 1$ so that

$$f(x) \equiv f(m) \equiv 0 \pmod{n}.$$

We tabulate the sieving process as follows:

(a, b)	Norm$(a + bi)$	$a + bm$
$\vdots$	$\vdots$	$\vdots$
$-50, 1$	$2501 = 41 \cdot 61$	$240 = 2^4 \cdot 3 \cdot 5$
$\vdots$	$\vdots$	$\vdots$
$-50, 3$	$2509 = 13 \cdot 193$	$820 = 2^2 \cdot 5 \cdot 41$
$\vdots$	$\vdots$	$\vdots$
$-49, 43$	$4250 = 2 \cdot 5^3 \cdot 17$	$12421 = 12\,421$
$\vdots$	$\vdots$	$\vdots$
$-38, 1$	$1445 = 5 \cdot 17^2$	$252 = 2^2 \cdot 3^2 \cdot 7$
$\vdots$	$\vdots$	$\vdots$
$-22, 19$	$845 = 5 \cdot 13^2$	$5488 = 2^4 \cdot 7^3$
$\vdots$	$\vdots$	$\vdots$
$-118, 11$	$14045 = 5 \cdot 53^2$	$3072 = 2^{10} \cdot 3$
$\vdots$	$\vdots$	$\vdots$
$218, 59$	$51005 = 5 \cdot 101^2$	$17328 = 2^4 \cdot 3 \cdot 19^2$
$\vdots$	$\vdots$	$\vdots$

Clearly, $-38 + i$ and $-22 + 19i$ can produce a product square, since

$$\begin{aligned}
(-38 + i)(-22 + 19i) &= (31 - 12i)^2, \\
f(31 - 12i) &= 31 - 12m \\
&= -3449, \\
&\updownarrow \\
&x \\
252 \cdot 5488 &= (2^3 \cdot 3 \cdot 7^2)^2 \\
&= 1176^2, \\
&\updownarrow \\
&y \\
\gcd(x \pm y, n) &= \gcd(-3449 \pm 1176, 84\,101) \\
&= (2273, 37).
\end{aligned}$$

In fact, $84101 = 2273 \times 37$. Note that $-118 + 11i$ and $218 + 59i$ can also produce a product square, since

$$\begin{aligned}
(-118 + 11i)(218 + 59i) &= (14 - 163i)^2, \\
f(14 - 163i) &= 14 - 163m \\
&= -47256, \\
&\updownarrow \\
&x \\
3071 \cdot 173288 &= (2^7 \cdot 3 \cdot 19)^2 \\
&= 7296^2, \\
&\updownarrow \\
&y \\
\gcd(x \pm y, n) &= \gcd(-47\,256 \pm 7296, 84\,101) \\
&= (37, 2273).
\end{aligned}$$

Example 4.15 Next we present a slightly more complicated example. Use NFS to factor $n = 1098413$. First notice that $n = 1098413 = 12 \cdot 45^3 + 17^3$, which is in a special form and can be factored by using SNFS.

[1] (Polynomials Selection) Select the two irreducible polynomials $f(x)$ and $g(x)$ and the integer m as follows:

$$m = \frac{17}{45},$$

$$f(x) = x^3 + 12 \Longrightarrow f(m) = \left(\frac{17}{45}\right)^3 + 12 \equiv 0 \pmod{n},$$

$$g(x) = 45x - 17 \Longrightarrow g(m) = 45\left(\frac{17}{45}\right) - 17 \equiv 0 \pmod{n}.$$

[2] (Sieving) Suppose after sieving, we get $U = \{a_i, b_i\}$ as follows:

$$U = \{(6, -1), (3, 2), (-7, 3), (1, 3), (-2, 5), (-3, 8), (9, 10)\}.$$

That is, the chosen polynomial that produces a product square can be constructed as follows (as an exercise, readers may wish to choose some other polynomial which can also produce a product square):

$$\prod_U (a_i + b_i x) = (6 - x)(3 + 2x)(-7 + 3x)(1 + 3x)(-2 + 5x)(-3 + 8x)(9 + 10x).$$

Let $\alpha = \sqrt[3]{-12}$ and $\beta = \frac{17}{45}$. Then

$$\begin{aligned}\prod_U (a - b\alpha) &= 7400772 + 1138236\alpha - 10549\alpha^2 \\ &= (2694 + 213\alpha - 28\alpha^2)^2 \\ &= \left(\frac{5610203}{2025}\right) \\ &= 270729^2, \\ \prod_U (a - b\beta) &= \frac{2^8 \cdot 11^2 \cdot 13^2 \cdot 23^2}{3^{12} \cdot 5^4} \\ &= \left(\frac{52624}{18225}\right)^2 \\ &= 875539^2.\end{aligned}$$

So, we get the required square of congruence:

$$270729^2 \equiv 875539^2 \pmod{1098413}.$$

Thus,

$$\gcd(270729 \pm 875539, 1098413) = (563, 1951).$$

That is,

$$1098413 = 563 \cdot 1951.$$

Example 4.16 Finally, we give some large factoring examples using NFS.

(1) SNFS examples: One of the largest numbers factored by SNFS is

$$n = (12^{167} + 1)/13 = p_{75} \times p_{105}$$

It was announced by P. Montgomery, S. Cavallar, and H. te Riele at CWI in Amsterdam on September 3 1997. They used the polynomials $f(x) = x^5 - 144$ and $g(x) = 12^{33}x + 1$ with common root $m \equiv 12^{134} \pmod{n}$. The factor base bound was 4.8 million for f and 12 million for g. Both large prime bounds were 150 million, with two large primes allowed on each side. They sieved over $|a| \leq 8.4$ million and $0 < b \leq 2.5$ million. The sieving lasted 10.3 calendar days; 85 SGI machines at CWI contributed a combined 13027719 relations in 560 machine-days. It took 1.6 more calendar days to process the data. This processing included 16 CPU-hours on a Cray C90 at SARA in Amsterdam to

process a 1969262×1986500 matrix with 57942503 nonzero entries. The other large number factorized by using SNFS is the 9th Fermat number:

$$F_9 = 2^{2^9} + 1 = 2^{512} + 1 = 2424833 \cdot p_{49} \cdot p_{99},$$

a number with 155 digits; it was completely factored in April 1990. The most wanted factoring number of special form at present is the 12th Fermat number $F_{12} = 2^{2^{12}} + 1$; we only know its partial prime factorization:

$$F_{12} = 114689 \cdot 26017793 \cdot 63766529 \cdot 190274191361 \cdot 1256132134125569 \cdot c_{1187}$$

and we want to find the prime factors of the remaining 1187-digit composite.

(2) GNFS examples: Three large general numbers RSA-130 (in April 1996), RSA-140 (in February 1999), RSA-155 (August 1999), and RSA-174 (December 2003) were factorized using GNFS:
   (a) RSA-130 $= p_{65} \cdot q_{65}$:
   3968599945959745429016112616288378606757644911281006483255515724 3_
   4553449864673597218840368689727440886435630126320506960099904459 9,
   (b) RSA-140 $= p_{70} \cdot q_{70}$:
   3398717423028438554530123627613875835633986495969597423 4_
   90929302771479,
   6264200187401285096151654948264442219302037178623509019 1_
   11660653946049,
   (c) RSA-155 $= p_{78} \cdot q_{79}$:
   1026395928297411057720541965739916759007165678080380668 0_
   3341933521790711307779,
   1066034883801684548209272203600128786792079585759892915 2_
   2270608237193062808643,
   (d) RSA-174 $= p_{87} \cdot q_{87}$:
   3980750864240649373971255005503864911990643623425267084 0_
   6385189575946388957261768583317,
   4727721461074353025362230719730482246329146953020971164 5_
   9852171130520711256363590397527.

Remark 4.7 Prior to the NFS, all modern factoring methods had an expected running time of at best

$$\mathcal{O}\left(\exp\left((c + o(1))\sqrt{\log n \log\log n}\right)\right).$$

For example, Dixon's Random Square Method has the expected running time

$$\mathcal{O}\left(\exp\left((\sqrt{2} + o(1))\sqrt{\log n \log\log n}\right)\right).$$

whereas the Multiple Polynomial Quadratic Sieve (MPQS) takes time

$$\mathcal{O}\left(\exp\left((1+o(1))\sqrt{\log\log n/\log n}\right)\right).$$

Because of the Canfield-Erdös-Pomerance theorem, some people even believed that this could not be improved upon except maybe for the term $(c+o(1))$, but the invention of the NFS has changed all this.

Conjecture 4.5 (Complexity of NFS) Under some reasonable heuristic assumptions, the NFS method can factor an integer N in time

$$\mathcal{O}\left(\exp\left((c+o(1))\sqrt[3]{\log n}\sqrt[3]{(\log\log n)^2}\right)\right) \tag{4.54}$$

where $c=(64/9)^{1/3}\approx 1.922999427$ if GNFS is used to factor an arbitrary integer N, whereas $c=(32/9)^{1/3}\approx 1.526285657$ if SNFS is used to factor a special integer N.

Problems for Section 4.7

1. (General Factoring Challenge Problems) Try to complete the prime factorization, either individually or in group, of the following RSA challenge numbers (Note that in these numbers, the value for x in RSA-x represents the number of bits, not the number of digits; this is just the RSA convention for these numbers):

   (1) RSA-704 (212 digits, 704 bits)
   740375634795617128280467960974295731425931888892312890849362 3_
   263897276503402826627689199641962511784399589433050212758537 0_
   119689809828673317327310893090055250511687706329907239638078 67_
   100860969625379346505637963 59,

   (2) RSA-768 (232 digits, 768 bits)
   123018668453011775513049495838496272077285356959533479219732 2_
   452151726400507263657518745202199786469389956474942774063845 9_
   251925573263034537315482685079170261221429134616704292143116 0_
   222124047927473779408066535141959745985690214341 3,

   (3) RSA-896 (270 digits, 896 bits)
   412023436986659543855531365332575948179811699844327982845455 6_
   264338764455652484261980988704231618418792614202471888694925 6_
   093177637503342113098239748515094490910691026986103186270411 4_
   880866970564902903653658867433731720813104105190864254793282 6_
   013912576240339463732693 91,

   (4) RSA-1024 (309 digits, 1024 bits)
   135066410865995223349603216278805969938881475605667027524485 1_
   438515265106048595338339402871505719094417982072821644715513 7_
   368041970396419174304649658927425623934102086438320211037295 8_

725762358509643110564073501508187510676594629205563685529475 2_
135008528794163773285339061097505443349998111500569772368909 2_
7563,

(5) RSA-1536 (463 digits, 1536 bits)
184769970321174147430683562020016440301854933866341017147178 5_
774910651696711161249859337684305435744585616061544571794052 2_
297177325246609606469460712496237204420222697567566873784275 6_
238950876467844093328515749657884341508847552829818672645133 9_
863364931908084671990431874381283363502795470282653297802934 9_
161558118810498449083195450098483937752272570525785919449938 7_
007369575568843693381277961308923039256969525326162082367649 0_
316036551371447913932347169566988069,

(6) RSA-2048 (617 digits, 2048 bits)
251959084756578934940271832400483985714292821262040320277771 3_
783604366202070759555626401852588078440691829064124951508218 9_
298559149176184502808489120072844992687392807287767359714183 _
472702618963750149718246911650776133798590957000973304597488 0_
842840179742910064245869181719511874612151517265463228221686 9_
987549182422433637259085141865462043576798423387184774447920 7_
399342365848238242811981638150106748104516603773060562016196 7_
625613384414360383390441495263443219011465754445418424020924 _
616515723350778707749817125772467962926386356373289912154831 4_
381678998850404453640235273819513786365643912120103971228221 2_
0720357.

2. (Knuth's Factoring Challenge Problem) Knuth in 1998 proposed the following 211 digits factoring challenge number [4], marked its difficulty degree as 50, one of the hardest problems in his book:

779030228851015954236247565470557836248576762097398394108440 2_
222135728725117099985850483876481319443405109322651368151685 7_
411993477558685427409422564450008791272325857493370618539583 4_
027843405820888108548507873 7.

Try to complete the prime factorization, either individually or in group, of the above Knuth factoring challenge number.

3. (Rivest's Factoring Challenge Problem) In April 1999, when the MIT Laboratory for Computer Science celebrated its 35 anniversary, Prof Ron Rivest proposed the following factoring challenge problem as a part of his Secret-Key Computing Challenge Problem:

$n =$ 631446608307288889379935712613129233236329 88_
183308413755889907727019571289248855473084 46_
055753206513618346628848948088663500368480 39_
658817136198766052189726781016228055747539 38_
383082617597132189266686117769545263915701 20_
690939973680089721274464666423319187806830 55_

2067951253070082020241246233982410737753705 1_
2734449416950118097524189066796385875485631 9_
8055072737099043971197336146667015439053601 5_
2543373982524579313575317653646331989064651 4_
0213398526580034199190398219284471021246488 7_
4593888535820703180842890232097109070323969 3_
4919962778995323320184064522476463966355937 3_
6700936921275809208629319872700829243124368 1.

Try to complete the prime factorization, either individually or in group, of the above Rivest's factoring challenge number.

4. In this problem, we list the smallest unfactored (not completely factored) Fermat numbers for you to try to find the complete factorization for each of these numbers:

$$F_{12} = 114689 \cdot 26017793 \cdot 63766529 \cdot 190274191361 \cdot 1256132134125569 \cdot c_{1187},$$

$$F_{13} = 2710954639361 \cdot 2663848877152141313 \cdot 36031098445229199 \cdot 319546020820551643220672513 \cdot c_{2391},$$

$$F_{14} = c_{4933},$$

$$F_{15} = 1214251009 \cdot 2327042503868417 \cdot 1687688170295169723830241270169 61 \cdot c_{9808},$$

$$F_{16} = 825753601 \cdot 188981757975021318420037633 \cdot c_{19694},$$

$$F_{17} = 31065037602817 \cdot c_{39444},$$

$$F_{18} = 13631489 \cdot 81274690703860512587777 \cdot c_{78884},$$

$$F_{19} = 70525124609 \cdot 646730219521 \cdot c_{157804},$$

$$F_{20} = c_{315653},$$

$$F_{21} = 4485296422913 \cdot c_{631294},$$

$$F_{22} = c_{1262612},$$

$$F_{23} = 167772161 \cdot c_{2525215},$$

$$F_{24} = c_{5050446}.$$

4.8 Bibliographic Notes and Further Reading

In this chapter, we discussed some of the most popular algorithms for integer factorization. For general references in this field, it is suggested that readers consult, for example, [14–38].

Shanks' Square Forms Factorization was studied in [39]. The $p-1$ Factoring Method was proposed by Pollard in [40] and ρ Factoring Method in [2], respectively. An improved version of ρ was proposed by Brent in [3]. The $p+1$ Factoring Method was proposed by Williams in [41].

The ECM factoring method was first proposed by Lenstra in [42]. More information on ECM may be found in [5, 6]. The CFRAC factoring method was first proposed and implemented in [7]. The QS factoring method was first proposed by Pomerance in the 1980s [9, 43, 44]. The idea of NFS was first proposed by Pollard and subsequently improved by many authors; more information about the NFS can be found in [11, 45, 46]. Readers who are interested in parallel and distributed factoring may consult [47–50].

References

1. P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1484–1509.
2. J. M. Pollard, "A Monte Carlo Method for Factorization", *BIT*, **15**, 1975, pp. 331–332.
3. R. P. Brent, "An Improved Monte Carlo Factorization Algorithm", *BIT*, **20**, 1980, pp. 176–184.
4. D. E. Knuth, *The Art of Computer Programming II – Seminumerical Algorithms*, 3rd Edition, Addison-Wesley, 1998.
5. P. L. Montgomery, "Speeding Pollard's and Elliptic Curve Methods of Factorization", *Mathematics of Computation*, **48**, 1987, pp. 243–264.
6. R. P. Brent, "Some Integer Factorization Algorithms using Elliptic Curves", *Australian Computer Science Communications*, **8**, 1986, pp. 149–163.
7. M. A. Morrison and J. Brillhart, "A Method of Factoring and the Factorization of F_7", *Mathematics of Computation*, **29**, 1975, pp. 183–205.
8. C. Pomerance, "Analysis and Comparison of some Integer Factoring Algorithms", in H. W. Lenstra, Jr. and R. Tijdeman, eds, *Computational Methods in Number Theory*, No **154/155**, Mathematical Centrum, Amsterdam, 1982, pp. 89–139.
9. H. J. J. te Riele, W. Lioen, and D. Winter, "Factoring with the Quadrtaic Sieve on Large Vector Computers", *Journal of Computational and Applied Mathematics*, **27**, 1989, pp. 267–278.
10. R. M. Huizing, "An Implementation of the Number Field Sieve", *Experimental Mathematics*, **5** 3, 1996, pp. 231–253.
11. A. K. Lenstra and H. W. Lenstra, Jr. (editors), *The Development of the Number Field Sieve*, Lecture Notes in Mathematics **1554**, Springer-Verlag, 1993.
12. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd Edition, Graduate Texts in Mathematics **84**, Springer, 1990.
13. P. L. Montgomery, "A Survey of Modern Integer Factorization Algorithms", *CWI Quarterly*, **7**, 4, 1994, pp. 337–394.
14. L. M. Adleman, "Algorithmic Number Theory – The Complexity Contribution", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1994, pp. 88–113.
15. D. Atkins, M. Graff, A. K. Lenstra, and P. C. Leyland, "The Magic Words are Squeamish Ossifrage", *Advances in Cryptology – ASIACRYPT'94*, Lecture Notes in Computer Science **917**, 1995, pp. 261–277.
16. R. P. Brent, "Primality Testing and Integer Factorization", *Proceedings of Australian Academy of Science Annual General Meeting Symposium on the Role of Mathematics in Science*, Canberra, 1991, pp. 14–26.
17. R. P. Brent, "Recent Progress and Prospects for Integer Factorization Algorithms", *Proc. COCOON 2000 (Sydney, July 2000)*, Lecture Notes in Computer Science **1858**, Springer, 2000, pp. 3–22.
18. D. M. Bressound, *Factorization and Primality Testing*, Springer, 1989.
19. H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer-Verlag, 1993.
20. T. H. Cormen, C. E. Ceiserson and R. L. Rivest, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009.

21. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer-Verlag, 2005.
22. J. D. Dixon, "Factorization and Primality tests", *The American Mathematical Monthly*, June–July 1984, pp. 333–352.
23. M. Gardner, "Mathematical Games – A New Kind of Cipher that Would Take Millions of Years to Break", *Scientific American*, **237**, 2, 1977, pp. 120–124.
24. C. F. Gauss, *Disquisitiones Arithmeticae*, G. Fleischer, Leipzig, 1801. English translation by A. A. Clarke, Yale University Press, 1966. Revised English translation by W. C. Waterhouse, Springer-Verlag, 1975.
25. A. Granville, "Smooth Numbers: Computational Number Theory and Beyond", *Algorithmic Number Theory*. Edited by J. P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 267–323.
26. T. Kleinjung, et al., "Factorization of a 768-Bit RSA Modulus", In: T. Rabin (Ed.), CRYPTO 2010, Lecture Notes in Computer Science **6223**, Springer, 2010, pp. 333–350.
27. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer-Verlag, 1994.
28. A. K. Lenstra, "Integer Factoring", *Design, Codes and Cryptography*, **19**, 2/3, 2000, pp. 101–128.
29. R. A. Mullin, *Algebraic Number Theory*, 2nd Edition, CRC Press, 2011.
30. C. Pomerance "Integer Factoring", *Cryptology and Computational Number Theory*. Edited by C. Pomerance, Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1990, pp. 27–48.
31. C. Pomerance, "On the Role of Smooth Numbers in Number Theoretic Algorithms", *Proceedings of the Intenational Congress of Mathematicians*, Zurich, Switzerland 1994, Birkhauser Verlag, Basel, 1995, pp. 411–422.
32. C. Pomerance, "A Tale of Two Sieves", *Notice of the AMS*, **43**, 12, 1996, pp. 1473–1485.
33. H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, Boston, 1990.
34. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
35. H. C. Williams, "The Influence of Computers in the Development of Number Theory", *Computers & Mathematics with Applications*, **8**, 2, 1982, pp. 75–93.
36. H. C. Williams, "Factoring on a Computer", *Mathematical Intelligencer*, **6**, 3, 1984, pp. 29–36.
37. S. Y. Yan, "Computing Prime Factorization and Discrete Logarithms: From Index Calculus to Xedni Calculus", *International Journal of Computer Mathematics*, **80**, 5, 2003, pp. 573–590.
38. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.
39. S. McMath, *Daniel Shanks' Square Forms Factorization*, United States Naval Academy, Annapolis, Maryland, 24 November 2004.
40. J. M. Pollard, "Theorems on Factorization and Primality Testing", *Procedings of Cambridge Philosophy Society*, **76**, 1974, pp. 521–528.
41. H. C. Williams, "A $p + 1$ Method of Factoring", *Mathematics of Computation*, **39**, 1982, pp. 225–234.
42. H. W. Lenstra, Jr., "Factoring Integers with Elliptic Curves", *Annals of Mathematics*, **126**, 1987, pp. 649–673.
43. C. Pomerance, "The Quadratic Sieve Factoring Algorithm", *Proceedings of Eurocrypt 84*, Lecture Notes in Computer Science **209**, Springer, 1985, pp. 169–182.
44. C. Pomerance, "Smooth Numbers and the Quadratic Sieve", *Algorithmic Number Theory*. Edited by J. P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 69–81.
45. C. Pomerance, "The Number Field Sieve", *Mathematics of Computation, 1943–1993, Fifty Years of Computational Mathematics*. Edited by W. Gautschi, Proceedings of Symposium in Applied Mathematics **48**, American Mathematical Society, Providence, 1994, pp. 465–480.
46. P. Stevenhagen, "The Number Field Sieve", *Algorithmic Number Theory*. Edited by J.P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 83–100.
47. R. P. Brent, "Parallel Algorithms for Integer Factorisation", *Number Theory and Cryptography*. Edited by J. H. Loxton, London Mathematical Society Lecture Notes i **154**, Cambridge University Press, 1990, pp. 26–37.
48. R. P. Brent, "Some Parallel Algorithms for Integer Factorisation", *Proceedings of 5th International Euro-Par Conference(Toulouse, France)*, Lecture Notes in Computer Science **1685**, Springer, 1999, pp. 1–22.
49. A. K. Lenstra and M. S. Manases "Factoring by Electronic Mail", *Advances in Cryptology – EUROCRYPT Ô89*. Edited by J. J. Quisquater and J. Vandewalle, Lecture Notes in Computer Science **434**, Springer, 1990, pp. 355–371.
50. R. D. Silverman, "Massively Distributed Computing and Factoring Large Integers", *Communications of the ACM*, **34**, 11, 1991, pp. 95–103.

5

Discrete Logarithms

The Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP), along with the Integer Factorization Problem (IFP), are the three most important infeasible computational problems in computational number theory and modern cryptography. In this chapter we discuss several popular and widely used modern algorithms for DLP/ECDLP, including:

- Baby-step Giant-step Method for DLP
- Pohlig–Hellman Algorithm for DLP/ECDLP
- The index calculus for DLP
- The xedni calculus for ECDLP.

5.1 Basic Concepts

The *Discrete Logarithm Problem* (DLP) can be described as follows:

$$\left.\begin{array}{ll} \text{Input :} & a, b, n \in \mathbb{Z}^+ \\ \text{Output :} & x \in \mathbb{Z}_{>1} \text{ with } a^x \equiv b \pmod{n} \\ & \text{if such an } x \text{ exists} \end{array}\right\} \tag{5.1}$$

where the modulus n can either be a composite or a prime.

According to Adleman [1], the Russian mathematician Bouniakowsky developed a clever algorithm to solve the congruence $a^x \equiv b \pmod{n}$, with the asymptotic complexity $\mathcal{O}(n)$ in 1870. Despite its long history, no efficient algorithm has ever emerged for the Discrete Logarithm Problem. It is believed to be hard, a little bit harder than the Integer Factorization Problem (IFP) even in the average case. The best known algorithm for DLP at present, using NFS and due to Gordon [2], requires an expected running time

$$\mathcal{O}\left(\exp\left(c(\log n)^{1/3}(\log\log n)^{2/3}\right)\right).$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

There are essentially three different categories of algorithms in use for computing discrete logarithms:

(1) Algorithms that work for arbitrary groups, that is, those that do not exploit any specific properties of groups; Shanks' Baby-step Giant-step Method, Pollard's ρ Method (an analogue of Pollard's ρ Factoring Method) and the λ Method (also known as wild and tame Kangaroos) are in this category.
(2) Algorithms that work well in finite groups for which the order of the groups has no large prime factors; more specifically, algorithms that work for groups with smooth orders. A positive integer is called *smooth* if it has no large prime factors; it is called y-smooth if it has no large prime factors exceeding y. The well-known Silver–Pohlig–Hellman algorithm based on the Chinese Remainder theorem is in this category.
(3) Algorithms that exploit methods for representing group elements as products of elements from a relatively small set (also making use of the Chinese Remainder theorem); the typical algorithms in this category are Adleman's index calculus algorithm and Gordon's NFS algorithm.

In this chapter, we shall introduce the basic ideas of the algorithms in each of these three categories; more specifically, we shall introduce Shanks' Baby-step Giant-step algorithm, the Silver–Pohlig–Hellman algorithm, Adleman's index calculus algorithm, as well as Gordon's NFS algorithm for computing discrete logarithms. In the last section of this chapter, we shall also deal with algorithms for ECDLP.

Problems for Section 5.1

1. What are the main differences between the familiar logarithms over $\mathbb{R}$ and the discrete logarithms?
2. Let

$$\ln x = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^n}{n},$$

then

$$\log_a b = \frac{\ln b}{\ln a}.$$

Find the logarithm k over $\mathbb{R}$:

$$k = \log_2 5.$$

3. Use the exhaustive method to find the following discrete logarithms k over $\mathbb{Z}_{1009}^*$, if there exists:
   (1) $k \equiv \log_3 57 \pmod{1009}$.
   (2) $k \equiv \log_{11} 57 \pmod{1009}$.
   (3) $k \equiv \log_3 20 \pmod{1009}$.

4. Use the exhaustive method to find the following elliptic curve discrete logarithms k over $E(\mathbb{F}_{1009})$:

$$(190, 271) \equiv k(1, 237) \pmod{1009},$$

that is,

$$k \equiv \log_{(1,237)}(190, 271) \pmod{1009}.$$

where E is the elliptic curve defined by

$$E : y^2 \equiv x^3 + 71x + 602 \pmod{1009}.$$

5.2 Baby-Step Giant-Step Method

The Baby-step Giant-step Method is a meet-in-the-middle algorithm for computing the discrete logarithm. It was first studied in 1968 to calculate the class number of an imaginary quadratic field [3]. Let G be a finite cyclic group of order n, a a generator of G and $b \in G$. The *obvious* algorithm for computing successive powers of a until b is found takes $\mathcal{O}(n)$ group operations. For example, to compute $x = \log_2 15 \pmod{19}$, we compute $2^x \bmod 19$ for $x = 0, 1, 2, \ldots, 19 - 1$ until $2^x \bmod 19 = 15$ for some x is found, that is:

x	0	1	2	3	4	5	6	7	8	9	10	11
a^x	1	2	4	8	16	13	7	14	9	18	17	15

So $\log_2 15 \pmod{19} = 11$. It is clear that when n is large, the algorithm is inefficient. In this section, we introduce a type of square root algorithm, called the baby-step giant-step algorithm, for taking discrete logarithms, which is better than the above mentioned *obvious* algorithm. The algorithm works for every finite cyclic group.

Let $m = \lfloor \sqrt{n} \rfloor$. The baby-step giant-step algorithm is based on the observation that if $x = \log_a b$, then we can uniquely write $x = i + jm$, where $0 \leq i, j < m$. For example, if $11 = \log_2 15 \bmod 19$, then $a = 2, b = 15, m = 5$, so we can write $11 = i + 5j$ for $0 \leq i, j < m$. Clearly here $i = 1$ and $j = 2$ so we have $11 = 1 + 5 \cdot 2$. Similarly, for $14 = \log_2 6 \bmod 19$ we can write $14 = 4 + 5 \cdot 2$, for $17 = \log_2 10 \bmod 19$ we can write $17 = 2 + 5 \cdot 3$, etc. The following is a description of the algorithm:

Algorithm 5.1 (Shanks' baby-step giant-step algorithm) This algorithm computes the discrete logarithm x of y to the base a, modulo n, such that $y = a^x \pmod{n}$:

[1] (Initialization) Computes $s = \lfloor \sqrt{n} \rfloor$.
[2] (Computing the Baby Step) Compute the first sequence (list), denoted by S, of pairs (ya^r, r), $r = 0, 1, 2, 3, \ldots, s - 1$:

$$S = \{(y, 0), (ya, 1), (ya^2, 2), (ya^3, 3), \ldots, (ya^{s-1}, s - 1) \bmod n\} \tag{5.2}$$

and sort S by ya^r, the first element of the pairs in S.

[3] (Computing the Giant Step) Compute the second sequence (list), denoted by T, of pairs (a^{ts}, ts), $t = 1, 2, 3, \ldots, s$:

$$T = \{(a^s, 1), (a^{2s}, 2), (a^{3s}, 3), \ldots, (a^{s^2}, s) \bmod n\} \tag{5.3}$$

and sort T by a^{ts}, the first element of the pairs in T.

[4] (Searching, comparing and computing) Search both lists S and T for a match $ya^r = a^{ts}$ with ya^r in S and a^{ts} in T, then compute $x = ts - r$. This x is the required value of $\log_a y \pmod n$.

This algorithm requires a table with $\mathcal{O}(m)$ entries ($m = \lfloor\sqrt{n}\rfloor$, where n is the modulus). Using a sorting algorithm, we can sort both the lists S and T in $\mathcal{O}(m \log m)$ operations. Thus this gives an algorithm for computing discrete logarithms that uses $\mathcal{O}(\sqrt{n} \log n)$ time and space for $\mathcal{O}(\sqrt{n})$ group elements. Note that Shanks' idea was originally for computing the order of a group element g in the group G, but here we use his idea to compute discrete logarithms. Note also that although this algorithm works on arbitrary groups, if the order of a group is larger than 10^{40}, it will be infeasible.

Example 5.1 Suppose we wish to compute the discrete logarithm $x = \log_2 6 \bmod 19$ such that $6 = 2^x \bmod 19$. According to Algorithm 5.5, we perform the following computations:

[1] $y = 6$, $a = 2$ and $n = 19$, $s = \lfloor\sqrt{19}\rfloor = 4$.

[2] Computing the baby step:

$$\begin{aligned} S &= \{(y, 0), (ya, 1), (ya^2, 2), (ya^3, 3) \bmod 19\} \\ &= \{(6, 0), (6 \cdot 2, 1), (6 \cdot 2^2, 2), (6 \cdot 2^3, 3) \bmod 19\} \\ &= \{(6, 0), (12, 1), (5, 2), (10, 3)\} \\ &= \{(5, 2), (6, 0), (10, 3), (12, 1)\}. \end{aligned}$$

[3] Computing the giant step:

$$\begin{aligned} T &= \{(a^s, s), (a^{2s}, 2s), (a^{3s}, 3s), (a^{4s}, 4s) \bmod 19\} \\ &= \{(2^4, 4), (2^8, 8), (2^{12}, 12), (2^{16}, 16) \bmod 19\} \\ &= \{(16, 4), (9, 8), (11, 12), (5, 16)\} \\ &= \{(5, 16), (9, 8), (11, 12), (16, 4)\}. \end{aligned}$$

[4] Matching and computing: The number 5 is the common value of the first element in pairs of both lists S and T with $r = 2$ and $st = 16$, so $x = st - r = 16 - 2 = 14$. That is, $\log_2 6 \pmod{19} = 14$, or equivalently, $2^{14} \pmod{19} = 6$.

Example 5.2 Suppose now we wish to find the discrete logarithm $x = \log_{59} 67 \bmod 113$, such that $67 = 59^x \bmod 113$. Again by Algorithm 5.1, we have:

[1] $y = 67$, $a = 59$ and $n = 113$, $s = \lfloor\sqrt{113}\rfloor = 10$.

[2] Computing the baby step:

$$\begin{aligned} S &= \{(y, 0), (ya, 1), (ya^2, 2), (ya^3, 3), \ldots, (ya^9, 9) \bmod 113\} \\ &= \{(67, 0), (67 \cdot 59, 1), (67 \cdot 59^2, 2), (67 \cdot 59^3, 3), (67 \cdot 59^4, 4), \\ &\quad (67 \cdot 59^5, 5), (67 \cdot 59^6, 6), (67 \cdot 59^7, 7), (67 \cdot 59^8, 8), \\ &\quad (67 \cdot 59^9, 9) \bmod 113\} \\ &= \{(67, 0), (111, 1), (108, 2), (44, 3), (110, 4), (49, 5), (66, 6), \\ &\quad (52, 7), (17, 8), (99, 9)\} \\ &= \{(17, 8), (44, 3), (49, 5), (52, 7), (66, 6), (67, 0), (99, 9), \\ &\quad (108, 2), (110, 4), (111, 1)\}. \end{aligned}$$

[3] Computing the giant-step:

$$\begin{aligned} T &= \{(a^s, s), (a^{2s}, ss), (a^{3s}, 3s), \ldots (a^{10s}, 10s) \bmod 113\} \\ &= \{(59^{10}, 10), (59^{2\cdot 10}, 2 \cdot 10), (59^{3\cdot 10}, 3 \cdot 10), (59^{4\cdot 10}, 4 \cdot 10), \\ &\quad (59^{5\cdot 10}, 5 \cdot 10), (59^{6\cdot 10}, 6 \cdot 10), (59^{7\cdot 10}, 7 \cdot 10), (59^{8\cdot 10}, 8 \cdot 10), \\ &\quad (59^{9\cdot 10}, 9 \cdot 10) \bmod 113\} \\ &= \{(72, 10), (99, 20), (9, 30), (83, 40), (100, 50), (81, 60), \\ &\quad (69, 70), (109, 80), (51, 90), (56, 100)\} \\ &= \{(9, 30), (51, 90), (56, 100), (69, 70), (72, 10), (81, 60), (83, 40), \\ &\quad (99, 20), (100, 50), (109, 80)\}. \end{aligned}$$

[4] Matching and computing: The number 99 is the common value of the first element in pairs of both lists S and T with $r = 9$ and $st = 20$, so $x = st - r = 20 - 9 = 11$. That is, $\log_{59} 67 \pmod{113} = 11$, or equivalently, $59^{11} \pmod{113} = 67$.

Remark 5.1 Shanks' baby-step giant-step algorithm is a type of *Square Root Method* for computing discrete logarithms. In 1978 Pollard [4] also gave two other types of Square Root Methods, namely the ρ Method (an analogue of Pollard's ρ Factoring Method) and the λ Method (also known as the Wild Kangaroo Method) for computing discrete logarithms. Pollard's methods are probabilistic but remove the necessity of precomputing the lists S and T, as with Shanks' Baby-step Giant-step Method. Again, Pollard's algorithm requires $\mathcal{O}(n)$ group operations and hence is infeasible if the order of the group G is larger than 10^{40}.

Problems for Section 5.2

1. Use the baby-step giant-step algorithm to find the following DLP(k):
   (1)

$$k \equiv \log_3 5 \pmod{29}.$$

(2)

$$k \equiv \log_3 17 \pmod{29}.$$

(3)

$$k \equiv \log_3 26 \pmod{29}.$$

2. Use the baby-step giant-step algorithm to find the following DLP(k):
(1)

$$k \equiv \log_2 11 \pmod{29}.$$

(2)

$$k \equiv \log_2 12 \pmod{53}.$$

(3)

$$k \equiv \log_3 26 \pmod{227}.$$

3. Use the baby-step giant-step algorithm to compute DLP(k):

$$k \equiv \log_5 96 \pmod{317}.$$

4. Use the baby-step giant-step algorithm to compute DLP(k):

$$k \equiv \log_{37} 15 \pmod{123}.$$

5. Use the baby-step giant-step algorithm to compute DLP(k):

$$k \equiv \log_5 57105961 \pmod{58231351}.$$

6. Implement the baby-step giant-step algorithm to compute DLP(k) over a large finite field.
7. Explain how to modify the baby-step giant-step algorithm to find the order of an integer a modulo a prime number p.

5.3 Pohlig–Hellman Method

In 1978, Pohlig and Hellman [5] proposed an important special algorithm, now widely known as the Silver–Pohlig–Hellman algorithm for computing discrete logarithms over GF(q) with

$\mathcal{O}(\sqrt{p})$ operations and a comparable amount of storage, where p is the largest prime factor of $q-1$. Pohlig and Hellman showed that if

$$q-1=\prod_{i=1}^{k} p_i^{\alpha_i}, \tag{5.4}$$

where the p_i are distinct primes and the α_i are natural numbers, and if $r_1, \ldots, r_k$ are any real numbers with $0 \le r_i \le 1$, then logarithms over GF(q) can be computed in

$$\mathcal{O}\left(\sum_{i=1}^{k}\left(\log q + p_i^{1-r_i}\left(1+\log p_i^{r_i}\right)\right)\right)$$

field operations, using

$$\mathcal{O}\left(\log q \sum_{i=1}^{k}\left(1+p_i^{r_i}\right)\right)$$

bits of memory, provided that a precomputation requiring

$$\mathcal{O}\left(\sum_{i=1}^{k} p_i^{r_i}\log p_i^{r_i} + \log q\right)$$

field operations is performed first. This algorithm is very efficient if q is "smooth," that is, all the prime factors of $q-1$ are small. We shall give a brief description of the algorithm as follows:

Algorithm 5.2 (Silver–Pohlig–Hellman Algorithm) This algorithm computes the discrete logarithm

$$x=\log_a b \bmod q. \tag{5.5}$$

[1] Factor $q-1$ into its prime factorization form:

$$q-1=\prod_{i=1}^{k} p_1^{\alpha_1} p_2^{\alpha_2}\cdots p_k^{\alpha_k}. \tag{5.6}$$

[2] Precompute the table $r_{p_i,j}$ for a given field:

$$r_{p_i,j}=a^{j(q-1)/p_i} \bmod q, \quad 0 \le j < p_i. \tag{5.7}$$

This only needs to be done once for any given field.

[3] Compute the discrete logarithm of b to the base a modulo q, that is, compute $x = \log_a b \bmod q$:

[3-1] Use an idea similar to that in the baby-step giant-step algorithm to find the individual discrete logarithms $x \bmod p_i^{\alpha_i}$: To compute $x \bmod p_i^{\alpha_i}$, we consider the representation of this number to the base p_i:

$$x \bmod p_i^{\alpha_i} = x_0 + x_1 p_i + \cdots + x_{\alpha_i - 1} p_i^{\alpha_i - 1}, \tag{5.8}$$

where $0 \le x_n < p_i - 1$.

(a) To find x_0, we compute $b^{(q-1)/p_i}$ which equals $r_{p_i,j}$ for some j, and set $x_0 = j$ for which

$$b^{(q-1)/p_i} \bmod q = r_{p_i,j}. \tag{5.9}$$

This is possible because

$$b^{(q-1)/p_i} \equiv a^{x(q-1)/p} \equiv a^{x_0(q-1)/p} \bmod q = r_{p_i,x_0}. \tag{5.10}$$

(b) To find x_1, compute $b_1 = ba^{-x_0}$. If

$$b_1^{(q-1)/p_i^2} \bmod q = r_{p_i,j}, \tag{5.11}$$

then set $x_1 = j$. This is possible because

$$\begin{aligned} b_1^{(q-1)/p_i^2} &\equiv a^{(x-x_0)(q-1)/p_i^2} \\ &\equiv a^{(x_1+x_2 p_i+\cdots)(q-1)/p_i} \\ &\equiv a^{x_1(q-1)/p} \bmod q \\ &= r_{p_i,x_1}. \end{aligned}$$

(c) To obtain x_2, consider the number $b_2 = ba^{-x_0 - x_1 p_i}$ and compute

$$b_2^{(q-1)/p_i^3} \bmod q.$$

The procedure is carried on inductively to find all $x_0, x_1, \ldots, x_{\alpha_i - 1}$.

[3-2] Use the Chinese Remainder theorem to find the unique value of x from the congruences $x \bmod p_i^{\alpha_i}$.

We now give an example of how the above algorithm works:

Example 5.3 Suppose we wish to compute the discrete logarithm $x = \log_2 62 \bmod 181$. Now we have $a = 2, b = 62$ and $q = 181$ (2 is a generator of $\mathbb{F}_{181}^*$). We follow the computation steps described in the above algorithm:

[1] Factor $q - 1$ into its prime factorization form:

$$180 = 2^2 \cdot 3^2 \cdot 5.$$

[2] Use the following formula to precompute the table $r_{p_i,j}$ for the given field $\mathbb{F}^*_{181}$:

$$r_{p_i,j} = a^{j(q-1)/p_i} \bmod q, \quad 0 \le j < p_i.$$

This only needs to be done once for this field.

(a) Compute

$$r_{p_1,j} = a^{j(q-1)/p_1} \bmod q = 2^{90j} \bmod 181 \text{ for } 0 \le j < p_1 = 2:$$
$$r_{2,0} = 2^{90 \cdot 0} \bmod 181 = 1,$$
$$r_{2,1} = 2^{90 \cdot 1} \bmod 181 = 180.$$

(b) Compute

$$r_{p_2,j} = a^{j(q-1)/p_2} \bmod q = 2^{60j} \bmod 181 \text{ for } 0 \le j < p_2 = 3:$$
$$r_{3,0} = 2^{60 \cdot 0} \bmod 181 = 1,$$
$$r_{3,1} = 2^{60 \cdot 1} \bmod 181 = 48,$$
$$r_{3,2} = 2^{60 \cdot 2} \bmod 181 = 132.$$

(c) Compute

$$r_{p_3,j} = a^{j(q-1)/p_3} \bmod q = 2^{36j} \bmod 181 \text{ for } 0 \le j < p_3 = 5:$$
$$r_{5,0} = 2^{36 \cdot 0} \bmod 181 = 1,$$
$$r_{5,1} = 2^{36 \cdot 1} \bmod 181 = 59,$$
$$r_{5,2} = 2^{36 \cdot 2} \bmod 181 = 42,$$
$$r_{5,3} = 2^{36 \cdot 3} \bmod 181 = 125,$$
$$r_{5,4} = 2^{36 \cdot 4} \bmod 181 = 135.$$

Construct the $r_{p_i,j}$ table as follows:

	j				
p_i	0	1	2	3	4
2	1	180			
3	1	48	132		
5	1	59	42	125	135

This table is manageable if all p_i are small.

[3] Compute the discrete logarithm of 62 to the base 2 modulo 181, that is, compute $x = \log_2 62 \bmod 181$. Here $a = 2$ and $b = 62$:
[3-1] Find the individual discrete logarithms $x \bmod p_i^{\alpha_i}$ using

$$x \bmod p_i^{\alpha_i} = x_0 + x_1 p_i + \cdots + x_{\alpha_i - 1} p_i^{\alpha_i - 1}, \quad 0 \le x_n < p_i - 1.$$

(a-1) Find the discrete logarithms $x \bmod p_1^{\alpha_1}$, that is, $x \bmod 2^2$:

$$x \bmod 181 \Longleftrightarrow x \bmod 2^2 = x_0 + 2x_1.$$

(i) To find x_0, we compute

$$b^{(q-1)/p_1} \bmod q = 62^{180/2} \bmod 181 = 1 = r_{p_1,j} = r_{2,0}$$

hence $x_0 = 0$.
(ii) To find x_1, compute first $b_1 = ba^{-x_0} = b = 62$, then compute

$$b_1^{(q-1)/p_1^2} \bmod q = 62^{180/4} \bmod 181 = 1 = r_{p_1,j} = r_{2,0}$$

hence $x_1 = 0$. So

$$x \bmod 2^2 = x_0 + 2x_1 \Longrightarrow x \bmod 4 = 0.$$

(a-2) Find the discrete logarithms $x \bmod p_2^{\alpha_2}$, that is, $x \bmod 3^2$:

$$x \bmod 181 \Longleftrightarrow x \bmod 3^2 = x_0 + 2x_1.$$

(i) To find x_0, we compute

$$b^{(q-1)/p_2} \bmod q = 62^{180/3} \bmod 181 = 48 = r_{p_2,j} = r_{3,1}$$

hence $x_0 = 1$.
(ii) To find x_1, compute first $b_1 = ba^{-x_0} = 62 \cdot 2^{-1} = 31$, then compute

$$b_1^{(q-1)/p_2^2} \bmod q = 31^{180/3^2} \bmod 181 = 1 = r_{p_2,j} = r_{3,0}$$

hence $x_1 = 0$. So

$$x \bmod 3^2 = x_0 + 2x_1 \Longrightarrow x \bmod 9 = 1.$$

(a-3) Find the discrete logarithms $x \bmod p_3^{\alpha_3}$, that is, $x \bmod 5^1$:

$$x \bmod 181 \Longleftrightarrow x \bmod 5^1 = x_0.$$

To find x_0, we compute

$$b^{(q-1)/p_3} \bmod q = 62^{180/5} \bmod 181 = 1 = r_{p_3,j} = r_{5,0}$$

hence $x_0 = 0$. So we conclude that

$$x \bmod 5 = x_0 \Longrightarrow x \bmod 5 = 0.$$

[3-2] Find the x in

$$x \bmod 181,$$

such that

$$\begin{cases} x \bmod 4 = 0, \\ x \bmod 9 = 1, \\ x \bmod 5 = 0. \end{cases}$$

To do this, we just use the Chinese Remainder theorem to solve the following system of congruences:

$$\begin{cases} x \equiv 0 \pmod 4, \\ x \equiv 1 \pmod 9, \\ x \equiv 0 \pmod 5. \end{cases}$$

The unique value of x for this system of congruences is $x = 100$. (This can be easily done by using, for example, the Maple function `chrem([0,1,0],[4,9,5])`.) So the value of x in the congruence $x \bmod 181$ is 100. Hence $x = \log_2 62 = 100$.

Problems for Section 5.3

1. Use the Silver–Pohliq–Hellman algorithm to solve the DLP

$$k \equiv \log_7 12 \pmod{41}$$

such that

$$7^k \equiv 12 \pmod{41}.$$

2. Use the Silver–Pohliq–Hellman algorithm to solve the DLP

$$k \equiv \log_5 57105961 \pmod{58231351}$$

such that

$$5^k \equiv 57105961 \pmod{58231351}.$$

3. Use the Silver–Pohliq–Hellman algorithm to find the discrete logarithm k such that

$$3^k \equiv 2 \pmod{65537}.$$

4. Use the Silver–Pohliq–Hellman algorithm to find the discrete logarithm k:

$$k \equiv \log_{11} 2 \pmod{65537}.$$

5. Suppose that g and h are primitive roots modulo n. Show that

$$\log_h y \equiv \log_h g \log_g y \pmod{\phi(n)}.$$

6. Let the prime factorization of $n = p - 1$ be given. Give a complete computational complexity analysis of the Silver–Pohliq–Hellman algorithm for the DLP.

5.4 Index Calculus

In 1979, Adleman [1] proposed a general purpose, subexponential algorithm for computing discrete logarithms, called the *index calculus*, with the following expected running time:

$$\mathcal{O}\left(\exp\left(c\sqrt{\log n \log\log n}\right)\right).$$

The index calculus is, in fact, a wide range of methods, including CFRAC, QS, and NFS for IFP. In what follows, we discuss a variant of Adleman's index calculus for DLP in $(\mathbb{Z}/p\mathbb{Z})^*$.

Algorithm 5.3 (Index calculus for DLP) This algorithm tries to find an integer k such that

$$k \equiv \log_\beta \alpha \pmod{p} \quad \text{or} \quad \alpha \equiv \beta^k \pmod{p}. \tag{5.12}$$

[1] Precomputation

[1-1] (Choose Factor Base) Select a factor base Γ, consisting of the first m prime numbers,

$$\Gamma = \{p_1, p_2, \ldots, p_m\}, \tag{5.13}$$

with $p_m \leq B$, the bound of the factor base.

[1-2] (Compute $\beta^e \bmod p$) Randomly choose a set of exponent $e \leq p - 2$, compute $\beta^e \bmod p$, and factor it as a product of prime powers.

[1-3] (Smoothness) Collect only those relations $\beta^e \bmod p$ that are smooth with respect to B. That is,

$$\beta^e \bmod p = \prod_{i=1}^{m} {p_i}^{e_i}, e_i \geq 0. \tag{5.14}$$

When such relations exist, get

$$e \equiv \sum_{j=1}^{m} e_j \log_\beta p_j \pmod{p-1}. \tag{5.15}$$

[1-4] (Repeat) Repeat [1-3] to find at least m such e in order to find m relations as in (5.15) and solve $\log_\beta p_j$ for $j = 1, 2, \ldots, m$.

[2] Compute $k \equiv \log_\beta \alpha \pmod{p}$

[2-1] For each e in (5.15), determine the value of $\log_\beta p_j$ for $j = 1, 2, \ldots, m$ by solving the m modular linear equations with unknown $\log_\beta p_j$.

[2-2] (Compute $\alpha\beta^r \bmod p$) Randomly choose exponent $r \leq p-2$ and compute $\alpha\beta^r \bmod p$.

[2-3] (Factor $\alpha\beta^r \bmod p$ over Γ)

$$\alpha\beta^r \bmod p = \prod_{j=1}^{m} {p_j}^{r_i}, r_j \geq 0. \tag{5.16}$$

If (5.16) is unsuccessful, go back to to Step [2-2]. If it is successful, then

$$\log_\beta \alpha \equiv -r + \sum_{j=1}^{m} r_j \log_\beta p_j \pmod{p-1}. \tag{5.17}$$

Example 5.4 (Index calculus for DLP) Find

$$x \equiv \log_{22} 4 \pmod{3361}$$

such that

$$4 \equiv 22^x \pmod{3361}.$$

[1] Precomputation

[1-1] (Choose Factor Base) Select a factor base Γ, consisting of the first 4 prime numbers,

$$\Gamma = \{2, 3, 5, 7\},$$

with $p_4 \leq 7$, the bound of the factor base.

[1-2] (Compute $22^e \bmod 3361$) Randomly choose a set of exponent $e \leq 3359$, compute $22^e \bmod 3361$, and factor it as a product of prime powers:

$$\begin{aligned} 22^{48} &\equiv 2^5 \cdot 3^2 \pmod{3361}, \\ 22^{100} &\equiv 2^6 \cdot 7 \pmod{3361}, \\ 22^{186} &\equiv 2^9 \cdot 5 \pmod{3361}, \\ 22^{2986} &\equiv 2^3 \cdot 3 \cdot 5^2 \pmod{3361}. \end{aligned}$$

[1-3] (Smoothness) The above four relations are smooth with respect to $B = 7$. Thus

$$\begin{aligned} 48 &\equiv 5\log_{22} 2 + 2\log_{22} 3 \pmod{3360}, \\ 100 &\equiv 6\log_{22} 2 + \log_{22} 7 \pmod{3360}, \\ 186 &\equiv 9\log_{22} 2 + \log_{22} 5 \pmod{3360}, \\ 2986 &\equiv 3\log_{22} 2 + \log_{22} 3 + 2\log_{22} 5 \pmod{3360}. \end{aligned}$$

[2] Compute $k \equiv \log_\beta \alpha \pmod{p}$

[2-1] Compute

$$\begin{aligned} \log_{22} 2 &\equiv 1100 \pmod{3360}, \\ \log_{22} 3 &\equiv 2314 \pmod{3360}, \\ \log_{22} 5 &\equiv 366 \pmod{3360}, \\ \log_{22} 7 &\equiv 220 \pmod{3360}. \end{aligned}$$

[2-2] (Compute $4 \cdot 22^r \bmod p$) Randomly choose exponent $r = 754 \leq 3659$ and compute $4 \cdot 22^{754} \bmod 3361$.

[2-3] (Factor $4 \cdot 22^{754} \bmod 3361$ over Γ)

$$4 \cdot 22^{754} \equiv 2 \cdot 3^2 \cdot 5 \cdot 7 \pmod{3361}.$$

Thus,

$$\begin{aligned} \log_{22} 4 &\equiv -754 + \log_{22} 2 + 2\log_{22} 3 + \log_{22} 5 + \log_{22} 7 \\ &\equiv 2200. \end{aligned}$$

That is,

$$22^{2200} \equiv 4 \pmod{3361}.$$

Example 5.5 Find $k \equiv \log_{11} 7 \pmod{29}$ such that $\beta^k \equiv 11 \pmod{29}$.

[1] (Factor Base) Let the factor base $\Gamma = \{2, 3, 5\}$.

[2] (Compute and Factor $\beta^e \bmod p$) Randomly choose $e < p$, compute and factor $\beta^e \bmod p = 11^e \bmod 29$ as follows:

(1) $11^2 \equiv 5 \pmod{29}$ (success),
(2) $11^3 \equiv 2 \cdot 13 \pmod{29}$ (fail),
(3) $11^5 \equiv 2 \cdot 7 \pmod{29}$ (fail),
(4) $11^6 \equiv 3^2 \pmod{29}$ (success),
(5) $11^7 \equiv 2^3 \cdot 3 \pmod{29}$ (success),
(6) $11^9 \equiv 2 \cdot 7 \pmod{29}$ (success).

[3] (Solve the systems of congruences for the quantities $\log_\beta p_i$)

(1) $\log_{11} 5 \equiv 2 \pmod{28}$,
(4) $\log_{11} 3 \equiv 3 \pmod{28}$,
(6) $\log_{11} 2 \equiv 9 \pmod{28}$,
(5) $2 \cdot \log_{11} 2 + \log_{11} 3 \equiv 7 \pmod{28}$,
$\log_{11} 3 \equiv 17 \pmod{28}$.

[4] (Compute and Factor $\alpha\beta^e \bmod p$) Randomly choose $e < p$, compute and factor $\alpha\beta^e \bmod p = 7 \cdot 11^e \bmod 29$ as follows:

$7 \cdot 11 \equiv 19 \pmod{29}$ (fail),
$7 \cdot 11^2 \equiv 2 \cdot 3 \pmod{29}$ (success).

Thus

$$\log_{11} 7 \equiv \log_{11} 2 + \log_{11} 3 - 2 \equiv 24 \pmod{28}.$$

This is true since

$$11^{24} \equiv 7 \pmod{29}.$$

For more than ten years since its invention, Adleman's method and its variants were the fastest algorithms for computing discrete logarithms. But the situation changed in 1993 Gordon [2] proposed an algorithm for computing discrete logarithms in GF(p). Gordon's algorithm is based on the Number Field Sieve (NFS) for integer factorization, with the heuristic expected running time

$$\mathcal{O}\left(\exp\left(c(\log p)^{1/3}(\log\log p)^{2/3}\right)\right),$$

the same as that used in factoring. The algorithm can be briefly described as follows:

Algorithm 5.4 (Gordon's NFS) This algorithm computes the discrete logarithm x such that $a^x \equiv b \pmod{p}$ with input a, b, p, where a and b are generators and p is prime:

[1] (Precomputation): Find the discrete logarithms of a factor base of small rational primes, which must only be done once for a given p.
[2] (Compute Individual Logarithms): Find the logarithm for each $b \in \mathbb{F}_p$ by finding the logarithms of a number of "medium-sized" primes.

[3] (Compute the Final Logarithm): Combine all the individual logarithms (by using the Chinese Remainder theorem) to find the logarithm of b.

Problems for Section 5.4

1. Let the factor base $\Gamma = \{2, 3, 5\}$. Use the index calculus method to find the discrete logarithm k:

$$k \equiv \log_{11} 7 \pmod{29}.$$

2. Let the factor base $\Gamma = \{2, 3, 5, 7\}$. Use the index calculus method to find the discrete logarithm k:

$$k \equiv \log_2 37 \pmod{131}.$$

3. Use the index calculus with factor base $\Gamma = (2, 3, 5, 7, 11)$ to solve the DLP problem

$$k \equiv \log_7 13 \pmod{2039}.$$

4. Let

$$\begin{aligned} p &= 31415926535897932384626433832795028841971693993751058209 \\ &= 74944592307816406286208998628034825342117067982148086513 \\ &= 28230664709384460955058223172535940812848123729 9, \\ x &= 2, \\ y &= 27182818284590452353602874713526624977572470936999595749 \\ &\quad 66967627724076630353547594571382178525166427427466391932 \\ &\quad 00305992181741359662904357290033429526059563073 8. \end{aligned}$$

   Use Gordon's index calculus method (Algorithm 5.4) to compute the k such that

$$y \equiv x^k \pmod{p}.$$

5. Give a heuristic argument for the expected running time

$$\mathcal{O}\left(\exp\left(c(\log p)^{1/3}(\log\log p)^{2/3}\right)\right)$$

   of Gordon's index calculus method (based on NFS) for DLP.
6. Let the group $G = (\mathbb{Z}/p\mathbb{Z})^*$ with p prime, let also $x, y \in G$ such that $y \equiv x^k \pmod{p}$. Use smooth numbers to show that the discrete logarithm $k \equiv \log_x y \pmod{p}$ can be computed in expected time $L(p)^{\sqrt{2}+o(1)}$.

5.5 Elliptic Curve Discrete Logarithms

The Elliptic Curve Discrete Logarithm Problem (ECDLP) is of fundamental importance to ECC (elliptic curve cryptography): Let $E/\mathbb{F}_p$ be an elliptic curve over a finite field $\mathbb{F}_p$, say, given by a Weierstrass equation

$$E : y^2 \equiv x^3 + ax + b \pmod{p}, \tag{5.18}$$

P and Q the two points in the elliptic curve group $E(\mathbb{F}_p)$. Then the ECDLP is to find the integer k (assuming that such an integer k exists)

$$k \equiv \log_T S \pmod{p} \tag{5.19}$$

such that

$$Q \equiv kP \pmod{p}. \tag{5.20}$$

Formally, the Elliptic Curve Discrete Logarithm Problem (ECDLP) could be defined as follows:

$$\left.\begin{array}{ll} \text{Input :} & P, Q \in E(\mathbb{F}_p) \\ \text{Output :} & k \in \mathbb{Z}_{>1} \text{ with } Q \equiv kP \pmod{q} \\ & \text{if such a } k \text{ exists.} \end{array}\right\} \tag{5.21}$$

The ECDLP is little bit more difficult than the DLP, on which the elliptic curve digital signature algorithm/elliptic curve digital signature standard (ECDSA/ECDSS) [37] is based on. As ECDLP is the generalization of DLP, which extends, for example, the multiplicative group $\mathbb{F}_p^*$ to the elliptic curve group $E(\mathbb{F}_p)$, many methods for DLP, even for IFP, can be extended to ECDLP, for example, the Baby-step Giant-step for DLP, Pollard's ρ and λ Methods for IFP and DLP; the Silver–Pohlig–Hellman Method for DLP, can also be naturally extended to ECDLP. In what follows, we present an example of solving ECDLP by an analog of the Silver–Pohlig–Hellman Method for elliptic curves over $\mathbb{F}_p^*$.

Example 5.6 Let

$$Q \equiv kP \pmod{1009}$$

where

$$\begin{cases} E : y^2 \equiv x^3 + 71x + 602 \pmod{1009} \\ P = (1, 237) \\ Q = (190, 271) \\ \text{order}(E(\mathbb{F}_{1009})) = 1060 = 2^2 \cdot 5 \cdot 53 \\ \text{order}(P) = 530 = 2 \cdot 5 \cdot 53 \end{cases}$$

Find k.

[1] Find the individual logarithm modulo 2: as $(530/2) = 265$, we have

$$\begin{cases} P_2 = 265P = (50, 0) \\ Q_2 = 265Q = (50, 0) \\ Q_2 = P_2 \\ k \equiv 1 \pmod{2} \end{cases}$$

[2] Find the individual logarithm modulo 5: as $530/5 = 106$, we have

$$\begin{cases} P_5 = 106P = (639, 160) \\ Q_5 = 106Q = (639, 849) \\ Q_5 = -P_5 \\ k \equiv 4 \pmod{5} \end{cases}$$

[3] Find the individual logarithm modulo 53: as $530/53 = 10$, we have

$$\begin{cases} P_{53} = 10P = (32, 737) \\ Q_{53} = 10Q = (592, 97) \\ Q_{53} = 48P_{53} \\ k \equiv 48 \pmod{53} \end{cases}$$

[4] Use the Chinese Remainder theorem to combine the individual logarithms to get the final logarithm:

$$\text{CHREM}([1, 4, 48], [2, 5, 53]) = 419.$$

That is,

$$(190, 271) \equiv 419(1, 237) \pmod{1009},$$

or alternatively,

$$(190, 271) \equiv \underbrace{(1, 237) + \cdots + (1, 237)}_{419 \text{ summands}} \pmod{1009}.$$

The index calculus for DLP is however generally not suitable for ECDLP as it is not for general groups. In what follows, we introduce a method, called xedni calculus for ECDLP. The xedni calculus was first proposed by Joseph Silverman in [42] and [44], and analyzed in [45]. It is called *xedni calculus* because it "stands index calculus on its head." The xedni calculus is a new method that *might* be used to solve the ECDLP, although it has not yet been tested in practice. It can be described as follows:

[1] Choose points in $E(\mathbb{F}_p)$ and lift them to points in $\mathbb{Z}^2$.
[2] Choose a curve $E(\mathbb{Q})$ containing the lift points; use Mestre's Method (in reverse) to make rank $E(\mathbb{Q})$ small.

Whilst the index calculus works in reverse:

[1] Lift $E/\mathbb{F}_p$ to $E(\mathbb{Q})$; use Mestre's Method to make rank $E(\mathbb{Q})$ large.
[2] Choose points in $E(\mathbb{F}_p)$ and try to lift them to points in $E(\mathbb{Q})$.

A brief description of the xedni algorithm is as follows.

Algorithm 5.5 (Xedni calculus for the ECDLP) Let $\mathbb{F}_p$ be a finite field with p elements (p prime), $E/\mathbb{F}_p$ an elliptic curve over $\mathbb{F}_p$, say, given by

$$E : y^2 + a_{p,1}xy + a_{p,3}y = x^3 + a_{p,2}x^2 + a_{p,4}x + a_{p,6}. \tag{5.22}$$

N_p the number of points in $E(\mathbb{F}_p)$, S and T the two points in $E(\mathbb{F}_p)$. This algorithm tries to find an integer k

$$k = \log_T S \tag{5.23}$$

such that

$$S = kT \quad \text{in } E(\mathbb{F}_p). \tag{5.24}$$

[1] Fix an integer $4 \leq r \leq 9$ and an integer M which is a product of small primes.
[2] Choose r points:

$$P_{M,i} = [x_{M,i}, y_{M,i}, z_{M,i}], \quad 1 \leq i \leq r \tag{5.25}$$

having integer coefficients and satisfying
[a] the first 4 points are [1, 0, 0], [0, 1, 0], [0, 0, 1], and [1, 1, 1].

[b] for every prime $l \mid M$, the matrix $\mathbf{B}(P_{M,1}, \ldots, P_{M,r})$ has maximal rank modulo l.

Further choose coefficients $u_{M,1}, u_{M,2}, \ldots, u_{M,10}$ such that the points $P_{M,1}, P_{M,2}, \ldots, P_{M,r}$ satisfy the congruence:

$$u_{M,1}x^3 + u_{M,2}x^2y + u_{M,3}xy^2 + u_{M,4}y^3 + u_{M,5}x^2z + u_{M,6}xyz + u_{M,7}y^2z + u_{M,8}xz^2 + u_{M,9}yz^2 + u_{M,10}z^3 \equiv 0 \pmod{M}. \tag{5.26}$$

[3] Choose r random pair of integers (s_i, t_i) satisfying $1 \le s_i, t_i < N_p$, and for each $1 \le i \le r$, compute the point $P_{p,i} = (x_{p,i}, y_{p,i})$ defined by

$$P_{p,i} = s_i S - t_i T \quad \text{in } E(\mathbb{F}_p). \tag{5.27}$$

[4] Make a change of variables in $\mathbb{P}^2$ of the form

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{5.28}$$

so that the first four points become

$$P_{p,1} = [1, 0, 0], \ P_{p,2} = [0, 1, 0], \ P_{p,3} = [0, 0, 1], \ P_{p,4} = [1, 1, 1].$$

The equation for E will then have the form:

$$u_{p,1}x^3 + u_{p,2}x^2y + u_{p,3}xy^2 + u_{p,4}y^3 + u_{p,5}x^2z + u_{p,6}xyz + u_{p,7}y^2z + u_{p,8}xz^2 + u_{p,9}yz^2 + u_{p,10}z^3 = 0. \tag{5.29}$$

[5] Use the Chinese Remainder theorem to find integers $u'_1, \ldots, u'_{10}$ satisfying

$$u'_i \equiv u_{p,i} \pmod{p} \quad \text{and} \quad u'_i \equiv u_{M,i} \pmod{M}, \tag{5.30}$$

for all $1 \le i \le 10$.

[6] Lift the chosen points to $\mathbb{P}^2(\mathbb{Q})$. That is, choose points

$$P_i = [x_i, y_i, z_i], \quad 1 \le i \le r, \tag{5.31}$$

with integer coordinates satisfying

$$P_i \equiv P_{p,i} \pmod{p} \quad \text{and} \quad P_i \equiv P_{M,i} \pmod{M}, \tag{5.32}$$

for all $1 \le i \le r$. In particular, take

$$P_1 = [1, 0, 0], P_2 = [0, 1, 0], P_3 = [0, 0, 1], P_4 = [1, 1, 1].$$

[7] Let $\mathbf{B} = \mathbf{B}(P_1, \ldots, P_r)$ be the matrix of cubic monomials defined earlier. Consider the system of linear equations:

$$\mathbf{Bu} = 0. \tag{5.33}$$

Find a small integer solution $\mathbf{u} = [u_1, \ldots, u_{10}]$ to (5.33) which has the additional property

$$\mathbf{u} \equiv [u'_1, \ldots, u'_{10}] \pmod{M_p}, \tag{5.34}$$

where $u'_1, \ldots, u'_{10}$ are the coefficients computed in Step [5]. Let $C_{\mathbf{u}}$ denote the associated cubic curve:

$$\begin{aligned} C_{\mathbf{u}} : u_1x^3 + u_2x^2y + u_3xy^2 + u_4y^3 + u_5x^2z + u_6xyz \\ + u_7y^2z + u_8xz^2 + u_9yz^2 + u_{10}z^3 = 0. \end{aligned} \tag{5.35}$$

[8] Make a change of coordinates to put $C_{\mathbf{u}}$ into standard minimal Weierstrass form with the point $P_1 = [1, 0, 0]$ the point at infinity, $\mathcal{O}$. Write the resulting equation as

$$E_{\mathbf{u}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{5.36}$$

with $a_1, \ldots, a_6 \in \mathbb{Z}$, and let $Q_1, Q_2, \ldots, Q_r$ denote the images of $P_1, P_2, \ldots, P_r$ under this change of coordinates (so in particular, $Q_1 = \mathcal{O}$). Let $c_4(\mathbf{u})$, $c_6(\mathbf{u})$, and $\Delta(\mathbf{u})$ be the usual quantities in Silverman (2000) associated to the equation (5.36).

[9] Check if the points $Q_1, Q_2, \ldots, Q_r \in E_{\mathbf{u}}(\mathbb{Q})$ are independent. If they are, return to Step [2] or [3]. Otherwise compute a relation of dependence

$$n_2Q_2 + n_3Q_3 + \cdots + n_rQ_r = \mathcal{O}, \tag{5.37}$$

set

$$n_1 = -n_2 - n_3 - \cdots - n_r, \tag{5.38}$$

and continue with the next step.

[10] Compute

$$s = \sum_{i=1}^{r} n_i s_i \quad \text{and} \quad t = \sum_{i=1}^{r} n_i t_i. \tag{5.39}$$

If $\gcd(s, n_p) > 1$, go to Step [2] or [3]. Otherwise compute an inverse $ss' \equiv 1 \pmod{N_p}$. Then

$$\log_T S \equiv s't \pmod{N_p}, \tag{5.40}$$

and the ECDLP is solved.

Table 5.1 Algorithms for IFP, DLP, and ECDLP

IFP	DLP	ECDLP
Trial divisions	Baby-step Giant-step	Baby-step Giant-step
Pollard's ρ Method	Pollard's ρ, λ Methods	Pollard's ρ, λ Mmethods
CFRAC/MPQS	Index calculus	
NFS	NFS	
Xedni calculus	Xedni calculus	Xedni calculus
Quantum algorithms	Quantum algorithms	Quantum algorithms

As can be seen, the basic idea in the above algorithm is that we first choose points $P_1, P_2, \ldots, P_r$ in $E(\mathbb{F}_p)$ and lift them to points $Q_1, Q_2, \ldots, Q_r$ having integer coordinates, then we choose an elliptic curve $E(\mathbb{Q})$ that goes through the points $Q_1, Q_2, \ldots, Q_r$, and finally check if the points $Q_1, Q_2, \ldots, Q_r$ are *dependent*. If they are, the ECDLP is almost solved. Thus, the goal of the xedni calculus is to find an instance where an elliptic curve has *smaller* than expected rank. Unfortunately, a set of points $Q_1, Q_2, \ldots, Q_r$ as constructed above will usually be *independent*. So, it will not work. To make it work, a congruence method, due to Mestre, is used *in reverse* to produce the lifted curve E having smaller than expected rank.[1] Again unfortunately, Mestre's Method is based on some deep ideas and unproved conjectures in analytic number theory and arithmetic algebraic geometry, it is not possible for us at present to give even a rough estimate of the algorithm's running time. So, we know virtually nothing about the complexity of the xedni calculus. We also do not know if the xedni calculus will be practically useful; it may be completely useless from a practical point of view. Much needs to be done before we will have a better understanding of the xedni calculus.

The index calculus is probabilistic, subexponential-time algorithm applicable for both the Integer Factorization Problem (IFP) and the finite field Discrete Logarithm Problem (DLP). However, there is no known subexponential-time algorithm for the Elliptic Curve Discrete Logarithm Problem (ECDLP); the index calculus will not work for the ECDLP. The *xedni calculus*, on the other hand, is applicable to ECDLP (it is in fact also applicable to IFP and DLP), but unfortunately its complexity is essentially unknown. From a computability point of view, xedni calculus is applicable to IFP, DLP, and ECDLP, but from a complexity point of view, the xedni calculus may turn out to be useless (i.e., not at all practical). As for quantum algorithms, we now know that IFP, DLP, and ECDLP can all be solved in polynomial-time if a quantum computer is available for use. However, the problem with quantum algorithms is that a practical quantum computer is out of reach in today's technology. We summarize various algorithms for IFP, DLP, and ECDLP in Table 5.1.

Finally, we conclude that we do have algorithms to solve the IFP, DLP, and ECDLP; the only problem is that we do not have an efficient algorithm, nor has any one proved that such an efficient algorithm exists. From a computational complexity point of view, a $\mathcal{P}$-type problem is easy to solve, whereas an $\mathcal{NP}$-type problem is easy to verify [7], so the IFP, DLP, and ECDLP are clearly in $\mathcal{NP}$. For example, it might be difficult (indeed, it is difficult at present) to factor a large integer, but it is easy to verify whether or not a given factorization

[1] Mestre's original method is to produce elliptic curves of large rank.

is correct. If $\mathcal{P} = \mathcal{NP}$, then two types of the problems are the same, the factorization is difficult only because no one has yet been clever enough to find an easy/efficient algorithm (it may turn out that the integer factorization problem is indeed $\mathcal{NP}$-hard, regardless of the cleverness of the human beings). Whether or not $\mathcal{P} = \mathcal{NP}$ is one of the biggest open problems in both mathematics and computer science, and it is listed in the first of the seven Millennium Prize Problems by the Clay Mathematics Institute in Boston of May 24 2000 [6]. The struggle continues and more research needs to be done before we can say anything about whether or not $\mathcal{P} = \mathcal{NP}$!

Problems for Section 5.5

1. As Shank's Baby-step Giant-step Method works for arbitrary groups, it can be extended, of course, to elliptic curve groups.
   (1) Develop an elliptic curve analog of Shank's algorithm to solve the ECDLP.
   (2) Use the analog algorithm to solve the following ECDLP, that is, to find k such that

$$Q \equiv kP \pmod{41}$$

   where $E/\mathbb{F}_{41} : y^2 \equiv x^3 + 2x + 1 \pmod{41}$, $P = (0, 1)$ and $Q = (30, 40)$.
2. Poland's ρ and λ Methods for IFP/DLP can also be extended to ECDLP.
   (1) Develop an elliptic curve analog of Poland ρ algorithm to solve the ECDLP.
   (2) Use the analog algorithm to solve the following ECDLP: Find k such that

$$Q \equiv kP \pmod{p}$$

   where $E\mathbb{F}_{1093} : y^2 \equiv x^3 + x + 1 \pmod{1093}$, $P = (0, 1)$ and $Q = (413, 959)$.
3. (Extend the Silver–Pohlig–Hellman Method)
   (1) Develop an elliptic curve analog of Silver-Pohlig-Hellman method for ECDLP.
   (2) Use this analog method to solve the following ECDLP: Find k such that

$$Q \equiv kP \pmod{p}$$

   where $E\mathbb{F}_{599} : y^2 \equiv x^3 + 1 \pmod{1093}$, $P = (60, 19)$ and $Q = (277, 239)$.
4. In 1993, Menezes, Okamota, and Vanstone developed an algorithm for ECDLP over $\mathbb{F}_{p^m}$ with p^m prime power. Give a description and complexity analysis of this algorithm.
5. Let $E\backslash\mathbb{F}_p$ be the elliptic curve E over $\mathbb{F}_p$ with p prime, where E is defined by

$$y^2 = x^3 + ax + b.$$

   (1) Let $P, Q \in E$ with $P \neq \pm Q$ are two points on E. Find the addition formula for computing $P + Q$.
   (2) Let $P \in E$ with $P \neq -P$. Find the addition formula for computing $2P$.
   (3) Let $E\backslash\mathbb{F}_{23}$ be as follows:

$$E\backslash\mathbb{F}_{23} : y^2 \equiv x^3 + x + 4 \pmod{23}.$$

Find all the points, $E(\mathbb{F}_{23})$, including the point at infinity, on the E.

(4) Let $P = (7, 20)$ and $Q = (17, 14)$ be in $E\backslash\mathbb{F}_{23}$ defined above, find $P + Q$ and $2P$.

(5) Let $Q = (13, 11)$ and $P = (0, 2)$ such that $Q \equiv kP \pmod{23}$. Find $k = \log_P Q \pmod{23}$, the discrete logarithm over $E(\mathbb{F}_{23})$.

6. Let the elliptic curve be as follows:

$$E\backslash\mathbb{F}_{151} : y^2 \equiv x^3 + 2x \pmod{151}$$

with order 152. A point $P = (97, 26)$ with order 19 is given. Let also $Q = (43, 4)$ such that

$$Q \equiv kP \pmod{151}.$$

Find $k = \log_P Q \pmod{151}$, the discrete logarithm over $E(\mathbb{F}_{151})$.

7. Let the elliptic curve be as follows:

$$E\backslash\mathbb{F}_{43} : y^2 \equiv x^3 + 39x^2 + x + 41 \pmod{43}$$

with order 43. Find the ECDLP

$$k = \log_P Q \pmod{43},$$

where $P = (0, 16)$ and $Q = (42, 32)$.

8. Let the elliptic curve be as follows:

$$E\backslash\mathbb{F}_{1009} : y^2 \equiv x^3 + 71x + 602 \pmod{1009}.$$

Find the ECDLP

$$k' = \log'_P Q' \pmod{53}$$

in

$$Q' = (529, 97) = k'(32, 737) = k'P'$$

in the subgroup of order 53 generated by $P' = (32, 737)$.

9. In November 1997, Certicom, a computer security company in Waterloo, Canada (see the official webpage:
`http://www.certicom.com/index.php?action=ecc,ecc_challenge`) introduced the Elliptic Curve Cryptosystem (ECC) Challenge. These problems aim at increasing industry understanding and appreciation of the difficulty of ECDLP and encouraging and stimulating further research in the security analysis of ECC. The challenge is to compute the ECC private keys from the given list of ECC public keys and associated system parameters. This is the type of problem facing an adversary who wishes to attack ECC. These problems are defined on curves either over $\mathbb{F}_{2^m}$ or over $\mathbb{F}_p$ with p prime (see Table 5.2 and Table 5.3). Also there are

three levels of difficulty associated with the curves: Exercise level (with bits less than 109), rather easy level (with bits 109–131), and very hard level (with bits 163–359). Readers who are interested in solving real-world ECDLP problems are suggested to try to solve the problems listed in Table 5.2 and Table 5.3, particularly those with the question marks "?" as they are still open.

Table 5.2 Elliptic curves over $\mathbb{F}_{2^m}$

Curve	Field size (in bits)	Estimated number of machine days	Prize in US dollars	Status
ECC2K-95	97	8637	$5000	May 1998
ECC2-97	97	180448	$5000	Sept 1999
ECC2K-108	108	1.3×106	$10000	April 2000
ECC2-109	109	2.1×107	$10000	April 2004
ECC2K-130	131	2.7×109	$20000	?
ECC2-131	131	6.6×1010	$20000	?
ECC2-163	163	2.9×1015	$30000	?
ECC2K-163	163	4.6×1014	$30000	?
ECC2-191	191	1.4×1020	$40000	?
ECC2-238	239	3.0×1027	$50000	?
ECC2K-238	239	1.3×1026	$50000	?
ECC2-353	359	1.4×1045	$100000	?
ECC2K-358	359	2.8×1044	$100000	?

Table 5.3 Elliptic curves over $\mathbb{F}_p$

Curve	Field size (in bits)	Estimated number of machine days	Prize in US dollars	Status
ECCp-97	97	71982	$5000	March 1998
ECCp-109	109	9×107	$10000	Nov 2002
ECCp-131	131	2.3×1010	$20000	?
ECCp-163	163	2.3×1015	$30000	?
ECCp-191	191	4.8×1019	$40000	?
ECCp-239	239	1.4×1027	$50000	?
ECC2p-359	359	3.7×1045	$100000	?

10. In ECCp-109, given

$$
\begin{aligned}
&E(\mathbb{F}_p) : y^2 \equiv x^3 + ax + b \pmod{p},\\
&p = 564538252084441556247016902735257,\\
&a = 321094768129147601892514872825668,\\
&b = 430782315140218274262276694323197,\\
&x_p = 973390109870590665231561339089 35,
\end{aligned}
$$

$$y_p = 149670372846169285760682371978898,$$
$$x_q = 446467696974058610576308611884284,$$
$$y_q = 522968098895785888047540374779097,$$

show that the following value of k

$$k = 281183840311601949668207954530684$$

is the correct value satisfying

$$Q(x_q, y_q) \equiv k \cdot P(x_p, y_p) \pmod{p}.$$

11. In ECCp-131, given

$$E(\mathbb{F}_p) : y^2 \equiv x^3 + ax + b \pmod{p},$$
$$p = 1550031797834347859248576414813139942411,$$
$$a = 1399267573763578815877905235971153316710,$$
$$b = 1009296542191532464076260367525816293976,$$
$$x_p = 1317953763239595888465524145589872695690,$$
$$y_p = 434829348619031278460656303481105428081,$$
$$x_q = 1247392211317907151303247721489640699240,$$
$$y_q = 207534858442090452193999571026315995117,$$

find the correct value of k such that

$$Q(x_q, y_q) \equiv k \cdot P(x_p, y_p) \pmod{p}.$$

5.6 Bibliographic Notes and Further Reading

In this chapter, we presented some important modern algorithms for the Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP).

For general references on DLP and methods for solving DLP, it is suggested that readers consult: [8–26].

The Baby-step and Giant-step Method for DLP was originally proposed by Shanks in 1971 [3]. Pohlig–Hellman method of DLP was proposed in [5]. The ρ and λ Methods were proposed by Pollard in [4].

The currently most powerful method, the index calculus, for DLP is discussed in many references such as [1, 2, 27, 28]. Generally speaking, the index calculus belongs to a wide

range of methods, such as the Continued FRACtion Method (CFRAC), Quadratic Sieve (QS), and Number Field Sieve (NFS). As far as Number Field Sieve is concerned, there is an analog method, called Function Field Sieve, based on the algebraic function field which is just an analog of the number field. FFS, just the as NFS, can be used for solving both IFP and DLP. For more information on FFS, see [29, 30].

The ECDLP and methods for ECDLP are discussed in, for example, [31–43]. In particular, the xedni calculus for ECDLP was proposed in [44] and analyzed in [45].

References

1. L. M. Adleman, "A Subexponential Algorithmic for the Discrete Logarithm Problem with Applications to Cryptography", *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1979, pp. 55–60.
2. D. M. Gordon, "Discrete Logarithms in GF(p) using the Number Field Sieve", *SIAM Journal on Discrete Mathematics*, **6**, 1, 1993, pp. 124–138.
3. D. Shanks. "Class Number, A theory of Factorization and Genera". In: Proceedings of Symposium of Pure Mathematics **20**, AMS, Providence, R.I., 1971, pp. 415–440.
4. J. M. Pollard, "Monte Carlo Methods for Index Computation $\pmod p$", *Mathematics of Computation*, **32**, 1980, pp. 918–924.
5. S. C. Pohlig and M. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance", *IEEE Transactions on Information Theory*, **24**, 1978, pp. 106–110.
6. S. Cook, "The P versus NP Problem", *The Millennium Prize Problems*. Edited by J. Carlson, A. Jaffe, and A. Wiles, Clay Mathematical Institute and American Mathematical Institute, 2006, pp. 87–104.
7. M. R. Gary and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
8. H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer-Verlag, 1993.
9. H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, CRC Press, 2006.
10. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer-Verlag, 2005.
11. T. Hayashi, N. Shinohara, L. Wang, et al., "Solving a 676-Bit Discrete Logarithm Problem in GF(3^{6n})", *Public Key Cryptography – PKC 2010*, Lecture Notes in Computer Science **6056**, Springer, 2010, pp. 351–367.
12. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms", *IEEE Transactions on Information Theory*, **31**, 1985, pp. 496–472.
13. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer-Verlag, 1994.
14. N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics **3**, Springer, 1998.
15. K. S. McCurley, "The Discrete Logarithm Problem", *Cryptology and Computational Number Theory*. Edited by C. Pomerance, Proceedings of Symposia in Applied Mathematics **42**, American Mathematics Society, 1990, pp. 49–74.
16. K. S. McCurley, "Odds and Ends from Cryptology and Computational Number Theory", *Cryptology and Computational Number Theory*. Edited by C. Pomerance, Proceedings of Symposia in Applied Mathematics **42**, American Mathematics Society, 1990, pp. 49–74.
17. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptosystems*, CRC Press, 1996.
18. A. M. Odlyzko, "Discrete Logarithms in Finite Fields and their Cryptographic Significance", *Advances in Cryptography*, EUROCRYPT '84, Proceedings, Lecture Notes in Computer Science **209**, Springer, 1984, pp. 225–314.
19. A. M. Odlyzko, "Discrete Logarithms: the Past and the future", *Design, Codes, and Cryptography*, **19**, (2000), pp. 129–145.
20. C. Pomerance, "Elementary Thoughts on Discrete Logarithms", *Algorithmic Number Theory*. Edited by J. P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 385–395.
21. H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhäuser, Boston, 1990.

22. O. Schirokauere, "The Impact of the Number Field Sieve on the Discrete Logarithm Problem in Finite Fields", *Algorithmic Number Theory*. Edited by J. P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 421–446.
23. V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005.
24. S. S. Wagstaff, Jr., *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC Press, 2002.
25. S. Y. Yan, "Computing Prime Factorization and Discrete Logarithms: From Index Calculus to Xedni Calculus", *International Journal of Computer Mathematics*, **80**, 5, 2003, pp. 573–590.
26. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.
27. D. M. Gordon and K. S. McCurley, "Massively Parallel Computation of Discrete Logarithms", *Advances in Cryptology – Crypto '92*, Lecture Notes in Computer Science **740**, Springer, 1992, pp. 312–323.
28. O. Schirokauer, D. Weber and T. Denny, "Discrete Logarithms: The Effectiveness of the Index Calculus Method", *Algorithmic Number Theory (ANTS-II)*, Lecture Notes in Computer Science **1122**, Springer, 1996, pp. 337–362.
29. L. M. Adleman, "The Function Field Sieve", *Algorithmic Number Theory (ANTS-I)*, Lecture Notes in Computer Science **877**, Springer, 1994, pp. 108–121.
30. L. M. Adleman, "Function Field Sieve Method for Discrete Logarithms over Finite Fields", *AInformation and Computation*, **151**, 1999, pp. 5–16.
31. I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
32. I. Blake, G. Seroussi, and N. Smart, *Advances in Elliptic Curves Cryptography*, Cambridge University Press, 2005.
33. H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, CRC Press, 2006.
34. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer, 2005.
35. D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.
36. J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, 2008.
37. D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signatures Algorithm (ECDSA)", *International Journal of Information Security*, **1**, 1, 2001, pp. 36–63.
38. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer, 1994.
39. N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics **3**, Springer, 1998.
40. A. Menezes, T. Okamoto, and S. A. Vanstone, "Reducing Elliptic Curve Logarithms in a Finite Field", *IEEE Transactions on Information Theory*, **39**, 5, 1993, pp. 1639–1646.
41. J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics **106**, 2nd Edition, Springer, 2010.
42. J. H. Silverman and J. Suzuki, "Elliptic Curve Discrete Logarithms and the Index Calculus", *Advances in Cryptology – ASIACRYPT '98*, Lecture Notes in Computer Science **1514**, Springer, 1998, pp. 110–125.
43. L. Washington, *Elliptic Curves: Number Theory and Cryptography*, 2nd Edition, Chapman & Hall/CRC, 2008.
44. J. H. Silverman, "The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem", *Designs, Codes and Cryptography*, **20**, 2000, pp. 5-40.
45. M. J. Jacobson, N. Koblitz, J. H. Silverman, et al. "Analysis of the Xedni Calculus Attack", *Designs, Codes and Cryptography*, **20**, 2000, pp. 41-64.

Part III

Modern Cryptography

Cryptography (from the Greek *Kryptós*, "hidden", and *gráphein*, "to write") is the study of the principles and techniques by which information can be concealed in ciphertexts and later revealed by legitimate users employing the secret key, but in which it is either impossible or computationally infeasible for an unauthorized person to do so. Cryptanalysis (from the Greek *Kryptós* and *analýein*, "to loosen") is the science (and art) of recovering information from ciphertexts without knowledge of the key. Both terms are subordinate to the more general term *cryptology* (from the Greek *Kryptós* and *lógos*, "word"). That is,

$$\text{Cryptology} := \text{Cryptography} \oplus \text{Cryptanalysis}$$

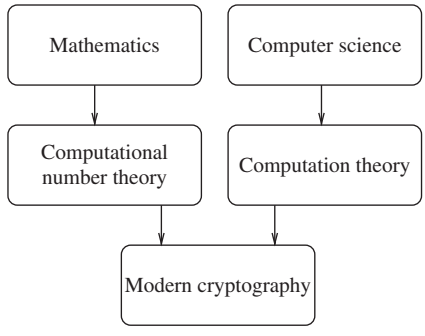
$$\swarrow \quad \searrow$$

$$\begin{array}{rr} \text{Public-Key} & \text{Secret-Key} \\ \text{Cryptography} & \text{Cryptography} \end{array}$$

$$\swarrow \quad \searrow$$

$$\begin{array}{rr} \text{Encryption} & \text{Decryption} \\ \text{(Public-Key)} & \text{(Private-Key)} \\ \text{Signature Generation} & \text{Signature Verification} \\ \text{(Private-Key)} & \text{(Public-Key)} \end{array}$$

The history of cryptography, more specifically, secret-key cryptography, is at least as old as the human civilization. The history for public-key cryptography is, however, rather short, and the official date of birth of public-key cryptography as well as digital signatures was in fact 1976 when Diffie and Hellman published their seminal paper *New Directions in Cryptography*. In this part of the book, we emphasize modern public-key cryptography and digital signatures. Our approach to cryptography will be more on the combined nature of mathematics and computer science; this can be seen from the following figure:



Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

6

Secret-Key Cryptography

Cryptography starts from secret-key cryptography. After a brief introduction to the basic concepts of cryptography and cryptanalysis, various types of secret-key cryptography shall be discussed in this chapter.

6.1 Cryptography and Cryptanalysis

Cryptography (from the Greek *Kryptós*, "hidden" or "secret", and gráphein, "writing") is the study of the processes of encryption (mapping the original message, called the plaintext, into a secret form, called the the ciphertext, using the encryption key), and decryption (inverting the ciphertext back to the plaintext, using the corresponding decryption key), in such a way that only the intended recipients can decrypt and read the original messages.

$$\text{Cryptography} \stackrel{\text{def}}{=} \text{Encryption} \oplus \text{Decryption}$$

The methods of encryption are often also called ciphers. Cryptanalysis (from the Greek *Kryptós* and *analýein*, "loosening"), on the other hand, is the study of breaking the encryptions without the knowledge of the key:

$$\text{Cryptanalysis} \stackrel{\text{def}}{=} \text{Cryptanalytic Attacks on Encryptions}$$

Cryptology (from the Greek *Kryptós* and lógos, "word") consists of both cryptography and cryptanalysis:

$$\text{Cryptology} \stackrel{\text{def}}{=} \text{Cryptography} \oplus \text{Cryptanalysis}$$

The idea of encryption, decryption, cryptanalysis, and secure communications over an insecure channel, usually a computer network and particularly the Internet, can be depicted as in Figure 6.1. Throughout the book, we shall assume that Bob sends a message to Alice, but Eve wants to cryptanalyze the message:

$$\begin{array}{ccc} \text{Bob} & \xrightarrow{\text{Message}} & \text{Alice.} \\ & \updownarrow & \\ & \text{Eve} & \end{array} \tag{6.1}$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

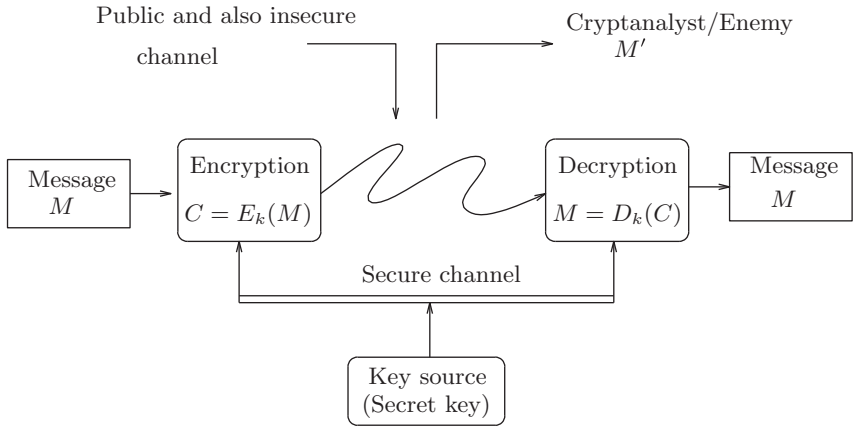


Figure 6.1 Cryptography and cryptanalysis

Modern cryptography, however, is the study of the mathematical systems of encryption and decryption, to solve the security, particularly the network security, problems as follows:

(1) *Confidentiality* or *privacy*: To stop Eve understanding Bob's message to Alice even if she can intercept and get the message.
(2) *Integrity*: To make sure that Bob's message has not been modified by Eve.
(3) *Authentication* or *authorization*: To make sure the message received by Alice is indeed from Bob, not from Eve.
(4) *Nonrepudiation*: To stop Bob later denying the sending of his message. Nonrepudiation is particularly important in electronic commerce since we need to make sure that a consumer cannot deny the authorization of a purchase. It must be noted that in some applications, however, such as in electronic voting, the nonrepudiation feature should, in fact, be avoided, since the voter does not want to disclose the authorization of a vote regardless whether of not he actually did the vote.

Such a mathematical system is called the *cryptographic system*, or *cryptosystem* for short.

Definition 6.1 A conventional *secret-key cryptosystem*, $\mathcal{SKC}$, may be formally defined as follow: (depicted in Figure 6.2):

$$\mathcal{SKC} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, k, E, D) \tag{6.2}$$

where

(1) $\mathcal{M}$ is the set of plaintexts, called the plaintext space.
(2) $\mathcal{C}$ is the set of cipherexts, called the ciphertext space.
(3) $\mathcal{K}$ is the set of keys, called the key space.
(4) $M \in \mathcal{M}$ is a piece of plaintext.

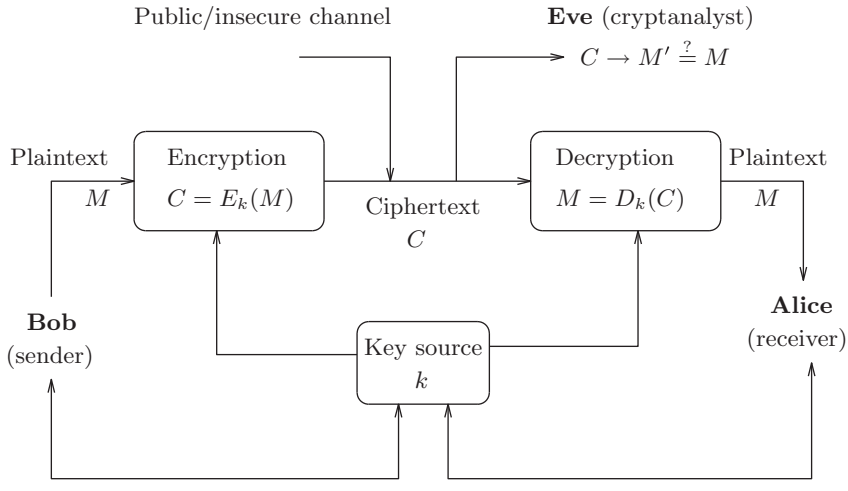


Figure 6.2 Secret-key cryptography

(5) $C \in \mathcal{C}$ is a piece of ciphertext.
(6) $k \in \mathcal{K}$ is the key for both encryption and decryption.
(7) E is the encryption function

$$E_k : M \mapsto C \tag{6.3}$$

where $M \in \mathcal{M}$ maps to $C \in \mathcal{C}$, using the key k, such that

$$C = E_k(M) \tag{6.4}$$

(8) D is the decryption function

$$D_k : C \mapsto M \tag{6.5}$$

where $C \in \mathcal{C}$ maps to $M \in \mathcal{M}$, using the *same* key k again such that

$$M = D_k(C) \tag{6.6}$$

satisfying

$$E_k D_k = 1 \text{ and } D_k(C) = D_k(E_k(M)) = M. \tag{6.7}$$

Cryptanalysis , on the other hand, is the study of the *cryptanalytic attacks* on cryptosystems, aiming at breaking the cryptosystems without using/knowing the keys, but according to the *Kerckhoff principle*, the cryptanalyst who wants to break the cryptosystem knows the cryptosystem. For example, the following is a ciphertext presented by Édouard Lucas at

the 1891 meeting of the French Association for Advancement of Science (see page 388 of Williams 1998) [1]), based on Étienne Bazeries' cylindrical cryptography (see pages 244–250 of Kahn 1976 [2]); it has never been decrypted, and hence is suitable as a good challenge to the interested reader:

XSJOD	PEFOC	XCXFM	RDZME
JZCOA	YUMTZ	LTDNJ	HBUSQ
XTFLK	XCBDY	GYJKK	QBSAH
QHXPE	DBMLI	ZOYVQ	PRETL
TPMUK	XGHIV	ARLAH	SPGGP
VBQYH	TVJYJ	NXFFX	BVLCZ
LEFXF	VDMUB	QBIJV	ZGGAI
TRYQB	AIDEZ	EZEDX	KS

The *security* or the *unbreakability* of any cryptographic system is of paramount importance. There are several different types of security measures for a cryptographic system:

(1) *Unconditionally secure*: A cryptosystem is unconditionally secure if a cryptanalyst cannot determine how to find the plaintext M regardless of how much ciphertext C and computer time/resources he has available to him. A one-time pad (OTP) can be shown to be unconditionally secure, as the key is used only one time (i.e., there are at least as many keys as the plaintexts), the key string is a random string, and the key size is at least as long as the plaintext string. Unconditional security for cryptosystems is called *perfect secrecy*, or *information-theoretic security*. A cryptosystem $\mathcal{S}$ is *unconditionally unbreakable* if $\mathcal{S}$ is unconditionally secure. In general, cryptosystems do not offer perfect secrecy, in particular public-key cryptosystems, such as the RSA cryptosystem described in later sections, cannot be unconditionally secure/breakable since once a ciphertext C is given, its corresponding plaintext M can in principle be recovered by computing all possible plaintexts until C is obtained, an attack called forward search, which will be discussed later. Nevertheless, unconditionally unbreakable cryptosystems exist; it was first proved by Shannon in his 1949 seminal paper in modern cryptography "Communication Theory of Secrecy Systems" [3]. Thus the prominent English mathematician J. E. Littlewood (1885–1977) commented:

> The legend that every cipher is breakable is of course absurd, though still widespread among people who should know better.

(2) *Computationally secure*: A cryptosystem $\mathcal{S}$ is computationally secure or *polynomially secure* if a cryptanalyst cannot decrypt C to get M in polynomial-time (or space). A cryptosystem $\mathcal{S}$ is *computationally unbreakable*, if it is unbreakable in polynomial-time, that is, it is computationally secure. According to the Cook–Karp thesis, any problem that cannot be solved in polynomial-time is *computationally infeasible*, thus,

if the cryptanalytic attack on a cryptosystem is computationally infeasible, then the cryptosystem is computationally secure and computationally unbreakable. There are several types of computationally security:

(a) *Provably secure*: A cryptosystem $\mathcal{S}$ is *provably secure* if the difficulty of breaking it can be shown to be essentially as difficult as solving a well-known and supposedly difficult mathematical problem such as the Integer Factorization Problem, IFP, or the Discrete Logarithm Problem, DLP. For example, the Rabin cryptosystem described later is provably secure, as the security of the Rabin cryptosystem is equivalent to the IFP.

(b) *Practical/conjectured secure*: A cryptosystem $\mathcal{S}$ is *practical secure* if the breaking of the system $\mathcal{S}$ is *conjectured* to be as difficult as solving a well-known and supposedly difficult mathematical problem such as the IFP or the DLP. For example, breaking the most popular public-key cryptosystem, RSA, is conjectured as being as hard as solving the IFP, but so far this has never been proved or disproved. Most of the public-key and secret-key cryptosystems in current use are of this type.

There are several types of possible cryptanalytic attacks on a cryptosystem $\mathcal{S}$, depending on what information the cryptanalyst might already have regarding $\mathcal{S}$:

(1) *Ciphertext-only attack*: Only a piece of ciphertext C is known to the cryptanalyst whose goal is to find the corresponding plaintext M and/or the key k. This is the most difficult type of attack; any cryptosystem vulnerable to this type of attack is considered to be *completely* insecure.
(2) *Known-plaintext attack*: The cryptanalyst has a piece of plaintext M and the corresponding ciphertext C. The goal is the find the key k so that other ciphertexts using the same encryption/key may be decrypted.
(3) *Chosen-plaintext attack*: The cryptanalyst has gained temporary access to the encryption machinery, so he can choose a piece of plaintext M and construct the corresponding ciphertext C. The goal here is to find the key k.
(4) *Chosen-ciphertext attack*: The cryptanalyst has gained temporary access to the decryption machinery, so can choose a piece of ciphertext C and construct the corresponding plaintext M. The goal here is also to find the key k.

A good cryptosystem should resist all of these types of attacks, so that it is impossible for a cryptanalysis to get the key k or to find the plaintext M in polynomial-time.

Remark 6.1 Public-key cryptosystems, such as the RSA cryptosystem described in later sections, give rise to the chosen-ciphertext attack, since the cryptanalyst may specify/obtain some ciphertext using the public key and learn the corresponding plaintext. In fact, all public-key cryptographic systems are vulnerable to a chosen-ciphertext attack, which, however, can be avoided by adding appropriate redundancy or randomness (padding or salting) prior to encryption.

Compared to secret-key cryptography, *public-key cryptography*, or *asymmetric key cryptography* (see Figure 6.3), is surprisingly almost the same (although the idea is different)

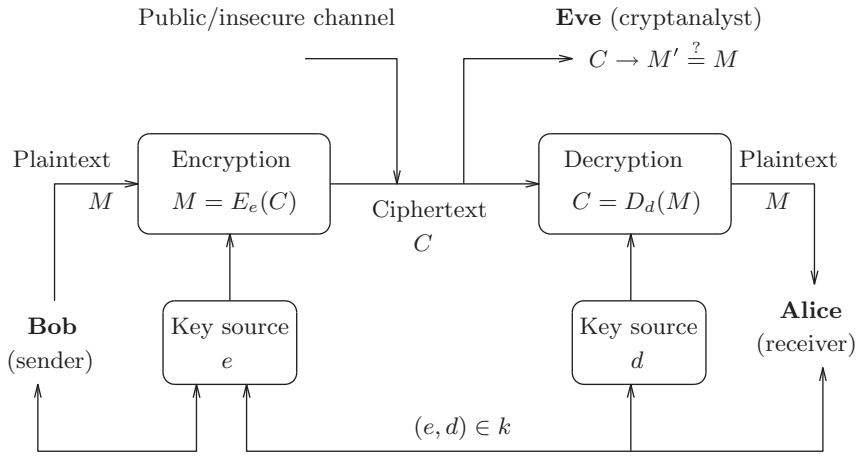


Figure 6.3 Public-key cryptography

as the *secret-key cryptography*, or *symmetric key cryptography*, except that the keys k for encryption and decryption are different. That is, we need two keys, e_k and d_k, such that e_k is used for encryption and d_k for decryption, respectively. As e_k is only used for encryption, it can be made public; only d_k must be kept a secret for decryption. To distinguish public-key cryptosystems from secret-key cryptosystems, e_k is called the *public key*, and d_k the *private key*; only the key used in secret-key cryptosystems is called the *secret key*. Remarkably enough, secret-key cryptography has a very long history, almost as long as our human civilization; whereas public-key cryptography has a rather short history. In fact, as mentioned above, the official date of birth of public-key cryptography was 1976, when Diffie and Hellman, then both at Stanford University, published their seminal paper *New Directions in Cryptography* [4] (see the first page of the paper in Figure 6.4). It was in this seminal paper that they *first* publicly proposed the completely new idea of public-key cryptography as well as digital signatures. Although Diffie and Hellman did not have a practical implementation of their idea, they did propose [4] an alternative key-exchange scheme over the insecure channel, based on the intractability of the DLP problem and using some of the ideas proposed earlier (although published later) by Merkle [5] (published in 1978, but submitted in 1975; see the first page of this paper in Figure 6.5).

Shortly after the publication of Diffie and Hellman's paper, in 1977 Rivest, Shamir, and Adleman, then all at Massachusetts Institute of Technology (MIT), proposed a first workable and practical public-key cryptosystem in 1977 ([6–8]), see the first page of the paper in Figure 6.6. The system is now known as RSA; it was first made public to the world and became famous probably because of Gardner's 1978 paper in Scientific American [7].

It is interesting to note that the British cryptographers Ellis, Cocks, and Williamson at the UK government's Communications-Electronics Security Group (CESG) of the Government Communications Headquarters (GCHQ) also claimed that they secretly discovered the public-key encryption years before the US scientists [9–13]. There are of course two

644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

Abstract—**Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.

Manuscript received June 3, 1976. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173. Portions of this work were presented at the IEEE Information Theory Workshop, Lenox, MA, June 23–25, 1975 and the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21–24, 1976.

W. Diffie is with the Department of Electrical Engineering, Stanford University, Stanford, CA, and the Stanford Artificial Intelligence Laboratory, Stanford, CA 94305.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a multiple access cipher. A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other enciphered in the receiver's public enciphering key and deciphers the messages he receives using his own secret deciphering key.

We propose some techniques for developing public key cryptosystems, but the problem is still largely open.

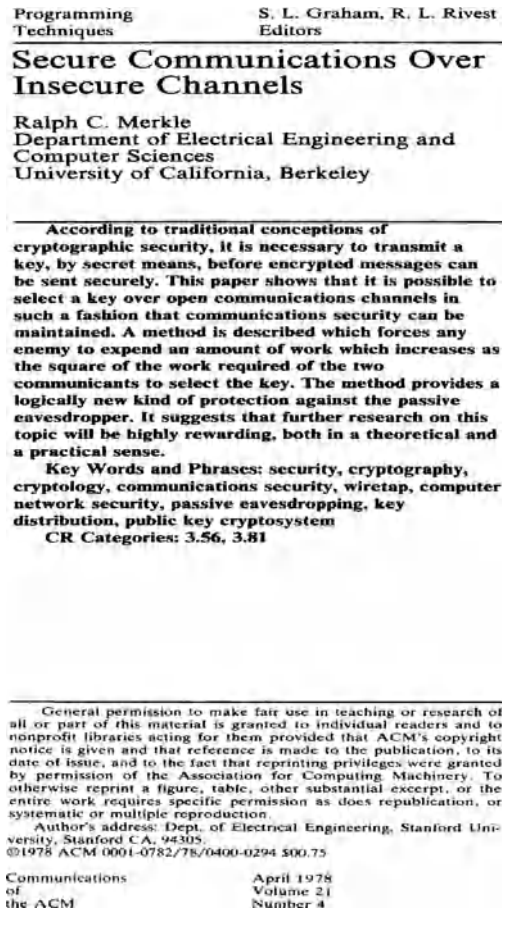
Public key distribution systems offer a different approach to eliminating the need for a secure key distribution channel. In such a system, two users who wish to exchange a key communicate back and forth until they arrive at a key in common. A third party eavesdropping on this exchange must find it computationally infeasible to compute the key from the information overheard. A possible solution to the public key distribution problem is given in Section III, and Merkle [1] has a partial solution of a different form.

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary busi-

Figure 6.4 First page of Diffie and Hellman's paper (Courtesy of IEEE)

different universes of cryptography: public (particularly for people working in academic institutions) and secret (particularly for people working for militaries and governments). Ellis-Cocks-Williamson certainly deserve some credit for their contribution to the development of public-key cryptography. It should be noted that Hellman and his colleagues not only invented the public-key encryption, but also digital signatures which were not mentioned in any of Ellis-Cocks-Williamson's documents/papers.

The implementation of public-key cryptosystems is based on *trapdoor one-way functions*.



Programming Techniques — S. L. Graham, R. L. Rivest Editors

Secure Communications Over Insecure Channels

Ralph C. Merkle
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

According to traditional conceptions of cryptographic security, it is necessary to transmit a key, by secret means, before encrypted messages can be sent securely. This paper shows that it is possible to select a key over open communications channels in such a fashion that communications security can be maintained. A method is described which forces any enemy to expend an amount of work which increases as the square of the work required of the two communicants to select the key. The method provides a logically new kind of protection against the passive eavesdropper. It suggests that further research on this topic will be highly rewarding, both in a theoretical and a practical sense.

Key Words and Phrases: security, cryptography, cryptology, communications security, wiretap, computer network security, passive eavesdropping, key distribution, public key cryptosystem
CR Categories: 3.56, 3.81

Author's address: Dept. of Electrical Engineering, Stanford University, Stanford CA, 94305.

Communications of the ACM — April 1978 Volume 21 Number 4

Figure 6.5 First page of Merkle's paper (Courtesy of ACM)

Definition 6.2 Let S and T be finite sets. A *one-way function*

$$f : S \to T \tag{6.8}$$

is an invertible function satisfying

(1) f is easy to compute, that is, given $x \in S$, $y = f(x)$ is easy to compute.
(2) f^{-1}, the inverse function of f, is difficult to compute, that is, given $y \in T$, $x = f^{-1}(y)$ is difficult to compute.
(3) f^{-1} is easy to compute when a trapdoor (i.e., a secret string of information associated with the function) becomes available.

A function f satisfying only the first two conditions is also called a one-to-one one-way function. If f satisfies further the third condition, it is called a *trapdoor one-way function*.

Programming Techniques S.L. Graham, R.L. Rivest* Editors

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences: (1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key. (2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where e * d ≡ 1(mod (p − 1) * (q − 1)). The security of the system rests in part on the difficulty of factoring the published divisor, n.

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

* Note. This paper was submitted prior to the time that Rivest became editor of the department, and editorial consideration was completed under the former editor, G. K. Manacher.

Authors' Address: MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

$$D(E(M)) = M. \tag{1}$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

$$E(D(M)) = M. \tag{2}$$

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext* C. Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a very *inefficient* method of computing D(C): testing all possible messages M until one such that E(M) = C is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)–(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but

Figure 6.6 First page of RSA's paper (Courtesy of ACM)

Example 6.1 The following functions are one-way functions:

(1) $f : pq \mapsto n$ is a one-way function, where p and q are prime numbers. The function f is easy to compute since the multiplication of p and q can be done in polynomial time. However, the computation of f^{-1}, the inverse of f is hard (this is the IFP).

(2) $f : x \mapsto g^x \bmod N$ is a one-way function. The function f is easy to compute since the modular exponentiation $g^x \bmod N$ can be performed in problem polynomial time. But the computation of f^{-1}, the inverse of f is hard (this is the DLP problem).
(3) $f : x \mapsto x^k \bmod N$ is a trapdoor one-way function, where $N = pq$ with p and q primes, and $kk' \equiv 1 (\bmod \phi(N))$. It is obvious that f is easy to compute since the modular exponentiation $x^k \bmod N$ can be done in polynomial time, but f^{-1}, the inverse of f (i.e., the kth root of x modulo N) is difficult to compute. However, if k', the trapdoor is given, and f can be easily inverted, since $(x^k)^{k'} = x$.

Now we are in a position to introduce the formal definition of public-key cryptography.

Definition 6.3 A public-key cryptosystem , $\mathcal{PKC}$, may be formally defined as follows:

$$\mathcal{PKC} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, e, d, E, D) \tag{6.9}$$

where

(1) $\mathcal{M}$ is the set of plaintexts, called the plaintext space.
(2) $\mathcal{C}$ is the set of cipherexts, called the ciphertext space.
(3) $\mathcal{K}$ is the set of keys, called the key space.
(4) $M \in \mathcal{M}$ is a piece of particular plaintext.
(5) $C \in \mathcal{C}$ is a piece of particular ciphertext.
(6) $e \neq d$ and $(e, d) \in \mathcal{K}$ is the key.
(7) E is the encryption function

$$E_{e_k} : M \mapsto C \tag{6.10}$$

where $M \in \mathcal{M}$ maps to $C \in \mathcal{C}$, using the public-key e_k, such that

$$C = E_{e_k}(M) \tag{6.11}$$

(8) D is the decryption function

$$D_{d_k} : C \mapsto M \tag{6.12}$$

where $C \in \mathcal{C}$ maps to $M \in \mathcal{M}$, using the private key d_k such that

$$M = D_{d_k}(C) \tag{6.13}$$

satisfying

$$E_{e_k} D_{d_k} = 1 \text{ and } D_{d_k}(C) = D_{d_k}(E_{e_k}(M)) = M. \tag{6.14}$$

The main task in public-key cryptography is to find a suitable trap-door one-way function, so that both encryption and decryption are easy to perform for authorized users, whereas

decryption, the inverse of the encryption, should be computationally infeasible for an unauthorized user. One of the major advantages of public-key cryptography is that it can be used not only for encryption busting also for digital signatures, a feature which is useful for Internet security and which is not provided by the traditional secret-key cryptography. Recall that in public-key cryptography, we perform

$$C = E_{e_k}(M), \tag{6.15}$$

where M is the message to be encrypted, for message encryption, and

$$M = D_{d_k}(C), \tag{6.16}$$

where C is the encrypted message that needs to be decrypted, for decryption. In digital signatures, we perform the operations in exactly the opposite direction. That is, we perform (see also Figure 6.7)

$$S = D_{d_k}(M), \tag{6.17}$$

where M is the message to be signed, for signature generation,

$$M = E_{e_k}(S), \tag{6.18}$$

where S is the signed message that needs to be verified, for signature verification.

Now we are able to give a formal definition for digital signatures using public-key cryptography.

Definition 6.4 A digital signature system, $\mathcal{DSS}$, may be formally defined as follows:

$$\mathcal{DSS} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, e, d, E, D) \tag{6.19}$$

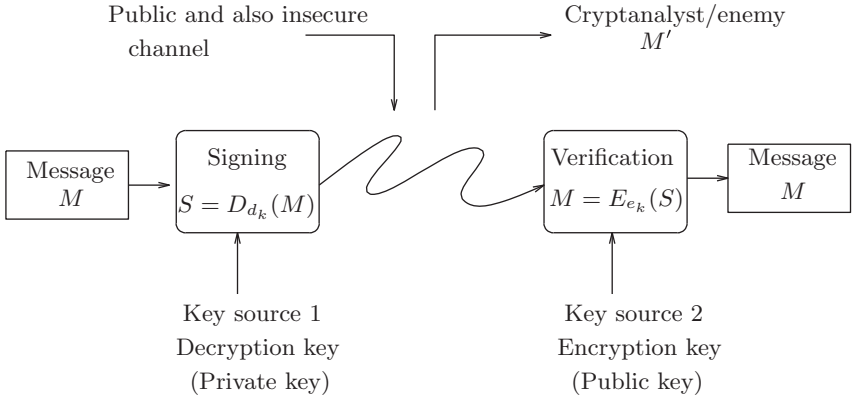


Figure 6.7 Digital signatures

where

(1) $\mathcal{M}$ is the set of plain documents to be signed, called the plain document space.
(2) $\mathcal{S}$ is the set of signed documents, called the signed document space.
(3) $\mathcal{K}$ is the set of keys, called the key space.
(4) $M \in \mathcal{M}$ is a piece of particular plain document.
(5) $S \in \mathcal{S}$ is a piece of particular signed document
(6) $e \neq d$ and $(e, d) \in \mathcal{K}$ is the key.
(7) D is the signature generation function

$$D_{d_k} : M \mapsto S \tag{6.20}$$

where $M \in \mathcal{M}$ maps to $S \in \mathcal{S}$, using the private key d_k, such that

$$S = D_{d_k}(M). \tag{6.21}$$

(8) E is the signature verification function

$$E_{e_k} : S \mapsto M \tag{6.22}$$

where $S \in \mathcal{S}$ maps to $M \in \mathcal{M}$, using the public key e_k such that

$$M = E_{e_k}(M) \tag{6.23}$$

satisfying

$$D_{d_k} E_{e_k} = 1 \text{ and } E_{e_k}(S) = E_{e_k}(D_{d_k}(M)) = M. \tag{6.24}$$

Problems for Section 6.1

1. Explain the following basic concepts in cryptography:
   (1) Cryptography;
   (2) Cryptanalysis;
   (3) Cryptology;
   (4) Public-key cryptography;
   (5) Secret-key cryptography;
   (6) Encryption;
   (7) Decryption;
   (8) Digital Signatures.
2. Explain the following four basic concepts in information security:
   (1) Confidentiality;
   (2) Integrity;
   (3) Authentication;
   (4) Nonrepudiation.

3. Explain the main difference between secret-key cryptography and public-key cryptography.
4. Explain the main difference between public-key cryptography and digital signatures.
5. Explain why the encryption key and the decryption key are the same key in secret-key cryptography, whereas the encryption key and the decryption key are the two different keys in public-key cryptography.
6. Write an essay on the history and the development of public-key cryptography.

6.2 Classic Secret-Key Cryptography

Earlier cryptosystems were based on transforming each letter of the plaintext into a different letter to produce the ciphertext. Such ciphers are called *character*, *substitution* or *monographic ciphers*, since each letter is shifted individually to another letter by a substitution. First of all, let us define the numerical equivalents, as in Table 6.1, of the 26 English capital letters, since our operations will be on the numerical equivalents of letters, rather than the letters themselves. The following are some typical character ciphers.

(1) Caesar cipher: A simple *Caesar cipher* uses the following substitution transformation:

$$f_3 = E_3(m) \equiv m + 3 \pmod{26}, \quad 0 \le m \in \mathcal{M} \le 25, \tag{6.25}$$

and

$$f_3^{-1} = D_3(c) \equiv c - 3 \pmod{26}, \quad 0 \le c \in \mathcal{C} \le 25, \tag{6.26}$$

where 3 is the key for both encryption and decryption. Clearly, the corresponding letters of the Caesar cipher will be obtained from those in Table 6.1 by moving three letters forward, as described in Table 6.2. Mathematically, in encryption we just perform a mapping $m \mapsto m + 3 \mod 26$ on the plaintext, whereas in decryption we perform a mapping $c \mapsto c - 3 \mod 26$ on the ciphertext.

Table 6.1 Numerical equivalents of English capital letters

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

Table 6.2 The corresponding letters of the Caesar cipher

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

(2) Shift transformations: Slightly more general transformations are the following so-called *shift transformations*:

$$f_k = E_k(m) \equiv m + k \pmod{26}, \quad 0 \leq k, m \leq 25, \tag{6.27}$$

and

$$f_k^{-1} = D_k(c) \equiv c - k \pmod{26}, \quad 0 \leq k, c \leq 25, \tag{6.28}$$

(3) Affine transformations: More general transformations are the following so-called *affine transformations*:

$$f_{(a,b)} = E_{(a,b)}(m) \equiv am + b \pmod{26}, \tag{6.29}$$

with $a, b \in \mathbb{Z}$ the key, $0 \leq a, b, m \leq 26$ and $\gcd(a, 26) = 1$, together with

$$f_{(a,b)}^{-1} = D_{(a,b)}(c) \equiv a^{-1}(c - b) \pmod{26}, \tag{6.30}$$

where a^{-1} is the multiplicative inverse of a modulo 26 (even more generally, the modulus 26 could be any number greater than 26, but normally chosen to be a prime number).

Example 6.2 In character ciphers, we have

$$E_3(\text{IBM}) = \text{LEP},$$
$$E_4(\text{NIST}) = \text{RMWX},$$
$$E_7(\text{ENCRYPTION}) = \text{LUJYFWAPVU}.$$

$$D_4(\text{GEPMJSVRME}) = \text{CALIFORNIA},$$
$$D_5(\text{JSLQFSI}) = \text{ENGLAND},$$
$$D_6(\text{JKIXEVZOUT}) = \text{DECRYPTION}.$$

Exercise 6.1 Decrypt the following character ciphertexts:

$$D_7(\text{JVTTBUPJHAPVU}),$$
$$D_9(\text{BNLDARCH}).$$

Example 6.3 Use the following affine transformations

$$f_{(7,21)} \equiv 7m + 21 \pmod{26}$$

and

$$f^{-1}_{(7,21)} \equiv 7^{-1}(c - 21) \pmod{26}$$

to encrypt the message SECURITY and decrypt the message VLXIJH. To encrypt the message, we have

$$\begin{array}{lll}
S = 18, & 7 \cdot 18 + 21 \bmod 26 = 17, & S \Rightarrow R, \\
E = 4, & 7 \cdot 4 + 21 \bmod 26 = 23, & E \Rightarrow X, \\
C = 2, & 7 \cdot 2 + 21 \bmod 26 = 9, & C \Rightarrow J, \\
U = 20, & 7 \cdot 20 + 21 \bmod 26 = 5, & U \Rightarrow F, \\
R = 17, & 7 \cdot 17 + 21 \bmod 26 = 10, & R \Rightarrow K, \\
I = 8, & 7 \cdot 8 + 21 \bmod 26 = 25, & I \Rightarrow Z, \\
T = 19, & 7 \cdot 19 + 21 \bmod 26 = 24, & T \Rightarrow Y, \\
Y = 24, & 7 \cdot 24 + 21 \bmod 26 = 7, & Y \Rightarrow H.
\end{array}$$

Thus, $E_{(7,21)}(\text{SECURITY}) = \text{RXJFKZYH}$. Similarly, to decrypt the message VLXIJH, we have

$$\begin{array}{lll}
V = 21, & 7^{-1} \cdot (21 - 21) \bmod 26 = 0, & V \Rightarrow A, \\
L = 11, & 7^{-1} \cdot (11 - 21) \bmod 26 = 6, & L \Rightarrow G, \\
X = 23, & 7^{-1} \cdot (13 - 21) \bmod 26 = 4, & X \Rightarrow E, \\
I = 8, & 7^{-1} \cdot (8 - 21) \bmod 26 = 13, & I \Rightarrow N, \\
J = 9, & 7^{-1} \cdot (9 - 21) \bmod 26 = 2, & J \Rightarrow C, \\
H = 7, & 7^{-1} \cdot (7 - 21) \bmod 26 = 24, & H \Rightarrow Y.
\end{array}$$

Thus, $D_{(7,21)}(\text{VLXIJH}) = \text{AGENCY}$.

Exercise 6.2 Use the affine transformation

$$f_{(11,23)} = 11m + 23 \pmod{26}$$

to encrypt the message THE NATIONAL SECURITY AGENCY. Use also the inverse transformation

$$f^{-1}_{(11,23)} = 11^{-1}(c - 23) \pmod{26}$$

to verify your result.

Monographic cryptography can be made more secure by splitting the plaintext into groups of letters (rather than a single letter) and then performing the encryption and decryption on these groups of letters. This block technique is called *block ciphering*. Block ciphers are also called a *polygraphic cipher*. Block ciphers may be described as follows:

(1) Split the message M into blocks of n-letters (when $n = 2$ it is called a *digraphic cipher*) $M_1, M_2, \cdots, M_j$; each block M_i for $1 \leq i \leq j$ is a block consisting of n letters.
(2) Translate the letters into their numerical equivalents and form the ciphertext:

$$\mathbf{C}_i \equiv \mathbf{A}\mathbf{M}_i + \mathbf{B} \pmod{N}, \quad i = 1, 2, \cdots, j \tag{6.31}$$

where $(\mathbf{A}, \mathbf{B})$ is the key, $\mathbf{A}$ is an invertible $n \times n$ matrix with $\gcd(\det(\mathbf{A}), N) = 1$, $\mathbf{B} = (B_1, B_2, \cdots, B_n)^T$, $\mathbf{C}_i = (c_1, c_2, \cdots, c_n)^T$ and $\mathbf{M}_i = (m_1, m_2, \cdots, m_n)^T$. For simplicity, we just consider

$$\mathbf{C}_i \equiv \mathbf{A}\mathbf{M}_i \pmod{26}. \tag{6.32}$$

(3) For decryption, we perform

$$\mathbf{M}_i \equiv \mathbf{A}^{-1}(\mathbf{C}_i - \mathbf{B}) \pmod{N}, \tag{6.33}$$

where $\mathbf{A}^{-1}$ is the inverse matrix of $\mathbf{A}$. Again, for simplicity, we just consider

$$\mathbf{M}_i \equiv \mathbf{A}^{-1}\mathbf{C}_i \pmod{26}. \tag{6.34}$$

Example 6.4 Let

$$M = \text{YOUR PIN NO IS FOUR ONE TWO SIX}$$

be the plaintext and $n = 3$. Let also the encryption matrix be

$$\mathbf{A} = \begin{pmatrix} 11 & 2 & 19 \\ 5 & 23 & 25 \\ 20 & 7 & 17 \end{pmatrix}.$$

Then the encryption and decryption of the message can be described as follows:

(1) Split the message M into blocks of 3-letters and translate these letters into their numerical equivalents:

Y	O	U	R	P	I	N	N	O	I	S	F
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
24	14	20	17	15	8	13	13	14	8	18	5
O	U	R	O	N	E	T	W	O	S	I	X
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
14	20	17	14	13	4	19	22	14	18	8	23

(2) Encrypt these nine blocks in the following way:

$$\mathbf{C}_1 = \mathbf{A}\begin{pmatrix}24\\14\\20\end{pmatrix} = \begin{pmatrix}22\\6\\8\end{pmatrix}, \quad \mathbf{C}_2 = \mathbf{A}\begin{pmatrix}17\\15\\8\end{pmatrix} = \begin{pmatrix}5\\6\\9\end{pmatrix},$$

$$\mathbf{C}_3 = \mathbf{A}\begin{pmatrix}13\\13\\14\end{pmatrix} = \begin{pmatrix}19\\12\\17\end{pmatrix}, \quad \mathbf{C}_4 = \mathbf{A}\begin{pmatrix}8\\18\\5\end{pmatrix} = \begin{pmatrix}11\\7\\7\end{pmatrix},$$

$$\mathbf{C}_5 = \mathbf{A}\begin{pmatrix}14\\20\\17\end{pmatrix} = \begin{pmatrix}23\\19\\7\end{pmatrix}, \quad \mathbf{C}_6 = \mathbf{A}\begin{pmatrix}14\\13\\4\end{pmatrix} = \begin{pmatrix}22\\1\\23\end{pmatrix},$$

$$\mathbf{C}_7 = \mathbf{A}\begin{pmatrix}19\\22\\14\end{pmatrix} = \begin{pmatrix}25\\15\\18\end{pmatrix}, \quad \mathbf{C}_8 = \mathbf{A}\begin{pmatrix}18\\8\\23\end{pmatrix} = \begin{pmatrix}1\\17\\1\end{pmatrix}.$$

(3) Translating these into letters, we get the ciphertext C:

22	6	8	5	6	9	19	12	17	11	7	7
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
W	G	I	F	G	J	T	M	R	L	H	H
23	19	7	22	1	23	25	15	18	1	17	1
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
X	T	H	W	B	X	Z	P	S	B	R	B

(4) To recover the message M from C, we first compute A^{-1} modulo 26:

$$\mathbf{A}^{-1} = \begin{pmatrix}11 & 2 & 19\\5 & 23 & 25\\20 & 7 & 17\end{pmatrix}^{-1} = \begin{pmatrix}10 & 23 & 7\\15 & 9 & 22\\5 & 9 & 21\end{pmatrix}.$$

and then perform $\mathbf{M}_i = \mathbf{A}^{-1}\mathbf{C}_i$ as follows:

$$\mathbf{M}_1 = \mathbf{A}^{-1}\begin{pmatrix}22\\6\\8\end{pmatrix} = \begin{pmatrix}24\\14\\20\end{pmatrix}, \quad \mathbf{M}_2 = \mathbf{A}^{-1}\begin{pmatrix}5\\6\\9\end{pmatrix} = \begin{pmatrix}17\\15\\8\end{pmatrix},$$

$$\mathbf{M}_3 = \mathbf{A}^{-1}\begin{pmatrix}19\\12\\17\end{pmatrix} = \begin{pmatrix}13\\13\\14\end{pmatrix}, \quad \mathbf{M}_4 = \mathbf{A}^{-1}\begin{pmatrix}11\\7\\7\end{pmatrix} = \begin{pmatrix}8\\18\\5\end{pmatrix},$$

$$\mathbf{M}_5 = \mathbf{A}^{-1}\begin{pmatrix}23\\19\\7\end{pmatrix} = \begin{pmatrix}14\\20\\17\end{pmatrix}, \quad \mathbf{M}_6 = \mathbf{A}^{-1}\begin{pmatrix}22\\1\\23\end{pmatrix} = \begin{pmatrix}14\\13\\4\end{pmatrix},$$

$$\mathbf{M}_7 = \mathbf{A}^{-1}\begin{pmatrix}25\\15\\18\end{pmatrix} = \begin{pmatrix}19\\22\\14\end{pmatrix}, \quad \mathbf{M}_8 = \mathbf{A}^{-1}\begin{pmatrix}1\\17\\1\end{pmatrix} = \begin{pmatrix}18\\8\\23\end{pmatrix}.$$

So, we have:

24	14	20	17	15	8	13	13	14	8	18	5
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
Y	O	U	R	P	I	N	N	O	I	S	F
14	20	17	14	13	4	19	22	14	18	8	23
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
O	U	R	O	N	E	T	W	O	S	I	X

which is the original message.

Exercise 6.3 Let

$$\mathbf{A} = \begin{pmatrix}3 & 13 & 21 & 9\\15 & 10 & 6 & 25\\10 & 17 & 4 & 8\\1 & 23 & 7 & 2\end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix}1\\21\\8\\17\end{pmatrix}.$$

Use the block transformation

$$C_i \equiv \mathbf{A}M_i + \mathbf{B} \pmod{26}$$

to encrypt the following message

PLEASE SEND ME THE BOOK, MY CREDIT CARD NO IS
SIX ONE TWO ONE THREE EIGHT SIX ZERO
ONE SIX EIGHT FOUR NINE SEVEN ZERO TWO.

Use

$$M_i \equiv \mathbf{A}^{-1}(C_i - \mathbf{B}) \pmod{26}$$

to verify your result, where

$$\mathbf{A}^{-1} = \begin{pmatrix} 23 & 13 & 20 & 5 \\ 0 & 10 & 11 & 0 \\ 9 & 11 & 15 & 22 \\ 9 & 22 & 6 & 25 \end{pmatrix}.$$

Problems for Section 6.2

1. The matrix encryption was studied by Lester S. Hill the in late 1920s [14] and early 1930s [15], based on some ideas from linear algebra. Such a cipher (cryptographic system) is now called Hill cipher. If the plaintext is grouped into sets of n letters and encrypted by an $n\times$ matrix with integer entries, then the Hill cipher is referred to as the Hill n-cipher. Show that
   (1) A square matrix A with entries in $\mathbb{Z}_n$ is inventible modulo n if and only if the residue of $\det(A)$ modulo n has a multiplicative inverse (reciprocal) modulo n.
   (2) A square matrix A with entries in $\mathbb{Z}_n$ is inventible modulo n if and only if n and the residue of $\det(A)$ modulo n has no common prime factors.
   (3) A square matrix A with entries in $\mathbb{Z}_{26}$ is inventible modulo 26 if and only if the residue of $\det(A)$ modulo 26 is not divisible by 2 or 3.
2. Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ be linear independent plaintext vector, and let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n$ be the corresponding ciphertext vectors. If

$$P = \begin{pmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_n^{\mathrm{T}} \end{pmatrix}$$

is the $n \times n$ matrix with row vector $\mathbf{p}_1^{\mathrm{T}}, \mathbf{p}_2^{\mathrm{T}}, \ldots, \mathbf{p}_n^{\mathrm{T}}$ and if

$$C = \begin{pmatrix} \mathbf{c}_1^{\mathrm{T}} \\ \mathbf{c}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{c}_n^{\mathrm{T}} \end{pmatrix}$$

is the $n \times n$ matrix with row vector $\mathbf{c}_1^{\mathrm{T}}, \mathbf{c}_2^{\mathrm{T}}, \ldots, \mathbf{c}_n^{\mathrm{T}}$, then the sequence of elementary row operations that reduces C ti I transforms P to $(A^{-1})^T$.

3. Give a complete complexity analysis of the Hill cipher.
4. Let the Hill encryption matrix be as follows:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 1 \\ 2 & 0 & 1 \end{pmatrix}$$

   (1) Find the inverse $A^{-1} \bmod 26$.
   (2) Find the ciphertext of the plaintext SENDTANKS, using the above encryption matrix.
5. (A challenge problem) As mentioned in Section 6.1, the following cryptogram has never been decrypted, and hence is suitable as a challenge to the interested reader.

XSJOD	PEFOC	XCXFM	RDZME
JZCOA	YUMTZ	LTDNJ	HBUSQ
XTFLK	XCBDY	GYJKK	QBSAH
QHXPE	DBMLI	ZOYVQ	PRETL
TPMUK	XGHIV	ARLAH	SPGGP
VBQYH	TVJYJ	NXFFX	BVLCZ
LEFXF	VDMUB	QBIJV	ZGGAI
TRYQB	AIDEZ	EZEDX	KS

6. Stream cipher is another simple cryptosystem, in which the message units are bits, and the key is usually produced by a random bit generator (see Figure 6.8).

   The plaintext is encrypted on a bit-by-bit basis:

M	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	1	$0\cdots$
K	1	0	0	1	1	0	0	1	0	0	0	1	0	1	1	1	0	$1\cdots$
C	1	1	1	1	1	0	1	0	1	1	1	0	1	1	0	1	1	$1\cdots$

   The key is fed into the random bit generator to create a long sequence of binary signals. This "key stream" K is then mixed with the plaintext stream M, usually by a bit-wise XOR (Exclusive-OR, or modulo-2 addition) to produce the ciphertext stream C. The

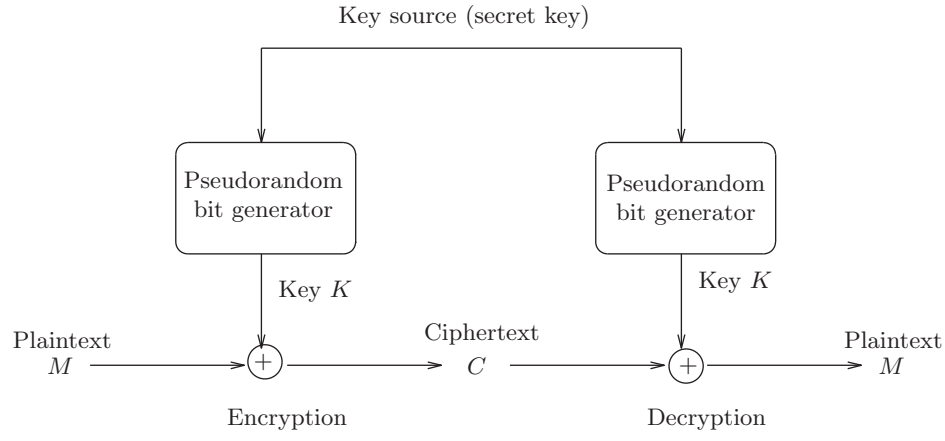


Figure 6.8 Stream cipher

decryption is done by XORing with the same key stream, using the same random bit generator and seed:

C	1	1	1	1	1	0	1	0	1	1	1	0	1	1	0	1	1	$1\cdots$
K	1	0	0	1	1	0	0	1	0	0	0	1	0	1	1	1	0	$1\cdots$
M	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	1	$0\cdots$

Show that

(1) The steam cipher can be easily converted to be a One-Time Pad, satisfying
   (a) The key K is randomly generated.
   (b) The key K can only be used once.
   (c) The key size must be at least as long as the plaintext M.
(2) Show that the One-Time Pad defined above is absolutely and unconditionally unbreakable.

6.3 Modern Secret-Key Cryptography

The exponentiation cipher, invented by Pohlig and Hellman in 1976, may be described as follows. Let p be a prime number, M the numerical equivalent of the plaintext, where each letter of the plaintext is replaced by its two digit equivalent, as defined in Table 6.3. Subdivide M into blocks M_i such that $0 < M_i < p$. Let k be an integer with $0 < k < p$ and $\gcd(k,\ p-1) = 1$. Then the encryption transformation for M_i is defined by

$$C_i = E_k(M_i) \equiv M_i^k \pmod{p}, \tag{6.35}$$

Table 6.3 Two digit equivalents of letters

␣	A	B	C	D	E	F	G	H	I	J	K	L	M
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
00	01	02	03	04	05	06	07	08	09	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	
14	15	16	17	18	19	20	21	22	23	24	25	26	

and the decryption transformation by

$$M_i = D_{k^{-1}}(C_i) \equiv C_i^{k^{-1}} \equiv (M_i^k)^{k^{-1}} \equiv M_i \pmod{p}, \tag{6.36}$$

where $k \cdot k^{-1} \equiv 1 \pmod{p-1}$.

Example 6.5 Let $p = 7951$ and $k = 91$ such that $\gcd(7951 - 1, 91) = 1$. Suppose we wish to encrypt the message

$$M = \text{ENCRYPTION REGULATION MOVES TO A STEP CLOSER}$$

using the exponentiation cipher. First, we convert all the letters in the message to their numerical equivalents via Table 6.3

05 14 03 18 25 16 20 09 15 14 00 18 05 07 21 12 01 20 09 15 14 00
13 15 22 05 19 00 20 15 00 01 00 19 20 05 16 00 03 12 15 19 05 18

and group them into blocks with four digits

0514 0318 2516 2009 1514 0018 0507 2112 0120 0915 1400
1315 2205 1900 2015 0001 0019 2005 1600 0312 1519 0518

Then we perform the following computation

$C_1 = 0514^{91} \bmod 7951 = 2174$ $C_2 = 0318^{91} \bmod 7951 = 4468$
$C_3 = 2516^{91} \bmod 7951 = 7889$ $C_4 = 2009^{91} \bmod 7951 = 6582$
$C_5 = 1514^{91} \bmod 7951 = 924$ $C_6 = 0018^{91} \bmod 7951 = 5460$
$C_7 = 0507^{91} \bmod 7951 = 7868$ $C_8 = 2112^{91} \bmod 7951 = 7319$
$C_9 = 0120^{91} \bmod 7951 = 726$ $C_{10} = 915^{91} \bmod 7951 = 2890$
$C_{11} = 1400^{91} \bmod 7951 = 7114$ $C_{12} = 1315^{91} \bmod 7951 = 5463$
$C_{13} = 2205^{91} \bmod 7951 = 5000$ $C_{14} = 1900^{91} \bmod 7951 = 438$
$C_{15} = 2015^{91} \bmod 7951 = 2300$ $C_{16} = 0001^{91} \bmod 7951 = 1$
$C_{17} = 0019^{91} \bmod 7951 = 1607$ $C_{18} = 2005^{91} \bmod 7951 = 3509$
$C_{19} = 1600^{91} \bmod 7951 = 7143$ $C_{20} = 0312^{91} \bmod 7951 = 5648$
$C_{21} = 1519^{91} \bmod 7951 = 3937$ $C_{22} = 0518^{91} \bmod 7951 = 4736.$

So, the ciphertext of M is

2174 4468 7889 6582 0924 5460 7868 7319 0726 2890 7114
5463 5000 0438 2300 0001 1607 3509 7143 5648 3937 5064.

To decrypt the ciphertext C back to the plaintext M, since the secret key $k = 91$ and the prime modulus $p = 7951$ are known, we compute the multiplicative inverse k^{-1} of k modulo $p - 1$ as follows:

$$k^{-1} \equiv \frac{1}{k} \pmod{p-1} \equiv \frac{1}{91} \pmod{7950} \equiv 961 \pmod{7950}.$$

Thus, we have

$$\begin{array}{ll}
M_1 = 2174^{961} \bmod 7951 = 514 & M_2 = 4468^{961} \bmod 7951 = 318 \\
M_3 = 7889^{961} \bmod 7951 = 2516 & M_4 = 6582^{961} \bmod 7951 = 2009 \\
M_5 = 924^{961} \bmod 7951 = 1514 & M_6 = 5460^{961} \bmod 7951 = 18 \\
M_7 = 7868^{961} \bmod 7951 = 507 & M_8 = 7319^{961} \bmod 7951 = 2112 \\
M_9 = 726^{961} \bmod 7951 = 120 & M_{10} = 2890^{961} \bmod 7951 = 915 \\
M_{11} = 7114^{961} \bmod 7951 = 1400 & M_{12} = 5463^{961} \bmod 7951 = 1315 \\
M_{13} = 5000^{961} \bmod 7951 = 2205 & M_{14} = 438^{961} \bmod 7951 = 1900 \\
M_{15} = 2300^{961} \bmod 7951 = 2015 & M_{16} = 1^{961} \bmod 7951 = 1 \\
M_{17} = 1607^{961} \bmod 7951 = 19 & M_{18} = 3509^{961} \bmod 7951 = 2005 \\
M_{19} = 7143^{961} \bmod 7951 = 1600 & M_{20} = 5648^{961} \bmod 7951 = 312 \\
M_{21} = 3937^{961} \bmod 7951 = 1519 & M_{22} = 4736^{961} \bmod 7951 = 518.
\end{array}$$

Therefore, we have recovered the original message.

Remark 6.2 In the next chapter, we shall see that exponential cryptography is the base of the first practical RSA public-key cryptography, where the modulus is a product of several (the simplest being two) prime numbers rather than just one prime number.

The most popular secret-key cryptographic scheme in use (by both governments and private companies) is the Data Encryption Standard (DES) — DES was designed at IBM and approved in 1977 as a standard by the US National Bureau of Standards (NBS), now called the National Institute of Standards and Technology (NIST). This standard, first issued in 1977 (FIPS 46 – Federal Information Processing Standard 46), is reviewed every five years. It is currently specified in FIPS 46-2. NIST is proposing to replace FIPS 46-2 with FIPS 46-3 to provide for the use of Triple DES (TDES) as specified in the American National Standards Institute (ANSI) X9.52 standard. Comments were sought from industry, government agencies, and the public on the draft of FIPS 46-3 before April 15 1999.

The standard (algorithm) uses a product transformation of transpositions, substitutions, and nonlinear operations. They are applied for 16 iterations to each block of a message; the message is split into 64-bit message blocks. The key used is composed of 56 bits taken from a 64-bit key which includes 8 parity bits. The algorithm is used in reverse to decrypt each ciphertext block and the same key is used for both encryption and decryption. The algorithm itself is shown schematically in Figure 6.9, where the $\oplus$ is the "exclusive or" (XOR) operator. The DES algorithm takes as input a 64-bit message (plaintext) M and a 56-bit key K, and produces a 64-bit ciphertext C. DES first applies an initial fixed bit-permutation (IP) to M to obtain M'. This permutation has no apparent cryptographic significance. Second, DES

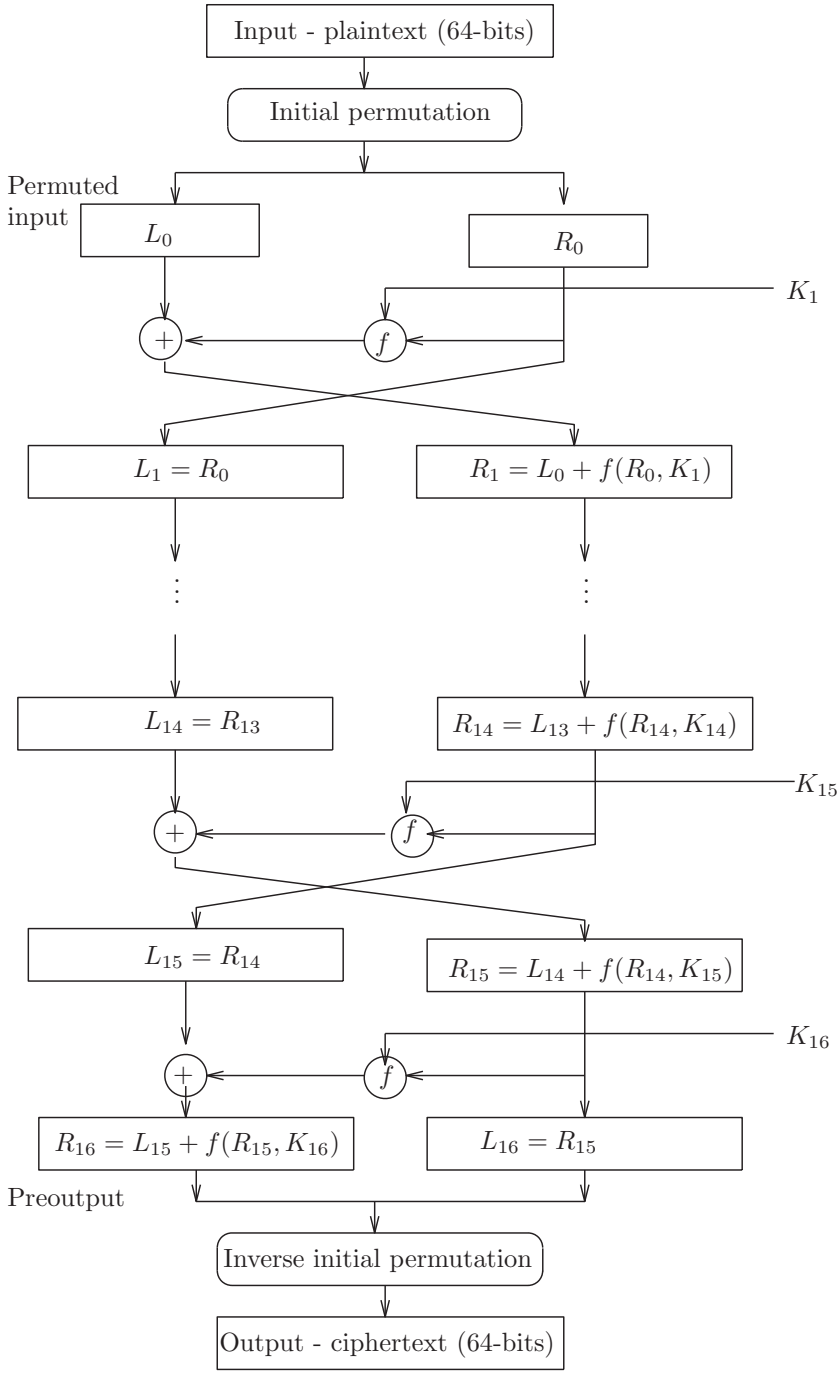


Figure 6.9 The Data Encryption Standard (DES) algorithm

divides M' into a 32-bit left half L_0 and 32-bit right half R_0. Third, DES executes the following operations for $i = 1, 2, \cdots, 16$ (there are 16 "rounds"):

$$\left.\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus f(R_{i-1},\ K_i), \end{aligned}\right\} \tag{6.37}$$

where f is a function that takes a 32-bit right half and a 48-bit "round key" and produces a 32-bit output. Each round key K_i contains a different subset of the 56-bit key bits. Finally, the pre-ciphertext $C' = (R_{16}, L_{16})$ is permuted according to IP^{-1} to obtain the final ciphertext C. To decrypt, the algorithm is run in reverse: a permutation, 16 XOR rounds using the round key in reverse order, and a final permutation that recovers the plaintext. All of this extensive bit manipulation can be incorporated into the logic of a single special-purpose microchip, so DES can be implemented very efficiently. However, the DES cracking project being undertaken by the Electronic Frontier Foundation is able to break the encryption for 56 bit DES in about 22 hours. As a result, NIST has recommended that businesses use Triple DES[1] (TDES), which involves three different DES encryption and decryption operations. Let $E_K(M)$ and $D_K(C)$ represent the DES encryption and decryption of M and C using DES key K, respectively. Each TDES encryption/decryption operation (as specified in ANSI X9.52) is a compound operation of DES encryption and decryption operations. The following operations are used in TDES:

(1) **TDES encryption operation**: the transformation of a 64-bit block M into a 64-bit block C is defined as follows:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(M))). \tag{6.38}$$

(2) **TDES decryption operation**: the transformation of a 64-bit block C into a 64-bit block M is defined as follows:

$$M = D_{K_1}(E_{K_2}(D_{K_3}(C))). \tag{6.39}$$

There are three options for the TDES *key bundle* (K_1, K_2, K_3):

(1) K_1, K_2, and K_3 are independent keys.
(2) K_1, K_2 are independent keys and $K_3 = K_1$.
(3) $K_1 = K_2 = K_3$.

[1]Triple DES is a type of *multiple encryption*. Multiple encryption is a combination technique aimed at improving the security of a block algorithm. It uses an algorithm to encrypt the same plaintext block multiple times with multiple keys. The simplest multiple encryption is the so-called *double encryption* in which an algorithm is used to encrypt a block twice with two different keys – first encrypt a block with the first key, and then encrypt the resulting ciphertext with the second key: $C = E_{k_2}(E_{k_1}(M))$. The decryption is just the reverse process of the encryption: $M = D_{k_1}(D_{k_2}(C))$.

For example, if option 2 is chosen, then the TDES encryption and decryption are as follows:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(M))), \tag{6.40}$$

$$M = D_{K_1}(E_{K_2}(D_{K_1}(C))). \tag{6.41}$$

Interested readers are suggested to consult the current NIST report FIPS 46-3 [32] for the new standard of the TDES.

It is interesting to note that some experts say DES is still secure at some level in e-commerce when used properly. However, Edward Roback at the NIST has said that the DES, which uses 56-bit encryption keys, is no longer sufficiently difficult to decrypt. For example, in February 1998, a team of engineers used a distributed "brute force" decryption program to break a 56-bit DES key in 39 days, about three times faster than it took another team just the year before, and more recently, the team cracked DES in just over 22 hours earlier this year.

The U.S. Department of Commerce's NIST issued a formal call on 12 September 1997 for companies, universities, and other organizations to submit algorithm proposals for a new generation encryption standard for protecting sensitive data well into the 21st century. This new Advanced Encryption Standard (AES) will replace the DES and support encryption key size up to 256 bits and must be available royalty-free throughout the world. On 20 August 1998 at the First AES Candidate Conference (AES1), NIST announced fifteen (15) official AES candidate algorithms submitted by researchers from twelve (12) different countries, including the United States, Australia, France, Germany, Japan, Norway, and the United Kingdom. Since then, cryptographers have tried to find ways to *attack* the different algorithms, looking for weaknesses that would compromise the encrypted information. Shortly after the Second AES Candidate Conference (AES2) on 22–23 March 1999 in Rome, Italy, NIST announced on 9 August 1999 that the following five (5) contenders had been chosen as finalists for the AES; all are block ciphers:

(1) **MARS**: Developed by International Business Machines (IBM) Corporation of Armonk, New York, USA.
(2) **RC6**: Developed by RSA Laboratories of Bedford, Massachusetts, USA.
(3) **Rijndael**: Developed by Joan Daemen and Vincent Rijmen of Belgium.
(4) **Serpent**: Developed by Ross Anderson, Eli Biham and Lars Knudsen of the United Kingdom, Israel, and Norway, respectively.
(5) **Twofish**: Developed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson, of Counterpane Systems, Minneapolis, USA.

These five finalist algorithms received further analysis during a second, more in-depth review period (August 1999–May 2000) in the selection of the final algorithm for the FIPS (Federal Information Processing Standard) AES. On 2 October 2000, the algorithm Rijndael, developed by Joan Daemen (Proton World International, Belgium) and Vincent Rijmen (Katholieke Universiteit Leuven, Belgium) was finally chosen to be the AES [16]. The strong points of Rijndael are a simple and elegant design, it is efficient and fast on modern processors, but also compact in hardware and on smartcards. These features make Rijndael suitable for a wide range of applications. It will be used to protect sensitive but 'unclassified' electronic information of the US government. During the last year, a large number of products

and applications has been AES-enabled. Therefore, it is very likely to become a worldwide de facto standard in numerous other applications such as Internet security, bank cards, and ATMs.

Problems for Section 6.3

1. Let $p = 9137$ and $k = 73$ so that $\gcd(p-1,\ k) = 1$ and $k^{-1} \bmod (p-1) = 750$.
   (1) Use the exponentiation transformation $C \equiv M^k (\bmod\ p)$ to encrypt the following plaintext message:

   THE CESG IS THE UK NATIONAL TECHNICAL AUTHORITY
   ON INFORMATION SECURITY.
   THE NSA IS THE OFFICIAL INTELLIGENCE-GATHERING
   ORGANIZATION OF THE UNITED STATES.

   (2) Use $M = C^{k^{-1}} \bmod\ p$ to verify your result.
2. Show that DES is conditionally unbreakable.
3. Show that AES is conditionally unbreakable.
4. Show that OTP is unconditionally unbreakable.

6.4 Bibliographic Notes and Further Reading

Cryptography is essentially the only automated tool for secure data communication. With the advent of modern Internet, it has become more and more important in network and information security. Today cryptography is used everywhere, from governments to private companies and individuals. It is suggested that everyone using the Internet should have certain knowledge about cryptography and information security. In recent years, there is an increasingly large number of references in cryptography and information security. Readers may consult the following references for more information about the basic concepts and history of cryptography, both secret-key and public-key: [17–31, 33–51].

References

1. H. C. Williams, *Édouard Lucas and Primality Testing*, John Wiley & Sons, 1998.
2. D. Kahn, *The Codebreakers: The Story of Secret Writing*, Macmillan, 1976.
3. C. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, **28**, 1949, pp. 656–715.
4. W. Diffie and E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, **22**, 5, 1976, pp. 644–654.
5. R. C. Merkle, "Secure Communications over Insecure Channels" *Communications of the ACM*, **21**, 1978, pp. 294–299. (Submitted in 1975.)
6. R. L. Rivest, A. Shamir, and L. Adleman, *On Digital Signatures and Public Key Cryptosystems*, Technical Memo 82, Laboratory for Computer Science, Massachusetts Institute of Technology, April 1977.
7. M. Gardner, "Mathematical Games – A New Kind of Cipher that Would Take Millions of Years to Break", *Scientific American*, **237**, 2, 1977, pp. 120–124.
8. R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM*, **21**, 2, 1978, pp. 120–126.
9. J. H. Ellis, *The Possibility of Non-Secret Encryption*, January 1970, 9 Pages.
10. J. H. Ellis, *The Story of Non-Secret Encryption*, 1987, 9 Pages.

11. C. C. Cocks, *A Note on Non-Secret Encryption*, 20 November 1973, 2 pages.
12. M. J. Williamson, *Non-Secret Encryption Using a Finite Field*, 21 January 1974, 2 Pages.
13. M. J. Williamson, *Thoughts on Cheaper Non-Secret Encryption*, 10 August 1976, 3 Pages.
14. L. S. Hill, "Cryptography in an Algebraic Alphabet", *American Mathematical Monthly*, **36**, 1929, pp. 306–312.
15. L. S. Hill, "Concerning Certain Linear Transforming Apparatus of Cryptography", *American Mathematical Monthly*, **38**, 1931, pp. 135–154.
16. J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer, 2002.
17. F. L. Bauer, *Decrypted Secrets – Methods and Maxims of Cryptology*, 3rd Edition, Springer-Verlag, 2002.
18. J. A. Buchmann, *Introduction to Cryptography*, 2nd Edition, Springer, 2004.
19. T. W. Cusick, D. Ding and A. Renvall, *Stream Cipher and Number Theory*, North-Holland, 1998.
20. H. Delfs and H. Knebl, *Introduction to Cryptography*, Springer, 2002.
21. N. Ferguson, B. Schneier and T. Kohno, *Cryptography Engineering*, Wiley, 2005.
22. P. Garrett, *Making, Breaking Codes: An Introduction to Cryptology*, Prentice-Hall, 2001.
23. O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001.
24. O. Goldreich, *Foundations of Cryptography: Basic Applications*, Cambridge University Press, 2004.
25. F. Guterl, "Suddenly, Number Theory Makes Sense to Industry", *International Business Week*, 20 June 1994, pp. 62–64.
26. J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer-Verlag, 2008.
27. N. Koblitz, "A Survey of Number Theory and Cryptography", *Number Theory*. Edited by P. Bambah, V. C. Dumir, and R. J. Hans-Gill, Birkhäser, 2000, pp. 217–239.
28. N. Koblitz, "Cryptography", in: *Mathematics Unlimited – 2001 and Beyond*, Edited by B. Enguist and W. Schmid, Springer, 2001, pp. 749–769.
29. Vaudenay:2010crypt, W. Mao, *Modern Cryptography*, Prentice-Hall, 2004.
30. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptosystems*, CRC Press, 1996.
31. R. A. Mollin, *Codes: The Guide to Secrecy from ancient to Modern Times*, Chapman & Hall/CRC Press, 2005.
32. NIST, "Data Encryption Standard", Federal Information Processing Standards Publication 46-3, National Institute of Standards and Technology, U.S. Department of Commerce, 1999.
33. R. A. Mollin, *An Introduction to Cryptography*, 2nd Edition, Chapman & Hall/CRC Press, 2006.
34. J. Pieprzyk, T. Hardjono and J. Seberry, *Fundamentals of Computer Security*, Springer, 2003.
35. M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", *Technical Report MIT/LCS/TR-212*, MIT Laboratory for Computer Science, 1979.
36. J. Rothe, *Complexity Theory and Cryptography*, Springer, 2005.
37. B. Schneier, *Applied Cryptography – Protocols, Algorithms, and Source Code in C*, 2nd Edition, Wiley, 1996.
38. B. Schneier, "The Secret Story of Non-Secret Encryption", *Crypto-Gram Newsletter*, Counterpane Systems, May 15, 1998.
39. G. J. Simmons (Editor), *Contemporary Cryptology – The Science of Information Integrity*, IEEE Press, 1992.
40. S. Singh, *The Code Book – The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, Fourth Estate, London, 1999.
41. S. Singh, *The Science of Secrecy – The History of Codes and Codebreaking*, Fourth Estate, London, 2000. Garrett:2001crypt
42. N. Smart, *Cryptography: An Introduction*, McGraw-Hill, 2003.
43. R. J. Spillman, *Classical and Contemporary Cryptology*, Prentice-Hall, 2005.
44. D. R. Stinson, *Cryptography: Theory and Practice*, 2nd Edition, Chapman & Hall/CRC Press, 2002.
45. J. C. A. van der Lubbe, *Basic Methods of Cryptography*, Cambridge University Press, 1998.
46. S. Vaudenay, *A Classical Introduction to Cryptography*, Springer, 2010.
47. H. C. A. van Tilborg, *Fundamentals of Cryptography*, Kluwer Academic Publishers, 1999.
48. W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, 2nd Edition, Prentice-Hall, 2006.
49. S. S. Wagstaff, Jr., *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC Press, 2002.
50. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer-Verlag, 2002.
51. S. Y. Yan, *Cryptanalyic Attacks on RSA*, Springer, 2009.

7

Integer Factorization Based Cryptography

This chapter studies the IFP-based cryptographic systems and protocols, including:

- The famous RSA cryptographic system
- The factoring-equivalent Rabin cryptographic system
- The quadratic residuosity based probabilistic encryption
- The zero-knowledge proof protocol.

7.1 RSA Cryptography

In 1977, Rivest, Shamir, and Adleman (see Figure 7.1), then all at MIT, proposed the first practical public-key cryptosystem, now widely known as the RSA public-key cryptosystem (see [1, 2]). The Association for Computing Machinery, ACM, offered the Year 2002 A. M. Turing Award, regarded as a Nobel Prize in computer science, to Adleman, Rivest, and Shamir for their contribution to the theory and practical application of public-key cryptography, particularly the invention of the RSA cryptosystem, as the RSA cryptosystem now

> "has become the foundation for an entire generation of technology security products and has also inspired important work in both theoretical computer science and mathematics."

The RSA cryptosystem is based on the following assumption:

> **RSA assumption**: It is easy to find two large prime numbers, but it is hard to factor a large composite number into its prime factorization form.

The RSA cryptosystem (see Figure 7.2) works as follows:

$$\left.\begin{aligned} C &\equiv M^e \pmod{n} \\ M &\equiv C^d \pmod{n} \end{aligned}\right\} \tag{7.1}$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

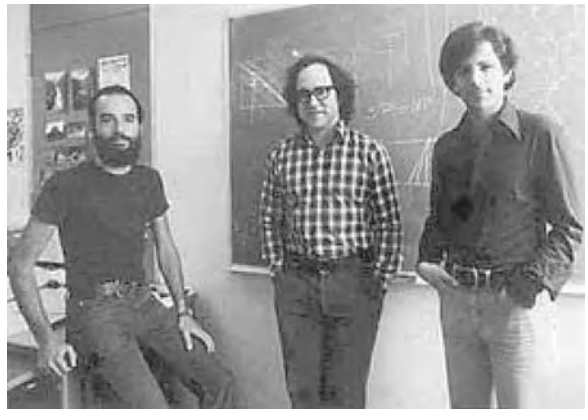


Figure 7.1 Shamir, Rivest, and Adleman in the 1970s (Courtesy of Prof Adleman)

where

(1) M is the plaintext;
(2) C is the ciphertext;
(3) $n = pq$ is the modulus, with p and q large and distinct primes;
(4) e is the *public* encryption exponent (key) and d the *private* decryption exponent (key), with $ed \equiv 1 \pmod{\phi(n)}$. $\langle n, e\rangle$ should be made public, but d (as well as $\phi(n)$) should be kept secret.

Clearly, the function $f : M \to C$ is a one-way trapdoor function, since it is easy to compute by the fast exponentiation method, but its inverse $f^{-1} : C \to M$ is difficult to

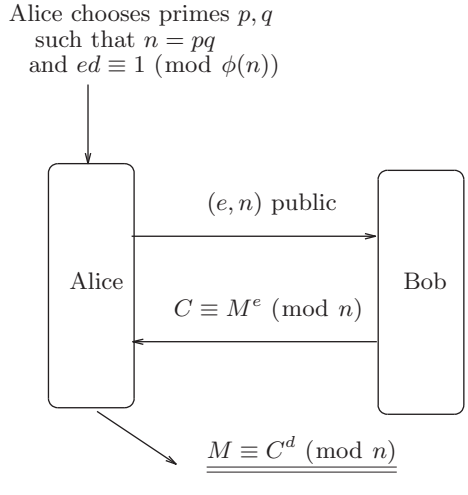


Figure 7.2 RSA cryptosystem

compute, because for those who do not know the private decryption key (the trapdoor information) d, they will have to factor n and compute $\phi(n)$ in order to find d. However, for those who know d, then the computation of f^{-1} is as easy as of f. This is the whole idea of the RSA cryptography.

Suppose now the sender, for example, Alice wants to send a message M to the receiver, for example Bob. Bob will have already chosen a one-way trapdoor function f described above, and published his *public-key* (e, n), so we can assume that both Alice and any potential adversary know (e, n). Alice splits the message M into blocks of $\lfloor \log n \rfloor$ bits or less (padded on the right with zeros for the last block), and treats each block as an integer $x \in \{0, 1, 2, \ldots, n-1\}$. Alice computes

$$y \equiv x^e \pmod{n} \tag{7.2}$$

and transmits y to Bob. Bob, who knows the private key d, computes

$$x \equiv y^d \pmod{n} \tag{7.3}$$

where $ed \equiv 1 \pmod{\phi(n)}$. An adversary who intercepts the encrypted message should not be able to decrypt it without knowledge of d. There is no known way of cracking the RSA system without essentially factoring N, so it is clear that the security of the RSA system depends on the difficulty of factoring N. Some authors, for example Woll in 1987 [3], observed that finding the RSA decryption key d is the random polynomial-time equivalent to factorization, and Pinch in 1997 [4] showed that an algorithm $A(n, e)$ for obtaining d given n and e can be turned into an algorithm which obtains p and q with positive probability.

Example 7.1 Suppose the message (plaintext) to be encrypted is "PLEASE WAIT FOR ME." Let $n = 5515596313 = 71593 \cdot 77041$. Let also $e = 1757316971$ with $\gcd(e, n) = 1$. Then $d \equiv 1/1757316971 \equiv 2674607171 \pmod{(71593-1)(77041-1)}$. To encrypt the message, we first translate the message into its numerical equivalent by the letter-digit encoding scheme defined by $A \to 01$, $B \to 02, \ldots, Z \to 26$ and space $\to 00$:

$$M = 16120501190500230109200615180013 05.$$

Then we split it into 4 blocks, each with 10 digits, padded on the right with zeros for the last block:

$$M = (M_1, M_2, M_3, M_4) = (1612050119\,0500230109\,2000061518\,0013050000).$$

Now, we have

$$\begin{aligned}
C_1 &\equiv 1612050119^{1757316971} \equiv 763222127 \pmod{5515596313},\\
C_2 &\equiv 0500230109^{1757316971} \equiv 1991534528 \pmod{5515596313},\\
C_3 &\equiv 2000061518^{1757316971} \equiv 74882553 \pmod{5515596313},\\
C_4 &\equiv 0013050000^{1757316971} \equiv 3895624854 \pmod{5515596313}.
\end{aligned}$$

That is,

$$C = (C_1, C_2, C_3, C_4) = (763222127, 1991534528, 74882553, 3895624854).$$

To decrypt the ciphertext, we perform:

$$\begin{aligned}
M_1 &\equiv 763222127^{2674607171} \equiv 1612050119 \pmod{5515596313},\\
M_2 &\equiv 1991534528^{2674607171} \equiv 500230109 \pmod{5515596313},\\
M_3 &\equiv 74882553^{2674607171} \equiv 2000061518 \pmod{5515596313},\\
M_4 &\equiv 3895624854^{2674607171} \equiv 13050000 \pmod{5515596313}.
\end{aligned}$$

By padding the necessary zeros on the left of some blocks, we get

$$M = (M_1, M_2, M_3, M_4) = (1612050119\,0500230109\,2000061518\,0013050000)$$

which is "Please wait for me," the original plain-text message.

Example 7.2 We now give a reasonably large RSA example. In one of his series of Mathematical Games, Martin Gardner [5] reported an RSA challenge with US$100 to decrypt the following message C:

$$\begin{aligned}
&96869613754622061477140922254355882905759991124574319874695120 93_\\
&08162982251457083569314766228839896280133919905518299451578151 54.
\end{aligned}$$

The public-key consists of a pair of integers (e, n), where $e = 9007$ and N is a "random" 129-digit number (called RSA-129):

$$\begin{aligned}
&11438162575788886766923577997614661201021829672124236256256184 29_\\
&35706935245733897830597123563958705058989075147599290026879543 541.
\end{aligned}$$

The RSA-129 was factored by Derek Atkins, Michael Graff, Arjen K. Lenstra, Paul Leyland et al. on April 2 1994 [6] to win the $100 prize offered by RSA in 1977. Its two prime factors are as follows:

$$\begin{aligned}
&34905295108476509491478496199038981334177646384933878439908205 77,\\
&32769132993266709549961988190834461413177642967992942539798288533.
\end{aligned}$$

They used the double large prime variation of the Multiple Polynomial Quadratic Sieve (MPQS) factoring method. The sieving step took approximately 5000 mips years, and was carried out in 8 months by about 600 volunteers from more than 20 countries, on all continents except Antarctica. As we explained in the previous example, to encrypt an RSA-encrypted message, we only need to use the public-key (n, e) to compute

$$x^e \equiv y \pmod{n}.$$

But decrypting an RSA-message requires factorization of N if one does not know the secret decryption key. This means that if we can factor n, then we can compute the secret key d, and get back the original message by calculating

$$y^d \equiv x \pmod{n}.$$

Since we now know the prime factorization of n, it is trivial to compute the secret key $d = 1/e \bmod \phi(n)$, which in fact is

$$106698614368578024442868771328920154780709906633937862801226224 4\_$$
$$966310631259117744708733401685974623065539685445132771090536060 95.$$

So we shall be able to compute

$$C^d \equiv M \pmod{n}$$

without any problem. To use the fast exponential method to compute $C^d \bmod N$, we first write d in its binary form $d_1 d_2 \cdots d_{\text{size}}$ (where size is the number of the bits of d) as follows:

$$d = d_1 d_2 \cdots d_{426} =$$
$$100111011001111110010100110010001000010000011101001111001001 10\_$$
$$010011110100111000000000000011111110100001101010110001011101111\_$$
$$010100001111101100000010000011101101010101111010101001111110 10\_$$
$$110100001111110100000011110100110001011001011001101001010001 100\_$$
$$100111010110000101110100101011010000011100000000111000111010101 0\_$$
$$011011101000111101001110001101011010101010010011101010001001111\_$$
$$000000100111010011000110111110101100100011001111$$

and perform the following computation:

$$M \leftarrow 1$$
$$\text{for } i \text{ from } 1 \text{ to } 426 \text{ do}$$
$$\quad M \leftarrow M^2 \bmod n$$
$$\quad \text{if } d_i = 1 \text{ then } M \leftarrow M \cdot C \bmod n$$
$$\text{print } M$$

which gives the plain-text M:

$$200805001301070903002315180419000118050019172105011309190800151 9\_$$
$$19090618010705$$

and hence the original message:

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

via the encoding alphabet $\sqcup = 00$, $A = 01$, $B = 02, \ldots, Z = 26$. Of course, by the public encryption key $e = 9007$, we can compute $M^e \equiv C \pmod{n}$; first write e in the binary form $e = e_1 e_2 \cdots e_{14} = 10001100101111$, then perform the following procedure:

$C \leftarrow 1$
for i from 1 to 14 do
  $C \leftarrow C^2 \bmod n$
  if $e_i = 1$ then $C \leftarrow C \cdot M \bmod n$
print C

which gives the encrypted text C at the beginning of this example:

$$96869613754622061477140922254355882905759991124574319874695120 93\_$$
$$08162982251457083569314766228839896280133919905518299451578151 54.$$

Remark 7.1 In fact, anyone who can factor the integer RSA-129 can decrypt the message. Thus, decrypting the message is essentially factoring the 129-digit integer. The factorization of RSA-129 implies that it is possible to factor any random 129-digit integer. It should be also noted that, as we mentioned earlier, the current best known general factoring record is the RSA-768 (a 768-bit and 232-digit number). It follows that the composite number (i.e., the modulus) n used in the RSA cryptosystem should have more than 232 digits.

A better example [7] of a trapdoor one-way function of the form used in the RSA cryptosystem would use Carmichael's λ-function rather than Euler's ϕ-function, and is as follows:

$$y = f(x) \equiv x^e \pmod{n} \tag{7.4}$$

where

$$\left.\begin{aligned} & n = pq (p \text{ and } q \text{ are two large primes}), \\ & e > 1, \quad \gcd(e, \lambda) = 1, \\ & \lambda(n) = \operatorname{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1,\ q-1)}. \end{aligned}\right\} \tag{7.5}$$

We assume that e and n are publicly known but p, q and $\lambda(n)$ are not. The inverse function of $f(x)$ is defined by

$$x = f^{-1}(y) \equiv y^d \pmod{n} \ \text{ with } \ ed \equiv 1 \pmod{\lambda}. \tag{7.6}$$

To show it works, we see

$$\begin{aligned} x &\equiv y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{k\lambda(n)+1} \\ &\equiv (x^{\lambda(n)})^k \cdot x \equiv 1^k \cdot x \quad \text{(by Carmichael's theorem)} \\ &\equiv x \pmod{n}. \end{aligned}$$

It should be easy to compute $f^{-1}(y) \equiv y^d \pmod{n}$ if d is known, provided that $f^{-1}(y)$ exists (note that $f^{-1}(y)$ may not exist). The assumption underlying the RSA cryptosystem is that it is hard to compute $f^{-1}(y)$ without knowing d. However, knowledge of p, q, or $\lambda(n)$ makes it easy to compute d.

Algorithm 7.1 (Construction of the above trapdoor function) This algorithm constructs the trapdoor function and generates both the public and the secret keys suitable for RSA cryptography:

[1] Find two large primes p and q, each with at least 100 digits such that:
  [a] $|p - q|$ is large;
  [b] $p \equiv -1 \pmod{12}$, $q \equiv -1 \pmod{12}$;
  [c] The following values of p', p'', q', and q'' are all primes:

$$\left.\begin{aligned} p' &= (p-1)/2, \\ p'' &= (p+1)/12, \\ q' &= (q-1)/2, \\ q'' &= (q+1)/12. \end{aligned}\right\} \tag{7.7}$$

[2] Compute $n = pq$ and $\lambda = 2p'q'$.
[3] Choose a random integer e relatively prime to λ such that $e - 1$ is not a multiple of p' or q'.
[4] Apply the extended Euclidean algorithm to e and λ to find d and λ' such that $0 < d < \lambda$ and

$$ed + \lambda\lambda' = 1. \tag{7.8}$$

[5] Destroy all evidence of p, q, λ, and λ'.
[6] Make (e, n) public but keep d secret.

According to the Prime Number theorem, the probability that a randomly chosen integer in $[1, n]$ is prime is $\sim 1/\ln n$. Thus, the expected number of random trials required to find p (or p', or p''; assume that p, p', and p'' are independent) is conjectured to be $\mathcal{O}\left((\log n)^3\right)$. Based on this assumption, the expected time required to construct the above one-way trapdoor function is $\mathcal{O}\left((\log n)^6\right)$.

Problems for Section 7.1

1. The RSA function $M \mapsto C \bmod n$ is a trapdoor one-way, as it is computationally intractable to invert the function if the prime factorization $n = pq$ is unknown. Give your own trapdoor one-way functions that can be used to construct public-key cryptosystems. Justify your answer.
2. Show that

$$M \equiv M^{ed} \pmod{n},$$

where $ed \equiv 1 \pmod{\phi(n)}$.

3. Let the ciphertexts $C_1 \equiv M_1^e \pmod{n}$ and $C_2 \equiv M_2^e \pmod{n}$ be as follows, where $e = 9137$ and n is the following RSA-129 number:

$$\begin{aligned}
&466049064350600963923911223871120237360391634700827 68_\\
&243410383296685073462027217982000297925067088337283 56_\\
&780453238389114071957 9,
\end{aligned}$$

$$\begin{aligned}
&650640969385110697415283133424753966489785517358138 36_\\
&777963503738147209287793861787878189741574391857183 60_\\
&819612416009343883015 8.
\end{aligned}$$

Find M_1 and M_2.

4. Let

$$\begin{aligned}
e_1 &= 9007,\\
e_2 &= 65537,\\
n &= 114381625757888867669235779976146612010218296721242362_\\
&\quad 562561842935706935245733897830597123563958705058989075_\\
&\quad 1475992900268795435 41,\\
C_1 &\equiv M^{e_1} \pmod{n},\\
&\equiv 104202250941196238413638382607974125774449084724929 59_\\
&\quad 125743374588926529777171718241302464293807835197908 99_\\
&\quad 453434074641613779772 12,\\
C_2 &\equiv M^{e_2} \bmod n\\
&\equiv 764527507291887001807199705175445747109447573179098 96_\\
&\quad 041340987488285573190280783480309084978021563396490 75_\\
&\quad 975060051949607130434 8.
\end{aligned}$$

Find the plain-text M.

5. (Rivest) Let

$$k = 2^{2^t} \pmod{n}$$

where

$$\begin{aligned} n = {} & 631446608307288889379935712613129233236329 88\_ \\ & 183308413755889907727019571289248855473084 46\_ \\ & 055753206513618346628848948088663500368480 39\_ \\ & 658817136198766052189726781016228055747539 38\_ \\ & 383082617597132189266686117769545263915701 20\_ \\ & 690939973680089721274464666423319187806830 55\_ \\ & 206795125307008202024124623398241073775370 51\_ \\ & 273444941695011809752418906679638587548563 19\_ \\ & 805507273709904397119733614666701543905360 15\_ \\ & 254337398252457931357531765364633198906465 14\_ \\ & 021339852658003419919039821928447102124648 87\_ \\ & 459388853582070318084289023209710907032396 93\_ \\ & 491996277899532332018406452247646396635593 73\_ \\ & 670093692127580920862931987270082924312436 81, \\ t = {} & 79685186856218. \end{aligned}$$

Find k.

6. The original version of the RSA cryptosystem:

$$C \equiv M^e \pmod{n}, \quad M \equiv C^d \pmod{n},$$

with

$$ed \equiv 1 \pmod{\phi(n)}$$

is a type of deterministic cryptosystem, in which the same ciphertext is obtained for the same plaintext even at a different time. That is,

$$\begin{aligned} M_1 &\xrightarrow{\text{Encryption at Time 1}} C_1, \\ M_1 &\xrightarrow{\text{Encryption at Time 2}} C_1, \\ &\vdots \\ M_1 &\xrightarrow{\text{Encryption at Time } t} C_1. \end{aligned}$$

A randomized cryptosystem is one in which different ciphertext is obtained at a different time even for the same plaintext

$$M_1 \xrightarrow{\text{Encryption at Time 1}} C_1,$$

$$M_1 \xrightarrow{\text{Encryption at Time 2}} C_2,$$

$$\vdots$$

$$M_1 \xrightarrow{\text{Encryption at Time } t} C_t,$$

with $C_1 \neq C_2 \neq \cdots \neq C_t$. Describe a method to make RSA a randomized cryptosystem.

7. Describe a man-in-the-middle attack on the original version of the RSA cryptosystem.
8. Show that cracking RSA is generally equivalent to solving the IFP problem.

7.2 Cryptanalysis of RSA

The most straightforward attacks on RSA are the integer factorization attack and discrete logarithm attack. If there are efficient algorithms for the integer factorization problem and the discrete logarithm problem, then RSA can be completely broken in polynomial-time. Unfortunately, there are no such efficient algorithm exists yet. The search for such an efficient algorithm is the most important unsolved problem in computational number theory. In this section, we introduce some elementary attacks on RSA, based on some elementary number-theoretic properties and the weakness of RSA. Of course, such weaknesses can be avoided if RSA is used properly.

Our first type of elementary algorithmic attack is concerned with attacks on guessing the values of the plaintext M. Suppose (e, n, C) is given, and the cryptanalyst, Eve, wishes to find M. That is,

$$\{e, n, C \equiv M^e \pmod{n}\} \xrightarrow[\text{guessing } M]{\text{find}} \{M\}.$$

If the plaintext space $\mathcal{M} = \{M_1, M_2, \cdots, M_k\}$ is small or predicable, Eve can decrypt C by simply encrypting all possible plaintext messages $M_1, M_2, \cdots, M_k$ to get $C_1', C_2', \cdots, C_k'$, and check, at each step, if $C_i' = C$. If yes, then $M = M_i$, the plaintext M is found. This simple process can be described as follows:

$$\begin{aligned}
C_1' &\equiv (M_1)^e \pmod{n}, && \text{if } C_1' = C, \text{ then } M = M_1,\\
C_2' &\equiv (M_2)^e \pmod{n}, && \text{if } C_2' = C, \text{ then } M = M_2,\\
&\vdots && \vdots\\
C_k' &\equiv (M_k)^e \pmod{n}, && \text{if } C_1' = C, \text{ then } M = M_k.
\end{aligned}$$

This attack is known as a *forward search attack*, or a *guessing plaintext attack*. The attack will be impractical if the message space $\mathcal{M}$ is large. So to prevent such an attack, the message space $\mathcal{M}$ is necessarily very large.

A closely related attack to the forward search attack is the *short plaintext attack*. If the plaintext message M is small although the corresponding C can be as big as N (this is the general case for public-key cryptography as it is usually only used to encrypt short massages particularly the encryption keys used for secret-key cryptographic systems such as DES and AES [8]), then the cryptanalyst can perform two sequences of the operations as follows:

$$\begin{aligned} U &\equiv Cx^{-e} \pmod{n}, \quad \text{for all } 1 \le x \le 19^9 \\ V &\equiv y^e \pmod{n}, \quad \text{for all } 1 \le y \le 19^9 \end{aligned}$$

If for some of the pair (x, y), we have $U = V$, then $C \equiv (xy)^e \pmod{n}$. Thus $M = xy$. This attack is much more efficient than the forward attack that would try all 10^{17} possible values of M, because it only needs to perform 2×10^9 computations and to compare the elements in the two sequences up to 10^9 times.

The above two attacks can be easily prevented by a *salting process* (i.e., appending some random digits to the plaintext message M prior to encryption), or by a *padding process* (i.e., adjoin some random digits to the beginning and the end of the plaintext message M prior to encryption) such as the one discussed in [9], so that a large random plaintext M can be formed; these randomly added digits can be simply removed after decryption.

Our second type of elementary algorithmic attacks is on RSA signatures. Suppose (e, n, M) is given, and the cryptanalyst, Eve, wishes to find the digital signature S. That is,

$$\{e, n, M \equiv S^e \pmod{n}\} \xrightarrow[\text{forging } S]{\text{find}} \{S\}. \tag{7.9}$$

The RSA function enjoys a certain kind of *self-reducibility* [10], which, on the one hand is good (as it provides assurance that all random ciphertexts are equally hard to decrypt) but on the other hand is bad (as it provides an avenue for an attacker, Eve, to gain information about the decryption of one ciphertext from the decryption of other ciphertexts). The following attack, called the *blinding attack* [11] is based, unfortunately, on this self-reducibility and can be used to obtain someone's valid digital signature.

Let (e, N) and (d, n) be Bob's public and secret keys, respectively. Suppose the cryptanalyst, Eve, wants to know Bob's signature S on a message $M \in \mathbb{Z}_n^*$, which is computed by:

$$S \equiv M^d \pmod{n}.$$

Then Eve can try the following:

[1] Eve picks up a random number $r \in \mathbb{Z}_n^*$, and computes $M' \equiv r^e M \pmod{n}$.
[2] Even asks Bob to sign the random message (looks like a hashed value, as usual) M'.
[3] Suppose Bob is willing to sign the message M', which means that Eve can get

$$S' \equiv (M')^d \pmod{n}.$$

[4] Now it is simple for Eve to get Bob's valid signature S as follows:

$$S \equiv S'/r \pmod{n},$$

which is so because

$$\begin{aligned} S^e &\equiv (S'/r)^e \\ &\equiv (S')^e/r^e \\ &\equiv ((M')^d)^e/r^e \\ &\equiv M'/r^e \\ &\equiv (r^e M)/r^e \\ &\equiv M \pmod{n}. \end{aligned}$$

Thus, Eve can forge Bob's valid signature without knowing his private exponent d, and Bob will not detect the forgery since $M \equiv S^e \pmod{n}$, as we have just shown.

Again, this *chosen plaintext attack* can be avoided by using random padding techniques. Note that the random padding techniques are also countermeasures against the following *chosen-ciphertext attack* [12]. Suppose the cryptanalyst, Eve, intercepts a ciphertext C from Bob to Alice. Then Eve chooses at random a positive integer r, computes $\tilde{M} \equiv C \cdot r^e \pmod{n}$ and sends it to Alice. Alice then decrypts the ciphertext $\tilde{C} \equiv \tilde{M}^e \equiv C^d \cdot r \pmod{n}$. Suppose now Eve can get this $\tilde{C}$, then she can get the original plaintext M by computing

$$M \equiv r^{-1} C^d \cdot r \pmod{n}.$$

Our third type of attack is based on the fact that if one can guess the value of $\phi(N)$, one can recover the RSA plaintext M from its corresponding ciphertext C in polynomial-time. That is,

$$\phi(N) \stackrel{\mathcal{P}}{\Longrightarrow} \{M\}. \tag{7.10}$$

First of all, we show that the computation of $\phi(n)$ and the factorization of N, IFP(n), are deterministic polynomial-time equivalent.

Theorem 7.1 (The equivalence of $\phi(n)$ and IFP(n))

$$\phi(n) \stackrel{\mathcal{P}}{\Longleftrightarrow} \text{IFP}(n). \tag{7.11}$$

Proof: Note first that if $(N, \phi(n))$ is known and n is assumed to be the product of two primes p and q, then N can be easily factored. Assume

$$n = pq,$$

then

$$\phi(n) = (p-1)(q-1),$$

thus

$$pq - p - q + 1 - \phi(n) = 0 \tag{7.12}$$

substituting $q = n/p$ into (7.12) gives

$$p^2 - (n - \phi(n) + 1)p + n = 0. \tag{7.13}$$

Let $A = n - \phi(n) + 1$, then

$$(p, q) = \frac{A \pm \sqrt{A^2 - 4n}}{2}$$

will be the two roots of (7.13), and hence, the two prime factors of n.

On the other hand, if the two prime factors p and q of n are known, then $\phi(n) = (p-1)(q-1)$ immediately from

$$\phi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

if

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}.$$

■

What this theorem says is that if an enemy cryptanalyst could compute $\phi(n)$ then he could break RSA by computing d as the multiplicative inverse of e modulo $\phi(n)$. That is, $d \equiv 1/e \pmod{\phi(n)}$. On the other hand, the knowledge of $\phi(n)$ can lead to an easy way of factoring n, since

$$\begin{aligned} p + q &= n - \phi(N) + 1, \\ (p-q)^2 &= (p+q)^2 - 4n, \\ p &= \frac{(p+q) + (p-q)}{2}, \\ q &= \frac{(p+q) - (p-q)}{2}. \end{aligned}$$

In other words, computing $\phi(N)$ is no easier than factoring n.

Example 7.3 Let

$$n = 74153950911911911.$$

Suppose the cryptanalyst knows by guessing, interception or whatever that

$$\phi(n) = 74153950339832712.$$

Then

$$\begin{aligned} A &= n - \phi(n) + 1 \\ &= 74153950911911911 - 74153950339832712 + 1 \\ &= 572079200 \end{aligned}$$

Thus

$$p^2 - 572079200p + 74153950911911911 = 0.$$

Solving this equation gives the two roots

$$\{p, q\} = \{198491317,\ 373587883\},$$

and hence the complete prime factorization of n

$$\begin{aligned} n &= 74153950911911911 \\ &= 198491317 \cdot 373587883. \end{aligned}$$

Theorem 7.2 *The RSA encryption is breakable in polynomial-time if the cryptanalyst knows $\phi(n)$. That is,*

$$\phi(n) \overset{\mathcal{P}}{\Longrightarrow} \text{RSA}(M). \tag{7.14}$$

Proof: If $\phi(n)$ is known, then $d \equiv 1/e \pmod{\phi(n)}$, hence recovers M from C: $M \equiv C^d \pmod{n}$. ∎

Thus, we have:

$$\text{IFP}(n) \overset{\mathcal{P}}{\Longleftrightarrow} \phi(n) \overset{\mathcal{P}}{\Longrightarrow} \text{RSA}(M). \tag{7.15}$$

Thus, breaking the RSA encryption by computing $\phi(n)$ is no easier than breaking the RSA encryption by factoring n. However, if someone can intelligently and efficiently guess/find the value of $\phi(n)$, or someone already knows $\phi(n)$ by some means, then he can break RSA completely without factoring.

Our fourth attack is the guessing d attack. We show that given the RSA private exponent d, the prime factorization of n can be made in polynomial-time.

Theorem 7.3 (Coron and May [13]) *Let $n = pq$ with p and q prime numbers. Let also e and d be the public and private exponent, respectively, satisfying $ed \equiv 1 \pmod{\phi(n)}$.*

(1) *If p and q have the same bit size and $1 < ed \leq n^{3/2}$, then given (n, e, d), the prime factorization of N can be computed deterministically in time $\mathcal{O}((\log n)^2)$.*
(2) *If p and q have the same bit size and $1 < ed \leq n^2$, then given (n, e, d), the prime factorization of n can be computed deterministically in time $\mathcal{O}((\log n)^9)$.*
(3) *Let β and $0 < \delta \leq 1/2$ be real numbers such that $2\beta\delta(1 - \delta) \leq 1$. Let $n = pq$ with p and q primes such that $p < n^{\delta}$ and $q < 2n^{1-\delta}$. Let $1 < ed \leq n^{\beta}$. Then given (n, e, d), the prime factorization of n can be computed deterministically in time $\mathcal{O}((\log n)^9)$.*

Proof: The results follow by applying Coppersmith's technique [14] of finding small solutions to the univariable modular polynomial equations using lattice reduction [15]. For more details, see [13]. ∎

Corollary 7.1 *If d is known, then the prime factorization n can be found in deterministic polynomial-time. That is,*

$$\{d\} \stackrel{\mathcal{P}}{\Longrightarrow} \text{IFP}(n). \tag{7.16}$$

Combining Theorem (7.3) and Corollary (7.1), we have

Theorem 7.4 (The equivalence of RSA(d) and IFP(n)) *Computing the private exponent d by giving the prime factorization N and computing the prime factorization of n by giving the private exponent d are deterministic polynomial-time equivalent. That is,*

$$\{d\} \stackrel{\mathcal{P}}{\Longleftrightarrow} \text{IFP}(n). \tag{7.17}$$

Remark 7.2 It was known from the moment the RSA cryptographic system was designed in 1977, that if the RSA private exponent d is given, then the prime factorization of the RSA modulus n can be computed in *random* polynomial-time by using Miller's techniques developed in 1976 [16]. That is,

$$\{d\} \stackrel{\mathcal{RP}}{\Longrightarrow} \text{IFP}(n). \tag{7.18}$$

The proof proceeds as follows. First note that

$$ed \equiv 1 \pmod{\phi(n)}.$$

then

$$ed = t\phi(n) + 1, t \in \mathbb{Z}.$$

Pick at random $x \in \mathbb{Z}_{\neq 0}$, this is guaranteed to satisfy

$$x^{ed-1} \equiv 1 \pmod{n}.$$

Then computing y_1 of 1 modulo N yields:

$$y_1 \equiv \sqrt{x^{ed-1}} \equiv x^{(ed-1)/2} \pmod{n}.$$

Therefore,

$$y_1^2 - 1 \equiv 0 \pmod{n}.$$

Thus, n can be factorized by computing

$$\gcd(y_1 \pm 1, n).$$

But this will only work when $y_1 \not\equiv \pm 1 \pmod{n}$. Suppose we are unlucky and obtain $y_1 \equiv \pm 1 \pmod{n}$ rather than a factor of n. If $y_1 \equiv -1 \pmod{n}$, we return to the beginning and pick another integer x. If $y_1 \equiv 1 \pmod{n}$, we take another square root of one via

$$\begin{aligned} y_2 &\equiv \sqrt{y_1} \\ &\equiv x^{(ed-1)/4} \pmod{n}. \end{aligned}$$

Hence,

$$y_2^2 - 1 \equiv 0 \pmod{n}.$$

Computing

$$\gcd(y_2 \pm 1, n).$$

Again, this will give a factor of n unless

$$y_2 \equiv \pm 1 \pmod{n}.$$

If we are unlucky, repeat the above process again (and again) until we have either factorized n or found $2 \nmid (ed - 1)/2^s$ for some $s \in \mathbb{Z}$. Clearly, the above process can be done in random polynomial-time. On the other hand, if IFP(n) is known then one can easily and deterministically find d in polynomial-time just by computing $d \equiv 1/e \pmod{(p-1)(q-1)}$.

Finally, we show that if the prime factorization of the RSA modulus n is known, the RSA plaintext M can be computed in polynomial-time from the corresponding ciphertext C.

Theorem 7.5 *If the prime factorization of the RSA modulus N is known, then RSA can be broken in polynomial-time. That is,*

$$\mathrm{IFP}(n) \overset{\mathcal{P}}{\Longrightarrow} M. \tag{7.19}$$

Proof: By Theorem 7.3, if the prime factorization of n is known, then d can be calculated in polynomial-time. Once d is found, then the following decryption process

$$M \equiv C^d \pmod{n},$$

can be done in polynomial-time, as (C, n) is known. ∎

This is the same as saying that computing d from the public key (e, N) is as hard as factoring the modulus N.

Example 7.4 Let

$$\begin{aligned} e &= 17579, \\ n &= 639784868795271438588314150 41, \\ d &= 106636877272320846243282850 19. \end{aligned}$$

We wish to find the prime factors p and q of N from the secret key d. We follow the procedure given in the proof of Theorem 7.2. Let $x = 2$ and perform the following computations:

for i from 2 to 100 do
  $s_i := (e * d - 1)/i$
  $y_i := 2_i^t \bmod n$
  print $(i, s_i, y_i, \gcd(y_i - 1, n))$
  end_do

We find that only those numbers when $i = 13, 26, 39, 52, 65, 78, 81, 91$ for $i \leq 100$ are lucky and give rise to $y_i \not\equiv 1 \pmod{n}$ and hence each leads to the complete prime factorization of $n = 145295143558111 \cdot 440334654777631$:

$$\begin{cases} s_{13} = (ed - 1)/13 = 144197666582317550470051478730 00 \\ y_{13} \equiv x^{s_{13}} \pmod{n} = 884402922605406885617295920 5 \\ \gcd(y_{13} - 1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{26} = (ed - 1)/26 = 720988332911587752350257393650 0 \\ y_{26} \equiv x^{s_{26}} \pmod{n} = 275980226045905369154668028 6 \\ \gcd(y_{13} - 1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{39} = (ed-1)/39 = 480658888607725168233504929 1000 \\ y_{39} \equiv x^{s_{39}} \pmod{n} = 1408741962175144428049208 7156 \\ \gcd(y_{39}-1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{52} = (ed-1)/52 = 360494166455793876175128696 8250 \\ y_{52} \equiv x^{s_{52}} \pmod{n} = 336619459358138613915602285 98 \\ \gcd(y_{52}-1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{65} = (ed-1)/65 = 288395333164635100940102957 4600 \\ y_{65} \equiv x^{s_{65}} \pmod{n} = 590173541933594942197795734 22 \\ \gcd(y_{65}-1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{78} = (ed-1)/78 = 240329444303862584116752464 5500 \\ y_{78} \equiv x^{s_{78}} \pmod{n} = 956417112215818257085919533 2 \\ \gcd(y_{78}-1, n) = 440334654777631 \end{cases}$$

$$\begin{cases} s_{81} = (ed-1)/81 = 231428353774089895816132002 9000 \\ y_{81} \equiv x^{s_{81}} \pmod{n} = 355905005236966211763919095 59 \\ \gcd(y_{81}-1, n) = 145295143558111 \end{cases}$$

$$\begin{cases} s_{91} = (ed-1)/91 = 205996666546167929242930683 9000 \\ y_{91} \equiv x^{s_{91}} \pmod{n} = 425913211637207795520959446 36 \\ \gcd(y_{91}-1, n) = 440334654777631 \end{cases}$$

The unlucky values of i in the range give rise to either $y_i \equiv 1 \pmod{n}$ or $s_i \notin \mathbb{Z}$.

Thus, to avoid the guessing d attack, N must be large, and d should be chosen from a large set such that the cryptanalyst cannot easily choose the correct d from the large set.

The fifth type of elementary attack is the common modulus attack The four RSA parameters $\{d, p, q, \phi(n)\}$ form the RSA trapdoor. These four pieces of information are equally important. Knowledge of any one of them reveals the knowledge of the remaining three, and hence break the RSA encryption completely. If RSA is not used properly, however, it may well be possible to break the RSA encryption without use of any knowledge of $\{d, p, q, \phi(n)\}$. One such improper use is the use of common modulus n in RSA encryption. Suppose that Bob sends Alice two ciphertexts C_1 and C_2 as follows:

$$C_1 \equiv M^{e_1} \pmod{n}$$
$$C_2 \equiv M^{e_2} \pmod{n}$$

where $\gcd(e_1, e_2) = 1$. Then as the following theorem shows, Eve can recover the plaintext M without factoring N and without using any of the trapdoor information $\{d, p, q, \phi(n)\}$.

Theorem 7.6 *Let $n_1 = n_2$ and $M_1 = M_2$ but $e_1 \neq e_2$ and $\gcd(e_1, e_2) = 1$ such that*

$$C_1 \equiv M^{e_1} \pmod{n}$$
$$C_2 \equiv M^{e_2} \pmod{n}$$

Then M can be recovered easily; that is,

$$\{[C_1, e_1, n],\ [C_2, e_2, n]\} \stackrel{\mathcal{P}}{\Longrightarrow} \{M\} \tag{7.20}$$

Proof: Since $\gcd(e_1, e_2) = 1$, then $e_1 x + e_2 y = 1$ with $x, y \in \mathbb{Z}$, which can be done by the extended Euclid's algorithm (or the equivalent continued fraction algorithm) in polynomial-time. Thus,

$$\begin{aligned} C_1^x C_2^y &\equiv (M_1^{e_1})^x (M_2^{e_2})^y \\ &\equiv M^{e_1 x + e_2 y} \\ &\equiv M \pmod{n} \end{aligned}$$

■

Example 7.5 Let

$$\begin{aligned} e_1 &= 9007, \\ e_2 &= 65537, \\ M &= 190503211809202519051822090305 19. \\ n &= 114381625757888867669235779976146612010218296721242362\_ \\ &\quad 562561842935706935245733897830597123563958705058989075\_ \\ &\quad 147599290026879543541 \end{aligned}$$

Then

$$\begin{aligned} C_1 &\equiv M^{e_1} \bmod n \\ &\equiv 104202250941196238413638382607974125774449084724929 59\_ \\ &\quad 125743374588926529777171718241302464293807835197908 99\_ \\ &\quad 45343407464161377977212 \\ C_2 &\equiv M^{e_2} \bmod n \\ &\equiv 764527507291887001807199705175445747109447573179098 96\_ \\ &\quad 041340987488285573190280783480309084978021563396490 75\_ \\ &\quad 9750600519496071304348 \end{aligned}$$

Now we determine x and y in

$$9007x + 65537y = 1.$$

First, we get the continued fraction expansion of 9007/65537 as follows:

$$9007/65537 = [0, 7, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 7].$$

Then we get the convergents of the continued fraction of 9007/65537 as follows:

$$\left[0, \frac{1}{7}, \frac{3}{22}, \frac{4}{29}, \frac{7}{51}, \frac{11}{80}, \frac{18}{131}, \frac{29}{211}, \frac{76}{553}, \frac{105}{764}, \frac{181}{1317}, \frac{286}{2081}, \frac{467}{3398}, \frac{1220}{8877}, \frac{9007}{65537}\right]$$

Thus,

$$\begin{cases} x = (-1)^{k-1} q_{k-1} = (-1)^{13} 8877 = -8877, \\ y = (-1)^{k} p_{k-1} = (-1)^{14} 1220 = 1220. \end{cases}$$

Therefore,

$$\begin{aligned} M &\equiv C_1^x C_2^y \\ &\equiv 104202250941196238413638382607974125774449084724929 59\_ \\ &\quad 125743374588926529777171718241302464293807835197908 99\_ \\ &\quad 453434074641613779772 12^{-8877} \cdot \\ &\quad 764527507291887001807199705175445747109447573179098 96\_ \\ &\quad 041340987488285573190280783480309084978021563396490 75\_ \\ &\quad 975060051949607130434 8^{1220} \\ &\equiv 190503211809202519051822090305 19 \pmod{n} \end{aligned}$$

So, we can recover the plaintext M without factoring N and/or using any of the trapdoor information $d, p, q, \phi(n)$.

This attack suggests that to defend RSA, one should never use common modulus in RSA encryption.

Finally we discuss the short d attack, based on the fact that if the RSA private exponent d that is chosen is too small, for example, $d < N^{0.25}$, then by Weiner's Diophantine attack [17], d can be efficiently recovered (in polynomial-time) from the public exponent e. That is,

$$\{e, n\} \xrightarrow[d<N^{0.25}]{\mathcal{P}} \{d\} \tag{7.21}$$

Weiner's idea is based on the properties of continued fractions and the idea of Diophantine approximation. In about 1842 Dirichlet showed that:

Theorem 7.7 *For any real α and any integer $Q > 1$, there exist integers p and q with $0 < q < Q$ such that*

$$|q\alpha - p| \leq \frac{1}{Q}. \tag{7.22}$$

This can be generalized to:

Corollary 7.2 *For any irrational α, there exist infinitely many integers p/q with $q > 0$ such that*

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{q^2}. \tag{7.23}$$

Theorem 7.8 *For any real α, each convergent p/q satisfies*

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{2q^2}. \tag{7.24}$$

Moreover,

$$\frac{p}{q} = \frac{p_i}{q_i}, \text{ for some } i. \tag{7.25}$$

Theorem 7.9 *Let $n = pq$ with p, q primes and $q < p < 2q$. Let $1 < e, d < \phi(n)$ with $ed \equiv 1 \pmod{\phi(n)}$. If $d < \frac{1}{3}\sqrt[4]{n}$, then d can be computed in polynomial-time.*

First we give a lemma relating to a property of continued fraction expansion of a rational number.

Lemma 7.1 Suppose that $\gcd(e, n) = \gcd(k, d) = 1$ and

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{2d^2}.$$

Then k/d is one of the convergents of the continued fraction expansion of e/n.

Proof: Let $n = pq$ with p, q prime and $1 < p < 2q$. Let also the private exponent d be small, say, for example,

$$d < \frac{1}{3}\sqrt[4]{n}.$$

Then, as we shall show, d will be the denominator of a convergent to the continued fraction expansion of e/n. ■

Theorem 7.10 (Weiner) *Let $n = pq$ with p and q primes such that*

$$\begin{cases} q < p < 2q \\ d < \frac{1}{3}t\sqrt[4]{n} \end{cases} \tag{7.26}$$

then given e with $ed \equiv 1 \pmod{\phi(n)}$, d can be efficiently calculated.

Proof: Since $ed \equiv 1 \pmod{\phi(n)}$,

$$ed - k\phi(n) = 1$$

for some $k \in \mathbb{Z}$. Therefore

$$\left|\frac{e}{\phi(n)} - \frac{k}{d}\right| = \frac{1}{d\phi(n)}.$$

Since $n = pq > q^2$, we have $q < \sqrt{n}$. Also since $\phi(n) = n - p - q + 1$,

$$0 < n - \phi(n) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

Now,

$$\begin{aligned} \left|\frac{e}{n} - \frac{k}{d}\right| &= \left|\frac{ed - kn}{dn}\right| \\ &= \left|\frac{ed - kn + k\phi(n) - k\phi(n)}{dn}\right| \\ &= \left|\frac{1 - k(n - \phi(N))}{dn}\right| \\ &< \frac{3k\sqrt{n}}{dn} \\ &= \frac{3k}{d\sqrt{n}} \\ &< \frac{1}{2d^2}. \end{aligned}$$

Thus, by Lemma 7.1, k/d must be one of convergents of the simple continued fraction e/n. Therefore, if $d < \frac{1}{3}\sqrt[4]{n}$, the d can be computed via the elementary task of computing a few convergent of e/N, which can be done in polynomial-time. ∎

Theorem 7.10 tells us that the private key d should be large enough (nearly as many bits as the modulus N); or otherwise, by the properties of continued fractions, the private key d can be found in time polynomial in the length of the modulus n, and hence decrypt RSA(M).

Example 7.6 Suppose that $n = 160523347$ and $e = 60728973$. Then the continued fraction expansion of e/n is as follows:

$$\frac{e}{N} = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{12 + \cfrac{1}{102 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{2 + \frac{1}{36}}}}}}}}}}}}}}$$

$$= [0, 2, 1, 1, 1, 4, 12, 102, 1, 1, 2, 3, 2, 2, 36]$$

and the convergents of the continued fraction are as follows:

$$\left[0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{14}{37}, \frac{171}{452}, \frac{17456}{46141}, \frac{17627}{46593}, \frac{35083}{92734}, \frac{87793}{232061}, \frac{298462}{788917},\right.$$

$$\left.\frac{684717}{1809895}, \frac{1667896}{4408707}, \frac{60728973}{160523347}\right].$$

If condition (7.26) is satisfied, then the unknown fraction k/d is a close approximation to the known fraction of e/n. Lemma 7.1 tells us that k/d must be one of the covergents of the continued fraction expansion of e/n. As can be seen, there are 15 such convergents; we need to find the "right" one. There are two methods to find the right value for d.

(1) The first method is to use the following trial-and-error procedure: If k/d is a convergent of e/N, then we compute $\phi(n) = (ed - 1)/k$ and solve the quadratic equation $p^2 - (n - \phi(n) + 1)p + n = 0$, and check if it leads to a factorization of n. If yes, use the factors p and q just found to compute $d \equiv 1/e \pmod{\phi(n)}$ and to recover M from C.

If no, then pick up the next convergent of e/n and try again. The following is the trial process starting from the second convergents:

Suppose $\frac{k}{d} = \frac{1}{2}$, then

$$\phi(n) = \frac{ed-1}{k} = 60728973 \cdot 2 - 1 = 121457945,$$
$$p^2 - (n - \phi(n) + 1)p + n = 0$$
$$\Longrightarrow p = \frac{39065403}{2} - \frac{\sqrt{1526105069459021}}{2}$$

which is impossible, thus $d \neq 2$.

Suppose $\frac{k}{d} = \frac{1}{3}$, then

$$\phi(n) = \frac{ed-1}{k} = 60728973 \cdot 3 - 1 = 182186918,$$
$$p^2 - (n - \phi(n) + 1)p + n = 0$$
$$\Longrightarrow p = -10831785 - \sqrt{117327405762878}$$

which is impossible, thus $d \neq 3$.

Suppose $\frac{k}{d} = \frac{2}{5}$, then

$$\phi(N) = \frac{ed-1}{k} = \frac{60728973 \cdot 5 - 1}{2} = 151822432,$$
$$p^2 - (n - \phi(n) + 1)p + n = 0$$
$$\Longrightarrow p = 4350458 - 3\sqrt{2102924920713}$$

which is impossible, thus $d \neq 5$.

Suppose $\frac{k}{d} = \frac{3}{8}$, then

$$\phi(n) = \frac{ed-1}{k} = \frac{60728973 \cdot 8 - 1}{3} = \frac{485831783}{3},$$
$$p^2 - (n - \phi(n) + 1)p + n = 0$$
$$\Longrightarrow p = -\frac{4261739}{6} - \frac{18156640463629}{6}$$

which is impossible, thus $d \neq 8$.

Suppose $\frac{k}{d} = \frac{14}{37}$, then

$$\phi(n) = \frac{ed-1}{k} = \frac{60728973 \cdot 37 - 1}{14} = 160498000,$$
$$p^2 - (n - \phi(n) + 1)p + n = 0$$
$$\Longrightarrow p = 12347,\ q = n/p = 13001.$$

Sucessful, since $n = pq = 12347 \cdot 13001 = 160523347$.

Given p, q, we can now find $d \equiv 1/e \equiv 37 \pmod{(12347-1)(13001-1)}$ easily, and hence find the plaintext M. Of course, the above procedure to find d by finding (p, q) after the convergents of e/N is known is not the only possible procedure.

(2) The second possible method is to test whether or not $a^{ed} \equiv a \pmod{n}$ for some randomly chosen a, since d should be one of the denominators

$$\{2, 3, 5, 8, 37, 452, 46141, 46593, 92734, 232061, 788917, 1809895, 4408707, 160523347\}$$

of the e/n convergents, but of course we do not know which one. Thus, we can simply test:

$$\begin{aligned}
2^{60728973 \cdot 2} \bmod 160523347 &= 137369160 \neq 2 \\
2^{60728973 \cdot 3} \bmod 160523347 &= 93568289 \neq 2 \\
2^{60728973 \cdot 5} \bmod 160523347 &= 73692312 \neq 2 \\
2^{60728973 \cdot 8} \bmod 160523347 &= 30860603 \neq 2 \\
2^{60728973 \cdot 37} \bmod 160523347 &= 2
\end{aligned}$$

which gives $d = 37$, as required.

Remark 7.3 The above attack in fact gives all the trapdoor information $\{d, \phi(n), p, q\}$ as follows:

$$\begin{aligned}
\phi(n) &= (ed - 1)/p_i \\
&= (60728973 \cdot 37 - 1)/14 \\
&= 160498000, \\
\{p, q\} &\Leftarrow x^2 - (N - \phi(n) + 1) + n \\
&\Rightarrow x^2 - (160523347 - 160498000 + 1)x + 160523347 \\
&\Rightarrow \{p, q\} = 12347 \cdot 13001
\end{aligned}$$

Thus, for $n = 160523347$, Wiener's attack works well for

$$d < \frac{\sqrt[4]{n}}{3} \approx 37.52.$$

Example 7.7 Let

$$\begin{aligned}
n &= 28562942440499 \\
e &= 7502876735617.
\end{aligned}$$

Then the continued fraction expansion of e/n is as follows:

$$\frac{e}{n} = [0, 3, 1, 4, 5, 1, 1, 3, 16, 1, 7, 1, 7, 1, 4, 1, 1, 2, 1, 1, 3, 3, 4, 2, 2, 4, 12, 3, 2, 1, 2]$$

and the convergents of the continued fraction are as follows:

$$\left[0, \frac{1}{3}, \frac{1}{4}, \frac{5}{19}, \frac{26}{99}, \frac{31}{118}, \frac{57}{217}, \frac{202}{769}, \frac{3289}{12521}, \frac{3491}{13290}, \frac{27726}{105551}, \frac{31217}{118841}, \frac{246245}{937438}, \frac{277462}{1056279}, \frac{1356093}{5162554}, \frac{1633555}{6218833}, \frac{2989648}{11381387}, \frac{7612851}{28981607}, \frac{10602499}{40362994}, \frac{18215350}{69344601}, \frac{65248549}{248396797}, \frac{213960997}{814534992}, \frac{921092537}{3506536765}, \frac{2056146071}{7827608522}, \frac{5033384679}{19161753809}, \frac{22189684787}{84474623758}, \frac{271309602123}{1032857238905}, \frac{836118491156}{3183046340473}, \frac{1943546584435}{7398949919851}, \frac{2779665075591}{10581996260324}, \frac{7502876735617}{28562942440499}\right].$$

So the required d must be one of the denominates of the above covergents, but we do not know which one, so we just try

$$\begin{aligned}
(3^{7502876735617})^3 &\equiv 17387646817554 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^4 &\equiv 7072755623312 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^{19} &\equiv 11902526494611 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^{99} &\equiv 5513494147015 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^{118} &\equiv 8201089526821 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^{217} &\equiv 8739051274402 \not\equiv 3 \pmod{28562942440499}\\
(3^{7502876735617})^{769} &\equiv 3 \pmod{28562942440499}
\end{aligned}$$

So, $d = 769$. Clearly, for $n = 28562942440499$, Wiener's attack works well for

$$d < \frac{\sqrt[4]{n}}{3} \approx 770.6.$$

Problems for Section 7.2

1. Show that given $n, N \in \mathbb{Z}^+$, with $N > 1, n > N$, then there exists a unique $r \in \mathbb{Z}_{\geq 0}$ such that $N^r \leq n \leq N^{r+1}$.
2. (RSA conjecture) Cryptanalyzing RSA is as hard as solving the Integer Factorization Problem. Prove or disprove this conjecture.
3. Reduce factoring integers to cracking RSA in probabilistic polynomial-time.
4. Prove or disprove that solving the Integer Factorization Problem (IFP) is equivalent to breaking the RSA cryptosystem.
5. Let $0 \leq x < N$. If

$$x^{e^k} \equiv x \pmod{N},\ k \in \mathbb{Z}^+$$

then x is called the fixed-point of RSA(e, N) and k is the order of the fixed-point. Suppose C is the fixed-point of RSA(e, N) with order k:

$$C^{e^k} \equiv C \pmod{N},\ k \in \mathbb{Z}^+.$$

Show that

$$C^{e^{k-1}} \equiv M \pmod{N},\ k \in \mathbb{Z}^+$$

(Note: The above result gives rise to an attack, called the fixed-point attack.)

6. (Coppersmith) Let $n = pq$ be a β-bit RSA modulo such that p has about $\beta/2$-bits. Show that if either $\beta/4$ least (or most) significant bits of p is given, then n can be efficiently factored in polynomial-time, and hence, RSA can be efficiently broken in polynomial-time.
7. (Boneh, et al.) Show that if $d/4$ least significant bits of d are given, then d can be computed in time $\mathcal{O}(e \log e)$, where e is the RSA public exponent.
8. Show that the following three problems are polynomial-time equivalent:
   (1) Factoring n.
   (2) Computing $\phi(n)$.
   (3) Computing $d \in \mathbb{Z}^+$ such that $ed \equiv 1 \pmod{\phi(n)}$, where the pair (e, n) is given.
9. Explain why all the known attacks on RSA do not actually threaten the security of RSA. Justify your answer.

7.3 Rabin Cryptography

As can be seen from the previous sections, RSA uses M^e for encryption, with $e \geq 3$ (3 is the smallest possible public exponent in RSA); in this way, we might call RSA encryption M^e encryption. In 1979, Michael Rabin [18] proposed a scheme based on M^2 encryption rather than the M^e for $e \geq 3$ encryption used in RSA. A brief description of the Rabin cryptosystem is as follows (see also Figure 7.3).

[1] **Key generation:** Let $n = pq$ with p, q odd primes satisfying

$$p \equiv q \equiv 3 \pmod{4}. \tag{7.27}$$

[2] **Encryption:**

$$C \equiv M^2 \pmod{n}. \tag{7.28}$$

[3] **Decryption:** Use the Chinese Remainder theorem to solve the system of congruences:

$$\begin{cases} M_p \equiv \sqrt{C} \pmod{p} \\ M_q \equiv \sqrt{C} \pmod{q} \end{cases} \tag{7.29}$$

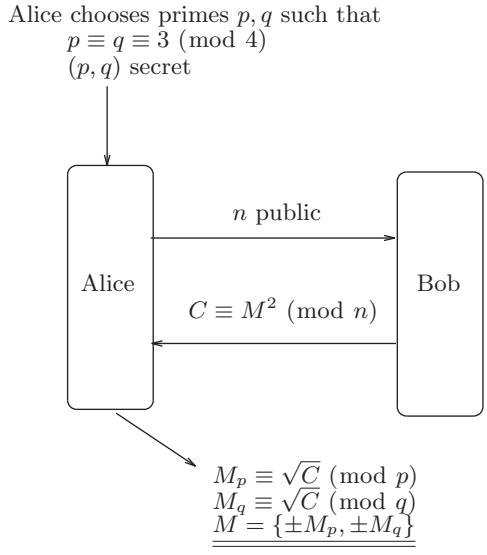


Figure 7.3 Rabin cryptosystem

to get the four solutions: $\{\pm M_p, \pm M_q\}$. The true plaintext M will be one of these four values.

[4] **Cryptanalysis:** A cryptanalyst who can factor n can compute the four square roots of C modulo n, and hence can recover M from C. Thus, breaking the Rabin system is equivalent to factoring n.

Example 7.8 Let $M = 31$.

[1] **Key generation:** Let $n = 11 \cdot 19$ be the public key, but keep the prime factors $p = 11$ and $q = 19$ of n secret.

[2] **Encryption:**

$$C \equiv 31^2 \equiv 125 \,(\text{mod } 209).$$

[3] **Decryption:** Compute

$$\begin{cases} M_p \equiv \sqrt{125} \equiv \pm 2 \,(\text{mod } p) \\ M_q \equiv \sqrt{125} \equiv \pm 7 \,(\text{mod } q). \end{cases}$$

Now use the Chinese Remainder theorem to solve

$$\begin{cases} M \equiv 2 \,(\text{mod } 11) \\ M \equiv 7 \,(\text{mod } 19) \end{cases} \Longrightarrow M = 178$$

$$\begin{cases} M \equiv -2 \,(\text{mod } 11) \\ M \equiv 7 \,(\text{mod } 19) \end{cases} \Longrightarrow M = 64$$

$$\begin{cases} M \equiv -2 \,(\mathrm{mod}\ 11) \\ M \equiv 7 \,(\mathrm{mod}\ 19) \end{cases} \Longrightarrow M = 145$$
$$\begin{cases} M \equiv -2 \,(\mathrm{mod}\ 11) \\ M \equiv -7 \,(\mathrm{mod}\ 19) \end{cases} \Longrightarrow M = 31$$

The true plaintext M will be one of the above four values, and in fact, $M = 31$ is the true value.

Unlike the RSA cryptosystem, whose security was only conjectured to be equivalent to the intractability of IFP, the security of Rabin system, and its variants such as the Rabin–Williams system, is proved to be equivalent to the intractability of IFP. First notice that there is a fast algorithm to compute the square roots modulo N if $n = pq$ is known. Consider the following quadratic congruence

$$x^2 \equiv y \,(\mathrm{mod}\ p) \tag{7.30}$$

there are essentially three cases for the prime p:

(1) $p \equiv 3 \,(\mathrm{mod}\ 4)$,
(2) $p \equiv 5 \,(\mathrm{mod}\ 8)$,
(3) $p \equiv 1 \,(\mathrm{mod}\ 8)$.

All three cases may be solved by the following process:

$$\begin{cases} \text{if p} \equiv 3 \,(\mathrm{mod}\ 4), & x \equiv \pm y^{\frac{p+1}{4}} \,(\mathrm{mod}\ p), \\ \text{if } p \equiv 5 \,(\mathrm{mod}\ 8), & \begin{cases} \text{if y}^{\frac{p+1}{4}} = 1, & x \equiv \pm y^{\frac{p+3}{8}} \,(\mathrm{mod}\ p) \\ \text{if } y^{\frac{p+1}{4}} \neq 1, & x \equiv \pm 2y(4y)^{\frac{p-5}{8}} \,(\mathrm{mod}\ p). \end{cases} \end{cases} \tag{7.31}$$

Algorithm 7.2 (Computing square roots modulo pq) Let $n = pq$ with p and q odd prime and $y \in \mathrm{QR}_n$. This algorithm will find all the four solutions in x to congruence $x^2 \equiv y \,(\mathrm{mod}\ pq)$ in time $\mathcal{O}((\log p)^4)$.

[1] Use (7.31) to find a solution r to $x^2 \equiv y \,(\mathrm{mod}\ p)$.
[2] Use (7.31) to find a solution s to $x^2 \equiv y \,(\mathrm{mod}\ q)$.
[3] Use the Extended Euclid's algorithm to find integers c and d such that $cp + dq = 1$.
[4] Compute $x \equiv \pm(rdq \pm scp) \,(\mathrm{mod}\ pq)$.

On the other hand, if there exists an algorithm to find the four solutions in x to $x^2 \equiv y \,(\mathrm{mod}\ n)$, then there exists an algorithm to find the prime factorization of n. The following is the algorithm.

Algorithm 7.3 (Factoring via square roots) This algorithm seeks to find a factor of n by using an existing square root finding algorithm (namely, Algorithm 7.2).

[1] Choose at random an integer x such that $\gcd(x, n) = 1$, and compute $x^2 \equiv a \pmod{n}$.
[2] Use Algorithm 7.5 to find four solutions in x to $x^2 \equiv a \pmod{n}$.
[3] Choose one of the four solutions, say y such that $y \not\equiv \pm x \pmod{n}$, then compute $\gcd(x \pm y, n)$.
[4] If $\gcd(x \pm y, n)$ reveals p or q, then go to Step [5], or otherwise, go to Step [1].
[5] Exit.

Theorem 7.11 *Let $N = pq$ with p, q odd prime. If there exists a polynomial-time algorithm A to factor $n = pq$, then there exists an algorithm B to find a solution to $x^2 \equiv y \pmod{n}$, for any $y \in \mathrm{QR}_N$.*

Proof: If there exists an algorithm A to factor $n = pq$, then there exists an algorithm (in fact, Algorithm 7.2), which determines $x = \pm(rdq \pm scp) \pmod{pq}$, as defined in Algorithm 7.2, for $x^2 \equiv y \pmod{n}$. Clearly, Algorithm 7.2 runs in polynomial-time. ∎

Theorem 7.12 *Let $n = pq$ with p, q odd prime. If there exists a polynomial-time algorithm A to find a solution to $x^2 \equiv a \pmod{n}$, for any $a \in \mathrm{QR}_n$, then there exists a probabilistic polynomial time algorithm B to find a factor of n.*

Proof: First note that for n composite, x and y integer, if $x^2 \equiv y^2 \pmod{n}$ but $x \not\equiv \pm y \pmod{n}$, then $\gcd(x + y, n)$ are proper factors of n. If there exists an algorithm A to find a solution to $x^2 \equiv a \pmod{n}$ for any $a \in \mathrm{QR}_n$, then there exists an algorithm (in fact, Algorithm 7.3), which uses algorithm A to find four solutions in x to $x^2 \equiv a \pmod{n}$ for a random x with $\gcd(x, n) = 1$. Select one of the solutions, say, $y \not\equiv \pm x \pmod{n}$, then by computing $\gcd(x \pm y, n)$, the probability of finding a factor of N will be $\geq 1/2$. If Algorithm 7.3 runs for k times and each time randomly chooses a different x, then the probability of not factoring n is $\leq 1/2^k$. ∎

So, finally, we have

Theorem 7.13 *Factoring integers, computing the modular square roots, and breaking the Rabin cryptosystem are computationally equivalent. That is,*

$$\mathrm{IFP}(n) \overset{\mathcal{P}}{\Longleftrightarrow} \mathrm{Rabin}(M). \tag{7.32}$$

Williams [19] proposed a modified version of the RSA cryptographic system, particularly Rabin's M^2 system, in order to make it suitable as a public-key encryption scheme (Rabin's original system was intended to be used as a digital signature scheme). A description of

Williams' M^2 encryption is as follows (suppose Bob wishes to send Alice a ciphertext $C \equiv M^2 \pmod{n}$):

[1] Key generation: Let $n = pq$ with q and q primes such that

$$\begin{cases} p \equiv 3 \pmod{8}, \\ q \equiv 7 \pmod{8}. \end{cases} \tag{7.33}$$

So, $n \equiv 5 \pmod{8}$ and $(n, 2)$ is used as the public key. The private key d is defined by

$$d = \frac{(p-1)(q-1)}{4} + 1. \tag{7.34}$$

[2] Encryption: Let $\mathcal{M}$ be plaintext space containing all possible plaintexts M such that

$$2(2M+1) < n \tag{7.35}$$

if the Jacobi symbol

$$\left(\frac{2M+1}{n}\right) = -1, \tag{7.36}$$

and

$$4(2M+1) < n \tag{7.37}$$

if the Jacobi symbol

$$\left(\frac{2M+1}{n}\right) = 1. \tag{7.38}$$

The first step in encryption is for all $M \in \mathcal{M}$, put

$$\begin{aligned} M' &= E_1(M) \\ &= \begin{cases} 2(2M+1) \\ \text{if the Jacobi symbol } \left(\frac{2M+1}{n}\right) = -1, \\ 4(2M+1) \\ \text{if the Jacobi symbol } \left(\frac{2M+1}{n}\right) = 1. \end{cases} \end{aligned} \tag{7.39}$$

The last step in encryption is just the same as Rabin's encryption:

$$C \equiv (M')^2 \pmod{n}. \tag{7.40}$$

[3] Decryption: On the reverse order of the encryption, the first step in decryption is as follows:

$$C' = D_2(C) \equiv C^d \,(\mathrm{mod}\, n) \tag{7.41}$$

and the last step in decryption is defined by:

$$M = D_1(C') = \begin{cases} \dfrac{\frac{M'}{4} - 1}{2} & \text{if } M' \equiv 0 \,(\mathrm{mod}\, 4) \\ \dfrac{\frac{N - M'}{4} - 1}{2} & \text{if } M' \equiv 1 \,(\mathrm{mod}\, 4) \\ \dfrac{\frac{M'}{2} - 1}{2} & \text{if } M' \equiv 2 \,(\mathrm{mod}\, 4) \\ \dfrac{\frac{N - M'}{2} - 1}{2} & \text{if } M' \equiv 3 \,(\mathrm{mod}\, 4). \end{cases} \tag{7.42}$$

The whole process of encryption and decryption is as follows:

$$M \xrightarrow{E_1} M' \xrightarrow{E_2} C \xrightarrow{D_2} M' \xrightarrow{D_1} M.$$

[4] Cryptanalysis: A cryptanalyst who can factor n can find d, and hence can recover M from C. Thus, breaking the Williams' system is equivalent to factoring n.

Theorem 7.14 (Correctness of Williams' M^2 encryption) *Let $M \in \mathcal{M}$. Then*

$$M = D_1(D_2(E_2(E_1(M)))). \tag{7.43}$$

Theorem 7.15 (Equivalence of Williams(M) and IFP(n)) *Breaking Williams' M^2 encryption (i.e., finding M from C) is equivalent to factoring the modulus n. That is,*

$$\text{IFP(n)} \overset{\mathcal{P}}{\Longleftrightarrow} \text{Williams}(M). \tag{7.44}$$

Just the same as Rabin's system, Williams' M^2 encryption is also provably secure, as breaking the Williams' M^2 mod n encryption is equivalent to factoring n, where the N is a special form of $N = pq$, with p, q primes and $p \equiv 3 \,(\mathrm{mod}\, 8)$ and $q \equiv 7 \,(\mathrm{mod}\, 8)$. Note that this special integer factorization problem is not the same as the general IFP, although there is no any known reason to believe this special factoring problem is any easier than the general factoring problem. But unlike Rabin's system, Williams' M^2 encryption can be easily generalized to the general M^e encryption with $e > 2$, as in RSA. Thus, Williams' M^2 encryption is not just a variant of Rabin system, but also a variant of the general RSA system.

Williams' M^2 encryption improved Rabin's M^2 encryption by eliminating the 4 : 1 ciphertext ambiguity problem in decryption without adding extra information for removing

the ambiguity. Williams in [20] also proposed a M^3 *encryption* variant to Rabin but eliminated the 9 : 1 ciphertext ambiguity problem. The encryption is also proved to be as hard as factoring, although it is again still not the general IFP, since $n = pq$ was chosen to be

$$p \equiv q \equiv 1 \,(\mathrm{mod}\, 3) \tag{7.45}$$

and

$$\frac{(p-1)(q-1)}{9} \equiv -1 \,(\mathrm{mod}\, 3). \tag{7.46}$$

Problems for Section 7.3

1. Let $M = 31, n = pq = 11 \cdot 19$ and $b = 187$. Let also the Rabin encryption be as follows:

   $$C \equiv M^2 + bM \,(\mathrm{mod}\, n).$$

   then M should be recovered by solving the quadratic congruence

   $$M^2 + bM - c \equiv 0 \,(\mathrm{mod}\, n).$$

   (1) Find C.
   (2) Use the Chinese Remainder theorem to find the four possible values M (the correct value should be one of the four possible values).
2. Show that breaking the Rabin encryption is equivalent to factoring the Rabin modulo n.
3. Give a method to eliminate the 4 : 1 ciphertext ambiguity problem in deciphering Rabin's codes without adding extra information.
4. Show that breaking the Williams M^2 encryption is equivalent to factoring the Williams modulo n.
5. Generalize Williams' M^2 encryption to M^e ($e \geq 3$) encryption.
6. Let

   $$\begin{aligned} n = {} & 21290246318258757547497882016271517497806703963277216278233 \\ & 38321538470570413250102890108976982548192582551350925260960 \\ & 2369983944024335907529 \\ C \equiv {} & M^2 \,(\mathrm{mod}\, n) \\ = {} & 51285205060243481188122109876540661122140906807437327290641 \\ & 60633920242479741450841196687149365272035106423411648279360 \\ & 3932042884271651389234 \end{aligned}$$

   Find the plaintext M.

7.4 Residuosity Based Cryptography

The RSA cryptosystem discussed in the previous sections is *deterministic* in the sense that under a fixed public key, a particular plaintext M is always encrypted to the same ciphertext C. Some of the drawbacks of a deterministic scheme are:

[1] It is not secure for all probability distributions of the message space. For example, in RSA encryption, the messages 0 and 1 always get encrypted to themselves, and hence are easy to detect.
[2] It is easy to obtain some partial information about the secret key (p, q) from the public modulus n (assume that $n = pq$). For example, when the least-significant digit of n is 3, then it is easy to obtain the partial information that the least-significant digits of p and q are either 1 and 3 or 7 and 9, and are indicated as follows:

$$\begin{aligned} 183 &= 3 \cdot 61 \qquad & 253 &= 11 \cdot 23 \\ 203 &= 7 \cdot 29 \qquad & 303 &= 3 \cdot 101 \\ 213 &= 3 \cdot 71 \qquad & 323 &= 17 \cdot 19 \end{aligned}$$

[3] It is sometimes easy to compute partial information about the plaintext M from the ciphertext C. For example, given (C, e, n), the Jacobi symbol of M over n can be easily deduced from C:

$$\left(\frac{C}{n}\right) = \left(\frac{M^e}{n}\right)\left(\frac{M}{n}\right)^e = \left(\frac{M}{n}\right). \tag{7.47}$$

[4] It is easy to detect when the same message is sent twice.

Probabilistic encryption, or randomized encryption, however, utilizes randomness to attain a strong level of security, namely, the *polynomial security* and *semantic security*, defined as follows:

Definition 7.1 A public-key encryption scheme is said to be *polynomially* secure if no passive adversary can, in expected polynomial-time, select two plaintexts M_1 and M_2 and then correctly distinguish between encryptions of M_1 and M_2 with a probability significantly greater that $1/2$.

Definition 7.2 A public-key encryption scheme is said to be *semantically* secure if, for all probability distributions over the message space, whatever a passive adversary can compute in expected polynomial-time about the plaintext given the ciphertext, it can also be computed in expected polynomial-time without the ciphertext.

Intuitively, a public-key encryption scheme is semantically secure if the ciphertext does not leak any partial information whatsoever about the plaintext that can be computed in expected polynomial time. That is, given (C, e, n), it should be intractable to recover any

information about M. Clearly, a public-key encryption scheme is semantically secure if and only if it is polynomially secure.

In this section, we shall introduce a semantically secure cryptosystem based on the *quadratic residuosity problem*. Recall that an integer a is a quadratic residue modulo n, denoted by $a \in Q_n$, if $\gcd(a, n) = 1$ and there exists a solution x to the congruence $x^2 \equiv a \pmod{n}$, otherwise a is a quadratic nonresidue modulo n, denoted by $a \in \overline{Q}_n$. The Quadratic Residuosity Problem may be stated as:

> Given positive integers a and n, decide whether or not $a \in Q_n$.

It is believed that solving QRP is equivalent to computing the prime factorization of n, so it is computationally infeasible. If n is prime then

$$a \in Q_n \Longleftrightarrow \left(\frac{a}{n}\right) = 1, \tag{7.48}$$

and if n is composite, then

$$a \in Q_n \Longrightarrow \left(\frac{a}{n}\right) = 1, \tag{7.49}$$

but

$$a \in Q_n \quad \not\Longleftarrow \quad \left(\frac{a}{n}\right) = 1, \tag{7.50}$$

however

$$a \in \overline{Q}_n \Longleftarrow \left(\frac{a}{n}\right) = -1. \tag{7.51}$$

Let $J_n = \{a \in (\mathbb{Z}/n\mathbb{Z})^* : \left(\frac{a}{n}\right) = 1\}$, then $\tilde{Q}_n = J_n - Q_n$. Thus, $\tilde{Q}_n$ is the set of all pseudosquares modulo n; it contains those elements of J_n that do not belong to Q_n. Readers may wish to compare this result to Fermat's little theorem, namely (assuming $\gcd(a, n) = 1$),

$$n \text{ is prime} \Longrightarrow a^{n-1} \equiv 1 \pmod{n}, \tag{7.52}$$

but

$$n \text{ is prime} \quad \not\Longleftarrow \quad a^{n-1} \equiv 1 \pmod{n}, \tag{7.53}$$

however

$$n \text{ is composite} \Longleftarrow a^{n-1} \not\equiv 1 \pmod{n}. \tag{7.54}$$

The Quadratic Residuosity Problem can then be further restricted to:

> Given a composite n and an integer $a \in J_n$, decide whether or not $a \in Q_n$.

For example, when $n = 21$, we have $J_{21} = \{1, 4, 5, 16, 17, 20\}$ and $Q_{21} = \{1, 4, 16\}$, thus $\tilde{Q}_{21} = \{5, 17, 20\}$. So, the QRP problem for $n = 21$ is actually to distinguish squares $\{1, 4, 16\}$ from pseudosquares $\{5, 17, 20\}$. The only method we know for distinguishing squares from pseudosquares is to factor n; since integer factorization is computationally infeasible, the QRP is computationally infeasible. In what follows, we shall present a cryptosystem whose security is based on the infeasibility of the Quadratic Residuosity Problem; it was first proposed by Goldwasser and Micali in 1984 [21] in 1984, under the term *probabilistic encryption*.

Algorithm 7.4 (Quadratic residuosity based cryptography) This algorithm uses the randomized method to encrypt messages and is based on the Quadratic Residuosity Problem (QRP). The algorithm divides into three parts: Key generation, message encryption, and decryption.

[1] Key generation: Both Alice and Bob should do the following to generate their public and secret keys:
  [a] Select two large distinct primes p and q, each with roughly the same size, say, each with β-bits.
  [b] Compute $n = pq$.
    Select a $y \in \mathbb{Z}/n\mathbb{Z}$, such that $y \in \overline{Q}_n$ and $\left(\frac{y}{n}\right) = 1$. ($y$ is thus a pseudosquare modulo n).
  [c] Make (n, y) public, but keep (p, q) secret.

[2] Encryption: To send a message to Alice, Bob should do the following:
  [a] Obtain Alice's public key (n, y).
  [b] Represent the message m as a binary string $m = m_1 m_2 \cdots m_k$ of length k.
  [c] For i from 1 to k

    [d-1] Choose at random an $x \in (\mathbb{Z}/n\mathbb{Z})^*$ and call it x_i.
    [d-2] Compute c_i:

$$c_i = \begin{cases} x_i^2 \bmod n, & \text{if } m_i = 0, \quad \text{(r.s.)} \\ y x_i^2 \bmod n, & \text{if } m_i = 1, \quad \text{(r.p.s.)}, \end{cases} \tag{7.55}$$

    where r.s. and r.p.s. represent random square and random pseudosquare, respectively.

  Send the k-tuple $c = (c_1, c_2, \ldots, c_k)$ to Alice. (Note first that each c_i is an integer with $1 \le c_i < n$. Note also that since n is a 2β-bit integer, it is clear that the ciphertext c is a much longer string than the original plaintext m.)

[3] Decryption: To decrypt Bob's message, Alice should do the following:
  [1] For i from 1 to k

    [a-1] Evaluate the Legendre symbol:

$$e_i' = \left(\frac{c_i}{p}\right). \tag{7.56}$$

[a-2] Compute m_i:

$$m_i = \begin{cases} 0, & \text{if } e_i' = 1 \\ 1, & \text{if otherwise.} \end{cases} \tag{7.57}$$

That is, $m_i = 0$ if $c_i \in Q_n$, otherwise, $m_i = 1$.

Finally, get the decrypted message $m = m_1 m_2 \cdots m_k$.

Remark 7.4 The above encryption scheme has the following interesting features:

(1) The encryption is random in the sense that the same bit is transformed into different strings depending on the choice of the random number x. For this reason, it is called *probabilistic* (or *randomized*) encryption.
(2) Each bit is encrypted as an integer modulo n, and hence is transformed into a 2β-bit string.
(3) It is semantically secure against any threat from a polynomially bounded attacker, provided that the QRP is hard.

Example 7.9 In what follows we shall give an example of how Bob can send the message "HELP ME" to Alice using the above cryptographic method. We use the binary equivalents of letters as defined in Table 7.1. Now both Alice and Bob proceed as follows:

[1] Key generation:
- Alice chooses $(n, y) = (21,\ 17)$ as a public key, where $n = 21 = 3 \cdot 7$ is a composite, and $y = 17 \in \tilde{Q}_{21}$ (since $17 \in J_{21}$ but $17 \notin Q_{21}$), so that Bob can use the public key to encrypt his message and send it to Alice.

Table 7.1 The binary equivalents of letters

Letter	Binary code	Letter	Binary code	Letter	Binary code
A	00000	B	00001	C	00010
D	00011	E	00100	F	00101
G	00110	H	00111	I	01000
J	01001	K	01010	L	01011
J	01001	K	01010	L	01011
M	01100	N	01101	O	01110
P	01111	Q	10000	R	10001
S	10010	T	10011	U	10100
V	10101	W	10110	X	10111
Y	11000	Z	11001	⊔	11010

- Alice keeps the prime factorization $(3, 7)$ of 21 secret; since $(3, 7)$ will be used as a private decryption key. (Of course, here we just show an example; in practice, the prime factors p and q should be at last 100 digits.)

[2] Encryption:

- Bob converts his plaintext HELP ME to the binary stream $M = m_1 m_2 \cdots m_{35}$:

$$00111\,00100\,01011\,01111\,11010\,01100\,00100.$$

(To save space, we only consider how to encrypt and decrypt $m_2 = 0$ and $m_3 = 1$; it is suggested that readers encrypt and decrypt the whole binary stream.)

- Bob randomly chooses integers $x_i \in (\mathbb{Z}/21\mathbb{Z})^*$. Suppose he chooses $x_2 = 10$ and $x_3 = 19$ which are elements of $(\mathbb{Z}/21\mathbb{Z})^*$.
- Bob computes the encrypted message $C = c_1 c_2 \cdots c_k$ from the plaintext $M = m_1 m_2 \cdots m_k$ using Equation (7.55). To get, for example, c_2 and c_3, Bob performs:

$$\begin{aligned} c_2 &= x_2^2 \bmod 21 = 10^2 \bmod 21 = 16, && \text{since } m_2 = 0, \\ c_3 &= y \cdot x_3^2 \bmod 21 = 17 \cdot 19^2 \bmod 21 = 5, && \text{since } m_3 = 1. \end{aligned}$$

(Note that each c_i is an integer reduced to 21, i.e., m_i is a bit, but its corresponding c_i is not a bit but an integer, which is a string of bits, determined by Table 7.1.)

- Bob then sends c_2 and c_3 along with all other c_i's to Alice.

[3] Decryption: To decrypt Bob's message, Alice evaluates the Legendre symbols $\left(\frac{c_i}{p}\right)$ and $\left(\frac{c_i}{q}\right)$. Since Alice knows the prime factorization (p, q) of n, it should be easy for her to evaluate these Legendre symbols. For example, for c_2 and c_3, Alice first evaluates the Legendre symbols $\left(\frac{c_i}{p}\right)$:

$$\begin{aligned} e_2' &= \left(\frac{c_2}{p}\right) = \left(\frac{16}{3}\right) = \left(\frac{4^2}{3}\right) = 1, \\ e_3' &= \left(\frac{c_3}{p}\right) = \left(\frac{5}{3}\right) = \left(\frac{2}{3}\right) = -1. \end{aligned}$$

then she gets

$$\begin{aligned} m_2 &= 0, \quad \text{since } e_2' = 1, \\ m_3 &= 1, \quad \text{since } e_3' = -1. \end{aligned}$$

Remark 7.5 The scheme introduced above is a good extension of the public-key idea, but encrypts messages bit by bit. It is completely secure with respect to semantic security as well

as bit security.[1] However, a major disadvantage of the scheme is the message expansion by a factor of $\log n$-bit. To improve the efficiency of the scheme, Blum and Goldwasser proposed in 1984 another randomized encryption scheme, in which the ciphertext is only longer than the plaintext by a constant number of bits; this scheme is comparable to the RSA scheme, both in terms of speed and message expansion.

Several other cryptographic schemes, including digital signature schemes and authentication encryption schemes, are based on the Quadratic Residuosity Problem (QRP).

Problems for Section 7.6

1. RSA encryption scheme is deterministic and not semantically secure, but it can be made semantically secure by adding randomness to the encryption process. Develop an RSA based probabilistic (randomized) encryption scheme that is semantically secure.
2. The number a is a quadratic residue modulo n if $\gcd(a, n) = 1$ and there is a solution to the congruence $x^2 \equiv a \pmod{n}$. If p is an odd prime, then

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}$$

   where $\left(\frac{a}{p}\right)$ is the Legendre symbol. Based on the above facts, design an efficient algorithm that takes $\mathcal{O}((\log p)^3)$-bit operations to decide if a is a quadratic residue modulo p.
3. Show that the Jacobi symbol can be computed in $\mathcal{O}((\log p)^2)$ bit operations using Gauss' Quadratic Reciprocity Law.
4. Show that if the integer factorization problem can be solved in polynomial-time, then the quadratic residuosity problem can be solved in polynomial-time.
5. Show that QRP is computationally equivalent to the IFP.
6. Use the binary-letter table in Exercise 7.9 to code the plaintext "HELP ME TO GET OUT OF THE DARK PLACE", then follow the steps in Algorithm 7.4 to encrypt the plaintext and decrypt the corresponding ciphertext. You need of course to choose the suitable values for p, q, and y.

7.5 Zero-Knowledge Proof

Zero-knowledge proof, originally studied by Goldwasser, Micali, and Rackoff [22] is a technique by which one can convince someone else that one has a certain knowledge (e.g., the two prime factors of n) without revealing any information about that knowledge (e.g., the prime factorization of n). To get a better understanding of the zero-knowledge technique, let us look at an example of a zero-knowledge proof based on the Square Root Problem (SQRTP). Recall that finding square roots modulo n is hard and equivalent to factoring.

[1] Bit security is a special case of semantic security. Informally, bit security is concerned not only that the whole message is not recoverable but also that individual bits of the message are not recoverable. The main drawback of the scheme is that the encrypted message is much longer than its original plaintext.

Algorithm 7.5 (Zero-knowledge proof) Let $n = pq$ be product of two large prime numbers. Let also $y \equiv x^2 \pmod{n}$ with $\gcd(y, n) = 1$. Now suppose that Alice claims to know x, the square root of y, she does not want to reveal the x. Now Bob wants to verify this.

[1] Alice first chooses two random numbers r_1 and r_2 with

$$r_1 r_2 \equiv x \pmod{n}. \tag{7.58}$$

(She can do so by first choosing r_1 with $\gcd(r_1, n) = 1$ and then letting $r_2 \equiv x r_1^{-1} \pmod{n}$). She then computes

$$x_1 \equiv r_1^2, \quad x_2 \equiv r_2^2 \pmod{n} \tag{7.59}$$

and sends x_1 and x_2 to Bob.

[2] Bob checks that $x_1 x_2 \equiv y \pmod{n}$, then chooses either x_1 or x_2 and asks Alice to supply a square root of it. He then checks that it is indeed a square root.

Example 7.10 Let $n = pq = 31 \cdot 61 = 1891$. Let also $\sqrt{56} \equiv x \pmod{1891}$. Now suppose that Alice claims to know x, the square root of 56, but she does not want to reveal it. Bob then wants to prove that Alice really knows x.

[1] Alice chooses $r_1 = 71$ such that $\gcd(71, 1891) = 1$. Then she finds $r_2 \equiv x(1/r_1) \equiv 408(1/71) \equiv 1151 \pmod{1891}$ (because Alice knows $x = 408$).
[2] Alice computes $x_1 \equiv r_1^2 \equiv 71^2 \equiv 1259 \pmod{1891}$, $x_2 \equiv r_2^2 \equiv 1151^2 \equiv 1101 \pmod{1891}$.
[3] Alice sends x_1 and x_2 to Bob.
[4] Bob checks $x_1 x_2 \equiv 1259 \cdot 1101 \equiv 56 \pmod{1891}$.
[5] Bob chooses either $x_1 = 1259$ or $x_2 = 1101$ and asks Alice to provide a square root of either x_1 or x_2.
[6] On request from Bob, Alice sends either $r_1 = 71$ or $r_2 = 1151$ to Bob, since $\sqrt{1259} \equiv 71 \pmod{1891}$ and $\sqrt{1101} \equiv 1151 \pmod{1891}$, i.e., $1259 \equiv 71^2 \pmod{1891}$ and $1101 \equiv 1151^2 \pmod{1891}$.
[7] Bob is now convinced that Alice really knows x, or otherwise she could not tell the square root of x_1 or x_2.

Algorithm 7.6 (Zero-knowlege identification scheme) Let $n = pq$ be the product of two large prime numbers. Let also Alice have the secret numbers $s_1, s_2, \ldots, s_k$ and $v_i \equiv s_i^{-2} \pmod{n}$ with $\gcd(s_i, n) = 1$. The numbers v_i are sent to Bob. Bob tries to verify that Alice knows the numbers $s_1, s_2, \ldots, s_k$. Both Alice and Bob proceed as follows:

[1] Alice first chooses a random number r, computes

$$x \equiv r^2 \pmod{n} \tag{7.60}$$

and sends x to Bob.

[2] Bob chooses numbers $\{b_1, b_2, \ldots, b_k\} \in \{0, 1\}$. He sends these to Alice.
[3] Alice computes

$$y \equiv r s_1^{b_1} s_2^{b_2} \cdots s_k^{b_k} \pmod{n} \tag{7.61}$$

and sends these to Bob.
[4] Bob checks that

$$x \equiv y^2 v_1^{b_1} v_2^{b_2} \cdots v_k^{b_k} \pmod{n}. \tag{7.62}$$

[5] Repeat Steps [1] to [4] several times (e.g., 20–30 times), each time with a different r.

The zero-knowledge technique is ideally suited to identification of an owner A (who e.g., has a ID number) of a smart card by allowing A to convince a merchant Bob of knowledge S without revealing even a single bit of S. Theoretically, the zero-knowledge technique can be based on any computationally intractable problem such as the IFP, DLP, ECDLP, KRTP, and SQRTP. The following is just an example.

Example 7.11 In this identification scheme, we assume that there is a smart card owned by, for example, Alice, a card reader machine owned by, for example, a bank, and a third party, called the third trust party (TTP).

[1] The TTP first chooses $n = pq$, where p and q are two large primes and $p \equiv q \equiv 3 \pmod 4$, and computes the PIN number for Alice's smart card such that

$$\text{PIN} \equiv s^2 \pmod{n}. \tag{7.63}$$

(ID is the quadratic residues of both q and q.)
[2] The TTP computes the square root s of ID (he can do so because he knows the prime factorization of n), and stores s in a segment of memory of the smart card that is not accessible from the outside world. The TTP should also made n public, but keep p and q secret. By now the smart card has the information (PIN, n, s), and the card reader has the information n.
[3] The Smart Card or the card holder Alice makes the PIN number to the card reader:

$$\text{Card/Alice} \xrightarrow{\text{PIN}} \text{Card Reader}. \tag{7.64}$$

[4] Card/Alice generates a random r and computes $t \equiv r^2 \pmod{n}$, and sends t to Bob:

$$\text{Card/Alice} \xrightarrow{t} \text{Card Reader}. \tag{7.65}$$

[5] The Card Reader selects a random $e \in \{0, 1\}$ and sends to Alice:

$$\text{Card/Alice} \xrightarrow{e} \text{Card Reader}. \tag{7.66}$$

[6] Card/Alice computes

$$u \equiv r \cdot s^e \pmod{n} \tag{7.67}$$

and sends it to the Card Reader:

$$\text{Card/Alice} \xrightarrow{\quad u \quad} \text{Card Reader.} \tag{7.68}$$

[7] The Card Reader checks whether or not

$$u^2 \equiv t \cdot \text{PIN}^e \pmod{n}. \tag{7.69}$$

[8] Repeat Steps [4]–[7] for different r. If each time,

$$u^2 \equiv t \cdot \text{PIN}^e \pmod{n}, \tag{7.70}$$

then the card is indeed issued by the TTP. That is, the Card Reader has been convinced that the Card has stored s, the square root of PIN modulo n.

Problems for Section 7.7

1. Suppose Alice knows:

$$\begin{aligned} k_1 &\equiv \log_{x_1} y_1 \pmod{n} \\ k_2 &\equiv \log_{x_2} y_2 \pmod{n} \\ \alpha &= k_1 = k_2 \end{aligned}$$

where $n = pq$. Suppose now that Alice wishes to convince Bob that she knows (k_1, k_2, α) as above. Design a zero-knowledge protocol that will convince Bob of what Alice claims.

2. Suppose Alice knows:

$$M \equiv C^d \pmod{n}$$

where

$$\begin{aligned} C &\equiv M^e \pmod{n} \\ ed &\equiv 1 \pmod{\phi(n)}) \\ n &= pq \\ (n, e, C) &\quad \text{public} \end{aligned}$$

Suppose now that Alice wishes to convince Bob that she knows M. Design a zero-knowledge protocol that Bob should be convinced that Alice knows M.

3. Let $n = pq$ with p, q primes. Given y, Alice wants to convince Bob that she knows x such that $x^2 \equiv y \pmod{n}$. Design a zero-knowledge protocol that will enable Bob to believe that Alice indeed knows x.
4. Design a zero-knowledge proof system based on the DLP problem.
5. Develop a zero-knowledge proof system based on the ECDLP problem.

7.6 Bibliographic Notes and Further Reading

This chapter discussed some of the most important and widely used public-key cryptographic systems, including the RSA, Rabin, probabilistic cryptography, and the zero-knowledge proof protocol, whose security relies on the infeasibility of the Integer Factorization Problem (IFP). For more information on IFP-based cryptography, readers are advised to consult: [23–58].

References

1. R. L. Rivest, A. Shamir, and L. Adleman, *On Digital Signatures and Public Key Cryptosystems*, Technical Memo 82, Laboratory for Computer Science, Massachusetts Institute of Technology, April 1977.
2. R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM*, **21**, 2, 1978, pp. 120–126.
3. H. Woll, "Reductions Among Number Theoretic Problems", *Information and Computation*, **72**, 1987, pp. 167–179.
4. R. G. E. Pinch, *Mathematics for Cryptography*, Queen's College, University of Cambridge, 1997.
5. M. Gardner, "Mathematical Games – A New Kind of Cipher that Would Take Millions of Years to Break", *Scientific American*, **237**, 2, 1977, pp. 120–124.
6. D. Atkins, M. Graff, A. K. Lenstra, and P. C. Leyland, "The Magic Words are Squeamish Ossifrage", *Advances in Cryptology – ASIACRYPT'94*, Lecture Notes in Computer Science **917**, 1995, pp. 261–277.
7. R. P. Brent, "Primality Testing and Integer Factorization", *Proceedings of Australian Academy of Science Annual General Meeting Symposium on the Role of Mathematics in Science*, Canberra, 1991, pp. 14–26.
8. W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, 2nd Edition, Prentice-Hall, 2006.
9. M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology EUROCRYPT'94*, Lecture Notes in Computer Science **950**, Springer, 1995, pp. 92–111.
10. R. L. Rivest and B. Kaliski, RSA Problem, *Encyclopedia of Cryptography and Security*. Edited by H. C. A. van Tilborg, Springer, 2005.
11. D. Boneh, "Twenty Years of Attacks on the RSA Cryptosystem", *Notices of the AMS*, **46** 2, 1999, pp. 203–213.
12. J. Rothe, *Complexity Theory and Cryptography*, Springer, 2005.
13. J. S. Coron and A. May, "Deterministic Polynomial-Time Equivalence of Computing the RSA Secret Key and Factoring", *Journal of Cryptology*, **20**, 1, 2007, pp. 39–50.
14. D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerability", *Journal of Cryptology*, **10**, 1997, pp. 233–260.
15. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring Polynomials with Rational Coefficients", *Mathematische Annalen*, **261**, 1982, pp. 515–534.
16. G. Miller, "Riemann's Hypothesis and Tests for Primality", *Journal of Systems and Computer Science*, **13**, 1976, pp. 300–317.
17. H. Wiener, "Cryptanalysis of Short RSA Secret Exponents", *IEEE Transactions on Information Theory*, **36**, 3, 1990, pp. 553–558.
18. M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", *Technical Report MIT/LCS/TR-212*, MIT Laboratory for Computer Science, 1979.
19. H. C. Williams, "A Modification of the RSA Public-Key Encryption Procedure", *IEEE Transactions on Information Theory*, **26**, 1980, pp. 726–729.

20. H. C. Williams, "An M^3 Public-Key Encryption Scheme", *Advances in Cryptology - CRYPTO '85*, Lecture Notes in Computer Science **218**, Springer, 1986, pp. 358–368.
21. S. Goldwasser and S. Micali, "Probabilistic Encryption", *Journal of Computer and System Sciences*, **28**, 1984, pp. 270–299.
22. S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof System", *SIAM Journal on Computing*, **18**, 1989, pp. 186–208.
23. J. A. Buchmann, *Introduction to Cryptography*, 2nd Edition, Springer, 2004.
24. T. H. Cormen, C. E. Ceiserson, and R. L. Rivest, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009.
25. T. W. Cusick, D. Ding, and A. Renvall, *Stream Cipher and Number Theory*, North-Holland, 1998.
26. H. Delfs and H. Knebl, *Introduction to Cryptography*, Springer, 2002.
27. N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering*, Wiley, 2005.
28. P. Garrett, *Making, Breaking Codes: An Introduction to Cryptology*, Prentice-Hall, 2001.
29. O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001.
30. O. Goldreich, *Foundations of Cryptography: Basic Applications*, Cambridge University Press, 2004.
31. M. J. Hinek, *Cryptanalysis of RSA and its Variants*, Chapman & Hall/CRC Press, 2009.
32. J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer-Verlag, 2008.
33. S. Katzenbeisser, "Recent Advances in RSA Cryptography", Kluwer Academic Publishers, 2001.
34. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer-Verlag, 1994.
35. N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics **3**, Springer, 1998.
36. N. Koblitz, "A Survey of Number Theory and Cryptography", *Number Theory*. Edited by P. Bambah, V. C. Dumir, and R. J. Hans-Gill, Birkhäser, 2000, pp. 217–239.
37. N. Koblitz, "Cryptography", *Mathematics Unlimited – 2001 and Beyond*. Edited by B. Enguist and W. Schmid, Springer, 2001, pp. 749–769.
38. N. Koblitz and A. J. Menezes, "A Survey of Public-Key Cryptosystems", *SIAM Review*, **46**, 4, 2004, pp. 599–634.
39. A. G. Konheim, *Computer Security and Cryptography*, Wiley, 2007.
40. D. E. Knuth, *The Art of Computer Programming II – Seminumerical Algorithms*, 3rd Edition, Addison-Wesley, 1998.
41. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptosystems*, CRC Press, 1996.
42. R. A. Mollin, *RSA and Public-Key Cryptography*, Chapman & Hall/CRC Press, 2003.
43. R. A. Mollin, *Codes: The Guide to Secrecy from ancient to Modern Times*, Chapman & Hall/CRC Press, 2005.
44. R. A. Mollin, *Introduction to Cryptography*, 2nd Edition, CRC Press, 2006.
45. J. Pieprzyk, T. Hardjono, and J. Seberry, *Fundamentals of Computer Security*, Springer, 2003.
46. B. Schneier, *Applied Cryptography – Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons, 1996.
47. S. Singh, *The Code Book – The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, Fourth Estate, London, 1999.
48. S. Singh, *The Science of Secrecy – The History of Codes and Codebreaking*, Fourth Estate, London, 2000.
49. N. Smart, *Cryptography: An Introduction*, McGraw-Hill, 2003.
50. M. Stamp and R. M. Low, *Applied Cryptanalysis*, Wiley, 2007.
51. D. R. Stinson, *Cryptography: Theory and Practice*, 2nd Edition, Chapman & Hall/CRC Press, 2002.
52. J. C. A. van der Lubbe, *Basic Methods of Cryptography*, Cambridge University Press, 1998.
53. H. C. A. van Tilborg, *Fundamentals of Cryptography*, Kluwer Academic Publishers, 1999.
54. O. N. Vasilenko, *Number-Theoretic Algorithms in Cryptography*, American Mathematical Society, 2006.
55. S. S. Wagstaff, Jr., *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC Press, 2002.
56. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer-Verlag, 2002.
57. S. Y. Yan, *Cryptanalyic Attacks on RSA*, Springer, 2009.
58. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.

8

Discrete Logarithm Based Cryptography

In this chapter, we shall introduce some of the well-known and widely used public-key cryptographic systems and protocols whose security relies on the infeasibility of the DLP; these include:

- The Diffie–Hellman–Merkle key-exchange protocol
- The ElGamal cryptographic system
- The Massey–Omura cryptographic system
- The US government's Digital Signature Standard.

8.1 Diffie–Hellman–Merkle Key-Exchange Protocol

In 1976 Diffie and Hellman [1] proposed for the first time the concept and idea of public-key cryptography, and the first public-key system based on the infeasible Discrete Logarithm Problem (DLP). Their system is not a public-key cryptographic system, but a public-key distribution system based on Merkle's seminal work in 1978 [2] (Figure 8.1 shows the DHM crypto years in the 1970s). Such a public-key distribution scheme does not send secret messages directly, but rather allows the two parties to agree on a common private key over public networks to be used later in exchanging messages through conventional secret-key cryptography. Thus, the Diffie–Hellman–Merkle scheme has the nice property that a very fast encryption scheme such as DES or AES can be used for actual encryption (just using the agreed key), yet it still enjoys one of the main advantages of public-key cryptography. The Diffie–Hellman–Merkle key-exchange protocol works in the following way (see also Figure 8.2):

[1] A prime q and a generator g are made public (assume all users have agreed upon a finite group over a fixed finite field $\mathbb{F}_q$),
[2] Alice chooses a random number $a \in \{1, 2, \ldots, q-1\}$ and sends $g^a \bmod q$ to Bob,
[3] Bob chooses a random number $b \in \{1, 2, \ldots, q-1\}$ and sends $g^b \bmod q$ to Alice,
[4] Alice and Bob both compute $g^{ab} \bmod q$ and use this as a private key for future communications.

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

Figure 8.1 Merkle, Hellman and Diffie (Courtesy of Prof Hellman)

Clearly, an eavesdropper has g, q, $g^a \bmod q$, and $g^b \bmod q$, so if he can take discrete logarithms, he can calculate $g^{ab} \bmod q$ and understand the communications. That is, if the eavesdropper can use his knowledge of g, q, $g^a \bmod q$, and $g^b \bmod q$ to recover the integer a, then he can easily break the Diffie–Hellman–Merkle system. So, the security of the Diffie–Hellman–Merkle system is based on the following assumption:

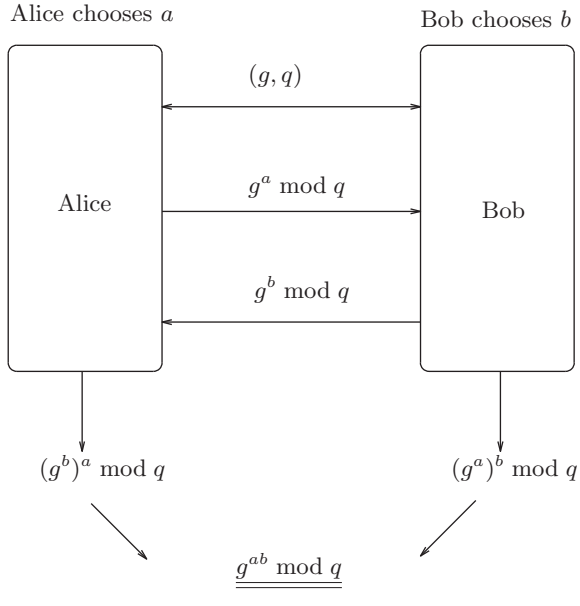


Figure 8.2 DHM key-exchange protocol

Diffie–Hellman–Merkle assumption : It is computationally infeasible to compute $g^{ab} \bmod q$ from $g, q, g^a \bmod q$ and $g^b \bmod q$. That is,

$$\{g, q, g^a \bmod q, g^b \bmod q\} \xrightarrow{\text{hard to find}} \{g^{ab} \bmod q\}.$$

The Diffie–Hellman–Merkle assumption, in turn, depends on the following Discrete Logarithm Problem assumption, that is,

$$\{g, q, g^a \bmod q\} \xrightarrow{\text{hard to find}} \{a\},$$

or

$$\{g, q, g^b \bmod q\} \xrightarrow{\text{hard to find}} \{b\}.$$

In theory, there could be a way to use knowledge of $g^a \bmod q$ and $g^b \bmod q$ to find $g^{ab} \bmod q$. But at present, we simply cannot imagine a way to go from $g^a \bmod q$ and $g^b \bmod q$ to $g^{ab} \bmod q$ without essentially solving the following Discrete Logarithm Problem:

$$\{g, q, g^a \bmod q\} \xrightarrow{\text{find}} \{a\},$$

or

$$\{g, q, g^b \bmod q\} \xrightarrow{\text{find}} \{b\}.$$

If either a or b can be found efficiently, then DHM can be broken easily, since

$$\{g, q, b, g^a \bmod q\} \xrightarrow{\text{easy to find}} \{(g^a)^b \equiv g^{ab} \pmod q\},$$

or

$$\{g, q, a, g^b \bmod q\} \xrightarrow{\text{easy to find}} \{(g^b)^a \equiv g^{ab} \pmod q\}.$$

Example 8.1 The following DHM challenge problem was proposed in [3].

[1] Let p be following prime number:

$$p = 2047062703855328380597445351669742748036083943401234595\_69579867459152659137268522951065284733970579762207550 5\_06983104348665168227 9.$$

[2] Alice chooses a random number a modulo p, computes $7^a \pmod p$, and sends the result to Bob, keeping a secret.

[3] Bob receives

$$7^a \equiv 127402180119973946824269244334322849749382042586931621654557735290322914679095998681860978813046595166455458144280588076766033781 \pmod{p}.$$

[4] Bob chooses a random number residue b modulo p, computes $7^b \pmod{p}$, and sends the result to Alice, keeping b secret.

[5] Alice receives

$$7^b \equiv 180162285287453102444782834836799895015967046695346697313025121734059953772058475958176910625380692101651848662362137934026803049 \pmod{p}.$$

[6] Now both Alice and Bob can compute the private key $7^{ab} \pmod{p}$.

McCurley offered a prize of $100 in 1989 to the first person or group to find the private key constructed from the above communication.

Example 8.2 McCurley's 129-digit discrete logarithm challenge was actually solved on January 25 1998 using the NFS method, by two German computer scientists, Weber at the Institut für Techno-und Wirtschaftsmathematik in Kaiserslautern and Denny at the Debis IT Security Services in Bonn [4]. Their solution to McCurley's DLP problem is as follows.

$$a \equiv 3812728041119001413807839150792963419399864355101867028556137516504552396692940392210217251405327092887266394263700635327977408 \pmod{p},$$

$$(7^b)^a \equiv 618586908596518832735933316652037904267987643069521713459146222184952599815614487782075749218290977740833879185045794674973 4.$$

As we have already mentioned earlier, the Diffie–Hellman–Merkle scheme is not intended to be used for actual secure communications, but for key-exchanges. There are, however, several other cryptosystems based on discrete logarithms that can be used for secure message transmissions.

Problems for Section 8.1

1. Suppose Alice and Bob decide to use DHM to establish a secret key for later secure communications. They first agree that

   $g = 64783087731,$
   $p = 80000000000000001239.$

   Then each of them chooses at random a secret random number

   $a = 476388475629$ by Alice,
   $b = 2243552788$ by Bob.

You are asked to compute

$$\begin{aligned} x &= g^a \pmod{p};\\ k_a &\equiv x^b \pmod{p},\\ x &= g^b \pmod{p},\\ k_b &\equiv y^a \pmod{p}, \end{aligned}$$

and then to check if

$$k_a \equiv k_b \pmod{p}.$$

2. In McCurley's DLP, we have

$$\begin{aligned} 7^b \equiv\ & 180162285287453102444782834836799895015967046695346697313025121734059953772058475958176910625380692101651848662362137934026803049 \pmod{p},\\ p =\ & 204706270385532838059744535166974274803608394340123459695798674591526591372685229510652847339705797622075505069831043486651682279. \end{aligned}$$

(1) Find the discrete logarithm b.
(2) Compute $(7^a)^b \bmod p$.
(3) Verify if your result $(7^a)^b \bmod p$ agrees with Weber and Denny's result, that is, check if $(7^a)^b \equiv (7^b)^a \pmod{p}$.

3. Let the DHM parameters be as follows:

$$\begin{aligned} p =\ & 100000000000000000000000000000000000000000000000000000000000000000000000000000204706270385532838059744535166974274803608394340123459695798674591526591372685229510652847339705797622075505069831043486651682889,\\ 13^x \equiv\ & 108519459267489303215368977875116015362914115512159637357974137547050028457782437666667887267761228059356952326614812573203747209862136106492028547633310541581302441198573774157137087441635299151446262 \pmod{p},\\ 13^y \equiv\ & 522002084001565230804843872480767603621983222550170142672568737458667077499227771880919869778498287278358483829459489565477648733256999972723227753686571233058307476978004178550365511987192742641223712 \pmod{p}. \end{aligned}$$

(1) Find the discrete logarithm x.
(2) Find the discrete logarithm y.
(3) Compute $(13^x)^y \pmod{p}$.
(4) Compute $(13^y)^x \pmod{p}$.

4. The man-in-the-middle attack is one of the attacks on DHM. Explain
   (1) What is the man-in-the-middle attack?
   (2) How does the man-in-the-middle attack work on DHM? (Use diagrams and mathematical formulas whenever possible.)
   (3) How can the man-in-the-middle attack on DHM can be prevented and detected?

8.2 ElGamal Cryptography

In 1985, ElGamal [5], then a PhD student of Hellman at Stanford, proposed the first DLP-based public-key cryptosystem, since the plaintext M can be recovered by taking the following discrete logarithms

$$M \equiv \log_{M^e} M \pmod{n}.$$

The ElGamal cryptosystem can be described as follows (see also Figure 8.3).

[1] A prime q and a generator $g \in \mathbb{F}_q^*$ are made public.
[2] Alice chooses at random a private integer

$$a \in \{1, 2, \ldots, q-1\}. \tag{8.1}$$

This a is the private decryption key. The public encryption key is $\{g, q, g^a \bmod q\}$.

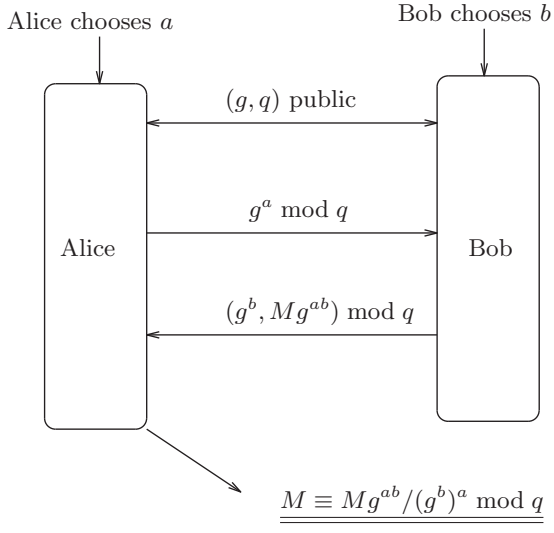


Figure 8.3 ElGamal Cryptography

[3] Suppose now Bob wishes to send a message to Alice. He chooses a random number $b \in \{1, 2, \ldots, q-1\}$ and sends Alice the following pair of elements of $\mathbb{F}_q$:

$$(g^b,\ Mg^{ab})$$

where M is the message.

[4] Since Alice knows the private decryption key a, she can recover M from this pair by computing $g^{ab} \pmod q$ and dividing this result into the second element. That is,

$$M \equiv Mg^{ab}/(g^b)^a \pmod q.$$

[5] Cryptanalysis: Find the private a by solving the DLP

$$a \equiv \log_g x \pmod{q-1}$$

such that

$$x \equiv g^a \pmod q.$$

Remark 8.1 Anyone who can solve the discrete logarithm problem in $\mathbb{F}_q$ breaks the cryptosystem by finding the secret decryption key a from the public encryption key g^a. In theory, there could be a way to use knowledge of g^a and g^b to find g^{ab} and hence break the cipher without solving the discrete logarithm problem. But as we have already seen in the Diffie–Hellman–Merkle scheme, there is no known way to go from g^a and g^b to g^{ab} without essentially solving the discrete logarithm problem. So, the ElGamal cryptosystem is equivalent to the Diffie–Hellman–Merkle key-exchange system.

Problems for Section 8.2

1. In ElGamal cryptosystem, Alice makes (p, g, g^a) public with p prime p:

$$\begin{aligned} p = {} & 10000000000000000000000000000000000000000000000000\_ \\ & 00000000000000000000204706270385532838059744535166974 2\_ \\ & 74804608394340123459695798674591526591372685229510652\_ \\ & 84733970579762207550506983104348665168328 1 \end{aligned}$$

$$g = 137$$

$$\begin{aligned} g^a \equiv {} & 15219266397668101959283316151426320683674451858111063\_ \\ & 45767690506157955692567935509944285656491006943855496\_ \\ & 14388735928661950422196794512676225936419253780225375\_ \\ & 37252639984353500071774531090027331523676 \end{aligned}$$

where $a \in \{1, 2, \cdots, p\}$ must be kept as a secret. Now Bob can send Alice an encrypted message $C = (g^b, Mg^{ab})$ to Alice by using her public-key information, where

$$\begin{aligned} g^b \equiv\ & 59547675601458322302365604133720220696052746940473 3\_ \\ & 55046049744137914374142183634043230653659070816467 4\_ \\ & 62466636904384382001528769925211730081006654249356 4\_ \\ & 12826389882146691842217779072611842406374051259 \\ Mg^{ab} \equiv\ & 49587861882815113830430418447664907530237264453603 2\_ \\ & 94479849527736721533557707864314686330644624599660 5\_ \\ & 60087834147651129038106201491085560126484952668340 8\_ \\ & 83323263742065525535496981642865216817002959760 \end{aligned}$$

(1) Find the discrete logarithm a, and compute $(g^b)^a \bmod p$.
(2) Find the discrete logarithm b, and compute $(g^a)^b \bmod p$.
(3) Decode the ciphertext C by computing either

$$M \equiv Mg^{ab}/(g^b)^a \pmod{p}$$

or

$$M \equiv Mg^{ab}/(g^a)^b \pmod{p}.$$

2. The ElGamal encryption can be randomized by the random choice of b or a. Describe a randomized version of the ElGamal cryptosystem.
3. The ElGamal can be implemented in any cyclic group. Describe a version of ElGamal cryptosystem in a class group of binary quadratic forms or more generally in a class group of algebraic number fields.
4. Suppose, in ElGamal cryptosystem, the random number b is revealed. Explain how the private key can be determined.
5. Compared to RSA, what is the disadvantage of ElGamal.

8.3 Massey–Omura Cryptography

The Massey–Omura cryptosystem is another popular public-key cryptosystem based on discrete logarithms over the finite field $\mathbb{F}_q$, with $p = p^r$ prime power. It was proposed by James Massey and Jim K. Omura in 1982 [6] as a possible improvement over Shamir's original three-pass cryptographic protocol was developed around 1980, in which the sender and the receiver do not exchange any keys, however, the protocol does require the sender and receiver to have two private keys for encrypting and decrypting messages. Thus, the Massey–Omura cryptosystem works in the following steps (see Figure 8.4):

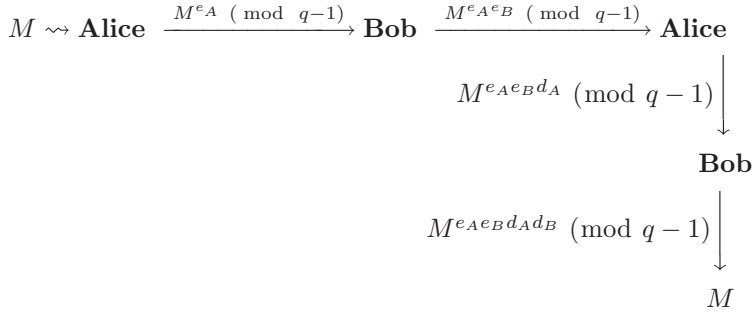


Figure 8.4 The Massey-Omura Cryptography

[1] All the users have agreed upon a finite group over a fixed finite field $\mathbb{F}_q$ with q a prime power.
[2] Each user secretly selects a random integer e between 0 and $q-1$ such that $\gcd(e, q-1) = 1$, and computes $d = e^{-1} \bmod (q-1)$ by using the extended Euclidean algorithm. At the end of this step, Alice gets (e_A, d_A) and Bob gets (e_B, d_B).
[3] Now suppose that user Alice wishes to send a secure message M to user Bob, then they follow the following procedure:
  [a] Alice first sends M^{e_A} to Bob,
  [b] On receiving Alice's message, Bob sends $M^{e_A e_B}$ back to Alice (note that at this point, Bob cannot read Alice's message M),
  [c] Alice sends $M^{e_A e_B d_A} = M^{e_B}$ to Bob,
  [d] Bob then computes $M^{d_B e_B} = M$, and hence recovers Alice's original message M.
[4] Cryptanalysis: Eve shall be hard to find M from the three-pass protocol between Alice and Bob unless she can solve the Discrete Logarithm Problem involved efficiently.

The Massey–Omura cryptosystem is also be described in detail in Figure 8.5 (suppose Alice wants to send Bob a secret message M, with Eve, the attacker in the middle).

Example 8.3 Let

$$\begin{aligned} p &= 80000000000000001239, \\ M &= 20210519040125 \ \text{(Tuesday)}, \\ e_A &= 6654873997, \\ e_B &= 7658494001. \end{aligned}$$

Then

$$\begin{aligned} d_A &\equiv \frac{1}{e_A} \equiv 70094446778448900393 \pmod{p-1}, \\ d_B &\equiv \frac{1}{e_B} \equiv 14252518250422012923 \pmod{p-1}, \\ M^{e_A} &\equiv 56964332403383118724 \pmod{p}, \end{aligned}$$

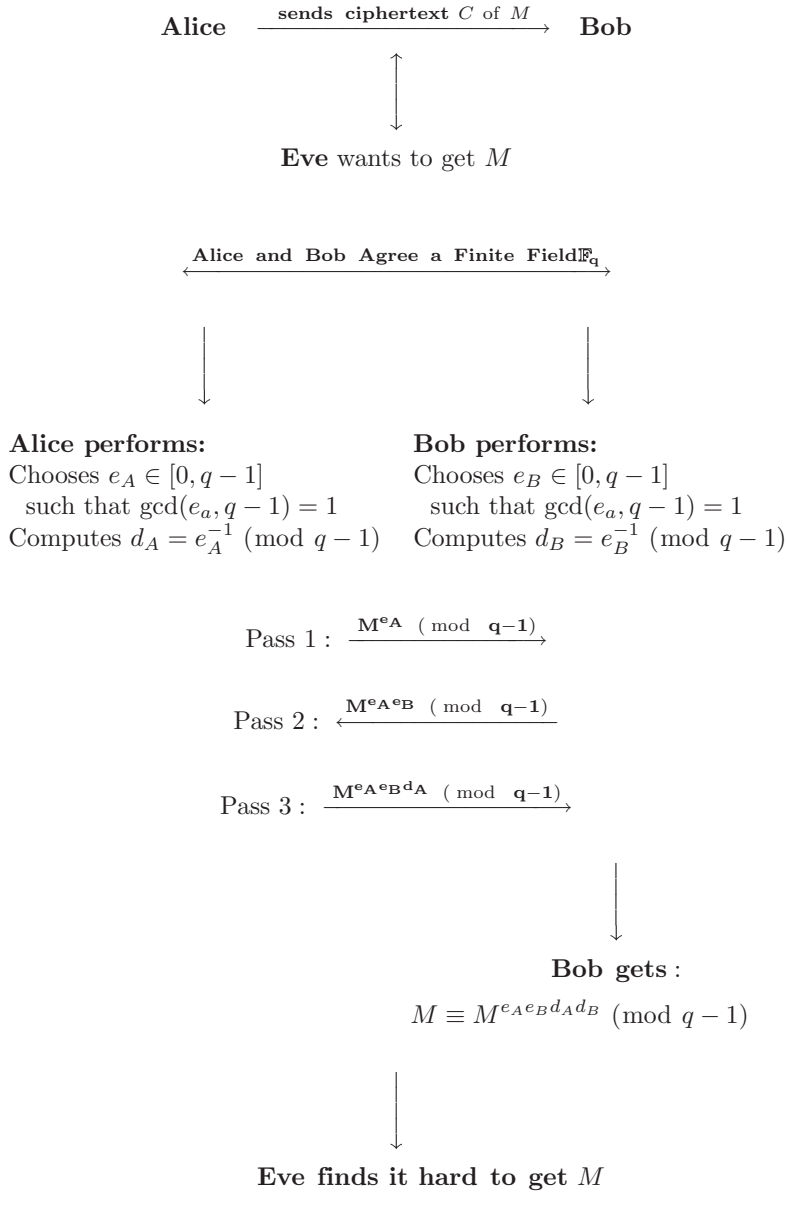


Figure 8.5 The Massey–Omura three-pass cryptographic protocol

$$M^{e_A e_B} \equiv 37671804887541585024 \bmod p,$$
$$M^{e_A e_B d_A} \equiv 50551151743565447865 \bmod p,$$
$$M^{e_A e_B d_A d_B} \equiv 20210519040125 \pmod{p},$$
$$\downarrow$$
$$M$$

Problems for Section 8.3

1. Is the Massey–Omura cryptosystem a public-key cryptosystem? What are the public key and secret key, respectively, in the Massey–Omura cryptosystem?
2. Explain why the security of Massey–Omura cryptosystem depends on the infeasibility of the Discrete Logarithm Problem.
3. Design a discrete logarithm attack on the Massey–Omura cryptosystem.
4. Let

$$\begin{aligned} &p = 14197, \\ &(e_A, d_A) = (13, 13105), \\ &(e_B, d_B) = (17, 6681), \\ &M = 1511 \quad \text{(OK)}, \end{aligned}$$

   Find

$$\begin{aligned} &M^{e_A} \bmod p, \\ &M^{e_A e_B} \bmod p, \\ &M^{e_A e_B d_A} \bmod p, \\ &M^{e_A e_B d_A d_B} \bmod p. \end{aligned}$$

   and check if $M \equiv M^{e_A e_B d_A d_B} \pmod{p}$.
5. Let

$$\begin{aligned} &p = 20000000000000002559, \\ &M = 201514042625151811 \quad \text{(To New York)}, \\ &e_A = 6654873997, \\ &e_B = 7658494001. \end{aligned}$$

   (1) Find

$$\begin{aligned} d_A &\equiv 1/e_A \pmod{p-1}, \\ d_B &\equiv 1/e_B \pmod{p-1}. \end{aligned}$$

   (2) Find

$$\begin{aligned} &M^{e_A} \bmod p, \\ &M^{e_A e_B} \bmod p, \\ &M^{e_A e_B d_A} \bmod p, \\ &M^{e_A e_B d_A d_B} \bmod p. \end{aligned}$$

   (3) Check if $M \equiv M^{e_A e_B d_A d_B} \pmod{p}$.

8.4 DLP-Based Digital Signatures

In this section, we shall introduce a very influential signature scheme based on ElGamal's cryptosystem; the security of such a signature scheme depends on the intractability of discrete logarithms over a finite field.

Algorithm 8.1 (ElGamal signature scheme) This algorithm tries to generate a digital signature $S = (a, b)$ for message m. Suppose that Alice wishes to send a signed message to Bob.

[1] [ElGamal key generation] Alice does the following:
  [1-1] Choose a prime p and two random integers g and x, such that both g and x are less than p.
  [1-2] Compute $y \equiv g^x \pmod{p}$.
  [1-3] Make (y, g, p) public (both g and p can be shared among a group of users), but keep x secret.

[2] [ElGamal signature generation] Alice does the following:
  [2-1] Choose at random an integer k such that $\gcd(k, p-1) = 1$.
  [2-2] Compute

$$\left.\begin{aligned} a &\equiv g^k \pmod{p}, \\ b &\equiv k^{-1}(m - xa) \pmod{(p-1)}. \end{aligned}\right\} \tag{8.2}$$

  Now Alice has generated the signature (a, b). She must keep the random integer, k, secret.

[3] [ElGamal signature verification] To verify Alice's signature, Bob confirms that

$$y^a a^b \equiv g^m \pmod{p}. \tag{8.3}$$

In August 1991, the US Government's National Institute of Standards and Technology (NIST) proposed an algorithm for digital signatures. The algorithm is known as DSA, for digital signature algorithm. The DSA has become the US Federal Information Processing Standard 186 (FIPS 186). It is called the Digital Signature Standard (DSS) [13], and is the first digital signature scheme recognized by any government. The role of DSA/DSS is expected to be analogous to that of the Data Encryption Standard (DES). The DSA/DSS is similar to a signature scheme proposed by Schnorr; it is also similar to a signature scheme of ElGamal. The DSA is intended for use in electronic mail, electronic funds transfer, electronic data interchange, software distribution, data storage, and other applications which require data integrity assurance and data authentication. The DSA/DSS consists of two main processes:

(1) Signature generation (using the private key).
(2) Signature verification (using the public key).

A one-way hash function is used in the signature generation process to obtain a condensed version of data, called a message digest. The message digest is then signed. The digital signature is sent to the intended receiver along with the signed data (often called the message). The receiver of the message and the signature verifies the signature by using the sender's public key. The same hash function must also be used in the verification process. In what follows, we shall give the formal specifications of the DSA/DSS.

Algorithm 8.2 (Digital signature algorithm, DSA) This is a variation of the ElGamal signature scheme. It generates a signature $S = (r, s)$ for the message m.

[1] [DSA key generation] To generate the DSA key, the sender performs the following:
  [1-1] Find a 512-bit prime p (which will be public).
  [1-2] Find a 160-bit prime q dividing evenly into $p - 1$ (which will be public).
  [1-3] Generate an element $g \in \mathbb{Z}/p\mathbb{Z}$ whose multiplicative order is q, that is, $g^q \equiv 1 \pmod{p}$.
  [1-4] Find a one-way function H mapping messages into 160-bit values.
  [1-5] Choose a secret key x, with $0 < x < q$.
  [1-6] Choose a public key y, where $y \equiv g^x \pmod{p}$. Clearly, the secret x is the discrete logarithm of y, modulo p, to the base g.

[2] [DSA signature generation] To sign the message m, the sender produces his signature as (r, s), by selecting a random integer $k \in \mathbb{Z}/q\mathbb{Z}$ and computing

$$\left.\begin{aligned} r &\equiv \left(g^k \pmod{p}\right) \pmod{q}, \\ s &\equiv k^{-1}(H(m) + xr) \pmod{q}. \end{aligned}\right\} \tag{8.4}$$

[3] [DSA signature verification] To verify the signature (r, s) for the message m from the sender, the receiver first computes:

$$t \equiv s^{-1} \pmod{q}, \tag{8.5}$$

and then accepts the signature as valid if the following congruence holds:

$$r \equiv \left(g^{H(m)t} y^{rt} \pmod{p}\right) \pmod{q}. \tag{8.6}$$

If the congruence (8.6) does not hold, then the message either may have been incorrectly signed, or may have been signed by an impostor. In this case, the message is considered to be invalid.

There are, however, many responses solicited by the (US) Association of Computing Machinery (ACM), positive and negative, to the NIST's DSA. Some positive aspects of the DSA include:

(1) The US government has finally recognized the utility and the usefulness of public-key cryptography. In fact, the DSA is the only signature algorithm that has been publicly proposed by any government.

(2) The DSA is based on reasonably familiar number-theoretic concepts, and it is especially useful to the financial services industry.
(3) Signatures in DSA are relatively short (only 320-bits), and the key generation process can be performed very efficiently.
(4) When signing, the computation of r can be done even before the message m is available, in a "precomputation" step.

Whilst some negative aspects of the DSA include:

(1) The DSA does not include key exchanges, and cannot be used for key distribution and encryption.
(2) The key size in DSA is too short; it is restricted to a 512-bit modulus or key size, which is too short and should be increased to at least 1024-bits.
(3) The DSA is not compatible with existing international standards; for example, the international standards organizations such as ISO, CCITT, and SWIFT all have accepted the RSA as a standard.

Nevertheless, the DSA is the only publicly known government digital signature standard.

Problems for Section 8.4

1. According to [7], the following numbers 26, 12, 22, 58, 61 are the signature of a former Special Agent for the FBI in the 1960s. It has been encoded using the agent's private key, which is none of your business. Find the signature by using their RSA public key $(e, n) = (7, 77)$. (As usual, the alphabetic-numeric encoding is just $A \to 1$, $B \to 2, \ldots, Z \to 26$.)
2. Suppose, in the ElGamal cryptosystem, the random number k is chosen to sign two different messages. Let

$$b_1 \equiv k^{-1}(m_1 - xa) \pmod{(p-1)},$$
$$b_2 \equiv k^{-1}(m_2 - xa) \pmod{(p-1)}$$

where

$$a \equiv g^k \pmod{p}.$$

(1) Show that k can be computed from

$$(b_1 - b_2)k \equiv (m_1 - m_2) \pmod{(p-1)}.$$

(2) Show that the private key x can be determined from the knowledge of k.
3. There are many variations to the ElGamal signature scheme by modifying the equation

$$b \equiv k^{-1}(m - xa) \pmod{(p-1)}.$$

Propose three such variations.

4. The US government's digital signature algorithm (DSA) is a variant of the ElGamal cryptosystem based on the intractability of the DLP.
   (1) Give a complete description of DSA.
   (2) Write an essay on the cryptanalytic attacks on DSA.
5. Just the same as encryption, digital signatures can be based on DLP, and also can be based on IFP. Design variations of RSA and Rabin digital signature schemes.

8.5 Bibliographic Notes and Further Reading

This chapter discussed some of the most important and widely used public-key cryptographic systems and digital signatures, whose security relies on the infeasibility of the Discrete Logarithm Problem (DLP). The first public-key system, namely, the key-exchange scheme, was first proposed by Diffie and Hellman in 1976 in [1], based on an idea of Merkle's [2] (although published later). The first Discrete Logarithm Problem based cryptographic system and digital signature scheme were proposed by ElGamal in 1985 [5]. For general references on Discrete Logarithm Problem based cryptographic systems and digital signature systems, it is suggested that readers consult [8–42].

References

1. W. Diffie and E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, **22**, 5, 1976, pp. 644–654.
2. R. C. Merkle, "Secure Communications over Insecure Channels" *Communications of the ACM*, **21**, 1978, pp. 294–299. (Submitted in 1975.)
3. K. S. McCurley, "The Discrete Logarithm Problem", *Cryptology and Computational Number Theory*. Edited by C. Pomerance, Proceedings of Symposia in Applied Mathematics **42**, American Mathematics Society, 1990, pp. 49–74.
4. D. Weber and T. F. Denny, "The Solution of McCurley's Discrete Log Challenge", *Advances in Cryptology - CRYPTO '98*, Lecture Notes in Computer Science **1462**, 1998, pp. 458–471.
5. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms", *IEEE Transactions on Information Theory*, **31**, 1985, pp. 496–472.
6. J. L. Massey and J.K. Omura, *Method and Apparatus for Maintainining the Privacy of Digital Message Conveyed by Public Transmission*, US Patent No 4677600, 28 Jan 1986.
7. R. E. Lewand, *Cryptological Mathematics*, The Mathematical Association of America, 2000.
8. L. M. Adleman, "A Subexponential Algorithmic for the Discrete Logarithm Problem with Applications to Cryptography", *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1979, pp. 55–60.
9. T. H. Barr, *Invitation to Cryptology*, Prentics-Hall, 2002.
10. F. L. Bauer, *Decrypted Secrets – Methods and Maxims of Cryptology*, 3rd Edition, Springer-Verlag, 2002.
11. D.Bishop, *Introduction to Cryptography with Java Applets*, Jones and Bartlett, 2003.
12. J. A. Buchmann, *Introduction to Cryptography*, 2nd Edition, Springer, 2004.
13. Communications of the ACM, "The Digital Signature Standard Proposed by NIST and Responses to NIST's Proposal", *Communications of the ACM*, **35**, 7, 1992, pp. 36–54.
14. T. H. Cormen, C. E. Ceiserson and R. L. Rivest, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009.
15. A. J. Elbirt, *Understanding and Applying Cryptography and Data Security*. CRC Press, 2009.
16. B. A. Forouzan, *Cryptography and Network Security*, McGraw-Hill, 2008.
17. J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer-Verlag, 2008.
18. J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. CRC Press, 2008.

19. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer-Verlag, 1994.
20. N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics **3**, Springer, 1998.
21. N. Koblitz, "A Survey of Number Theory and Cryptography", *Number Theory*. Edited by P. Bambah, V. C. Dumir, and R. J. Hans-Gill, Birkhäser, 2000, pp. 217–239.
22. N. Koblitz, "Cryptography", in: *Mathematics Unlimited – 2001 and Beyond*, Edited by B. Enguist and W. Schmid, Springer, 2001, pp. 749–769.
23. W. Mao, *Modern Cryptography*, Prentice-Hall, 2004.
24. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptosystems*, CRC Press, 1996.
25. R. A. Mollin, *An Introduction to Cryptography*, 2nd Edition, Chapman & Hall/CRC Press, 2006.
26. A. M. Odlyzko, "Discrete Logarithms in Finite Fields and their Cryptographic Significance", *Advances in Cryptography*, EUROCRYPT '84, Proceedings, Lecture Notes in Computer Science **209**, Springer, 1984, pp. 225–314.
27. S. C. Pohlig and M. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance", *IEEE Transactions on Information Theory*, **24**, 1978, pp. 106–110.
28. M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", *Technical Report MIT/LCS/TR-212*, MIT Laboratory for Computer Science, 1979.
29. J. Rothe, *Complexity Theory and Cryptography*, Springer, 2005.
30. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons, 1996.
31. S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, Fourth Estate, London, 1999.
32. S. Singh, *The Science of Secrecy – The History of Codes and Codebreaking*, Fourth Estate, London, 2000.
33. N. Smart, *Cryptography: An Introduction*, McGraw-Hill, 2003.
34. M. Stamp and R. M. Low, *Applied Cryptanalysis*, Wiley, 2007.
35. A. Stanoyevitch, *Introduction to Cryptography*, CRC Press, 2011.
36. D. R. Stinson, *Cryptography: Theory and Practice*, 3rd Edition, Chapman & Hall/CRC Press, 2006.
37. C.Swenson *Modern Cryptanalysis*, Wiley, 2008.
38. H. C. A. van Tilborg, *Fundamentals of Cryptography*, Kluwer Academic Publishers, 1999.
39. W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, 2nd Edition, Prentice-Hall, 2006.
40. S. S. Wagstaff, Jr., *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC Press, 2002.
41. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer-Verlag, 2002.
42. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.

9

Elliptic Curve Discrete Logarithm Based Cryptography

In this chapter, we shall give an account of the elliptic curve discrete logarithm-based (ECDLP-based, for short) pubic-key cryptosystems and digital signatures, these include:

- Elliptic curve Diffie–Helman–Meerkle key-exchange
- Elliptic curve Massey–Omura three-pass protocol
- Elliptic curve ElGamal cryptography
- Elliptic curve RSA cryptosystem
- Menezes–Vanstone elliptic curve cryptography
- Elliptic curve digital signature algorithm (ECDSA).

9.1 Basic Ideas

Elliptic curve is ubiquitous in mathematics and computing, for example, we have seen in this book that elliptic curves have novel applications to primality testing and integer factorization. However, the applications of elliptic curves to cryptography were not found until the following two seminal papers were published:

[1] Victor Miller [1], "Uses of Elliptic Curves in Cryptography", *Lecture Notes in Computer Science*, **218**, Springer, 1986, pp. 417–426.
[2] Neal Koblitz [2]:1987ECC, "Elliptic Curve Cryptography", *Mathematics of Computation*, **48**, 1987, pp. 203–209.

Since then, elliptic curves have been studied extensively for the purpose of cryptography, and many practically more secure encryption and digital signature schemes have been developed based on elliptic curves. Now elliptic curve cryptography (ECC) is a standard term in the field. There is even a computer company in Canada, Certicom, which is a leading provider of cryptographic technology based on elliptic curves. Now we will move on to discussing the basic ideas and computational methods of elliptic curve cryptography.

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

To implement elliptic curve cryptography, we need to do the following precomputations:

[1] Embed messages on elliptic curves: Our aim here is to do cryptography with elliptic curve groups in place of $\mathbb{F}_q$. More specifically, we wish to embed plaintext messages as points on an elliptic curve defined over a finite field $\mathbb{F}_q$, with $q = p^r$ and $p \in$ Primes. Let our message units m be integers $0 \le m \le M$, let also κ be a large enough integer for us to be satisfied with an error probability of $2^{-\kappa}$ when we attempt to embed a plaintext message m. In practice, $30 \le \kappa \le 50$. Now let us take $\kappa = 30$ and an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_q$. Given a message number m, we compute a set of values for x:

$$x = \{m\kappa + j,\ j = 0, 1, 2, \ldots\} = \{30m,\ 30m + 1,\ 30m + 2,\ \cdots\}$$

until we find $x^3 + ax + b$ is a square modulo p, giving us a point $(x, \sqrt{x^3 + ax + b})$ on E. To convert a point (x, y) on E back to a message number m, we just compute $m = \lfloor x/30 \rfloor$. Since $x^3 + ax + b$ is a square for approximately 50% of all x, there is only about a $2^{-\kappa}$ probability that this method will fail to produce a point on E over $\mathbb{F}_q$. In what follows, we shall give a simple example of how to embed a message number by a point on an elliptic curve. Let E be $y^2 = x^3 + 3x$, $m = 2174$ and $p = 4177$ (in practice, we select $p > 30m$). Then we calculate $x = \{30 \cdot 2174 + j,\ j = 0, 1, 2, \ldots\}$ until $x^3 + 3x$ is a square modulo 4177. We find that when $j = 15$:

$$\begin{aligned} x &= 30 \cdot 2174 + 15 \\ &= 65235, \\ x^3 + 3x &= (30 \cdot 2174 + 15)^3 + 3(30 \cdot 2174 + 15) \\ &= 277614407048580 \\ &\equiv 1444 \bmod 4177 \\ &\equiv 38^2. \end{aligned}$$

So we get the message point for $m = 2174$:

$$(x,\ \sqrt{x^3 + ax + b}) = (65235, 38).$$

To convert the message point (65235, 38) on E back to its original message number m, we just compute

$$m = \lfloor 65235/30 \rfloor = \lfloor 2174.5 \rfloor = 2174.$$

[2] Multiply points on elliptic curves over $\mathbb{F}_q$: We have discussed the calculation of $kP \in E$ over $\mathbb{Z}/n\mathbb{Z}$. In elliptic curve public-key cryptography, we are now interested in the calculation of $kP \in E$ over $\mathbb{F}_q$, which can be done in $\mathcal{O}(\log k(\log q)^3)$-bit operations by the *Repeated Doubling Method*. If we happen to know N, the number of points on our elliptic curve E, and if $k > N$, then the coordinates of kP on E can be computed in $\mathcal{O}((\log q)^4)$-bit operations; recall that the number N of points on E satisfies

$N \leq q + 1 + 2\sqrt{q} = \mathcal{O}(q)$ and can be computed by René Schoof's algorithm in $\mathcal{O}((\log q)^8)$-bit operations.

[3] Compute elliptic curve discrete logarithms: Let E be an elliptic curve over $\mathbb{F}_q$, and B a point on E. Then the *discrete logarithm* on E is the problem, given a point $P \in E$, find an integer $x \in \mathbb{Z}$ such that $xB = P$ if such an integer x exists. It is likely that the discrete logarithm problem on elliptic curves over $\mathbb{F}_q$ is more intractable than the discrete logarithm problem in $\mathbb{F}_q$. It is this feature that makes cryptographic systems based on elliptic curves even more secure than those based on the discrete logarithm problem. In the rest of this section, we shall discuss elliptic curve analogs of some important public-key cryptosystems.

In what follows, we shall present some elliptic curve analogs of four widely used public-key cryptosystems, namely the elliptic curve DHM, the elliptic curve Massey–Omura, the elliptic curve ElGamal, the elliptic curve RSA, and elliptic curve digital signature algorithm (ECDSA).

Problems for Section 9.1

1. Describe the advantages of ECC (elliptic curve cryptography) over integer factoring based and discrete logarithm based cryptography.
2. Give the complexity measures for the fastest known general algorithms for
   (1) the Integer Factorization Problem (IFP)
   (2) the Discrete Logarithm Problem (DLP)
   (3) the Elliptic Curve Discrete Logarithm Problem (ECDLP).
3. Give the complexity measures for
   (1) the Integer Factorization Problem (IFP)-based cryptosytems
   (2) the Discrete Logarithm Problem (DLP)-based cryptosytems
   (3) the Elliptic Curve Discrete Logarithm Problem (ECDLP)-based cryptosytems.
4. The exponential cipher, invented by Pohlig and Hellman in 1978 and based on the mod p arithmetic, is a secret-key cryptosystem, but it is very close to the RSA public-key cryptosystem based on mod n arithmetic, where $n = pq$ with p, q pime numbers. In essence, the Pohlig–Hellman cryptosystem works as follows:
   [1] Choose a large prime number p and the encryption key k such that $0 < k < p$ and $\gcd(k, p - 1) = 1$.
   [2] Compute the decryption key k' such that $k \cdot k' \equiv 1 (\mathrm{mod}\, p - 1)$.
   [3] Encryption: $C \equiv M^k (\mathrm{mod}\, p)$.
   [4] Decryption: $M \equiv C^{k'} (\mathrm{mod}\, p)$.
   Clearly, if you change the modulo p to modulo $n = pq$, then the Pohlig–Hellman cryptosystem is just the RSA cryptosystem.
   (1) Design an elliptic curve analog of the Pohlig–Hellman cryptosystem.
   (2) Explain why the original Pohlig–Hellman cryptosystem is easy to break whereas the elliptic curve Pohlig–Hellman cryptosystem is hard to break.

9.2 Elliptic Curve Diffie–Hellman–Merkle Key Exchange Scheme

The Diffie–Hellman–Merkle key-exchange scheme over a finite field $\mathbb{F}_p$ can be easily extended to elliptic curve E over a finite field $\mathbb{F}_p$ (denoted by $E\backslash(\mathbb{F}_p)$); such an elliptic curve analog may be described as follows (see Figure 9.1).

[1] Alice and Bob publicly choose a finite field $\mathbb{F}_q$ with $q = p^r$ and $p \in$ Primes, an elliptic curve E over $\mathbb{F}_q$, and a random *base* point $P \in E$ such that P generates a large subgroup of E, preferably of the same size as that of E itself. All of this is public information.
[2] To agree on a secret key, Alice and Bob choose two secret random integers a and b. Alice computes $aP \in E$ and sends aP to Bob; Bob computes $bP \in E$ and sends bP to Alice. Both aP and bP are, of course, public but a and b are not.
[3] Now both Alice and Bob compute the secret key $abP \in E$, and use it for further secure communications.
[4] Cryptanalysis: For the eavesdropper Eve to get abP, she has to either to find a from (abP, P) or b from (bP, P).

As everybody knows, there is no known fast way to compute abP if one only knows P, aP and bP – this is the infeasible Elliptic Curve Discrete Logarithm Problem (ECDLP).

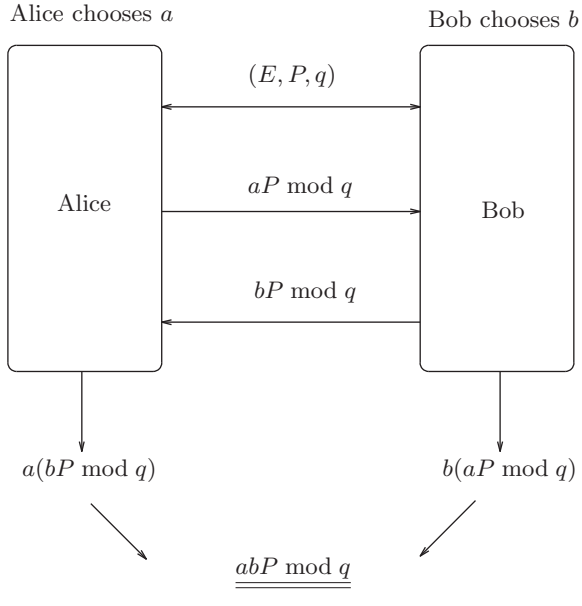
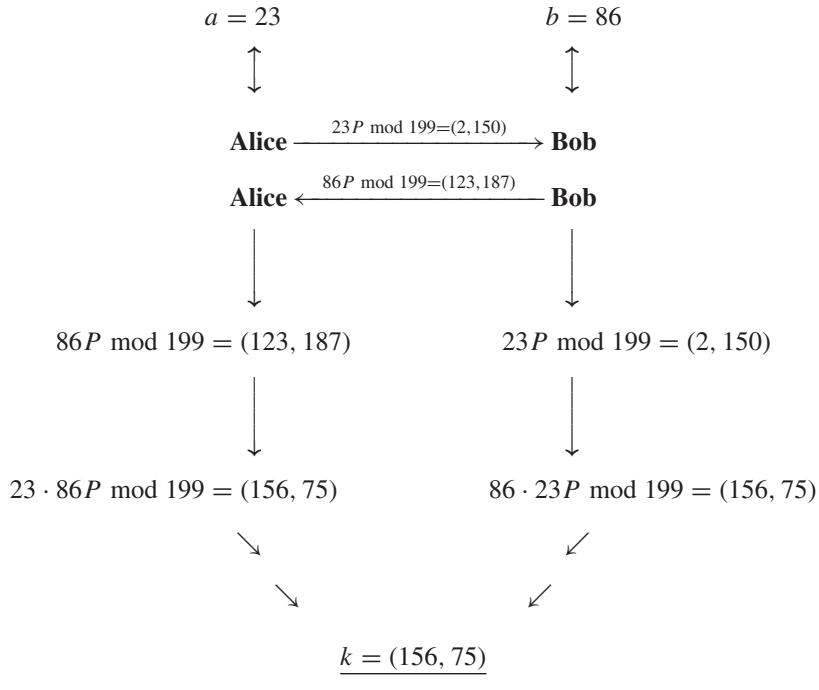


Figure 9.1 Elliptic curve Diffie–Hellman–Merkle key-exchange scheme

Example 9.1 The following is an elliptic curve analog of the DHM scheme. Let

$$E\backslash\mathbb{F}_{199} : y^2 \equiv x^3 + x - 3,$$
$$P = (1, 76) \in E(\mathbb{F}_{199}),$$
$$a = 23$$
$$b = 86$$

Then



Clearly, anyone who can find the discrete logarithm a or b such that

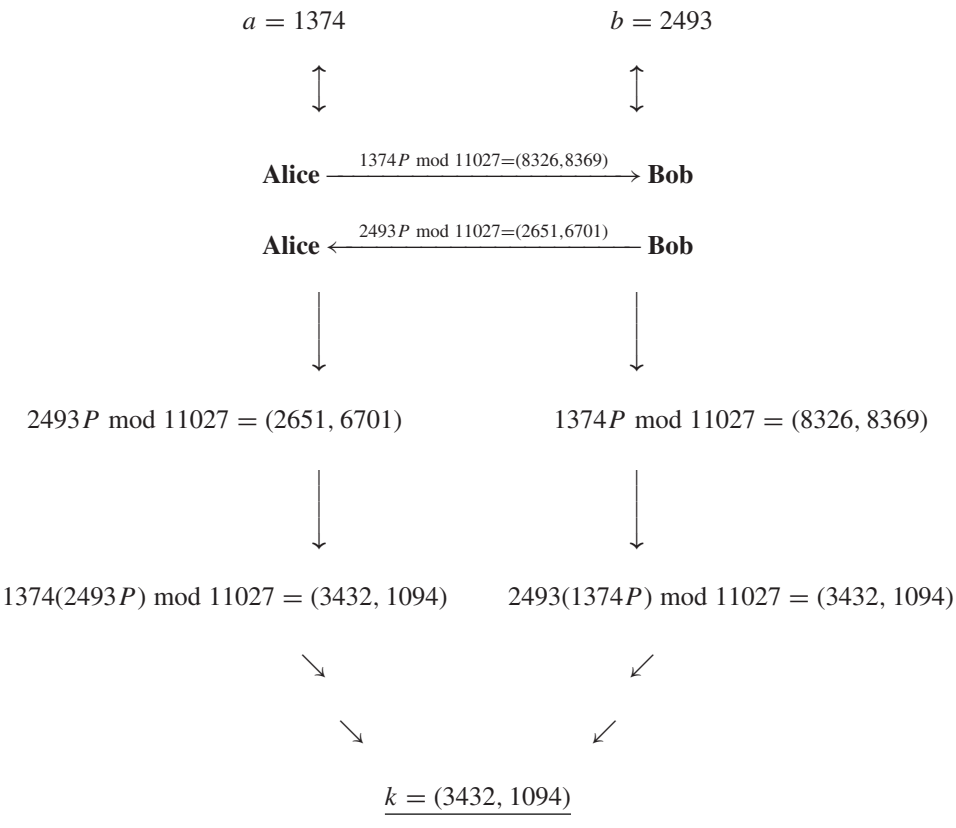
$$(2, 150) \equiv a(1, 76) \pmod{199}, \quad (123, 187) \equiv b(1, 76) \pmod{199}$$

can get the key $abP \equiv (156, 75) \pmod{199}$.

Example 9.2 We illustrate another example of the elliptic curve analog of the DHM scheme. Let

$$E\backslash\mathbb{F}_{11027} : y^2 \equiv x^3 + 4601x + 548,$$
$$P = (9954, 8879) \in E(\mathbb{F}_{11027})$$
$$a = 1374$$
$$b = 2493$$

Then



Anyone who can find the discrete logarithm a or b such that

$$(8326, 8369) \equiv a(9954, 8879) \pmod{11027}$$

or

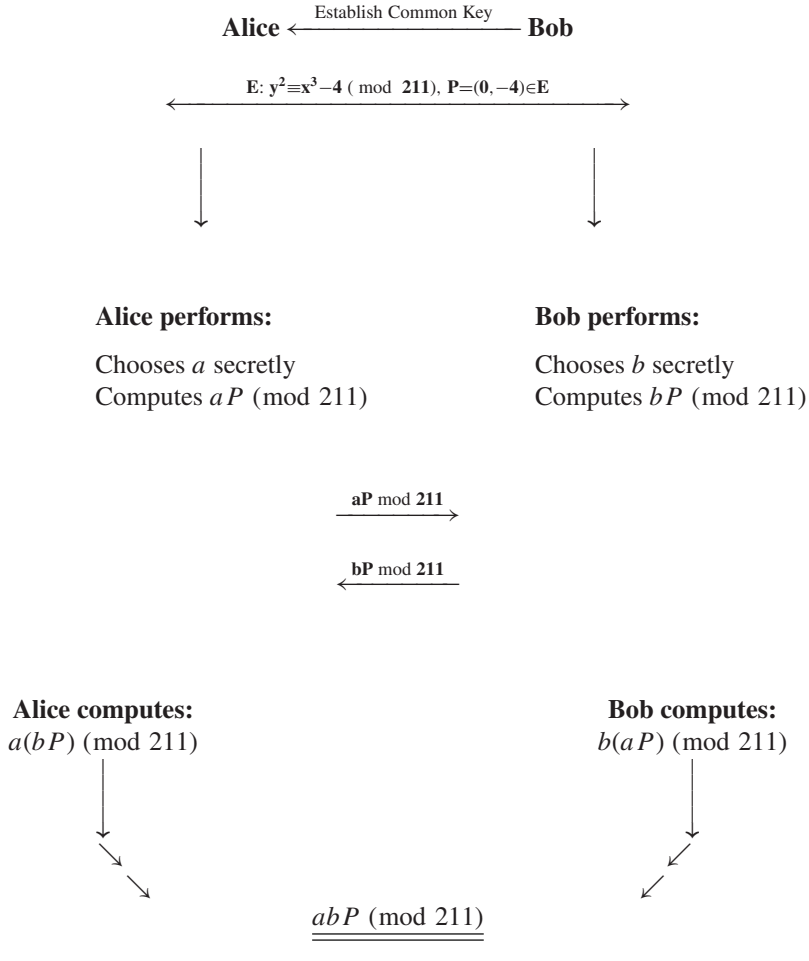
$$(2651, 6701) \equiv b(9954, 8879) \pmod{11027}$$

can get the key $abP \equiv ((3432, 1094)) \pmod{11027}$.

Problems for Section 9.2

1. Koyama et al. [3] proposed three trapdoor one-way functions; one of the functions claimed to be applicable to zero-knowledge identification protocols. Give an implementation of the elliptic curve trapdoor one-way function for the zero-knowledge identification protocol.

2. Suppose that Alice and Bob want to establish a secret key for future encryption in EC DHM key-exchange. Both Alice and Bob perform as follows:

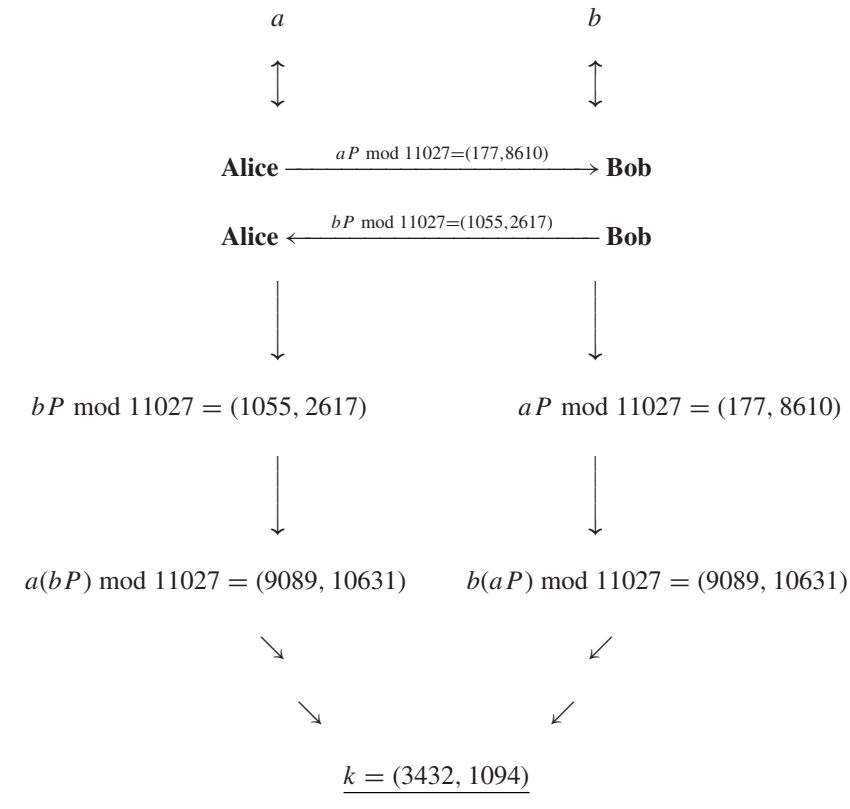


Find the actual values for
(1) aP mod 211.
(2) bP mod 211.
(3) abP mod 211.
(4) baP mod 211.

Verify if $abP \equiv baP \pmod{211}$.

3. Let the elliptic curve analog of a DHM scheme be as follows.

$$E \backslash \mathbb{F}_{11027} : y^2 \equiv x^3 + 4601x + 548$$
$$P = (2651, 6701) \in E(\mathbb{F}_{11027})$$



(1) Find the discrete logarithm a such that

$$aP \bmod 11027 = (177, 8610).$$

(2) Find the discrete logarithm b such that

$$bP \bmod 11027 = (1055, 2617).$$

9.3 Elliptic Curve Massey–Omura Cryptography

Recall that the Massey–Omura cryptographic scheme is a three-pass protocol for sending messages, allowing Alice to securely send a message to Bob without the need to exchange or distribute encryption keys. Let E be an elliptic curve over $\mathbb{F}_q$ with q a prime power, and $M = P \in E(\mathbb{F}_q)$ the original message point. Then the elliptic curve analog of the Massey–Omura cryptosystem may be described as follows (see also Figure 9.2).

[1] Alice and Bob publicly choose an elliptic curve E over $\mathbb{F}_q$ with $q = p^r$, a large prime power; as usual, we assume $q = p$ and we suppose also that the number of points on $E\backslash\mathbb{F}_q$ (denoted by N) is publicly known.

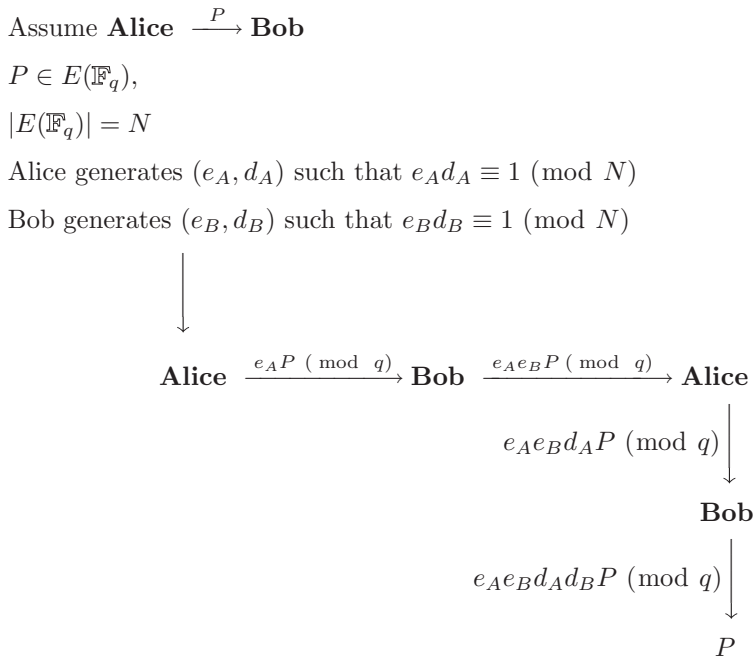


Figure 9.2 Elliptic Curve The Massey–Omura cryptography

[2] Alice chooses a secret pair of numbers $(e_A,\ d_A)$ such that $d_A e_A \equiv 1 \pmod{N}$. Similarly, Bob chooses $(e_B,\ d_B)$ such that $d_B e_B \equiv 1 \pmod{N}$.
[3] If Alice wants to send a secret message-point $P \in E$ to Bob, then the procedure should be as follows:
  [a] Alice sends $e_A P \bmod q$ to Bob,
  [b] Bob sends $e_B e_A P \bmod q$ to Alice,
  [c] Alice sends $d_A e_B e_A P \bmod q = e_B P$ to Bob,
  [d] Bob computes $d_B e_B P = P$ and hence recovers the original message point.

Note that an eavesdropper would know $e_A P$, $e_B e_A P$, and $e_B P$. So if he could solve the elliptic curve discrete logarithm problem on E, he could determine e_B from the first two points and then compute $d_B = e_B^{-1} \bmod q$ and hence get $P = d_B(e_B P)$.

Example 9.3 We follow closely the steps in the above discussed elliptic curve Massey–Omura cryptography. Let

$$
\begin{aligned}
&p = 13,\\
&E\backslash\mathbb{F}_{13} : y^2 \equiv x^3 + 4x + 4 \pmod{13}\\
&|E(\mathbb{F}_{13})| = 15\\
&M = (12, 8)),
\end{aligned}
$$

$$(e_A, d_A) \equiv (7, 13) \pmod{15},$$
$$(e_B, d_B) \equiv (2, 8) \pmod{15}.$$

Then

$$\begin{aligned}
e_A M &\equiv 7(12, 8) \pmod{13},\\
&\equiv (1, 10) \pmod{13},\\
e_A e_B M &\equiv e_B(1, 10) \pmod{13},\\
&\equiv 2(1, 10) \pmod{13},\\
&\equiv (12, 5) \pmod{13},\\
e_A e_B d_A M &\equiv d_A(12, 5) \pmod{13},\\
&\equiv 13(12, 5) \pmod{13},\\
&\equiv (6, 6) \pmod{13},\\
e_A e_B d_A d_B M &\equiv d_B(6, 6) \pmod{13},\\
&\equiv 8(6, 6) \pmod{13},\\
&\equiv (12, 8) \pmod{13},\\
&\downarrow\\
&M.
\end{aligned}$$

Example 9.4 Let

$$\begin{aligned}
&p = 13,\\
&E\backslash\mathbb{F}_{13} : y^2 \equiv x^3 + x \pmod{13}\\
&|E(\mathbb{F}_{13})| = 20\\
&M = (11, 9)),\\
&(e_A, d_A) \equiv (3, 7) \pmod{20},\\
&(e_B, d_B) \equiv (13, 17) \pmod{20}.
\end{aligned}$$

Then

$$\begin{aligned}
e_A M &\equiv 3(11, 9) \pmod{13},\\
&\equiv (7, 5) \pmod{13},\\
e_A e_B M &\equiv e_B(7, 5) \pmod{13},\\
&\equiv 13(7, 5) \pmod{13},\\
&\equiv (11, 4) \pmod{13},\\
e_A e_B d_A M &\equiv d_A(11, 4) \pmod{13},\\
&\equiv 17(11, 4) \pmod{13},\\
&\equiv (7, 5) \pmod{13},
\end{aligned}$$

$$\begin{aligned} e_A e_B d_A d_B M &\equiv d_B(7,5) \pmod{13}, \\ &\equiv 17(7,5) \pmod{13}, \\ &\equiv (11,9) \pmod{13}, \\ &\quad\downarrow \\ &\quad M. \end{aligned}$$

Problems for Section 9.3

1. In RSA public-key cryptosystem, (e, n) is the public key whereas d is the private key. What are the public key and secret key, respectively, in the Massey–Omura cryptosystem?
2. Consider the elliptic curve E

$$E : y^2 = x^3 + x - 3$$

over the field $\mathbb{F}_{199}$. Let $M = (1, 76) \in E(\mathbb{F}_{199})$ and $(e_A, e_B) = (23, 71)$.
(1) Find the number of points, N, in $E(\mathbb{F}_{199})$.
(2) Find

$$\begin{aligned} &e_A P \bmod q, \\ &e_A e_B M \bmod q, \end{aligned}$$

(3) Find

$$\begin{aligned} &e_A e_B d_A M \bmod q, \\ &e_A e_B d_A d_B M \bmod q. \end{aligned}$$

(4) Check if $e_A e_B d_A d_B M \bmod q = P$?
3. Consider the elliptic curve E

$$E : y^2 = x^3 + 1441x + 611$$

over the field $\mathbb{F}_{2591}$. Let $P = (1619, 2103) \in E(\mathbb{F}_{2591})$, $(e_A, e_B) = (107, 257)$.
(1) Find the number of points, N, in $E(\mathbb{F}_{2591})$.
(2) Find

$$\begin{aligned} &e_A P \bmod q, \\ &e_A(e_B M) \bmod q. \end{aligned}$$

(3) Find

$$\begin{aligned} &d_A(e_A e_B) M \bmod q, \\ &d_B(d_A e_A e_B M) \bmod q. \end{aligned}$$

(4) Check if $e_A e_B d_A d_B P \bmod q = M$?

4. Let p be a 200-digit prime number as follows:

$$p = 10000000000000000000000000000000000000000000000000\_$$
$$00000000000000000000000000000000000000000000000000\_$$
$$00000000000000000000000000000000000000000000000000\_$$
$$00000000000000000000000000000000000000000000000153$$

Let the elliptic curve over $\mathbb{F}_p$ be as follows:

$$E\backslash\mathbb{F}_p : y^2 \equiv x^3 + 105x + 78153 \pmod{p},$$

with a point order:

$$N = 10000000000000000000000000000000000000000000000000\_$$
$$00000000000000000000000000000000000000000000000000\_$$
$$06789750288004224118080314365460277641928049641888\_$$
$$39991591392960032210630561760029050858613689631753$$

(1) Let $e_A = 179$, compute $d_A \equiv \frac{1}{e_A} \bmod N$.

(2) Let $e_B = 983$, compute $d_B \equiv \frac{1}{e_B} \bmod N$.

5. Let p be a prime number

$$p = 12345678901234567890123456789065483337452508596673\_$$
$$7125236501$$

Let the elliptic curve over $\mathbb{F}_p$ be as follows:
$y^2 \equiv x^3 +$
$112507913528623610837613885503682230698868883572599681384335x$
$-112507913528623610837613885503682230698868883572599681384335$
$\pmod{p}$.
with order $|E(\mathbb{F}_p)| = N$ as follows:

$$12345678901234567890123456789012345678901234568197.$$

Let
(764298923297529289535635175490327802980486022328440631574 9,
100181741322448105444520871614464053169400529776945655771441)
be the plaintext point M. Suppose Alice wishes to send M to Bob.
Let

$e_A = 3$,
$d_A = 82304526008230452600823045260082304526008230452600823045465$,
$e_B = 7$,
$d_B = 17636684144620811271604938270017636684144620811271604938314$,

all modulo N. Compute:

(1) $e_A M \bmod p$.

(2) $e_B(e_A M) \bmod p$.

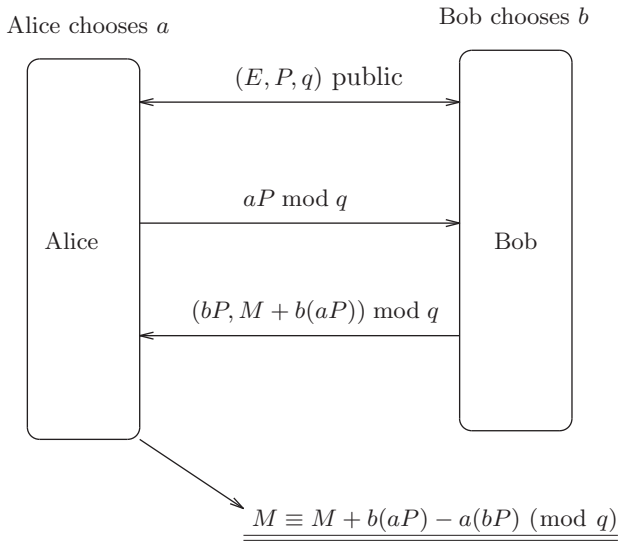


Figure 9.3 Elliptic curve ElGamal cryptography

(3) $d_A(e_B e_A M) \bmod p$.
(4) $d_B(d_A e_B e_A M) \bmod p$.
(5) Check if $d_B(d_A e_B e_A M) \bmod p = M$.

9.4 Elliptic Curve ElGamal Cryptography

Just the same as many other public-key cryptosystems, the famous ElGamal cryptosystem also has a very straightforward elliptic curve analog, which may be described as follows (see also Figure 9.3).

[1] Suppose Bob wishes to send a secret message to Alice:

$$\text{Bob} \xrightarrow{\text{Secrete Message}} \text{Alice}.$$

Alice and Bob publicly choose an elliptic curve E over $\mathbb{F}_q$ with $q = p^r$ a prime power, and a random *base* point $P \in E$. Suppose they also know the number of points on E, that is, they know $|E(\mathbb{F}_q)| = N$.

[2] Alice chooses a random integer a, computes $aP \bmod q$, and sends it to Bob.

[3] Encryption: Bob chooses at random an integer b and computes $bP \bmod q$. Bob also computes $M + b(aP) \bmod q$. Then Bob sends the secret encrypted message $(bP, M + b(aP)) \bmod q$ to Alice.

[4] Decryption: Since Alice has the secret key a, she can compute $a(bP) \bmod q$ and get

$$M \equiv M + a(bP) - b(aP) \pmod{q}, \tag{9.1}$$

the original plaintext message.

[5] Cryptanalysis: Eve, the eavesdropper, can only get M if she can solve the Elliptic Curve Discrete Logarithm Problem. That is, she can get M if she can find a from $aP \bmod q$ or b from $bP \bmod q$. But, as everybody knows, there is no efficient way to compute the elliptic curve discrete logarithms, so the ElGamal cryptosystem system is secure.

Example 9.5 Suppose Bob wishes to send Alice a secret message M by using the elliptic curve ElGamal cryptographic scheme.

[1] Set-up;

$$\begin{aligned}
&E\backslash\mathbb{F}_{29} : y^2 \equiv x^3 - x + 16 \pmod{29},\\
&N = |E(\mathbb{F}_{29})| = 31,\\
&P = (5, 7) \in E(\mathbb{F}_{29}),\\
&M = (28, 25).
\end{aligned}$$

[2] Public-key generation: Assume Bob sends the secret message M to Alice, so Alice:

$$\begin{aligned}
&\text{Chooses a random secret integer } a = 23,\\
&\text{Computes } aP = 23P = (21, 18) \pmod{29},\\
&\text{Sends } aP = (21, 18) \pmod{29} \text{ to Bob.}
\end{aligned}$$

[3] Encryption: Bob

$$\begin{aligned}
&\text{Chooses a random secret integer } b = 25,\\
&\text{Computes } bP = 25P = (13, 24) \pmod{29},\\
&\qquad b(aP) = 17(23P) = 17(21, 18) = (1, 25) \pmod{29},\\
&\qquad M + b(aP) = (28, 25) + (1, 25) = (0, 4) \pmod{29},\\
&\text{Sends } (bP = (1, 25),\ M + b(aP) = (0, 4)) \text{ to Alice.}
\end{aligned}$$

[4] Decryption: Alice computes

$$\begin{aligned}
a(bP) &= 23(25P) = 23(13, 24) = (1, 25),\\
M &= M + b(aP) - a(bP)\\
&= (0, 4) - (1, 25)\\
&= (0, 4) + (1, -25)\\
&= (28, 25).
\end{aligned}$$

So, Alice recovers the original secret message $M = (28, 25)$.

Example 9.6 Now we give one more example on the elliptic curve ElGamal cryptosystem.

[1] Set-up;

$$E\backslash\mathbb{F}_{523} : y^2 \equiv x^3 + 22x + 153 \pmod{523},$$
$$P = (167, 118) \in E(\mathbb{F}_{523}),$$
$$M = (220, 287) \text{ is the plaintext.}$$

[2] Public-key generation: Assume Bob sends the secret message M to Alice, so Alice:

$$\text{Chooses a random secret integer } a = 97,$$
$$\text{Computes } aP = 97(167, 118) = (167, 405) \pmod{523},$$
$$\text{Sends } aP = (167, 405) \pmod{523} \text{ to Bob.}$$

[3] Encryption: Bob

$$\text{Chooses a random secret integer } b = 263,$$
$$\text{Computes } bP = 263(167, 118) = (5, 503) \pmod{523},$$
$$b(aP) = 263(167, 405) = (5, 20) \pmod{523},$$
$$M + b(aP) = (220, 287) + (5, 20)$$
$$= (36, 158) \pmod{523},$$
$$\text{Sends } (bP = (5, 503),\ M + b(aP) = (36, 158)) \text{ to Alice.}$$

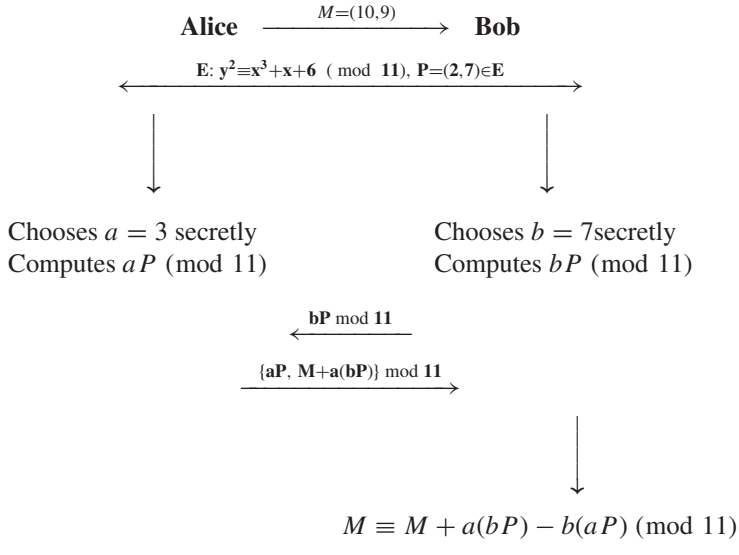
[4] Decryption: Alice computes

$$a(bP) = 97(5, 503) = (5, 20),$$
$$M = M + b(aP) - a(bP)$$
$$= (36, 158) - (5, 20)$$
$$= (36, 158) + (5, 503)$$
$$= (220, 287).$$

So, Alice recovers the original secret message $M = (220, 287)$.

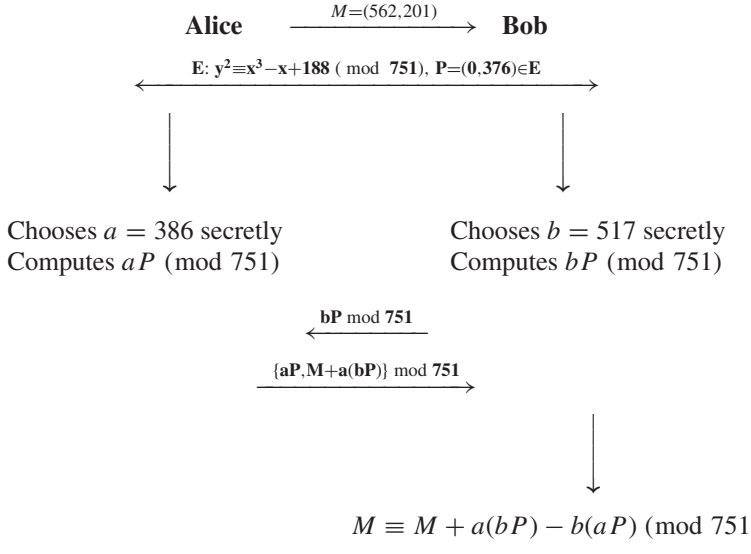
Problems for Section 9.4

1. Suppose that Alice wants to send Bob a secret message $M = (10, 9)$ using elliptic curve ElGamal cryptography. Both Alice and Bob perform as follows:



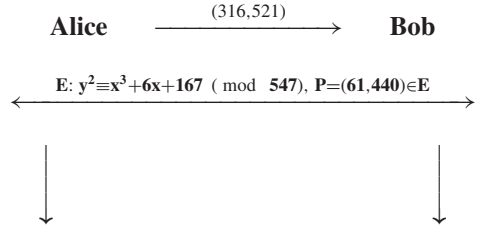
Compute the actual values for
(1) aP mod 11.
(2) bP mod 11.
(3) $b(aP)$ mod 11.
(4) $a(bP)$ mod 11.
(5) $M + a(bP)$ mod 11.
(6) $M + a(bP) - b(aP)$ (mod 11).
Check if $M + a(bP) - b(aP)$ (mod 11) $= (10, 9)$?

2. Suppose that Alice wants to send Bob a secret message $M = (562, 201)$ in elliptic curve ElGamal cryptography. Both Alice and Bob perform the following:



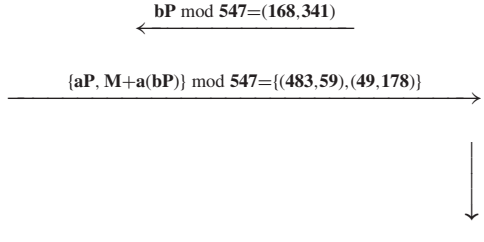
Compute the actual values for
(1) aP mod 751.
(2) bP mod 751.
(3) $a(bP)$ mod 751.
(4) $b(aP)$ mod 751.
(5) $M + a(bP)$ mod 751.
(6) $M + a(bP) - b(aP)$ mod 751.
Check if $M + a(bP) - b(aP)$ mod 751 $= (562, 201)$?

3. Suppose that Alice wants to send Bob a secret message $M = (316, 521)$ in elliptic curve ElGamal cryptography. Both Alice and Bob perform the following:



Chooses a secretly
Computes aP (mod 547)
$= (483, 59)$

Chooses b secretly
Computes bP (mod 547)
$= (168, 341)$



$$M \equiv M + a(bP) - b(aP) \pmod{547}$$
$$\equiv (49, 178) + (143, -443) \pmod{547}$$
$$\equiv (316, 521) \pmod{547}$$

Find
(1) a such that aP mod 547 $= (483, 59)$.
(2) b such that bP mod 547 $= (168, 341)$.
(3) $a(bP)$ mod 547.
(4) $b(aP)$ mod 547.
(5) Check if $a(bP) \equiv b(aP)$ (mod 547).

9.5 Elliptic Curve RSA Cryptosystem

The most widely used RSA cryptosystem can not only be used for both encryption and digital signatures, but also has a natural analog of elliptic curve cryptography, and in fact, several elliptic curve RSA cryptosystems have been developed. In what follows, we describe one of the analogs.

[1] $N = pq$ is a public key which is the product of the two large secret primes p and q.
[2] Choose two random integers a and b such that $E : y^2 = x^3 + ax + b$ defines an elliptic curve both mod p and mod q.
[3] To encrypt a message-point P, just perform $eP \mod N$, where e is the public (encryption) key. To decrypt, one needs to know the number of points on E modulo both p and q.

The above are some elliptic curve analogs of certain public-key cryptosystems. It should be noted that almost every public-key cryptosystem has an elliptic curve analogue; it is of course possible to develop new elliptic curve cryptosystems which do not rely on the existing cryptosystems.

It should be also noted that the digital signature schemes can also be analoged by elliptic curves over $\mathbb{F}_q$ or over $\mathbb{Z}/n\mathbb{Z}$ with $n = pq$ and $p, q \in$ Primes in exactly the same way as that for public-key cryptography; several elliptic curve analogs of digital signature schemes have already been proposed, for example [4].

Problems for Section 9.5

1. The following is an elliptic curve version of the RSA cryptosystems developed by Koyama, Maurer, Okamoto, and Vanstone. Suppose Bob wishes to send Alice a message. Both Bob and Alice perform as follows:
   (1) **Key generation**: Bob first selects two large distinct primes p and q such that $p \equiv q \equiv 2 \pmod 3$ and computes $n = pq$, then he selects integers e and d such that $ed \equiv 1 \pmod{\text{lcm}(p+1, q+1)}$, finally he announces (n, e) as his public key, but keeps (d, p, q) secret.
   (2) **Encryption**: Alice represents her message

$$M \equiv (m_1, m_2) \pmod n,$$

   where M is a point on the elliptic curve

$$y^2 \equiv x^3 + b \pmod n$$

   with

$$b \equiv m_2^3 - m_1^3 \pmod n.$$

   Alice then computes $C \equiv (em_1, em_2) \pmod n$. Finally she sends C to Bob.

(3) **Decryption**: Bob computes

$$M \equiv dC \equiv (dem_1, dem_2) \equiv (m_1.m_2).$$

Suppose now that Eve does not know the prime factorization of n but she knows in some way d. Show that she can factor n with high probability.

2. In 1993 Demytko [5] proposed an elliptic curve RSA cryptosystem, which uses a fixed randomly chosen elliptic curve E over $\mathbb{Z}_n$, where $n = pq$ is an RSA modulus. Give a full description of the Demytko cryptosystem, and extend it to a semantically secure elliptic curve RSA.

9.6 Menezes–Vanstone Elliptic Curve Cryptography

A serious problem with all the above mentioned elliptic curve cryptosystems is that the plaintext message units m lie on the elliptic curve E, and there is no convenient method known of deterministically generating such points on E. Fortunately, Menezes and Vanstone discovered a more efficient variation [6]; in this variation which we shall describe below, the elliptic curve is used for "masking," and the plaintext and ciphertext pairs are allowed to be in $\mathbb{F}_p^* \times \mathbb{F}_p^*$ rather than on the elliptic curve.

[1] Key generation: Alice and Bob publicly choose an elliptic curve E over $\mathbb{F}_p$ with $p > 3$ prime and a random *base* point $P \in E(\mathbb{F}_p)$ such that P generates a large subgroup H of $E(\mathbb{F}_p)$, preferably of the same size as that of $E(\mathbb{F}_p)$ itself. Assume that randomly chosen $k \in \mathbb{Z}_{|H|}$ and $a \in \mathbb{N}$ are secret.

[2] Encryption: Suppose now Alice wants to send message

$$m = (m_1, m_2) \in (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/p\mathbb{Z})^* \tag{9.2}$$

to Bob, then she does the following:

[a] $\beta = aP$, where P and β are public;

[b] $(y_1, y_2) = k\beta$;

[c] $c_0 = kP$;

[d] $c_j \equiv y_j m_j \pmod{p}$ for $j = 1, 2$;

[e] Alice sends the encrypted message c of m to Bob:

$$c = (c_0, c_1, c_2). \tag{9.3}$$

[3] Decryption: Upon receiving Alice's encrypted message c, Bob calculates the following to recover m:

[a] $ac_0 = (y_1, y_2)$;

[b] $m = \left(c_1 y_1^{-1} \pmod{p},\ c_2 y_2^{-1} \pmod{p}\right)$.

Example 9.7 The following is a nice example of the Menezes–Vanstone cryptosystem [8].

[1] Key generation: Let E be the elliptic curve given by $y^2 = x^3 + 4x + 4$ over $\mathbb{F}_{13}$, and $P = (1, 3)$ be a point on E. Choose $E(\mathbb{F}_{13}) = H$ which is cyclic of order 15, generated by P. Let also the private keys be $k = 5$ and $a = 2$, and the plaintext $m = (12, 7) = (m_1, m2)$.

[2] Encryption: Alice computes:

$$\begin{aligned}
&\beta = aP = 2(1, 3) = (12, 8),\\
&(y_1, y_2) = k\beta = 5(12, 8) = (10, 11),\\
&c_0 = kP = 5(1, 3) = (10, 2),\\
&c_1 \equiv y_1 m_1 \equiv 10 \cdot 2 \equiv 3 \pmod{13}\\
&c_2 \equiv y_2 m_2 \equiv 11 \cdot 7 \equiv 12 \pmod{13}.
\end{aligned}$$

Then Alice sends

$$c = (c_0, c_1, c_2) = ((10, 2), 3, 12)$$

to Bob.

[3] Decryption: Upon receiving Alice's message, Bob computes:

$$\begin{aligned}
&ac_0 = 2(10, 2) = (10, 11) = (y_1, y_2),\\
&m_1 \equiv c_1 y_1^{-1} \equiv 12 \pmod{13},\\
&m_2 \equiv c_2 y_2^{-1} \equiv 7 \pmod{13}.
\end{aligned}$$

Thus, Bob recovers the message $m = (12, 7)$.

Problems for Section 9.6

1. Let $E\backslash\mathbb{F}_{2^m}$ be the elliptic curve E over $\mathbb{F}_{2^m}$ with $m > 1$, where E is defined by

$$y^2 + xy = x^3 + ax^2 + b.$$

(1) Let $P, Q \in E$ with $P \neq \pm Q$ be two points on E. Find the addition formula for computing $P + Q$.
(2) Let $P \in E$ with $P \neq -P$. Find the addition formula for computing $2P$.
(3) Let $E\backslash\mathbb{F}_{2^m}$ be as follows:

$$E\backslash\mathbb{F}_{2^4} : y^2 \equiv x^3 + \alpha^4 x^2 + 1 \pmod{2^4}.$$

Find all the points, $E(\mathbb{F}_{2^4})$, including the point at infinity, on the E.
(4) Let $P = (\alpha^6, \alpha^8)$ and $Q = (\alpha^3, \alpha^{13})$ be in $E\backslash\mathbb{F}_{2^4}$ defined above, find $P + Q$ and $2P$.

2. Give an implementation, in any programming language, of the Menezes–Vanstone cryptosystem.
3. Develop an effective attack on the Menezes–Vanstone cryptosystem.

9.7 Elliptic Curve DSA

We have already noted that almost every public-key cryptosystem has an elliptic curve analog. It should also be noted that digital signature schemes can also be represented by elliptic curves over $\mathbb{F}_q$ with q a prime power or over $\mathbb{Z}/n\mathbb{Z}$ with $n = pq$ and $p, q \in$ Primes. In exactly the same way as that for public-key cryptography, several elliptic curve analogs of digital signature schemes have already been proposed (see, for example, Meyer and Müller [4]). In what follows we shall describe an elliptic curve analog of the DSA/DSS, called the ECDSA [7].

Algorithm 9.1 (Elliptic curve digital signature algorithm) Let E be an elliptic curve over $\mathbb{F}_p$ with p prime, and let P be a point of prime order q (note that the q here is just a prime number, not a prime power) in $E(\mathbb{F}_p)$. Suppose Alice wishes to send a signed message to Bob.

[1] [ECDSA key generation] Alice does the following:
  [1-1] select a random integer $x \in [1, q-1]$,
  [1-2] compute $Q = xP$,
  [1-3] make Q public, but keep x secret.
  Now Alice has generated the public key Q and the private key x.
[2] [ECDSA signature generation] To sign a message m, Alice does the following:
  [2-1] select a random integer $k \in [1, q-1]$,
  [2-2] compute $kP = (x_1, y_1)$, and $r \equiv x_1 \pmod{q}$. If $r = 0$, go to step [2-1].
  [2-3] compute $k^{-1} \bmod q$.
  [2-4] compute $s \equiv k^{-1}(H(m) + xr) \pmod{q}$, where $H(m)$ is the hash value of the message. If $s = 0$, go to step [2-1].
  The signature for the message m is the pair of integers (r, s).
[3] [ECDSA signature verification] To verify Alice's signature (r, s) of the message m, Bob should do the following:
  [3-1] obtain an authenticated copy of Alice's public key Q;
  [3-2] verify that (r, s) are integers in the interval $[1, q-1]$, computes $kP = (x_1, y_1)$, and $r \equiv x_1 \pmod{q}$.
  [3-3] compute $w \equiv s^{-1} \pmod{q}$ and $H(m)$.
  [3-4] compute $u_1 \equiv H(m)w \pmod{q}$ and $u_2 \equiv rw \pmod{q}$.
  [3-5] compute $u_1 P + u_2 Q = (x_0, y_0)$ and $v \equiv x_0 \pmod{q}$.
  [3-6] accept the signature if and only if $v = r$.

As a conclusion to elliptic curve cryptography (ECC), we provide two remarks about the comparison of ECC and other types of cryptography, particularly the famous and widely used RSA cryptography.

Table 9.1 Key size comparison between RSA and ECC

Security level	RSA	ECC
Low	512-bits	112-bits
Medium	1024-bits	161-bits
High	3027-bits	256-bits
Very high	15360-bits	512-bits

Remark 9.1 ECC provides a high level of security using smaller keys than that used in RSA. A comparison between the key sizes for an equivalent level of security for RSA and ECC is given in the following Table 9.1.

Remark 9.2 Just the same as there are weak keys for RSA, there are also weak keys for ECC, for example, as an acceptable elliptic curve for cryptography, it must satisfy the following conditions:

1. If N is the number of integer coordinates, it must be divisible by a large prime r such that $N = kr$ for some integer k.
2. It the curve has order p modulo p, then r must not be divisible by $p^i - 1$ for a small set of i, say, $0 \leq i \leq 20$.
3. Let N be the number of integer coordinates and $p = E(\mathbb{F}_p)$, then N must not be equal to p. The curve that satisfies the condition $p = N$ is called the anomalous curve.

Problems for Section 9.7

1. The elliptic curve digital signature algorithm (ECDSA) is the elliptic curve analog of the digital signature algorithm (DSA).
   (1) Give a complete description of the elliptic curve analog of the digital signature algorithm, ECDSA.
   (2) Explain the main advantages and features of ECDSA over DSA.
   (3) Give a critical analysis of the security of ECDSA.
2. Give an implementation, in any programming language, of the elliptic curve digital signature algorithm (ECDSA).
3. Give an elliptic curve version of the RSA digital signature system.
4. Give an elliptic curve version of Rabin's digital signature system.

9.8 Bibliographic Notes and Further Reading

This chapter introduced elliptic curve cryptography (ECC) and the digital signature algorithm (ECDSA), whose security are based on the infeasibility of the Elliptic Curve Discrete Logarithm Problem. Some of the most important and widely used public-key cryptographic

systems and the idea of using elliptic curves, more specifically the Elliptic Curve Discrete Logarithm Problem as the basis to construct cryptographic systems, were independently proposed by Miller [1] and Koblitz [2]. The following references provide more information on elliptic curves and elliptic curve cryptography: [4, 8–42].

References

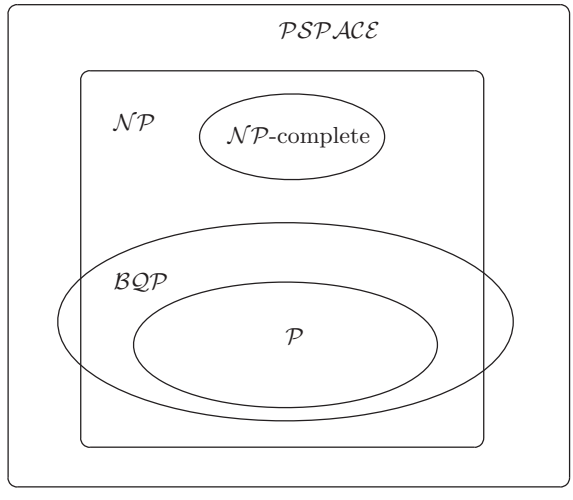
1. V. Miller, "Uses of Elliptic Curves in Cryptography", *Lecture Notes in Computer Science* **218**, Springer, 1986, pp. 417–426.
2. N. Koblitz, "Elliptic Curve Cryptography", *Mathematics of Computation*, **48**, 1987, pp. 203–209.
3. K. Koyama, U. M. Maurer, T. Okamoto, and S. A. Vanstone, "New Public-Key Schemes Based on Elliptic Curves over the Ring $\mathbb{Z}_n$", NTT Laboratories, Kyoto, Japan, 1991.
4. B. Meyer and V. Müller, "A Public Key Cryptosystem Based on Elliptic Curves over $\mathbb{Z}/n\mathbb{Z}$ Equivalent to Factoring", *Advances in Cryptology*, EUROCRYPT '96, Proceedings, Lecture Notes in Computer Science **1070**, Springer, 1996, pp. 49–59.
5. N. Demytko, "A New Elliptic Curve Based Analogue of RSA", *Advances in Cryptology – EUROCRYPT 93*, Lecture Notes in Computer Science **765**, Springer, 1994, pp. 40–49.
6. A. Menezes and S. A. Vanstone, "Elliptic Curve Cryptosystems and their Implementation", *Journal of Cryptology*, **6**, 1993, pp. 209–224. smallskip
7. D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", *International Journal of Information Security*, **1**, 1, 2001, pp. 36–63.
8. R. A. Mollin, *An Introduction to Cryptography*, 2nd Edition, Chapman & Hall/CRC, 2006.
9. G. Agnew, R. Mullin, and S. A. Vanstone, "An Implementation of Elliptic Curve Cryptosystems over $\mathbb{F}_{2^{155}}$", *IEEE Journal of Selected Areas in Communications*, **11**, 1993, pp. 804–813.
10. I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
11. I. F. Blake, G. Seroussi, and N. P. Smart. *Advances in Elliptic Curve Cryptography*, Cambridge University Press, 1999.
12. H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, CRC Press, 2006.
13. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer, 2005.
14. D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.
15. G. H. Hardy and E. M. Wright, *An Introduction to Theory of Numbers*, 6th Edition, Oxford University Press, 2008.
16. J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, 2008.
17. D. Husemöller, *Elliptic Curves*, Graduate Texts in Mathematics **111**, Springer, 1987.
18. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd Edition, Graduate Texts in Mathematics **84**, Springer, 1990.
19. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics **114**, Springer, 1994.
20. N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics **3**, Springer, 1998.
21. N. Koblitz, A. Menezes, and S. A. Vanstone, "The State of Elliptic Curve Cryptography", *Designs, Codes and Cryptography*, **19**, 2000, pp. 173–193.
22. H. W. Lenstra, Jr., *Elliptic Curves and Number-Theoretic Algorithms*, Mathematisch Instituut, Universiteit van Amsterdam, 1986.
23. A. J. Menezes, *Elliptic Curve Public Key Cryptography*. Kluwer Academic Publishers, 1993.
24. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptosystems*, CRC Press, 1996.
25. R. A. Mollin, *Algebraic Number Theory*, 2nd Edition Chapman & Hall/CRC, 2011.
26. M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning, 1999.
27. B. Schneier, *Applied Cryptography – Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons, 1996.
28. R. Schoof, "Elliptic Curves over Finite Fields and the Computation of Square Roots mod p", *Mathematics of Computation*, **44**, 1985, pp. 483–494.
29. J. H. Silverman, "The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem", *Designs, Codes and Cryptography*, **20**, 2000, pp. 5-40.

30. J. H. Silverman, "The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem", *Designs, Codes and Cryptography*, **20**, 2000, pp. 5-40.
31. J. H. Silverman, *The Arithmetic of Elliptic Curves*, 2nd Edition, Graduate Texts in Mathematics **106**, Springer, 2009.
32. J. H. Silverman and J. Suzuki, "Elliptic Curve Discrete Logarithms and the Index Calculus", *Advances in Cryptology – ASIACRYPT '98*, Lecture Notes in Computer Science **1514**, Springer, 1998, pp. 110–125.
33. N. Smart, *Cryptography: An Introduction*, McGraw-Hill, 2003.
34. M. Stamp and R. M. Low, *Applied Cryptanalysis*, Wiley, 2007.
35. A. Stanoyevitch, *Introduction to Cryptography*, CRC Press, 2011.
36. D. R. Stinson, *Cryptography: Theory and Practice*, 2nd Edition, Chapman & Hall/CRC Press, 2002.
37. H. C. A. van Tilborg, *Fundamentals of Cryptography*, Kluwer Academic Publishers, 1999.
38. W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, 2nd Edition, Prentice-Hall, 2006.
39. S. S. Wagstaff, Jr., *Cryptanalysis of Number Theoretic Ciphers*, Chapman & Hall/CRC, 2002.
40. L. C. Washinton, *Elliptic Curve: Number Theory and Cryptography*, 2nd Edition, Chapman & Hall/CRC, 2008.
41. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer, 2002.
42. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.

Part IV

Quantum Resistant Cryptography

Quantum computers have gained widespread interest because some problems of particular cryptographic interest are known to be in bounded error quantum polynomial-time ($\mathcal{BQP}$), but suspected to be outside polynomial-time ($\mathcal{P}$). The following figure shows the *conjectured* relationship of $\mathcal{BQP}$ to other common complexity classes. For example, the following three infeasible problems can all be solved in $\mathcal{BQP}$ on a quantum computer, but can only be solved in subexponential-time on a classical computer:



- Integer Factorization Problem (IFP),
- Discrete Logarithm Problem (DLP),
- Elliptic Curve Discrete Logarithm Problem (ECDLP).

Of course, the quantum algorithms for solving these problems are necessarily to be run on a practical quantum computer. Thus, once a practical quantum computer can be built, all the cryptographic schemes based on IFP, DLP, and ECDLP will be insecure. In this chapter, we shall first show some quantum algorithms for solving IFP, DLP, and ECDLP, and hence for breaking IFP-, DLP-, and ECDLP-based cryptography. Then we shall present

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

some quantum-computing resistant cryptographic schemes. By quantum-computing resistant cryptography, we mean that the quantum-computing attacks are invalid against these cryptographic schemes. This is possible, because quantum computers are not just faster versions of classical electronic computers, but use a different paradigm for computation. They would speed-up some problems such as IFP and DLP by a large factor and others problems such as the Linear Coding Problem not at all.

10

Quantum Computational Number Theory

This chapter gives an account of *efficient algorithms* for solving the three important infeasible problems in computational number theory: the Integer Factorization Problem (IFP), the Discrete Logarithm Problems (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP). As the security of many cryptographic systems and protocols relies on the infeasibility of IFP, DLP, and ECDLP, the quantum algorithms for IFP, DLP, and ECDLP form efficient attacks on the IFP-, DLP-, and ECDLP-based cryptographic systems and protocols (see Figure 10.1). In this chapter, we shall discuss the quantum algorithms for solving the infeasible number theoretic problems IFP, DLP, and ECDLP.

10.1 Quantum Algorithms for Order Finding

Definition 10.1 Let $G = \mathbb{Z}_n^*$ be a finite multiplicative group, and $x \in G$ a randomly chosen integer (element). Then an order of x in G, or an order of an element a modulo n, sometimes denoted by order(x, n), is the smallest positive integer r such that

$$x^r \equiv 1 \,(\mathrm{mod}\, n). \tag{10.1}$$

Definition 10.2 The Order Finding Problem (OFP) may be defined as follows:

$$\mathrm{OFP} \stackrel{\mathrm{def}}{=} \begin{cases} \text{Input :} & n \in \mathbb{Z}_{>1}^+, x \in \mathbb{Z}_n, \\ \text{Output :} & r \text{ such that } x^r \equiv 1 \,(\mathrm{mod}\, n). \end{cases} \tag{10.2}$$

Just the same as IFP, OFP is infeasible, as no efficient algorithm has been found for solving it.

Example 10.1 Let $5 \in \mathbb{Z}_{104}^*$. Then order$(5, 104) = 4$, since 4 is the smallest positive integer satisfying

$$5^4 \equiv 1 \,(\mathrm{mod}\, 104).$$

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

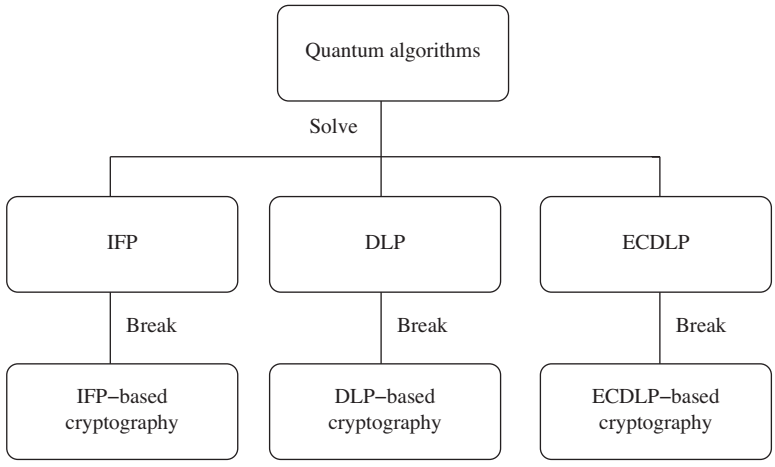


Figure 10.1 Quantum computing and quantum attacks

Theorem 10.1 *Let G be a finite group and suppose that $x \in G$ has finite order r. If $x^k = 1$, then $r \mid k$.*

Example 10.2 Let $5 \in \mathbb{Z}_{104}^*$. As $5^{24} \equiv 1 \pmod{104}$, so, $4 \mid 24$.

Definition 10.3 Let G be a finite group, then the number of elements in G, denoted by $|G|$, is called the *order* of G.

Example 10.3 Let $G = \mathbb{Z}_{104}^*$. Then there are 48 elements in G that are relatively prime to 104 (two numbers a and b are relatively prime if $\gcd(a, b) = 1$), namely;

1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 41, 43
45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 67, 69, 71, 73, 75, 77, 79, 81
83, 85, 87, 89, 93, 95, 97, 99, 101, 103

Thus, $|G| = 48$. That is, the order of the group G is 48.

Theorem 10.2 (Lagrange) *Let G be a finite group. Then the order of an element $x \in G$ divides the order of the group G.*

Example 10.4 Let $G = \mathbb{Z}_{104}^*$. Then the order of G is 48, whereas the order of the element $5 \in G$ is 4. Clearly $4 \mid 24$.

Corollary 10.1 *If a finite group G has order r, then $x^r = 1$ for all $x \in G$.*

Example 10.5 Let $G = \mathbb{Z}_{104}^*$ and $|G| = 48$. Then

$$\begin{aligned} 1^{48} &\equiv 1 \,(\text{mod } 104) \\ 3^{48} &\equiv 1 \,(\text{mod } 104) \\ 5^{48} &\equiv 1 \,(\text{mod } 104) \\ 7^{48} &\equiv 1 \,(\text{mod } 104) \\ &\vdots \\ 101^{48} &\equiv 1 \,(\text{mod } 104) \\ 103^{48} &\equiv 1 \,(\text{mod } 104). \end{aligned}$$

Now, we are in a position to present our two main theorems as follows.

Theorem 10.3 *Let C be the RSA ciphertext, and* $\text{order}(C, n)$ *the order of $C \in \mathbb{Z}_n^*$. Then*

$$d \equiv 1/e \,(\text{mod order}(C, n)). \tag{10.3}$$

Corollary 10.2 *Let C be the RSA ciphertext, and* $\text{order}(C, n)$ *the order of $C \in \mathbb{Z}_n^*$. Then*

$$M \equiv C^{1/e \,(\text{ mod } \text{order}(C,n))} \,(\text{mod } n). \tag{10.4}$$

Thus, to recover the RSA M from C, it suffices to just find the order of C modulo n.

Now we return to the actual computation of the order of an element x in $G = \mathbb{Z}_n^*$. Finding the order of an element x in G is, in theory, not a problem: Just keep multiplying until we get to "1," the identity element of the multiplicative group G. For example, let $n = 179359$, $x = 3 \in G$, and $G = \mathbb{Z}_{179359}^*$, such that $\gcd(3, 179359) = 1$. To find the order $r = \text{order}(3, 179359)$, we just keep multiplying until we get to "1":

$$\begin{array}{llll} 3^1 & \text{mod } 179359 & = & 3 \\ 3^2 & \text{mod } 179359 & = & 9 \\ 3^3 & \text{mod } 179359 & = & 27 \\ & \vdots & & \\ 3^{1000} & \text{mod } 179359 & = & 31981 \\ 3^{1001} & \text{mod } 179359 & = & 95943 \\ 3^{1002} & \text{mod } 179359 & = & 108470 \\ & \vdots & & \\ 3^{14716} & \text{mod } 179359 & = & 99644 \\ 3^{14717} & \text{mod } 179359 & = & 119573 \\ 3^{14718} & \text{mod } 179359 & = & 1. \end{array}$$

Thus, the order r of 3 in the multiplicative group $\mathcal{G} = n(\mathbb{Z}/179359\mathbb{Z})^*$ is 14718, that is, $\text{ord}_{179359}(3) = 14718$.

Example 10.6 Let

$$\begin{aligned} n &= 5515596313 \\ e &= 1757316971 \\ C &= 763222127 \\ r &= \text{order}(C, n) = 114905160 \end{aligned}$$

Then

$$\begin{aligned} M &\equiv C^{1/e \bmod r} \pmod{n} \\ &\equiv 763222127^{1/1757316971 \bmod 114905160} \pmod{5515596313} \\ &\equiv 1612050119 \end{aligned}$$

Clearly, this result is correct, since

$$\begin{aligned} M^e &\equiv 1612050119^{1757316971} \\ &\equiv 763222127 \\ &\equiv C \pmod{5515596313} \end{aligned}$$

It must be noted, however, that in practice, the above computation for finding the order of $x \in (\mathbb{Z}/n\mathbb{Z})^*$ may not work, since for an element x in a large group $\mathcal{G}$ with n having more than 200 digits, the computation of r may require more than 10^{150} multiplications. Even if these multiplications could be carried out at the rate of 1000 billion per second on a supercomputer, it would take approximately $3 \cdot 10^{80}$ years to arrive at the answer. There is however a "quick" way to find the order of an element x in the multiplicative group $\mathcal{G}$ modulo n if the order $|\mathcal{G}|$ (where $|\mathcal{G}| = |(\mathbb{Z}/N\mathbb{Z})^*| = \phi(n)$) of $\mathcal{G}$ as well as the prime factorization of $|\mathcal{G}|$ are known, since, by Lagrange's theorem, $r = \text{ord}_n(x)$ is a divisor of $|\mathcal{G}|$. Of course, as we know, the number $\lambda(n)$ is the largest possible order of an element x in the group $\mathcal{G}$. So, once we have the value of $\lambda(n)$, it is relatively easy to find $\text{ord}_n(x)$, the order of the element $x \in \mathcal{G}$. For example, let $n = 179359$, then $\lambda(179359) = 29436$. Therefore, $\text{ord}_{179359}(3) \leq 29436$. In fact, $\text{ord}_{179359}(3) = 14718$, which of course is a divisor of 29436. However, there are no efficient algorithms at present for calculating either $\phi(n)$ or $\lambda(n)$. Therefore, the "quick" ways for computing $\text{ord}_n(x)$ by either $\phi(n)$ or $\lambda(n)$ are essentially useless in practice. This partly explains why integer factorization is difficult. Fortunately, Shor [9] discovered in 1994 an efficient quantum algorithm to find the order of an element $x \in (\mathbb{Z}/n\mathbb{Z})^*$ and hence a quick way to factor n. The main idea of Shor's method is as follows [56].

First of all, we create two quantum registers for our quantum computer: Register-1 and Register-2. Of course, we can create just one single quantum memory register partitioned into two parts. Second, we create in Register-1 a superposition of the integers $a = 0, 1, 2, 3, \cdots$ which will be the arguments of $f(a) = x^a \pmod{n}$, and load Register-2 with all zeros. Third, we compute in Register-2, $f(a) = x^a \pmod{n}$ for each input a. (Since the values of a are

kept in Register-1, this can be done reversibly.) Fourth, we perform the discrete Fourier transform on Register-1. Finally we observe both registers of the machine and find the order r that satisfies $x^r \equiv 1 \pmod{n}$. The following is a brief description of the quantum algorithm for the order finding problem.

Algorithm 10.1 (Quantum order finding attack) Given RSA ciphertext C and modulus n, this attack will first find the order, r, of C in $\mathbb{Z}_n^*$, such that $C^r \equiv 1 \pmod{n}$, then recover the plaintext M from the ciphertext C. Assume the quantum computer has two quantum registers: Register-1 and Register-2, which hold integers in binary form.

[1] (Initialization) Find a number q, a power of 2, say 2^t, with $n^2 < q < 2n^2$.

[2] (Preparation for quantum registers) Put in the first t-qubit register, Register-1, the uniform superposition of states representing numbers $a \pmod{q}$, and load Register-2 with all zeros. This leaves the machine in the state $|\Psi_1\rangle$:

$$|\Psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,|0\rangle\,. \tag{10.5}$$

(Note that the joint states of both registers are represented by $|\mathsf{Register\text{-}1}\rangle$ and $|\mathsf{Register\text{-}2}\rangle$). What this step does is put each qubit in Register-1 into the superposition

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

[3] (Power creation) Fill in the second t-qubit register, Register-2, with powers $C^a \pmod{n}$. This leaves the machine in state $|\Psi_2\rangle$:

$$|\Psi_2\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,\big|\,C^a \pmod{n}\big\rangle. \tag{10.6}$$

This step can be done reversibly since all the a's were kept in Register-1.

[4] (Perform a quantum FFT) Apply FFT on Register-1. The FFT maps each state $|a\rangle$ to

$$\frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \exp(2\pi\, iac/q)\,|c\rangle\,. \tag{10.7}$$

That is, we apply the unitary matrix with the (a, c) entry equal to $\frac{1}{\sqrt{q}} \exp(2\pi\, iac/q)$. This leaves the machine in the state $|\Psi_3\rangle$:

$$|\Psi_3\rangle = \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \exp(2\pi\, iac/q)\,|c\rangle\,\big|\,C^a \pmod{n}\big\rangle. \tag{10.8}$$

[5] (Periodicity detection in C^a) Observe both $|c\rangle$ in Register-1 and $|C^a \pmod n\rangle$ in Register-2 of the machine, measure both arguments of this superposition, obtaining the values of $|c\rangle$ in the first argument and some $|C^k \pmod n\rangle$ as the answer for the second one ($0 < k < r$).

[6] (Extract r) Extract the required value of r. Given the pure state $|\Psi_3\rangle$, the probabilities of different results for this measurement will be given by the probability distribution:

$$\begin{aligned}\text{Prob}(c, C^k \pmod n) &= \left| \frac{1}{q} \sum_{\substack{a=0 \\ C^a \equiv a^k \,(\bmod\, N)}}^{q-1} \exp(2\pi i ac/q) \right|^2 \\ &= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i (br+k)c/q) \right|^2 \\ &= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i b\{rc\}/q) \right|^2 \end{aligned} \tag{10.9}$$

where $\{rc\}$ is $rc \mod N$. Since

$$\begin{aligned}\frac{-r}{2} \le \{rc\} \le \frac{-r}{2} &\Longrightarrow \frac{-r}{2} \le rc - dq \le \frac{-r}{2}, \text{ for some } d \\ &\Longrightarrow \text{Prob}(c, C^k \pmod n) > \frac{1}{3r^2}. \end{aligned} \tag{10.10}$$

then we have

$$\left| \frac{c}{q} - \frac{d}{r} \right| \le \frac{1}{2q}. \tag{10.11}$$

Since $\frac{c}{q}$ were known, r can be obtained by the continued fraction expansion of $\frac{c}{q}$.

[7] (Code breaking) Once the order r, $r = \text{order}(C, n)$, is found, then compute:

$$M \equiv C^{1/e \bmod r} \pmod n. \tag{10.12}$$

Hence, this decodes the RSA code C.

Theorem 10.4 (Complexity of quantum order finding attack) *Quantum order attack can find* order(C, n) *and recover M from C in time* $\mathcal{O}((\log n)^{2+\epsilon})$.

Remark 10.1 This quantum attack is for particular RSA ciphertexts C. In this special case, the factorization of the RSA modulus n is not needed. In the next section, we shall consider the more general quantum attack by factoring n.

Problems for Section 10.1

1. Let $n = pq$ be an odd integer with p and q distinct prime factors. Show that for a randomly chosen number $x \in (\mathbb{Z}/n\mathbb{Z})^*$ with multiplicative order r modulo n, the probability that r is even and $x^{r/2} \not\equiv -1 \pmod{n}$ is at least $1/2$.
2. Let $n > 1$ be an odd integer with k distinct prime factors. Show that for a randomly chosen number $x \in (\mathbb{Z}/n\mathbb{Z})^*$ with multiplicative order r modulo n, the probability that r is even and $x^{r/2} \not\equiv -1 \pmod{n}$ is at least $1 - 1/2^{k-1}$.
3. Let $n = pq$ with p and q prime. Use the Chinese Remainder theorem to show that with probability at least $3/4$, the order r of a modulo n is even. If r is even, show the probability that $x^{r/2} \equiv \pm 1 \pmod{n}$ is at most $1/2$.
4. Griffiths and Niu in 1996 [1] proposed a network of only one-qubit gates for performing quantum Fourier transforms. Construct a quantum circuit to perform the quantum Fourier transforms on two and four qubits, respectively.
5. Let u/v and s/t be two distinct rational numbers, with $0 < v, t < d$. Show that

$$\left|\frac{u}{v} - \frac{s}{t}\right| > \frac{1}{d^2}.$$

6. Show that

$$\left|\sum_{q=1^s} \exp\left(\frac{2\pi\, ipqr}{m}\right)\right|^2 = \frac{\sin^2 \frac{\pi pr(s+1)}{m}}{\sin^2 \frac{\pi pr}{m}}.$$

10.2 Quantum Algorithms for Integer Factorization

The previous section introduced a quantum algorithm for group order finding in order to break an IFP-based, particularly RSA, cryptosystem. Instead of finding the order of the ciphertext C in $\mathbb{Z}_n^*$, one can take this further to a more general case: Find the order of an element x in $\mathbb{Z}_n^*$, denoted by order(x, n), where n is the RSA modulus. Once the order of an element x in $\mathbb{Z}_n^*$ is found, and it is even, there will be a good chance to factor n, of course in polynomial-time, by just calculating

$$\left\{\gcd(x^{r/2} + 1, n),\ \ \gcd(x^{r/2} - 1, n)\right\}.$$

For instance, for $x = 3$, $r = 14718$ and $n = 179359$, we have

$$\left\{\gcd(3^{14718/2} + 1, 179359),\ \ \gcd(3^{14718/2} - 1, 179359)\right\} = (67, 2677),$$

and hence the factorization of n:

$$n = 179359 = 67 \cdot 2677.$$

The following theorem shows that the probability for r to work is high.

Theorem 10.5 *Let the odd integer $n > 1$ have exactly k distinct prime factors. For a randomly chosen $x \in \mathbb{Z}_n^*$ with multiplicative order r, the probability that r is even and that*

$$x^{r/2} \not\equiv -1 \pmod{n} \tag{10.13}$$

is least $1 - 1/2^{k-1}$. More specifically, if n has just two prime factors (this is often the case for the RSA modulus n), then the probability is at least $1/2$.

Algorithm 10.2 (Quantum algorithm for integer factorization) Given integers x and N, the algorithm will

- find the order of x, that is, the smallest positive integer r such that

$$x^r \equiv 1 \pmod{n},$$

- find the prime factors of N and compute the decryption exponent d,
- decode the RSA message.

Assume the machine has two quantum registers: Register-1 and Register-2, which hold integers in binary form.

[1] (Initialization) Find a number q, a power of 2, say 2^t, with $n^2 < q < 2n^2$.
[2] (Preparation for quantum registers) Put in the first t-qubit register, Register-1, the uniform superposition of states representing numbers $a \pmod{q}$, and load Register-2 with all zeros. This leaves the machine in the state $|\Psi_1\rangle$:

$$|\Psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,|0\rangle\,. \tag{10.14}$$

(Note that the joint state of both registers are represented by $|\mathsf{Register\text{-}1}\rangle$ and $|\mathsf{Register\text{-}2}\rangle$). What this step does is put each qubit in Register-1 into the superposition

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

[3] (Base selection) Choose a random $x \in [2, n-2]$ such that $\gcd(x, n) = 1$.
[4] (Power creation) Fill in the second t-qubit register, Register-2, with powers $x^a \pmod{n}$. This leaves the machine in state $|\Psi_2\rangle$:

$$|\Psi_2\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,|x^a \pmod{n}\rangle. \tag{10.15}$$

This step can be done reversibly since all the a's were kept in Register-1.

[5] (Perform a quantum FFT) Apply FFT on Register-1. The FFT maps each state $|\, a\rangle$ to

$$\frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \exp(2\pi\, iac/q)\,|\, c\rangle\,.$$

That is, we apply the unitary matrix with the (a, c) entry equal to $\frac{1}{\sqrt{q}} \exp(2\pi\, iac/q)$. This leaves the machine in the state $|\, \Psi_3\rangle$:

$$|\, \Psi_3\rangle = \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \exp(2\pi\, iac/q)\,|\, c\rangle\,\left|\, x^a \,(\mathrm{mod}\, n)\right\rangle. \tag{10.16}$$

[6] (Periodicity detection in x^a) Observe both $|\, c\rangle$ in Register-1 and $|\, x^a \,(\mathrm{mod}\, n)\rangle$ in Register-2 of the machine, measure both arguments of this superposition, obtaining the values of $|\, c\rangle$ in the first argument and some $|\, x^k \,(\mathrm{mod}\, n)\rangle$ as the answer for the second one $(0 < k < r)$.

[7] (Extract r) Extract the required value of r. Given the pure state $|\, \Psi_3\rangle$, the probabilities of different results for this measurement will be given by the probability distribution:

$$\begin{aligned}
\mathrm{Prob}(c, x^k \,(\mathrm{mod}\, n)) &= \left| \frac{1}{q} \sum_{\substack{a=0 \\ x^a \equiv a^k \,(\mathrm{mod}\ N)}}^{q-1} \exp(2\pi i ac/q) \right|^2 \\
&= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i (br + k)c/q) \right|^2 \\
&= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i b\{rc\}/q) \right|^2
\end{aligned} \tag{10.17}$$

where $\{rc\}$ is $rc \bmod n$. Since

$$\begin{aligned}
\frac{-r}{2} \le \{rc\} \le \frac{-r}{2} &\Longrightarrow \frac{-r}{2} \le rc - dq \le \frac{-r}{2}, \text{ for some } d \\
&\Longrightarrow \mathrm{Prob}(c, x^k \,(\mathrm{mod}\, n)) > \frac{1}{3r^2}.
\end{aligned}$$

then we have

$$\left| \frac{c}{q} - \frac{d}{r} \right| \le \frac{1}{2q}. \tag{10.18}$$

Since $\frac{c}{q}$ were known, r can be obtained by the continued fraction expansion of $\frac{c}{q}$.

[8] (Resolution) If r is odd, go to Step [3] to start a new base. If r is even, then try to compute. Once r is found, the factors of n can be found *possibly*

$$\{\gcd(x^{r/2} - 1, n),\ \gcd(x^{r/2} + 1, n)\} \tag{10.19}$$

Hopefully, this will produce two factors of n.

[9] (Computing d) Once N is factored and p and q are found, then compute

$$d \equiv 1/e \,(\mathrm{mod}\,(p-1)(q-1)). \tag{10.20}$$

[10] (Code break) As soon as d is found, RSA ciphertext encrypted by the public key (e, n), can be decrypted by this d as follows:

$$M \equiv C^d \,(\mathrm{mod}\, n). \tag{10.21}$$

Theorem 10.6 (Complexity of quantum factoring) *The quantum factoring algorithm can factor the RSA modulus N and break the RSA system in time $\mathcal{O}((\log n)^{2+\epsilon})$.*

Remark 10.2 The quantum fatoring attack discussed in Algorithm 10.2 is more general than that in Algorithm 10.1. Algorithm 10.2 also implies that if a practical quantum computer can be built, then the RSA cryptosystem can be completely broken, and a quantum resistent cryptosystem must be developed and used to replace the RSA cryptosystem.

Example 10.7 On 19 December 2001, IBM made the first demonstration of the quantum factoring algorithm [2], that correctly identified 3 and 5 as the factors of 15. Although the answer may appear to be trivial, it may have good potential and practical implication. In this example, we show how to factor 15 quantum-mechanically.

[1] Choose at random $x = 7$ such that $\gcd(x, N) = 1$. We aim to find $r = \mathrm{order}_{15}7$ such that $7^r \equiv 1 \,(\mathrm{mod}\, 15)$.

[2] Initialize two 4-qubit registers to state 0:

$$|\,\Psi_0\rangle = |\,0\rangle\,|\,0\rangle\,.$$

[3] Randomize the first register as follows:

$$|\,\Psi_0\rangle \to |\,\Psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |\,k\rangle\,|\,0\rangle\,.$$

[4] Unitarily compute the function $f(a) \equiv 13^a \pmod{15}$ as follows:

$$\begin{aligned} |\Psi_1\rangle \to |\Psi_2\rangle &= \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle \left| 13^k \pmod{15} \right\rangle \\ &= \frac{1}{\sqrt{2^t}} [|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + \\ &\qquad |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + |7\rangle|13\rangle + \\ &\qquad |8\rangle|1\rangle + |9\rangle|7\rangle + |10\rangle|4\rangle + |11\rangle|13\rangle + \\ &\qquad + \cdots] \end{aligned}$$

[5] We now apply the FFT to the second register and measure it (it can be done in the first), obtaining a random result from 1, 7, 4, 13. Suppose we incidently get 4, then the state input to FFT would be

$$\sqrt{\frac{4}{2^t}} [|2\rangle + |6\rangle + |10\rangle + |14\rangle + \cdots] .$$

After applying FFT, some state

$$\sum_{\lambda} \alpha_\lambda |\lambda\rangle$$

with the probability distribution for $q = 2^t = 2048$. The final measurement gives 0, 512, 1024, 2048, each with probability almost exactly $1/4$. Suppose that $\lambda = 1536$ was obtained from the measurement. Then we compute the continued fraction expansion

$$\frac{\lambda}{q} = \frac{1536}{2048} = \frac{1}{1 + \frac{1}{3}}, \quad \text{with convergents} \left[0, 1, \frac{3}{4}, \right]$$

Thus, $r = 4 = \text{order}_{15}(7)$. Therefore,

$$\gcd(x^{r/2} \pm 1, N) = \gcd(7^2 \pm 1, 15) = (5, 3).$$

Remark 10.3 Quantum factoring is still in its very earliest stages and will not threaten the security of RSA at least at present, as the current quantum computer can only factor a number with just 2 digits, such as 15, which is essentially hopeless.

Problems for Section 10.2

1. Show that if in Shor's factoring algorithm, we have

$$\left|\frac{c}{2^m} - \frac{d}{r}\right| < \frac{1}{2n^2}$$

and

$$\left|\frac{c}{2^m} - \frac{d_1}{r_1}\right| < \frac{1}{2n^2},$$

then

$$\frac{d}{r} = \frac{d_1}{r_1}.$$

2. There are currently many pseudo-simulations of Shor's quantum factoring algorithm; for example, the paper by Schneiderman, Stanley, and Aravind [3] gives one of the simulations in Maple, whereas Browne [15] presents an efficient classical simulation of the quantum Fourier transform based on [3]. Construct your own Java (C/C++, Mathematica, or Maple) program to simulate Shor's quantum factoring algorithm and discrete logarithm algorithm.
3. Show that in case $r \nmid 2^n$, Shor's factoring algorithm [4] needs to be repeated only $\mathcal{O}(\log\log r)$ steps in order to achieve a high probability of success.
4. The ECM (Elliptic Curve Method) factoring algorithm is very well suited to parallel implementation. Give a quantum parallel implementation of the ECM method.
5. The NFS (Number Field Sieve) factoring algorithm is also very well suited to parallel implementation. Give a quantum parallel implementation of the NFS method.

10.3 Quantum Algorithms for Discrete Logarithms

The quantum algorithms for order finding and factoring can be used, with some small modifications, to solve the Discrete Logarithm Problem (DLP) efficiently in polynomial-time. In this section, we give a brief description of a quantum algorithm for DLP.

Algorithm 10.3 (Quantum algorithm for discrete logarithms) Given $g, x \in \mathbb{F}_p$ with p prime, this algorithm will find the integer r

$$r \equiv \log_g x \,(\mathrm{mod}\ p) \tag{10.22}$$

such that

$$g^r \equiv x \,(\mathrm{mod}\ p) \tag{10.23}$$

if r exists. The algorithm uses three quantum registers.

[0] Initializing three required quantum registers as follows:

$$|\,\Psi_0\rangle = |\,0,\ 0,\ 0\rangle\,. \tag{10.24}$$

[1] Find q, a power of 2, such that q is close to p, that is, $p < q < 2p$.

[2] Put in the first two registers of the quantum computer the uniform superposition of all $|\,a\rangle$ and $|\,b\rangle \pmod{p-1}$, and compute $g^a x^{-b} \pmod{p}$ in the third register. This leaves the quantum computer in the state $|\,\Psi_1\rangle$:

$$\frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}\left|\,a,\ b,\ g^a x^{-b} \pmod{p}\right\rangle. \tag{10.25}$$

[3] Use the Fourier transform A_q to map $|\,a\rangle \to |\,c\rangle$ and $|\,b\rangle \to |\,d\rangle$ with probability amplitude

$$\frac{1}{q}\exp\left(\frac{2\pi i}{q}(ac+bd)\right).$$

Thus, the state $|\,a, b\rangle$ will be changed to the state:

$$\frac{1}{q}\sum_{c=0}^{q-1}\sum_{d=0}^{q-1}\exp\left(\frac{2\pi i}{q}(ac+bd)\right)|\,c,\ d\rangle\,. \tag{10.26}$$

This leaves the machine in the state $|\,\Psi_2\rangle$:

$$\frac{1}{(p-1)q}\sum_{a,b=0}^{p-2}\sum_{c,d=0}^{q-1}\exp\left(\frac{2\pi i}{q}(ac+bd)\right)\left|\,c,\ d,\ g^a x^{-b} \pmod{p}\right\rangle. \tag{10.27}$$

[4] Observe the state of the quantum computer and extract the required information. The probability of observing a state $|\,c,\ d,\ g^k \pmod{p}\rangle$ is

$$\left|\frac{1}{(p-1)q}\sum_{\substack{a,b\\ a-rb\equiv k \pmod{p-1}}}\exp\left(\frac{2\pi i}{q}(ac+bd)\right)\right|^2 \tag{10.28}$$

where the sum is over all (a, b) such that

$$a - rb \equiv k \pmod{p-1}. \tag{10.29}$$

[5] Use the relation

$$a = rb + k - (p-1)\left\lfloor\frac{br+k}{p-1}\right\rfloor. \tag{10.30}$$

to substitute in (10.28) to get the amplitude on $|\, c, d, g^k \pmod p\rangle$:

$$\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\frac{2\pi i}{q}\left(brc+kc+bd-c(p-1)\left\lfloor\frac{br+k}{p-1}\right\rfloor\right)\right). \quad (10.31)$$

This finally leaves the machine in the state $|\,\Psi_3\rangle$:

$$\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\frac{2\pi i}{q}\left(brc+kc+bd-c(p-1)\left\lfloor\frac{br+k}{p-1}\right\rfloor\right)\right)\Big|\, c, d, g^k \pmod p\Big\rangle. \quad (10.32)$$

The probability of observing the above state $|\, c, d, g^k \pmod p\rangle$ is thus:

$$\left|\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\frac{2\pi i}{q}\left(brc+kc+bd-c(p-1)\left\lfloor\frac{br+k}{p-1}\right\rfloor\right)\right)\right|^2. \quad (10.33)$$

Since $\exp(2\pi ikc/q)$ does not change the probability, (10.50) can be rewrite algebraically as follows:

$$\left|\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\frac{2\pi i}{q}bT\right)\exp\left(\frac{2\pi i}{q}V\right)\right|^2, \quad (10.34)$$

where

$$T = rc + d - \frac{r}{p-1}\{c(p-1)\}_q, \quad (10.35)$$

$$V = \left(\frac{br}{p-1} - \left\lfloor\frac{br+k}{p-1}\right\rfloor\right)\{c(p-1)\}_q. \quad (10.36)$$

The notation $\{z\}_q$ here denotes $z \bmod q$ with $-q/2 < \{z\}_q < q/2$.

[6] Finally, deduce r from (c, d). Let j be the closest integer to T/q and $b \in [0,\, p-2]$, then

$$|\{T\}_q| = |rc + d - \frac{r}{p-1}\{c(p-1)\}_q - jq| \le \frac{1}{2}. \quad (10.37)$$

Further, if

$$|\{c(p-1)\}_q| \le \frac{q}{12}, \quad (10.38)$$

then

$$|V| \leq \frac{q}{12}. \tag{10.39}$$

Therefore, given (c, d), r can be easily calculated with a high probability.

Remark 10.4 When p is smooth, there is an easy version of the DLP over $\mathbb{F}_p$, which can be solved in polynomial-time on a classical computer. The case here is the general one, for which the DLP is over $\mathbb{F}_q$ with $p \leq q \leq 2p$.

Remark 10.5 The quantum discrete logarithm algorithm discussed in this section can also be used to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial-time, which shall be discussed in the next section.

Problems for Section 10.3

1. Pollard's ρ Method for DLP (Discrete Logarithm Problem) admits parallelism, and in fact there are some parallel versions of the ρ Method for DLP. Develop a quantum version of the ρ method for DLP.
2. Pollard's λ Method for DLP is also very well suited to parallel implementation. Give a quantum implementation of the λ Method for DLP.
3. Develop a new quantum algorithm for DLP, based on an idea different from Shor's algorithm or Pollard's algorithms or any other known algorithms.

10.4 Quantum Algorithms for Elliptic Curve Discrete Logarithms

Shor's quantum algorithms for integer factorization and discrete logarithms can also be used to solve the elliptic curve discrete logarithms in $\mathcal{BQP}$. First, we introduce an quantum algorithm, due to Proos and Zalka in 2003 [5], to solve the ECDLP over $\mathbb{F}_p$ with p prime.

Algorithm 10.4 (Quantum algorithm for ECDLP in $\mathbb{F}_p$). The quantum algorithm tries to find

$$r \equiv \log_P Q \,(\mathrm{mod}\ p) \tag{10.40}$$

such that

$$Q \equiv rP \,(\mathrm{mod}\ p), \tag{10.41}$$

where $P, Q \in E(\mathbb{F}_p)$, and $E : y^2 \equiv x^2 + ax + b \pmod{p}$, for $p > 3$.

[0] Initialize three required quantum registers as follows:

$$|\Psi_0\rangle = |0,\ 0,\ 0\rangle\,. \tag{10.42}$$

[1] Write $q = 2^m$, a power of 2, such that q is close to p.

[2] Put in the first two registers of the quantum computer the uniform superposition of all $|a\rangle$ and $|b\rangle \pmod{p-1}$, and compute $aP + bQ \pmod{p}$ in the third register. This leaves the quantum computer in the state $|\Psi_1\rangle$:

$$\frac{1}{2^n} \sum_{a=0}^{2^n-1} \sum_{b=0}^{2^n-1} |a,\ b,\ aP + bQ \pmod{p}\rangle \tag{10.43}$$

Note that $aP + bQ \pmod{p}$ can be done classically as follows:

$$\frac{1}{2^n} \sum_{a=0}^{2^n-1} \sum_{b=0}^{2^n-1} \left| a,\ b,\ \sum a_i P_i + \sum b_i Q_i \pmod{p} \right\rangle, \tag{10.44}$$

where $a = \sum_i a_i 2^i,\ b = \sum_i y_i 2^i$, and $P_i = 2^i P$, $Q_i = 2^i Q$.

[3] Use the Fourier transform A_q to map $|a\rangle \to |c\rangle$ and $|b\rangle \to |d\rangle$ with probability amplitude

$$\frac{1}{q} \exp\left(\frac{2\pi i}{q}(ac + bd)\right).$$

Thus, the state $|a, b\rangle$ will be changed to the state:

$$\frac{1}{q} \sum_{c=0}^{q-1} \sum_{d=0}^{q-1} \exp\left(\frac{2\pi i}{q}(ac + bd)\right) |c,\ d\rangle\,. \tag{10.45}$$

This leaves the machine in the state $|\Psi_2\rangle$:

$$\frac{1}{(p-1)q} \sum_{a,b=0}^{p-2} \sum_{c,d=0}^{q-1} \exp\left(\frac{2\pi i}{q}(ac + bd)\right) |c,\ d,\ aP + bQ \pmod{p}\rangle\,. \tag{10.46}$$

[4] Observe the state of the quantum computer and extract the required information. The probability of observing a state $|c,\ d,\ kP \pmod{p}\rangle$ is

$$\left| \frac{1}{(p-1)q} \sum_{\substack{a,b \\ a - rb \equiv k \pmod{p-1}}} \exp\left(\frac{2\pi i}{q}(ac + bd)\right) \right|^2 \tag{10.47}$$

where the sum is over all (a, b) such that

$$aP + bQ \equiv kP \pmod{p-1}. \tag{10.48}$$

[5] Use the relation

$$a = rb + k - (p-1)\left\lfloor \frac{br+k}{p-1} \right\rfloor. \tag{10.49}$$

to substitute in (10.47) to get the amplitude on $|\, c, d, kP \pmod{p}\rangle$:

$$\frac{1}{(p-1)q} \sum_{b=0}^{p-2} \exp\left(\frac{2\pi i}{q}\left(brc + kc + bd - c(p-1)\left\lfloor \frac{br+k}{p-1} \right\rfloor\right)\right). \tag{10.50}$$

This finally leaves the machine in the state $|\, \Psi_3\rangle$:

$$\frac{1}{(p-1)q} \sum_{b=0}^{p-2} \exp\left(\frac{2\pi i}{q}\left(brc + kc + bd - c(p-1)\left\lfloor \frac{br+k}{p-1} \right\rfloor\right)\right) |\, c, d, kP \pmod{p}\rangle\,. \tag{10.51}$$

The probability of observing the above state $|\, c, d, kP \pmod{p}\rangle$ is thus:

$$\left| \frac{1}{(p-1)q} \sum_{b=0}^{p-2} \exp\left(\frac{2\pi i}{q}\left(brc + kc + bd - c(p-1)\left\lfloor \frac{br+k}{p-1} \right\rfloor\right)\right)\right|^2. \tag{10.52}$$

Since $\exp(2\pi ikc/q)$ does not change the probability, (10.50) can be rewrite algebraically as follows:

$$\left| \frac{1}{(p-1)q} \sum_{b=0}^{p-2} \exp\left(\frac{2\pi i}{q} bT\right) \exp\left(\frac{2\pi i}{q} V\right)\right|^2, \tag{10.53}$$

where

$$T = rc + d - \frac{r}{p-1}\{c(p-1)\}_q, \tag{10.54}$$

$$V = \left(\frac{br}{p-1} - \left\lfloor \frac{br+k}{p-1} \right\rfloor\right)\{c(p-1)\}_q. \tag{10.55}$$

The notation $\{z\}_q$ here denotes $z \bmod q$ with $-q/2 < \{z\}_q < q/2$.

[6] Finally, deduce r from (c, d). Let j be the closest integer to T/q and $b \in [0,\ p-2]$, then

$$|\{T\}_q| = |rc + d - \frac{r}{p-1}\{c(p-1)\}_q - jq| \leq \frac{1}{2}. \tag{10.56}$$

Further, if

$$|\{c(p-1)\}_q| \leq \frac{q}{12}, \tag{10.57}$$

then

$$|V| \leq \frac{q}{12}. \tag{10.58}$$

Therefore, given (c, d), r can be easily calculated with a high probability.

Remark 10.6 The quantum ECDLP for $E(\mathbb{F}_p)$ algorithm is essentially that for DLP over $\mathbb{F}_p$.

Remark 10.7 Cheung et al. in 2008 [6] considered an optimization of the quantum algorithm for the elliptic curve discrete logarithm problem over $\mathbb{F}_{2^n}$. They first constructed a quantum state

$$|\,\Psi\rangle = \frac{1}{2^n}\sum_{a=0}^{2^n-1}\sum_{b=0}^{2^n-1}|\,a,\ b,\ aP + bQ\,(\mathrm{mod}\ p)\rangle \tag{10.59}$$

then performed a two-dimensional quantum Fourier transform over the first two registers. The point addition for aP and bQ can be done classically by the standard "double-addition" method for each 2^P and $2^i Q$ with $0 \leq i \leq n$. The multiplications over $\mathbb{F}_{2^n}$ are performed on quantum circuits (see Figure 10.2 for a particular example, but the general techniques are applicable to any primitive polynomial), with depth $\mathcal{O}(m^2)$, previously $\mathcal{O}(m^3)$.

Problems for Section 10.4

1. The fastest known algorithm for solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) in $F(\mathbb{F}_p)$ is Pollard's ρ Method, which runs in $\mathcal{O}(\sqrt{q})$ steps. Give a quantum implementation of the ρ for ECDLP.
2. Extend Proos and Zalka's quantum ECDLP algorithm [5] for $E(\mathbb{F}_p)$ to $E(\mathbb{F}_{2^m})$, or more generally to $E(\mathbb{F}_{p^m})$.
3. Propose a new quantum algorithm, based on a different idea from that of Proos and Zalka, for solving the ECDLP problem in polynomial-time.

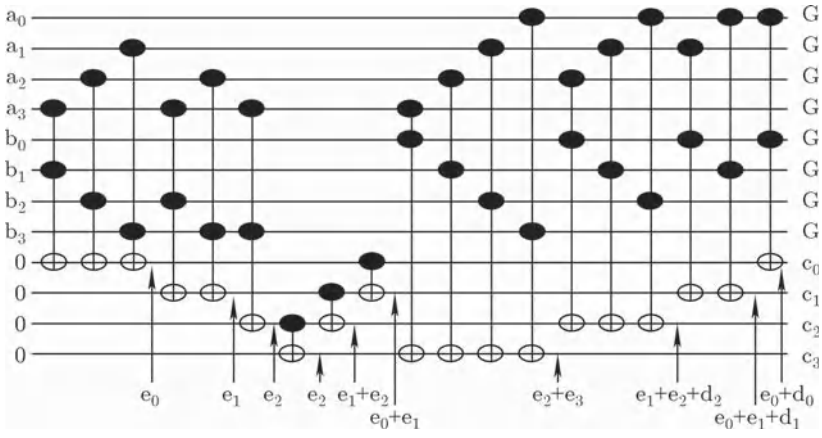


Figure 10.2 $\mathbb{F}_{2^4}$ multiplier with $P(x) = x^4 + x + 1$ (see [6])

10.5 Bibliographic Notes and Further Reading

In this chapter, we discussed quantum polynomial-time algorithms for solving the three infeasible number-theoretic problems: IFP, DLP, and ECDLP. Feynman is considered to have been the first to investigate the new quantum computing paradigm [7] and Deutsch the first to study the quantum Turing machine [8]. The world was surprised when Shor demonstrated in 1994 that the infeasible IFP and DLP can be solved in polynomial-time on a quantum computer [9]. Shor's algorithm was later extended to solve ECDLP [5]. For more information on quantum computing, particularly on quantum factoring and quantum discrete logarithms, including computability especially computability and complexity aspects of quantum computing, it is suggested that readers consult: [10–63].

References

1. R. B. Griffiths and C. Niu, "Semiclassical Fourier Transform for Quantum Computation", *Physical Review Letters*, **76**, 1996, pp. 3228–3231.
2. L. M. K. Vandersypen, M. Steffen, G. Breyta, et al. "Experiemental Realization of Shor's Quantum Factoring Algorithm Uisng Nuclear Magnetic Resonance", *Nature*, **414**, 20/27 December 2001, pp. 883–887.
3. J. F. Schneiderman, M. E. Stanley, and P. K. Aravind, "A pseudo-simulation of Shor's quantum factoring algorithm", **arXiv:quant-ph/0206101v1**, 20 pages, 2002.
4. P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1484–1509.
5. J. Proos and C. Zalka, "Shor's Discrete Logarithm Quantum Algorithm for Elliptic Curves", *Quantum Information & Computation*, **3**, 4, 2003, pp. 317–344.
6. D. Cheung and D. Maslo, et al., "On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography", *Theory of Quantum Computation, Communication, and Cryptography Third Workshop, Theory of Quantum Computing 2008*, Lecture Notes in Computer Science **5106**, Springer, 2008, pp. 96–104.
7. R. P. Feynman, "Simulating Physics with Computers", *International Journal of Theoretical Physics*, **21**, 1982, pp. 467–488.
8. D. Deutsch, "Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer", Proceedings of the Royal Society of London, Series **A400**, 1985, pp. 96–117.

9. P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1994, pp. 124–134.
10. L. M. Adleman, J. DeMarrais, and M. D. A. Huang, "Quantum Computability", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1524–1540.
11. D. Bacon and W. V. Dam, "Recent Progress in Quantum Algorithms", *Communications of the ACM*, **53**, 5, 2010, pp. 84–93.
12. C. H. Bennett and E. Bernstein, et al., "Strengths and Weakness of Quantum Computing", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1510–1523.
13. C. H. Bennett and D. P. DiVincenzo, "Quantum Information and Computation", *Nature*, **404**, 2000, pp. 247–255.
14. E. Bernstein and U. Vazirani, "Quantum Complexity Theory", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1411–1473.
15. D. E. Browne, "Efficient Classical Simulation of the Quantum Fourier Transform", *New Journal of Physics*, **9**, 146, 2007, pp. 1–7.
16. A. Childs and W. Van Dam, "Quantum Algorithms for Algebraic Problems", *Reviews of Modern Physics*, **82**, 1, 2010, pp. 1–52.
17. I. L Chuang, R. Laflamme, P, Shor, and W. H. Zurek, "Quantum Computers, Factoring, and Decoherence", *Science*, **270**, 1995, pp. 1633–1635.
18. R. Crandall and C. Pomerance, *Prime Numbers – A Computational Perspective*, 2nd Edition, Springer-Verlag, 2005. Desurvire:2009,
19. E. Desurvire, *Classical and Quantum Information Theory*, Cambridge University Press, 2009.
20. D. Deutsch, "Quantum Computational Networks", Proceedings of the Royal Society of London, Series **A425**, 1989, pp. 73–90.
21. J. Eicher and Y. Opoku, *Using the Quatum Computer to Break Elliptic Curve Cryptosystems*, University of Richmond, VA 23173, 1997.
22. A. Ekert and R. Jozsa, "Quantum Computation and Shor's Factoring Algorithm", *SIAM Journal on Computing*, **26**, 5, 1997, pp. 1510–1523.
23. R. P. Feynman, *Feynman Lectures on Computation*. Edited by A. J. G. Hey and R. W. Allen, Addison-Wesley, 1996.
24. Grustka, J. *Quantum Computing*, McGraw-Hill, 1999.
25. M. Hirvensalo, *Quantum Computing*, 2nd Edition, Springer, 2004.
26. R. Jain and Z. Ji, and S. Upahdhyay et al. "QIP = PSPACE", *Communications of the ACM*, **53**, 12(2010), pp. 102–109.
27. P. Kaye, *Techniques for Quantum Computing*, PhD Thesis, University of Waterloo, 2007, 151 pages.
28. P. Kaye and C. Zalka, "Optimized Quantum Implementation of Elliptic Curve Arithmetic over Binary Fields", University of Waterloo, 25 June 2006.
29. A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation*, American Mathematical Society, 2002.
30. E. Knill and R. Laflamme, A. Ashihkmin et al., "From factoring to Phase Estimation: A Discussion of Shor's Algorithm", Las Alamos Science, **27**, 2002, pp. 38–45. 25 June 2006.
31. B. P. Lanyou and T. J. Weinhold, et al., "Experimental Demonstration of a Compiled version of Shor's Algorithm' with Quantum Entanglement', *Physical Review letters*, **99**, 250504, 2007, 4 pages.
32. M. Le Bellac, *Quantum Information and Quantum Computation*, Cambridge University Press, 2005.
33. S. J. Lomonaco, Jr., "Shor's Quantum Factoring Algorithm", *AMS Proceedings of Symposium in Applied Mathematics*, **58**, 2002, 19 pages.
34. C. Lu, D. E. Brown, T. Yang, and J. Pan, "Demonstration of a Compiled version of Shor's Quantum Factoring Algorithm", *Physical Review letters*, **99**, 250505, 2007, 4 pages.
35. D. C. Marinescu and G. M. Marinescu, *Approaching Quantum Computing*, Prentice-Hall, 2005.
36. N. D. Mermin, *Quantum Computer Science*, Cambridge University Press, 2007.
37. D. M. McMahon, *Quantum Computing Explained*, Wiley, 2008.
38. M. A. Nielson and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University Press, 2010.
39. A. O. Pittenger, *An Introduction to Quantum Computing Algorithms*, Birkhäuser, 2001.
40. E. Rieffel and W. Polak, *Quantum Computing: A Gentle Introduction*, MIT Press, 2011.
41. M. Ross, "Quantum Computing", *Communications of the ACM*, *51*, 7, 2008, pp. 12–13.

42. A. Schmidt, "Quantum Algorithm for Solving the Discrete Logarithm Problem in the Class Group of an Imaginary Quadratic Field and Security Comparison of Current Cryptosystems at the Beginning of Quantum Computer Age", *Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006*. Edited by G. Müller, Lecture Notes in Computer Science **3995**, Springer, 2006, pp. 81–493.
43. P. Shor, "Quantum Computing", *Documenta Mathematica*, Extra Volume ICM 1998, I, pp. 467–486.
44. P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Review*, **41**, 3, 1999, pp. 303–332.
45. P. Shor, "Introduction to Quantum Algorithms", *AMS Proceedings of Symposium in Applied Mathematics*, **58**, 2002, 17 pages.
46. P. Shor, "Why Haven't More Quantum Algorithms Been Found?", *Journal of the ACM*, **50**, 1, 2003, pp. 87–90.
47. J. Smith and M. Mosca, "Algorithms for Quantum Computers", *e-print quant-ph*, 2010, 48 pages.
48. D. R. Simon, "On the Power of Quantum Computation", *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, 1994, pp. 116–123. See also: *SIAM Journal in Computing*, **26**, 5, 1997, pp. 1471–1483.
49. R. Van Meter and K. M. Itoh, "Fast Quantum Modular Exponentiation", *Physical Review A*, **71**, 052320, 2005, 12 pages.
50. R. Van Meter, W. J. Munro, and K. Nemoto, "Architecture of a Quantum Milticomputer Implementing Shor's Algorithm", *Theory of Quantum Computation, Communication and Cryptography*. Edited by Y. Kawano and M. Mosca, Lecture Note in Computer Science **5106**, 2008, pp. 105–114.
51. U. V. Vazirani, "On the Power of Quantum Computation", *Philosophical Transactions of the Royal Society London*, **A356**, (1998), pp. 1759–1768.
52. U. V. Vazirani, "Fourier Transforms and Quantum Computation", *Proceedings of Theoretical Aspects of Computer Science*, 2000, pp. 208–220.
53. U. V. Vazirani, "A Survey of Quantum Complexity Theory", *AMS Proceedings of Symposium in Applied Mathematics*, **58**, 2002, 28 pages.
54. J. Watrous, "Quantum Computational Complexity", . *Encyclopedia of Complexity and System Science*, Springer, 2009, pp. 7174–7201.
55. J. Watrous, "An Introduction to Quantum Information and Quantum Circuits", *ACM SIGACT News*, **42**, 2, 2011, pp. 52–67.
56. C. P. Williams, *Explorations in Quantum Computation*, 2nd Edition, Springer, 2011.
57. C. P. Williams and S. H. Clearwater, *Ultimate Zero and One: Computing at the Quantum Frontier* Copernicus, 2000.
58. S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer-Verlag, 2002.
59. S. Y. Yan, *Cryptanalyic Attacks on RSA*, Springer, 2009.
60. S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*, Advances in Information Security **11**, 2nd Edition, Springer, 2009.
61. N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*, Cambridge University Press, 2008.
62. A. C. Yao, "Quantum Circuit Complexity", *Proceedings of Foundations of Computer Science*, IEEE Press, 1993, pp. 352–361.
63. C. Zalka, "Fast Versions of Shor's Quantum Factoring Algorithm", *LANA e-print quant-ph 9806084*, 1998, 37 pages.

11

Quantum Resistant Cryptography

Once a practical quantum computer with several thousand quantum bits can be built, all the IFP-, DLP-, and ECDLP-based cryptographic systems and protocols can be broken in polynomial-time and hence are no more secure. However, there are still many problems that cannot be solved by a quantum computer in polynomial-time. These problems form a class of quantum-computing (or quantum-attack) resistant problems, and the cryptographic systems and protocols based on these problems will be quantum-computing resistant. In this chapter, we shall introduce:

- A coding-based quantum-computing resistant cryptographic system
- A lattice-based quantum-computing resistant cryptographic system
- A cryptographic protocol based on the idea of quantum mechanics
- DNA Based Biological Cryptography.

11.1 Coding-Based Cryptography

We first introduce the most famous code-based cryptosystem, the McEliece system, invented by McEliece in 1978 [1]. One of the most important features of the McEliece system is that it has resisted cryptanalysis to date; it is even quantum computer resistant. The idea of the McEliece system is based on coding theory and its security is based on the fact that decoding an arbitrary linear code is $\mathcal{NP}$-complete.

Algorithm 11.1 (McEliece's coding-based cryptography) Suppose Bob wishes to send an encrypted message to Alice, using Alice's public key. Alice generates her public key and the corresponding private key. Bob uses her public key to encrypt his message and sends it to Alice, Alice uses her own private key to decrypt Bob's message.

[1] Key generation: Alice performs:
  [1-1] Choose integers k, n, t as common system parameters.
  [1-2] Choose a $k \times n$ generator matrix G for a binary (n, k)-linear code which can correct t errors and for which an efficient decoding algorithm exists.
  [1-3] Select a random $k \times k$ binary nonsingular matrix S.
  [1-4] Select a random $k \times k$ permutation matrix P.

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.

[1-5] Compute the $k \times n$ matrix $\widehat{G} = SGP$.
[1-6] Now $(\widehat{G}, t)$ is Alice's public key whereas (S, G, P) is Alice's private key.

[2] Encryption: Bob uses Alice's public key to encrypt his message to Alice. Bob performs:
[2-1] Obtain Alice's authentic public key $(\widehat{G}, t)$.
[2-2] Represent the message in binary string m of length k.
[2-3] Choose a random binary error vector z of length n having at most t 1's.
[2-4] Compute the binary vector $c = m\widehat{G} + z$.
[2-5] Send the ciphertext c to Alice.

[3] Decryption: Alice receives Bob's message m and uses her private key to recover c from m. Alice performs:
[3-1] Compute $\widehat{c} = cP^{-1}$, where P^{-1} is the inverse of the matrix P.
[3-2] Use the decoding algorithm for the code generated by G to decode $\widehat{c}$ to $\widehat{m}$.
[3-3] Compute $m = \widehat{m}\widehat{S}^{-1}$. This m is thus the original plaintext.

Theorem 11.1 (Correctness of McEliece's cryptosystem) *In McEliece's cryptosystem, m can be correctly recovered from c.*

Proof: Since

$$\begin{aligned}\widehat{c} &= cP^{-1} \\ &= (m\widehat{G} + z)P^{-1} \\ &= (mSGP + z)P^{-1} \\ &= (mS)G + zP^{-1}, \quad (zP^{-1} \text{ is a vector with at most } t \text{ 1's})\end{aligned}$$

the decoding algorithm for the code generated by G corrects $\widehat{c}$ to $\widehat{m} = mS$. Now applying S^{-1} to $\widehat{m}$, we get $mSS^{-1} = m$, the required original plaintext. ∎

Remark 11.1 The security of McEliece's cryptosystem is based on error-correcting codes, particularly the Goppa; if the Goppa code is replaced by other error-correcting codes, the security will be severely weakened. The McEliece's cryptosystem has two main drawbacks:

(1) the public key is very large and
(2) there is a message expansion by a factor of n/k.

It is suggested that the values for the system parameters should be $n = 1024$, $t = 50$, and $k \geq 644$. Thus for these recommended values of system parameters, the public key has about 2^{19}-bits, and the message expansion is about 1.6. For these reasons, McEliece's cryptosystem receives little attention in practice. However, as McEliece's cryptosystem is the first probabilistic encryption and, more importantly, it has resisted all cryptanalysis including quantum cryptanalysis, it may be a good candidate to replace RSA in the post-quantum cryptography age.

Problems for Section 11.1

1. Compare the main parameters (such as encryption and decryption complexity, cryptographic resistance, ease of use, secret-key size, and public-key size, etc.) of RSA and McEliece systems.
2. Show that decoding a general algebraic code is NP-complete.
3. Write an essay on all possible attacks for the McEliece coding-based cryptosystem.

11.2 Lattice-Based Cryptography

Cryptography based on ring properties and particularly lattice reduction is another promising direction for post-quantum cryptography, as lattice reduction is a reasonably well-studied hard problem that is currently not known to be solved in polynomial-time, or even subexponential-time on a quantum computer. There are many types of cryptographic systems based on lattice reduction [2–4]. In this section, we give a brief account of one if the lattice based cryptographic systems, the NTRU encryption scheme. NTRU is rumored to stand for Nth-degree TRUncated polynomial ring, or Number Theorists eRe Us. Compared with RSA, it is a rather young cryptosystem, developed by Hoffstein, Pipher, and Silverman [5] in 1995. We give a brief introduction to NTRU, more information can be found in [6].

Algorithm 11.2 (NTRU encryption scheme) The NTRU encryption scheme works as follows:

[1] Key generation:
  [1-1] Randomly generate polynomials f and g in D_f and D_g, respectively, each of the form:

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-2} x^{k-2} + a_{k-1} x^{k-1}. \tag{11.1}$$

  [1-2] Invert f in $\mathcal{R}_p$ to obtain f_p, and check that g is invertible in f_q.
  [1-3] The public key is $h \equiv p \cdot g \cdot f_q \pmod{q}$. The private key is the pair (f, f_p).
[2] Encryption:
  [2-1] Randomly select a small polynomials r in D_r.
  [2-2] Compute the ciphertext

$$c \equiv r \cdot h + m \pmod{q}. \tag{11.2}$$

[3] Decryption:
  [3-1] Compute $a = \text{center}(f \cdot c)$,
  [3-2] Recover m from c by computing $m \equiv f_p \cdot a \pmod{q}$. This is true since

$$a \equiv p \cdot r \cdot \equiv + f \cdot m \pmod{q}. \tag{11.3}$$

In Table 11.1, we present some information comparing NTRU to RSA and McEliece.

Table 11.1 Comparison among NTRU, RSA, and McEliece

	NTRU	RSA	McEliece
Encryption speed	n^2	$n^2 \approx n^3$	n^2
Decryption speed	n^2	n^3	n^2
Public key	n	n	n^2
Secret key	n	n	n^2
Message expansion	$\log_p q - 1$	$1 - 1$	$1 - 1.6$

Problems for Section 11.2

1. Give a critical analysis of the computational complexity of the NTRU cryptosystem.
2. NTRU is currently considered quantum resistant. Show that NTRU is indeed quantum resistant, or may not be quantum resistant.
3. Lattice-based cryptography is considered to be quantum resistant. However, if not designed properly, it may be broken by traditional mathematical attacks without using any quantum techniques. For example, the Cai–Cusick lattice-based cryptosystem [8] was recently cracked completely by Pan and Deng [9]. Show that the Cai–Cusick lattice-based cryptosystem can be broken in polynomial-time by classical mathematical attacks.
4. It is widely considered that multivariate public key cryptosystems (MPKC, see [7]) are quantum resistant. The usual approach to polynomial evaluation is FFT-like, whereas quantum computation makes good use of FFT to sped-up the computation. With this in mind, show that MPKC may not be quantum resistant.

11.3 Quantum Cryptography

It is evident that if a practical quantum computer is available, then all public-key cryptographic systems based on the difficulty of IFP, DLP, and ECDLP will be insecure. However, the cryptographic systems based on quantum mechanics, called *quantum cryptography*, will still be secure even if a quantum computer is available. So, quantum cryptography is a type of cryptography using quantum mechanics against quantum mechanics. In this section some basic ideas of quantum cryptography are introduced. More specifically, a quantum analog of the Diffie–Hellman–Meikle key-exchange/distribution system, proposed by Bennett and Brassard in 1984 [10], will be addressed.

First let us define four *polarizations* as follows:

$$\{0^\circ, 45^\circ, 90^\circ, 135^\circ\} \stackrel{\text{def}}{=} \{\rightarrow, \nearrow, \uparrow, \nwarrow\}. \tag{11.4}$$

The quantum system consists of a transmitter, a receiver, and a quantum channel through which polarized photons can be sent. By the law of quantum mechanics, the receiver can either distinguish between the *rectilinear polarizations* $\{\rightarrow, \uparrow\}$, or reconfigure to discriminate

between the diagonal polarizations $\{\nearrow, \nwarrow\}$, but in any case, cannot distinguish both types. The system works in the following way:

[1] Alice uses the transmitter to send Bob a sequence of photons, each of them should be in one of the four polarizations $\{\rightarrow, \nearrow, \uparrow, \nwarrow\}$. For instance, Alice could choose, at random, the following photons

$$\uparrow \quad \nearrow \quad \rightarrow \quad \nwarrow \quad \rightarrow \quad \rightarrow \quad \nearrow \quad \uparrow \quad \uparrow$$

to be sent to Bob.

[2] Bob then uses the receiver to measure the polarizations. For each photon received from Alice, Bob chooses, at random, the following type of measurements $\{+, \times\}$:

$$+ \quad + \quad \times \quad \times \quad + \quad + \quad \times \quad \times \quad \times \quad +$$

[3] Bob records the result of his measurements but keeps it secret:

$$\uparrow \quad \rightarrow \quad \nearrow \quad \nwarrow \quad \rightarrow \quad \nearrow \quad \nearrow \quad \nearrow \quad \uparrow$$

[4] Bob publicly announces the type of measurements he made, and Alice tells him which measurements were of correct type:

$$\surd \qquad \surd \quad \surd \qquad \surd \qquad \surd$$

[5] Alice and Bob keep all cases in which Bob measured the correct type. These cases are then translated into bits {0, 1} and thereby become the key:

$\uparrow$	$\nwarrow$	$\rightarrow$	$\nearrow$	$\uparrow$
1	1	0	0	1

[6] Using this secret key formed by the quantum channel, Bob and Alice can now encrypt and send their ordinary messages via the classic public-key channel.

An eavesdropper is free to try to measure the photons in the quantum channel, but, according to the law of quantum mechanics, he cannot in general do this without disturbing them, and hence, the key formed by the quantum channel is secure.

Problems for Section 11.3

1. Explain what the main features of quantum cryptography are.
2. Explain why the quantum key distribution is quantum computing resistant.
3. Use the idea explained in this section to simulate the quantum key distribution and to generate a string of 56 characters for a DES key.
4. Use the idea explained in this section to simulate the quantum key distribution and to generate a stream of 128 or 256 characters for an AES key.

11.4 DNA Biological Cryptography

The world was shocked by a paper [12] of Adleman (the "A" in the RSA) , who demonstrated that an instance of the NP-complete problem, more specifically, the Hamiltonian Path Problem (HPP), can be solved in polynomial-time on a DNA biological computer (for more information on biological computing, see for example, [13] and [14]. The fundamental idea of DNA-based biological computation is that of a set of DNA strands. Since the set of DNA strands is usually kept in a test tube, the test tube is just a collection of pieces of DNA. In what follows, we shall first give a brief introduction to the DNA biological computation.

Definition 11.1 A *test tube* (or just tube for short) is a set of molecules of DNA (i.e., a multi-set of finite strings over the alphabet $\Sigma = \{A, C, G, T\}$). Given a tube, one can perform the following four elementary biological operations:

(1) **Separate** or **Extract**: Given a tube T and a string of symbols $S \in \Sigma$, produce two tubes $+(T, S)$ and $-(T, S)$, where $+(T, S)$ is all the molecules of DNA in T which contain the consecutive subsequence S and $-(T, S)$ is all of the molecules of DNA in T which do not contain the consecutive sequence S.
(2) **Merge**: Given tubes T_1, T_2, produce the multi-set union $\cup(T_1, T_2)$:

$$\cup (T_1, T_2) = T_1 \cup T_2 \tag{11.5}$$

(3) **Detect**: Given a tube T, output "yes" if T contains at least one DNA molecule (sequence) and output "no" if it contains none.
(4) **Amplify**: Given a tube T produce two tubes $T'(T)$ and $T''(T)$ such that

$$T = T'(T) = T''(T). \tag{11.6}$$

Thus, we can replicate all the DNA molecules from the test tube.

These operations are then used to write "programs" which receive a tube as input and return either "yes" or "no" or a set of tubes.

Example 11.1 Consider the following program:

(1) Input(T)
(2) $T_1 = -(T, C)$
(3) $T_2 = -(T_1, G)$
(4) $T_3 = -(T_2, T)$
(5) Output(Detect(T_3))

The model defined above is an unrestricted one. We now present a restricted biological computation model:

Definition 11.2 A tube is a multi-set of aggregates over an alphabet Σ which is not necessarily $\{A, C, G, T\}$. (An aggregate is a subset of symbols over Σ). Given a tube, there are three operations:

(1) **Separate**: Given a tube T and a symbol $s \in \sum$, produce two tubes $+(T, s)$ and $-(T, s)$ where $+(T, s)$ is all the aggregates of T which contains the symbols s and $-(T, s)$ is all of the aggregates of T which do not contain the symbol s.
(2) **Merge**: Given tube T_1, T_2, produce

$$\cup (T_1, T_2) = T_1 \cup T_2 \tag{11.7}$$

(3) **Detect**: Given a tube T, output "yes" if T contains at least one aggregate, or output "no" if it contains none.

Example 11.2 (3-colorability problem) Given an n vertex graph G with edges $e_1, e_2, \cdots, e_z$, let

$$\Sigma = \{r_1, b_1, g_1, r_2, b_2, g_2, \cdots, r_n, b_n, g_n\}.$$

and consider the following restricted program on input

$$\begin{aligned} T = \{\alpha | \alpha \subseteq \Sigma \\ \alpha = \{c_1, c_2, \cdots, c_n\} \\ [c_i = r_i \, or \, c_i = b_i \, or \, c_i = g_i], i = 1, 2, \cdots, n\} \end{aligned}$$

(1) Input(T).
(2) for $k = 1$ to z. Let $e_k = \langle i, j \rangle$:
  (a) $T_{\text{red}} = +(T, r_i)$ and $T_{\text{blue or green}} = -(T, r_i)$.
  (b) $T_{\text{blue}} = +(T_{\text{blue or green}}, b_i)$ and $T_{\text{green}} = -(T_{\text{blue or green}}, b_i)$.
  (c) $T^{\text{good}}_{\text{red}} = -(T_{\text{red}}, r_j)$.
  (d) $T^{\text{good}}_{\text{blue}} = -(T_{\text{blue}}, b_j)$.
  (e) $T^{\text{good}}_{\text{green}} = -(T_{\text{green}}, g_j)$.
  (f) $T' = \cup(T^{\text{good}}_{\text{red}}, T^{\text{good}}_{\text{blue}})$.
  (g) $T = \cup(T^{\text{good}}_{\text{green}}, T')$.
(3) Output(Detect(T)).

Theorem 11.2 *(Lipton, 1994) Any SAT problem in n variables and m clauses can be solved with at most $\mathcal{O}(m + 1)$ separations, $\mathcal{O}(m)$ merges, and one detection.*

The above theorem implies that biological computation can be used to solve all problems in $\mathcal{NP}$, although it does not mean all instances of $\mathcal{NP}$ can be solved in a feasible way. From a computability point of view, neither the quantum computation model nor the biological

computation model has more computational power than the Turing machine. Thus we have an analog of the Church–Turing Thesis for quantum and biological computations:

> **Quantum and biological computation thesis**: An arithmetic function is computable or a decision problem is decidable by a quantum computer or by a biological computer if and only if it is computable or decidable by a Turing machine.

This means that from a complexity point of view, both the quantum computation model and the biological computation model do indeed have some more computational power than the Turing machine. More specifically, we have the following complexity results about quantum and biological computations:

(1) Integer factorization and discrete logarithm problems are believed to be intractable in Turing machines; no efficient algorithms have been found for these two classical, number-theoretic problems, in fact, the best algorithms for these two problems have the worst-case complexity $\Theta\left((\log n)^2(\log\log n)(\log\log\log n)\right)$. But however, both of these two problems can be solved in polynomial-time by quantum computers.
(2) The famous Boolean Formula Satisfaction Problem (SAT) and directed Hamiltonian Path Problem (HPP) are proved to be $\mathcal{NP}$-complete, but these problems, and in fact any other $\mathcal{NP}$-complete problems, can be solved in polynomial biological steps by biological computers.

Now we are in a position to discuss the DNA-based cryptography. We first study a DNA analog of one-time pad (OTP) encryption; its idea may be described as follows.

(1) Plaintext encoding: The plaintext: M is encoded in DNA strands.
(2) Key generation: Assemble a large OTP in the form of DNA strands.
(3) OTP substitution: Generate a table that randomly maps all possible strings of $M \to C$ such that there is a unique reverse mapping $M \leftarrow C$.
(4) Encryption: Substitute each block of M with the ciphertext C given by the table, to get $M \to C$.
(5) Decryption: Reverse the substitutions to get $C \to M$.

The DNA implementation of the above scheme may be as follows:

(1) Plaintext in DNA: Set one test tube of short DNA strands for M.
(2) Ciphertext in DNA: Set another test tube of different short DNA strands for C.
(3) Key generation: Assemble a large OTP in the form of DNA strands.
(4) OTP substitution: Map M to C in a random yet reversible way.
(5) Encryption – DNA substitution OTDs: Use long DNA one-time pads containing many segments; each contains a cipher word followed by a plaintext word. These word-pair DNA strands are used as a lookup table in conversion of plaintext into ciphertext for $M \to C$.
(6) Decryption; Just do the opposite operation to the previous step for $C \to M$.

Just the same as stream cipher, we could use the operation XOR, denoted by $\oplus$ to implement the DNA OTP encryption as follows.

(1) DNA plaintext test tube: Set one test tube of short DNA strands for M.
(2) DNA ciphertext test tube: Set another test tube of different short DNA strands for C.
(3) Key Generation: Assemble a large OTP in the form of DNA strands.
(4) Encryption: Perform $M \oplus$ OTPs to get cipher strands; remove plaintext strands.
(5) Decryption: Perform $C \oplus$ OTPs to get back plaintext strands.

Problems for Section 11.4

1. Explain how DNA computing can be used to solve the Hamiltonian Path Problem (HPP).
2. Explain what the main features of DNA biological cryptography are.
3. Explain why DNA biological cryptography is quantum computing resistant.
4. DNA molecular biologic cryptography, for example, Reif's one-time pad DNA cryptosystem developed in 2004 [41], is a new development in cryptography. Give a description of the Reif's DNA-based one-time pads.
5. Write an essay to compare the main features of classic, quantum and DNA cryptography.

11.5 Bibliographic Notes and Further Reading

Quantum-computing resistant, or quantum-attack resistant, or just quantum resistant cryptography is an important research direction in modern cryptography, since once a practical quantum computer can be built, all the public-key cryptography based on IFP, DLP, and ECDLP can be broken in polynomial-time. As Bill Gates noted in his book [11]:

> We have to ensure that if any particular encryption technique proves fallible, there is a way to make an immediate transition to an alternative technique.

We need to have quantum resistant cryptographic systems ready at hand, so that we can use these cryptosystems to replace these quantum attackable cryptosystems. In this chapter, we only discussed some quantum resistant cryptographic systems, including quantum cryptography, interested readers should consult the following references for more information: [19–39]. Note that in the literature, quantum-computing resistant cryptography is also called *post-quantum cryptography*. Springer publishes the proceedings of the post-quantum cryptography conferences [42–45].

Just the same as quantum computing and quantum cryptography, DNA molecular computation is another type of promising computing paradigm and cryptographic scheme. Unlike the traditional computing model, DNA molecular computing is analog, not digital, so it opens a completely different phenomena to solve the hard computational problem. As can be seen from our above discussion, DNA computing has the potential to solve the NP-completeness problems such as the famous Hamiltonian Path Problem (HPP) and the Satisfiability Problem

(SAT). Of course there is a long way to go to truly build a practical DNA computer. The reader may consult the following references for more information on DNA computing and cryptography: [46–54].

Chaos-based cryptography [16–18] may be another good candidate for quantum resistant cryptography; it is suggested that readers consult [15] for more information.

References

1. R. J. McEliece, *A Public-Key Cryptosystem based on Algebraic Coding Theory*, JPL DSN Progress Report 42–44, 1978, pp. 583–584.
2. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring Polynomials with Rational Coefficients", *Mathematische Annalen*, **261**, 1982, pp. 515–534.
3. H. W. Lenstra, Jr., "Lattices", *Algorithmic Number Theory*, edited by J.P. Buhler and P. Stevenhagen, Cambridge University Press, 2008, pp. 127–182.
4. P. Q. Nguyen and B. Vallée, *The LLL Algorithm: Survey and Applications*, Springer, 2011.
5. J. Hoffstein, J. Pipher, and J. H. Silverman, "A Ring-Based Public-Key Cryptosystem", *Algorithmic Number Theory ANTS-III*, Lecture Notes in Computer Science **1423**, Springer, 1998, pp. 267–288.
6. J. Hoffstein, N. Howgrave-Graham, J. Pipher et al., "NTRUEncrypt and NTRUSign: Efficient Public Key Algorithmd for a Post-Quantum World", *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto 2006)*, 23–26 May 2006, pp. 71–77.
7. J. Ding, J. E. Gower, and D. S. Schmidt, *Multivariate Public Key Cryptosystems*, Springer, 2006.
8. J. Y. Cai and T. W. Cusick, "A Lattice-Based Public-Key Cryptosystem", *Information and Computation*, **151**, 1–2, 1999, pp. 17–31.
9. Y. Pan and Y Deng, "Cryptanalysis of the Cai-Cusick Lattice-Based Public-Key Cryptosystem", *IEEE Transactions on Information Theory*, **57**, 3, 2011, pp. 1780–1785.
10. C. H. Bennett and G. Brassard, "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Proceedings of the IEEE International Conference on Computers Systems and Singnal Processing*, IEEE Press, 1984, pp. 175–179.
11. B. Gates, *The Road Ahead*, Viking, 1995.
12. L. M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems", *Science*, **266**, 11 November 1994, pp. 1021–1024.
13. L. M. Adleman, "On Constructing a Molecular Computer", *DNA Based Computers*. Edited by R. Lipton and E. Baum, American Mathematical Society, 1996, pp. 1–21.
14. E. Lamm and R. Unger, *Biological Computation*, CRC Press, 2011.
15. L. Kocarev and S. Lian, *Chaos-Based Cryptography*, Springer, 2011.
16. I. MishkovskiK and L. Kocarev, "Chaos-Based Public-Key Cryptography", In: [15], *Chaos-Based Cryptography*. Edited by Kocarev and Lian, pp. 27–66.
17. D. Xiao, X. Liao, and S. Deng, "Chaos-Based Hash Function", In: [15], *Chaos-Based Cryptography*, Edited by Kocarev and Lian, 2011, pp. 137–204.
18. E. Solak, "Cryptanalysis of Chaotic Ciphers", In: [15], *Chaos-Based Cryptography*, Edited by Kocarev and Lian, 2011, pp. 227–254.
19. C. H. Bennett, "Quantum Cryptography using any two Nonorthogonal Sates", *Physics Review Letters*, **68**, 1992, pp. 3121–3124.
20. C. H. Bennett, "Quantum Information and Computation", *Physics Today*, October 1995, pp. 24–30.
21. C. H. Bennett, G. Brassard, and A. K. Ekert, "Quantum Cryptography", *Scientific American*, October 1992, pp. 26–33.
22. E. R. Berlekampe, R. J. McEliece, and H. van Tilburg, "On the Inherent Intractability of Certain Coding Problems", *IEEE Transaction on Information Theory*, **IT-24**, 1978, pp. 384–386.
23. D. Bruss, G. Erdélyi, T. Meyer, et al., "Quantum Cryptography: A Survey", *ACM Computing Surveys*, **39**, 2, 2007, Article 6, pp. 1–27.
24. E. F. Canteaut and N. Sendrier, "Cryptanalysis of the Original McEliece Cryptosystem", *Advances in Cryptology – AsiaCrypto'98*, Lecture Notes in Computer Science **1514**, Springer, 1989, pp. 187–199.

25. P-L. Cayrel and M. Meziani, "Post-Quantum Cryptography: Code-Based Signatures", *Advances in Computer Science and Information Technology – AST/UCMA/ISA/ACN 2010*, Lecture Notes in Computer Science **6059**, Springer, 2010, pp. 82–99.
26. H. Dinh, C. Moore, and A, Russell, "McEliece and Niederreiter Cryptosystems That Resist Quantum Fourier Sampling Attacks", *Advances in Cryptology – Crypto 2011*, Lecture Notes in Computer Science **6841**, Springer, 2011, pp. 761–779.
27. R. J. Hughes, "Cryptography, Quantum Computation and Trapped Ions", *Philosophic Transactions of the Royal Society London*, Series **A**, **356**, 1998, pp. 1853–1868.
28. H. Inamori, *A Minimal Introduction to Quantum Key Distribution*, Centre for Quantum Computation, Clarendon Laboratory, Oxford University, 1999.
29. H. K. Lo, "Quantum Cryptography", *Introduction to Quantum Computation and Information*. Edited by H. K. Lo, S. Popescu and T. Spiller, World Scientific, 1998, pp. 76–119.
30. H. Lo and H. Chau, "Unconditional Security of Quantum key Distribution over Arbitrary Long Distances", *Science*, **283**, 1999, pp. 2050–2056.
31. H. Niederreiter, "Knapsack Type Cryptosystems and Algebraic Coding Theory", *Problem of Control and Information Theory*, **15**, 1986, pp. 159–166.
32. M. A. Nielson and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University Press, 2010.
33. R. A. Perlner and D. A. Cooper, "Quantum Resistant Public Key Cryptography", *Proceedings of the 8th Symposium on Identity and Trust on the Internet*, Gaithersburg, MD, April 14–16, ACM Press, 2009, pp. 85–93.
34. W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, 2nd Edition, Prentice-Hall, 2006.
35. H. van Tilborg (editor), *Encyclopedia of Cryptography and Security*, Springer, 2005.
36. H. van Tilburg, "On the McEliece Public-Key Cryptography", *Advances in Cryptology – Crypto'88*, Lecture Notes in Computer Science **403**, Springer, 1989, pp. 119–131.
37. J. L. Walker, *Codes and Curves*, American Mathematical Society and Institute for Advanced Study, 2000.
38. C. P. Williams, *Explorations in Quantum Computation*, 2nd Edition, Springer, 2011.
39. S. Y. Yan, *Cryptanalyic Attacks on RSA*, Springer, 2009.
40. S. Y. Yan, *Quantum Attacks on Public-Key Cryptography*, Springer, 2012.
41. A. Gehani, T. H. LaBean, and J. H. Reif, "DNA-Based Cryptography", *Molecular Computing*, Lecture Notes in Computer Science **2950**, Springer, 2004, pp. 167–188.
42. D. J. Bernstein, J. Buchmann, and E. Dahmen (Editors), *Post-Quantum Cryptography*, Springer, 2010.
43. J. Buchmann and J. Ding (Editors), *Post-Quantum Cryptography*, Lecture Notes in Computer Science **5299**, Springer, 2008.
44. N. Sendrier (Editor), *Post-Quantum Cryptography*, Lecture Notes in Computer Science **6061**, Springer, 2010.
45. B. Yang (Editor), *Post-Quantum Cryptography*, Lecture Notes in Computer Science **7071**, Springer, 2011.
46. R. D. Barish, P. Rothemund, and E. Winfree, "Two Computational Primitives for Algorithmic Self-Assembly: Copying and Counting", *Nano Letters*, **5**, 12, 2005, pp. 2586–2592.
47. Y. Benenson, B. Gill, and U. Ben-Dor, et al., "An Autonomous Moleular Computer for Logical Control of Gene Expressions", **Nature**, **429**, 6990, 2004, pp. 423–429.
48. D. Boneh, C. Dunworth and R. Lipton, et al., "On the Computational Power of DNA", *Discrete Applied Mathematics*, **71**, 1, 1996, pp. 79–94.
49. D. Bray, "Pretein Molecular as Computational Elements in Living Cells", *Nature*, **376**, 6538, 1995, pp. 307–312.
50. T. Gramb, A. Bornholdt, and M. Grob, et al., *Non-Standard Computation*, Wiley-VCH, 1998.
51. R.Lipton, "DNA Solution of Hard Computational Problems", *Science*, **268**, 5210, 1995, pp. 542–545.
52. R. Unger and J. Moult, "Towards Computing with Protein", *Proteine*, **63**, 2006, pp. 53–64.
53. E. Winfree, F. Liu, and L. A. Wenzler, et al., "Design and Self-Assembly of Two-Dimensional DNA Crystals", *Nature*, **394**, 6693(1998), pp. 539–544.
54. N. Jonoska, G. Paun, and G. Rozenberg (editors), *Molecular Computing*, Lecture Notes in Computer Science **2950**, Springer, 2004.

INDEX

Computational Number Theory and Modern Cryptography, First Edition. Song Y. Yan.