Open Introduction to Cryptography

Eric Landquist

May 22, 2018

Acknowledgments. I gratefully acknowledge support for the initial development of this book by the Educational Advancement Foundation in the summer of 2015.

First release, May 2018

Contents

Notation

$\oplus$	Addition modulo m, for some given $m \in \mathbb{N}$: [§2.3.1]
$\mid$	Divides
$\mapsto$	"Maps to:" $a \mapsto b$ if for some function f, we have $f(a) = b$.
$\mathbb{F}$	A field
$\mathbb{F}_p$	The field of p elements: $\{0, 1, \ldots, p-1\}$
$\mathbb{F}_q$	The field of $q = p^k$ elements, for some $k \in \mathbb{N}$.
lg	The binary logarithm: $\log_2$.
$\mathcal{L}$	The set of letters $\{\mathtt{A}, \mathtt{B}, \ldots \mathtt{Z}\}$: [§2.4]
$\pmod m$	Modulo m: [§2.3]
$\mathbb{N}$	The set of natural numbers: $\{1, 2, \ldots\}$
$\mathcal{NP}$	Non-deterministic polynomial time decidable problems. [§1.6]
$\mathcal{P}$	Polynomial time decidable problems. [§1.6]
$\mathcal{PP}$	Probabilistic polynomial time decidable problems. [§1.6]
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integers: $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Z}_n$	The ring $\{0, 1, \ldots, n-1\}$
$\mathbb{Z}_n^*$	The set of invertible integers modulo n. [§4.3]

Chapter 1

What is Cryptography?

The history and study of cryptography begins with a need or desire for privacy. We begin by posing a question that is easy to understand, but which will take months to begin to unpack.

The Problem: How does Alice send Bob a message so that only Bob can read the message?

The history of cryptography is the history of solving problems of information security. Throughout human history, individuals and organizations have sought to conceal communications from all but the intended recipient. In recent decades, more complex problems of secrecy and privacy have arisen. Motivated by these needs, governments, militaries, corporations, academic institutions, and individuals have proposed various solutions. The future will certainly bring new problems and questions that will require a combination of existing knowledge and technology with creative ideas and breakthroughs in order to solve them. The material of the course will serve to partially equip you to utilize current technology and developing your problem-solving mindset will prepare you to develop new technology. Indeed, cryptography is a discipline of problem-solving. Mathematics as a whole is all about problem-solving. All of science and even every profession is motivated by problem-solving.

1.1 Problem-Solving

In light of this, it will help us to understand the problem-solving process. In his famous book *How to Solve It*, mathematician George Pólya described his problem-solving process in four steps.

1. Understand the problem.
2. Devise a plan and identify the tools that could apply to the problem.
3. Execute the plan and come up with a solution.
4. Finally, check the solution to see if it makes sense, solves the problem, if there are any weaknesses in the solution, and if the problem and solution could be extended or generalized.

In this course, it will help to be of the same mindset as you work through each problem, no matter how small or great it is. To facilitate describing the problems in this course, we will introduce some common terminology.

1.2 Foundational Terminology

Cryptography comes from the Greek words $\kappa\rho\upsilon\pi\tau o\varsigma$ (*kryptos*), meaning "hidden," and $\gamma\rho\alpha\varphi\epsilon\iota\nu$ (*graphein*), meaning "writing," and is the study of devising schemes to secure information. Informally, it is used interchangeably with **cryptology**, from the Greek word $\lambda o\gamma\iota\alpha$ (*-logia*), meaning "study." Technically, though, cryptology combines the study of securing information as well as **cryptanalysis**, the study of analyzing

cryptographic schemes in order to discover flaws or insecurities. Informally, you can think of cryptographers as "code writers" and cryptanalysts as "code breakers."

A **cryptosystem** is a scheme or set of methods intended to secure information. In the most basic application in which a message is to be sent securely from one person, Alice, to another, Bob, a cryptosystem will have two functions or algorithms: one function to **encrypt**, or secure, the message and another function to **decrypt** the encrypted message and thus reveal the original message. Such a cryptosystem is called a **cipher**. The message is often called the **plaintext** and an encrypted message is often called the **ciphertext**. Encryption functions generally take two arguments: a message and a **key**. Different keys will produce different ciphertexts. When the key is a very long list of letters or numbers, it is often called a **keystream**. If a cryptosystem uses the same key to encrypt and decrypt, then it is called a **symmetric** cipher. A symmetric cipher is also called a **private-key** cryptosystem because anyone with the key that was used to encrypt a message will be able to decrypt the ciphertext; the key must be kept private by both the sender and receiver.

1.3 Overview

In this course, we will begin by learning about some of the first ciphers ever used in Section 2. Before the 20th century, most encryption, decryption, and cryptanalysis was done by hand. With the need for speed in the 20th century, machines and computers were built to enable more sophisticated cryptosystems and cryptanalytic methods to be developed and implemented. Some of these methods will be discussed in Section 3. The obvious problem is that it may be difficult or impossible for two people to share an encryption key. In the 1970s, researchers began to tackle this problem and devised several different creative solutions that has forever changed electronic communications and cryptography as a whole. This resulted in two new branches of cryptography: secure key exchange and **public-key cryptography** (or **asymmetric-key** cryptography. This will be discussed in Section 5. Public-key cryptography is the result of creative applications of various results of discrete mathematics that had previously been considered to be "pure," in the sense that these results lacked any kind practical application. In light of this, a summary of necessary mathematics will be covered in Section 4. Public-key cryptography has opened the door for solutions to countless other problems and applications, many of which will be covered in Section 6. We will wrap up these notes with examples of personal encryption products, all of which are open-source, in Section 8 and with topics for further study and final projects in Section 9. Some of these topics concern problems that will arise if quantum computing becomes a reality. The advent of quantum computing will forever change the face of cryptography and will certainly introduce new problems needing innovative solutions.

1.4 A Selection of Cryptographic Problems

What are some applications of cryptography? More precisely, what are some problems that have arisen that cryptographic methods have been applied to solve?

- Confidentiality: How can you encrypt a message so that unintended recipients will be unable to decipher the message (at least with high probability)?
- Integrity: How can you be certain that a message or data set was not altered between the sender and receiver?
- Key Exchange: How can two people agree on a private key over an insecure channel?
- Public-key cryptography: How can you decouple the encryption and decryption process so that knowledge of the encryption function and encryption key makes it infeasible to decipher a message?
- Zero-knowledge proofs: How can you prove possession of information without revealing any portion of the information?
- Digital signatures, authentication, and identification: How do you digitally sign a message to prove that you're the one who sent it?

- Nonrepudiation: How is it possible to make it impossible to deny that you have sent a message?
- Coin-flipping: How do you digitally flip a coin and guarantee that neither side cheats?
- Secret sharing: How do you divide a piece of information among a group of people so that only a predetermined number of people in the group can recover the information.

1.5 Randomness Versus Structure

Discussion What is your idea of random or randomness? What are some ways to obtain truly random strings of digits, numbers, or letters? What is structure? What are examples of structures that appear to be random?

Problem 1.5.1. [10 points] The following ciphertext was intercepted by the enigmatic Quasi Lendrict. We don't believe that the encryption method that he chose was particularly sophisticated. Do your best to crack the message to reveal the plaintext, but any observation or partial solution will be helpful. Be mindful of some of the questions for discussion below.

```
NTLDX, EHRPLDXKHR XOP ZLUYFPB. XOPH TRPHXTNV KZZFTAKYFP XUUFD KHR BKGP K ZFKH. HPIX,

AKLLV UEX XOP ZFKH. NTHKFFV, AOPAG VUEL DUFEXTUH XU DPP XOKX TX BKGPD DPHDP KHR DUFMPD

XOP ZLUYFPB. PIXPHR UL QPHPLKFTCP XOP ZLUYFPB UL XOP DUFEXTUH.
```

Discussion. What observations of the ciphertext did you make? What are some tools that you used to reveal the plaintext, whether in whole or in part?

Problem 1.5.2. [20 points] The following ciphertext was intercepted by the enigmatic Quasi Lendrict. He apparently realized that he needed to eliminate some structure in his ciphertexts. Do your best to crack the message to reveal the plaintext, but any observation or partial solution will be helpful.

```
PUDXT TXOJP DXROP NLHPL NDJON ZMEXK MDOOZ MEZAX XEOBK KDPNJ HHNBT PXOBO PDKWJ IIXFR LOHZP

DOPNL HPLND OJMHD ZHNBT PZMZI BOPWJ IILOD ZMBOP NLHPL NDPXH NZHYP UDHNB TPXOB OPDKO HJDMP

JRJHZ IIBNZ MEXKM DOOJO HZIID EDMPN XTBPU DXTTX OJPDJ OOPNL HPLND XNXNE DNYHH
```

Discussion. What are some similarities and differences between Problem 1.5.1 and Problem 1.5.2?

Read Chapter 15 of [8], but rather than trying to absorb all of the notation, definitions, and theorems, get a feel for the big picture of what defines randomness and structure. The main motivation for this discussion on randomness and structure is that a cryptographer wants to obfuscate structure as much as possible so that any ciphertext that a cryptosystem produces appears random.

1.6 Mathematical Hardness and Computational Complexity

At times, we will describe the security of a cryptosystem based on the computational effort required to crack it using the best known attacks. When we talk of an algorithm, we mean some mathematical or computational recipe that gives you an output for a given input. The **(computational complexity** of an algorithm is a rough measure of the number of steps the algorithm requires to complete, as a function of the size of the input. A **decision problem** is a problem that asks a "yes or no" question. More generally, a **computational problem** is a problem that computes a solution to a problem. To distinguish this subtlety, consider the **integer factorization problem**: given an integer n, find all prime factors of n. This is a computational problem. We can transform this into the decision integer factorization problem. "Given some

$n \in \mathbb{Z}$, does n have a prime factor less than some $0 < c \in \mathbb{R}$?" This kind of rephrasing is one of the most common ways to turn a computational problem into a decision problem and shows that a computational problem is at least as hard as its corresponding decision problem in the sense that if a computational problem can be solved, then any corresponding decision problem is trivial. This leads us to formalize the concept of computational complexity somewhat rigorously.

Definition 1.6.1. An algorithm runs in **polynomial time** if the number of steps required to terminate is bounded above by a fixed polynomial function of the size (generally the number of **bits**) of the input. The set of all decision problems that can be determined with a polynomial time algorithm is denoted $\mathcal{P}$.

An algorithm runs in **probabilistic polynomial time** if it terminates in polynomial time with high probability. The set of all decision problems that can be solved with high probability by a probabilistic polynomial time algorithm is denoted $\mathcal{PP}$.

In the case of the integer factorization problem, the size of the input n is in fact the length of the binary representation of n, or the number of bits of n, which is also roughly $\lg(n)$, the binary logarithm of n. There is no known algorithm that solves either formulation of the integer factorization problem in deterministic or probabilistic polynomial time, so this problem is not in $\mathcal{P}$. However, multiplication and division only require a polynomial number of steps, as a function of the inputs, so the problem of verifying a solution to the integer factorization problem can be determined in polynomial time. This motivates more terminology.

Definition 1.6.2. A decision problem is decided in **non-deterministic polynomial time** if there is no known algorithm to determine a yes or no answer answer to the problem in polynomial time, but given a solution to the problem, it can be verified in polynomial time that it does indeed answer the problem. The set of all decision problems that can be decided in non-deterministic polynomial time is denoted $\mathcal{NP}$.

A decision problem that is at least as hard as any other problem in $\mathcal{NP}$ is said to be $\mathcal{NP}$**-complete** and any computational problem that is at least as hard as an $\mathcal{NP}$-complete problem is said to be $\mathcal{NP}$**-hard**.

The only known algorithms that solve $\mathcal{NP}$-complete decision problems require an exponential number of steps, as a function of the size of the input, to terminate with a solution. We say that such algorithms run in **exponential time**.

Under these definitions, the integer factorization problem is in $\mathcal{NP}$, but is not $\mathcal{NP}$-complete because there are integer factorization algorithms, such as the Quadratic Sieve and Number Field Sieve , that run faster than exponential time, but slower than polynomial time. Such algorithms are said to be **subexponential time** algorithms.

To illustrate the subtle distinction between $\mathcal{NP}$-complete and $\mathcal{NP}$-hard, consider the Traveling Salesman Problem (TSP). The computational version is the problem of finding the shortest (or fastest or cheapest) route for a salesman to travel such that he visits each location in a given set, beginning and ending at the same location. The decision version asks, is there a route, as described above, that is less than some fixed distance (or time or cost). The decision TSP is $\mathcal{NP}$-complete, because the

Any computational or decision problem that is in $\mathcal{P}$ or $\mathcal{PP}$ is often called **tractable** or **easy** because it's computationally easy to compute an answer either deterministically or with high probability. Deterministic or probabilistic polynomial time algorithms are often called **fast** or **efficient**. By contrast, any computational or decision problem that is not in $\mathcal{P}$ or $\mathcal{PP}$ is called **intractable** or **hard** because of the belief or expectation that any attempt to solve a large instance of such a problem will require too much computational effort.

Problem 1.6.1 (10 points)**.** Find an example of a computationally or mathematically hard problem. Is the problem a decision problem or computational problem? Does the problem have both a computational and decision description? If so, please describe both versions. Is the problem $\mathcal{NP}$-hard, $\mathcal{NP}$-complete, or just in $\mathcal{NP}$? Were you able to find if there are any cryptosystems based on the intractability of the problem? How fast is the best known algorithm that solves the problem?

Getting back to the subject at hand, when designing a cryptosystem, cryptographers want to base the security of the cipher or protocol on a computationally or mathematically hard problem. If cryptanalysis of a cryptosystem requires a polynomial number of steps, as a function of the size of the key, then the cryptosystem is insecure.

Chapter 2

Classical Cryptography

Cryptography has a handful of main branches including standard encryption and decryption of plaintexts, error detection and correction, and what is called **steganography**. Steganography comes from the Greek word $\sigma\tau\epsilon\gamma\alpha\nu o\varsigma$ (*steganos*), meaning "concealed," and its aim is to conceal the existence of a message. Well-known examples of steganography are invisible ink, microdots, and Bible codes. While the main focus of this course is on mathematical aspects of cryptography, the topic of steganography is significant enough to merit some attention. (We should note that methods to detect steganography in digital files can involve some advanced mathematics.)

After discussing steganography, we will explore two main encryption methods used in classical cryptography: transposition ciphers and substitution ciphers. A **transposition cipher** permutes the plaintext to obtain a ciphertext while a **substitution cipher** substitutes ciphertext for plaintext a bit, byte, letter, or a block of letters or bits at a time. Transposition and substitution ciphers are the foundation for modern symmetric ciphers.

2.1 Steganography

Steganography can be loosely categorized into two branches: visible and invisible. By *visible* steganography, we mean that the plaintext can be seen, but is obscured among other text or media. By *invisible* steganography, we mean that the plaintext is invisible to the unaided eye.

2.1.1 Visible - Hidden in Plain Sight

Problem 2.1.1. [10 points] Do some basic research on some method of visible steganography. It can be one of the methods below or something different. Specifically:

1. describe how the steganographic method is used,
2. give at least a couple key historical facts about the method, and
3. give an example of the method in practice, if it is feasible.

- Equidistant Letter Sequences: Bible Codes
- Anagrams
- Grille Cipher
- Bacon's Cipher
- Digital steganography

2.1.2 Invisible

Problem 2.1.2. [10 points] Do some basic research on some method of invisible steganography. It can be one of the methods below or something different. Specifically:

1. describe how the steganographic method is used,
2. give at least a couple key historical facts about the method, and
3. give an example of the method in practice, if it is feasible.

- Ancient ad-hoc techniques
- Invisible ink
- Microdots
- Printer steganography

2.2 Transposition Ciphers

We will not give transposition ciphers much attention, but they are included for historical purposes and also because the underlying concept, **permutations**, is an essential component of modern ciphers and hash functions.

As noted above, a transposition cipher or **permutation cipher** permutes the plaintext to obtain the ciphertext. The following are examples of classical transposition ciphers.

- Columnar Transposition
- Double Columnar Transposition
- Rail Fence Cipher
- Route Cipher
- Scytale

Problem 2.2.1. [10 points] Do some basic research on one of the transposition ciphers above. Specifically:

1. describe the cipher,
2. give at least one historical fact about the cipher, and
3. encrypt a message of your choice using one of the transposition ciphers above.

Problem 2.2.2 (15 points)**.** Recover the plaintext from a peer's ciphertext, which was encrypted using a transposition cipher in Probem 2.2.1.

Fractionation The concept of **fractionation** is to represent letters or other characters using a smaller set of symbols. This is akin to **encoding**, in which letters are converted into other symbols for transmission. Morse Code is an example of encoding and in the digital age, computers must represent letters and symbols using a binary encoding, such as ASCII or UTF-8 . The following are historical methods of fractionating plaintext so that a transposition cipher could easily break a letter apart.

- Polybius Square
- Straddling checkerboard

Problem 2.2.3. [10 points] Do some basic research on one of the fractionation methods above. Specifically:

1. describe the method,
2. give at least one historical fact about the method, and
3. encode a message of your choice using one of the fractionation methods above.

Fractionation and Transposition The subtle distinction between fractionation and encoding is that the purpose of encoding is for direct transmission over a particular channel, while fractionation is employed for the purpose of further encryption. The following are some historical methods that combined fractionation and transposition. Many of these were very difficult to cryptanalyze in their day.

- ADFGX (§2.6 of [8])
- ADFGVX
- Bifid
- Trifid
- VIC

Problem 2.2.4. [10 points] Do some basic research on one of the ciphers above. Specifically:

1. describe the cipher,
2. give at least one historical fact about the cipher, and
3. encrypt a message of your choice using one of the ciphers above.

Problem 2.2.5 (15 points)**.** Recover the plaintext from a peer's ciphertext, which was encrypted using a cipher in Probem 2.2.4.

2.2.1 Permutations

Definition 2.2.1. A **permutation** of a set S is a bijective (one-to-one and onto; or injective and surjective) function $\pi : S \to S$.

Example 2.2.1. Let $S = \{1, 2, 3, 4, 5, 6\}$. One possible permutation, π, of S is the following.

$$\pi(1) = 5, \pi(2) = 3, \pi(3) = 6, \pi(4) = 1, \pi(5) = 2, \pi(6) = 4$$

This can be written as an array, for convenience, with the input as the top row and the output as the bottom row.

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 3 & 6 & 1 & 2 & 4 \end{pmatrix}$$

From this representation, it is easy to see the inverse of π:

$$\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 2 & 6 & 1 & 3 \end{pmatrix}$$

There is a much more convenient way to represent a permutation, namely as a product of disjoint **cycles**.

Example 2.2.2. Notice that in Example 2.2.1, π maps $1 \mapsto 5 \mapsto 2 \mapsto 3 \mapsto 6 \mapsto 4 \mapsto 1$. We represent this series of mappings by the cycle $(1, 5, 2, 3, 6, 4)$, with the understanding that the last element of the cycle maps to the first.

We can now describe transposition ciphers as permutation ciphers.

Definition 2.2.2. Let $n \in \mathbb{N}$ and $S = \{1, 2, \cdots, n\}$. A *permutation cipher* is a permutation $\pi : S \to S$ in which blocks of n characters are considered and the location of character i is moved to location $\pi(i)$ in the block, for all $1 \leq i \leq n$. If π is used to encrypt the plaintext, then π^{-1} is used to recover the plaintext from the ciphertext.

Example 2.2.3. Suppose that we wanted to encrypt the six-letter message `CIPHER` using the permutation π in Example 2.2.1. The letter `C` is in position 1, so it is moved to position 5, etc. So the ciphertext is `HEIRCP`.

2.3 Mathematical Interlude: Modular Arithmetic

A fundamental concept to understand in cryptography, both classical and modern, is modular arithmetic. Here is the notation and definition that we will be working from.

2.3.1 Definition and Notation

Definition 2.3.1. Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{N}$. We say that a is **congruent** to b **modulo** m, and write $a \equiv b \pmod{m}$, if $m \mid (b - a)$.

We say that we **reduce** an integer a modulo m, written $a \pmod{m}$, if we want the unique nonnegative integer $b \in \mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ such that $a \equiv b \pmod{m}$. In this case, we write $b = a \pmod{m}$ or $a \pmod{m} = b$.

Example 2.3.1. $37 \equiv 11 \pmod{26}$ because $37 - 11 = 26$ and $26 \mid 26$.
$37 \equiv -15 \pmod{26}$ because $37 - (-15) = 52$ and $26 \mid 52$.

Definition 2.3.2. When the modulus, $m \in \mathbb{N}$ is understood, let $a \oplus b = (a+b) \pmod{m}$, for all $a, b \in \mathbb{Z}$. We will also extend this operator to vectors of integers modulo m, so that if $\vec{a}, \vec{b} \in \mathbb{Z}^n$, with $\vec{a} = \langle a_1, a_2, \ldots, a_n \rangle$ and $\vec{b} = \langle b_1, b_2, \ldots, b_n \rangle$, then $\vec{a} \oplus \vec{b} = \langle a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n \rangle$. In practice, $\vec{a}$, $\vec{b}$, and $\vec{a} \oplus \vec{b}$ will represent plaintext, a key stream, and the corresponding ciphertext.

Example 2.3.2. Let the modulus $m = 26$.

$4 \oplus 15 = 19$
$15 \oplus 15 = 30 \pmod{26} = 4$

2.3.2 Basic Properties

In this section, we will prove various useful and important properties of modular arithmetic.

Theorem 2.3.1. If $a, b, c, d \in \mathbb{Z}$, $m \in \mathbb{N}$, $a \equiv b \pmod{m}$, and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$.

Theorem 2.3.2. If $a, b, c, d \in \mathbb{Z}$, $m \in \mathbb{N}$, $a \equiv b \pmod{m}$, and $c \equiv d \pmod{m}$, then $ac \equiv bd \pmod{m}$.

Theorem 2.3.3. If $a, b, c \in \mathbb{Z}$, $m \in \mathbb{N}$, and $a \equiv b \pmod{m}$, then $ac \equiv bc \pmod{m}$.

Theorem 2.3.4. If $a, b, n \in \mathbb{Z}$, $m \in \mathbb{N}$, and $a \equiv b \pmod{m}$, then $a^n \equiv b^n \pmod{m}$.

Problem 2.3.1 (10 points)**.** Prove Theorem 2.3.1.

Problem 2.3.2 (10 points)**.** Prove Theorem 2.3.2.

Problem 2.3.3 (10 points)**.** Prove Theorem 2.3.3.

Problem 2.3.4 (10 points)**.** Prove Theorem 2.3.4.

2.3.3 Modular Inverses

Definition 2.3.3. The number 1 is called the **multiplicative identity** because for all $a \in \mathbb{Z}$, $a \cdot 1 = 1 \cdot a = a$. Likewise, $a \cdot 1 = 1 \cdot a \equiv a \pmod{m}$ for all $m \in \mathbb{Z}$. If $b \in \mathbb{Z}$ such that $ab \equiv 1 \pmod{m}$, then b is called an **inverse** of a modulo m and we write $a^{-1} \equiv b \pmod{m}$. If there exists some $b \in \mathbb{Z}_m$ such that $ab \equiv 1 \pmod{m}$, then b is called the inverse of a modulo m and we write $b = a^{-1} \pmod{m}$.

Example 2.3.3. Since $2 \cdot 4 = 8 \equiv 1 \pmod{7}$, $2^{-1} \pmod{7} = 4$ and $4^{-1} \pmod{7} = 2$.

Problem 2.3.5 (10 points)**.** Prove that if $a^{-1} \equiv b \pmod{m}$, then $b^{-1} \equiv a \pmod{m}$.

Problem 2.3.6 (10 points)**.** Let $m = 5$. Find the inverse of each $a \in \mathbb{Z}_m$ if an inverse exists. How many elements of $\mathbb{Z}_m$ have inverses?

Problem 2.3.7 (10 points)**.** Let $m = 30$. Find the inverse of each $a \in \mathbb{Z}_m$ if an inverse exists. How many elements of $\mathbb{Z}_m$ have inverses?

Problem 2.3.8 (10 points)**.** Prove that if $a \in \mathbb{Z}$ and $m \in \mathbb{N}$ such that there exists some $b \in \mathbb{Z}$ such that $a^{-1} \equiv b \pmod{m}$, then $b^{-1} \equiv a \pmod{m}$.

Problem 2.3.9 (10 points)**.** Under what conditions does an integer a have an inverse modulo m?

Theorem 2.3.5 (Euclidean Algorithm)**.** If $a, q, r \in \mathbb{Z}$, then $\gcd(a, aq + r) = \gcd(a, r)$.

Problem 2.3.10 (15 points)**.** Prove Theorem 2.3.5 and explain why it is the Euclidean Algorithm.

Problem 2.3.11 (10 points)**.** Apply the Euclidean Algorithm to compute $\gcd(30, 12)$.

Theorem 2.3.6. If $a, b \in \mathbb{Z}$, then there exist $x, y \in \mathbb{Z}$ such that $ax + by = \gcd(a, b)$.

Problem 2.3.12 (15 points)**.** Prove Theorem 2.3.6.

Problem 2.3.13. [10 points] Apply the Euclidean Algorithm to compute $\gcd(12, 17)$ and find $x, y \in \mathbb{Z}$ such that $12x + 17y = \gcd(12, 17)$.

Problem 2.3.14 (10 points)**.** As a follow up of Problem 2.3.13, determine $12^{-1} \pmod{17}$, given the statement $12x + 17y = \gcd(12, 17)$.

Problem 2.3.15 (10 points)**.** Explain how to apply Theorems 2.3.5 and 2.3.6 to compute modular inverses in general.

2.4 Substitution Ciphers

In this and subsequent sections, it will be convenient to represent the set of 26 English letters mathematically. To this end, we define the following set.

Definition 2.4.1. $\mathcal{L} = \{\texttt{A}, \texttt{B}, \ldots, \texttt{Z}\}$

It will also be convenient to assign each letter a number in order to work with these letters mathematically. To this end, we will associate each letter of $\mathcal{L}$ with the integers 0 through 25.

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 2.1: English letters and their corresponding shifts

2.4.1 Monoalphabetic Ciphers

Shift Ciphers and Caesar's Cipher Read the introduction to Chapter 2 and §2.1 of [8].

Affine Ciphers Read §2.2 of [8].

Problem 2.4.1 (10 points)**.** Show that any shift cipher is an affine cipher.

Atbash Atbash is possibly the oldest substitution cipher and was created by the ancient Hebrews. The name "Atbash" ("ATBaSh") is the key to the cryptosystem: the first and last letters are substituted for each other (Aleph (A) and Tav (T)), the second and second to last letters are substituted for each other (Bayt (B) and Sheen (Sh)), and so on. The equivalent in English would be to switch A and Z, B and Y, and so on. We could call this "Azby." The Hebrew prophet Jeremiah used Atbash a few times to conceal the fact that he was prophesying doom against the Babylonian Empire from everyone except his Hebrew readers in Jeremiah 25:26, 51:1, and 51:41.

Problem 2.4.2 (10 points)**.** Show that Atbash and Azby are examples of affine ciphers.

Full Generalization Each of the previous ciphers, shift ciphers, Caesar's cipher, affine ciphers, and Atbash, are all examples of a broader category of monoalphabetic substitution ciphers .

Definition 2.4.2. Let $\pi : \mathcal{L} \to \mathcal{L}$ be a permutation of the set $\mathcal{L}$ of English letters. A **monoalphabetic substitution cipher** is a cipher in which each plaintext letter, $p \in \mathcal{L}$ is mapped to a ciphertext letter $c = \pi(p)$.

Cryptanalysis of Monoalphabetic Ciphers

Problem 2.4.3. [5 points] Referring to Table 2.1 in [8], list the letters of the alphabet in descending order in terms of their occurrence in English.

Problem 2.4.4 (10 points)**.** §2.13 Exercise #2

Problem 2.4.5 (10 points)**.** §2.13 Exercise #3

Problem 2.4.6 (10 points)**.** §2.13 Exercise #5

Problem 2.4.7 (15 points)**.** §2.13 Exercise #6

Problem 2.4.8 (15 points)**.** §2.13 Exercise #7

Problem 2.4.9 (15 points)**.** Suppose that you encrypt a message using two different affine ciphers with two different moduli, m_1 and m_2. Is there any advantage to doing this instead of using a single affine cipher?

Problem 2.4.10 (10 points)**.** Aside from using the frequency of English letters, what other structural elements of the English language can be used to crack messages encrypted with a monoalphabetic cipher?

Problem 2.4.11 (10 points)**.** §2.14 Computer Problem #1

Problem 2.4.12 (10 points)**.** §2.14 Computer Problem #2

Problem 2.4.13 (10 points)**.** §2.14 Computer Problem #3

Problem 2.4.14 (10 points)**.** §2.14 Computer Problem #4

2.4.2 Polyalphabetic Ciphers

Vigenère's Cipher To make things more difficult, the French cryptographer Blaise de Vigenére used a multi-letter key in 1586. Today, we call his cipher **Vigenère's cipher**, but he called it *Le Chiffre Indéchiffrable*, which means "the indecipherable cipher." Sure, Vigenère's cipher is more difficult to crack than the most basic substitution ciphers, but guess what. It can be cracked.

Read §2.3 in [8].

A table, called a **tabula recta** to aid in encrypting, decrypting, and cryptanalysis with Vigenère's Cipher can be found in Appendix 9.

Problem 2.4.15. [10 points] Encrypt a plaintext of your choice with at least 100 characters using a key with 3-10 characters. The rest of the class will be challenged later to cryptanalyze the ciphertext.

Cryptanalysis of Vigenère's Cipher Charles Babbage and Friedrich Wilhelm Kasiski were the first to cryptanalyze Vigenère's cipher [7]. Babbage cryptanalyzed a ciphertext encrypted by John Thwaites after being challenged around 1854, but never published his results. Kasiski independently discovered a similar attack and published it in 1863. In 1920, William Friedman developed additional probabilistic techniques to determine the key length, called the **Friedman test** or the **kappa test**, using the **index of coincidence**. The common element of each cryptanalytic technique is to first determine the length of the key. Once the key length has been determined, the key itself can be recovered using techniques similar to those that we saw with shift ciphers.

Read §2.3.1, 2.3.2, and 2.3.3.

Problem 2.4.16. [15 points] Recover the plaintext a key from a peer's ciphertext, which was encrypted using Vigenère's Cipher in Problem 2.4.15. Explain your approach.

Problem 2.4.17. [20 points] Recover the plaintext and key from the following ciphertext, which was encrypted using Vigenère's Cipher. Explain your approach. [Note: The message is a Confederate letter written on September 14, 1862.]

```
UHP XVQAP VBPTWTK QFRDEY- MZMHZRH AE EWGWJZJLWX IZR TSWIY- ZBAB. KLFY LKM OCEJJDPGB AT

JSPN NKCEVZRH MJ YWDQV LFRP BB UGZQ QOCMIZH FYS CZFUGBZ GBTTHV IWKL FANA WFVVV THZNTP PV

OFPE HXQB Z WIAWE XDSJIOT L UWXR WVPNE BV AFUIS TZ WMOSZZF TSX MZSDC BS WHVS OJ TPSDBJXS

RRE WSXV OCDTFLWXL U GYEML QTTX PRGL UAHV KCL. QBR DAIXZ ZW TTTET ROI FFHTGL. QYJ
```

Problem 2.4.18 (15 points)**.** §2.14 Computer Problem #7

Problem 2.4.19 (15 points)**.** §2.14 Computer Problem #8

Problem 2.4.20 (15 points)**.** §2.14 Computer Problem #9

Running Key Cipher / "Book Cipher" So any substitution cipher with a relatively short key can be cracked and is insecure. To make things more difficult, someone had the idea to use text from a book as the key. The key is as long as the plaintext and nothing is repeated. In practice, the key would typically be taken from a popular book that the sender and receiver both had in their possession. Thus the key to the key (Should we call it a **metakey**?) would be the book and the starting page, chapter, or section of the book. While this is often called a "Book cipher," a book cipher is somewhat different; it is called the **running key cipher** because the key runs through a passage of some text without repeating. Here is an example.

Example 2.4.1. We will encrypt the message "Running key" with the key beginning with §2.3 of [8]: "A variation of the shift cipher was invented ...." The middle column will be elements of $\mathcal{L}$ and the right column will be the corresponding numbers given by Table 2.1.

	Letters	**Corresponding Numbers**									
Plaintext:	RUNNINGKEY	17	20	13	13	8	13	6	10	4	24
Key:	AVARIATION	0	21	0	17	8	0	19	8	14	13
Ciphertext:	RPNEQNZSSL	17	15	13	4	16	13	25	18	18	11

Recall that a tabula recta to aid in encrypting, decrypting, and cryptanalysis with the Running Key Cipher can be found in Appendix 9.

Problem 2.4.21. [10 points] Encrypt a plaintext of your choice with at least 100 characters using a key from a book of your choice. The rest of the class will be challenged later to cryptanalyze the ciphertext.

Cryptanalysis of the Running Key Cipher The Running key cipher is more difficult to cryptanalyze, but it can be cracked by applying the same probabilistic tools that we have used for other substitution ciphers. Friedman published a method to attack the running key cipher in 1918 [3, 6]. The following problems give us some ideas behind an approach.

Problem 2.4.22 (5 points)**.** If you are analyzing a ciphertext encrypted with the running key cipher, and you have determined that the plaintext or keystream begins with FOURSCO, what letters are likely to follow these?

Problem 2.4.23 (5 points)**.** If you are analyzing a ciphertext encrypted with the running key cipher, and you have determined that the plaintext or keystream begins with WECHOO, what letters are likely to follow these?

Problem 2.4.24 (5 points)**.** If you are analyzing a ciphertext encrypted with the running key cipher, and you have determined that the plaintext or keystream begins with `THI`, what letters are likely to follow these?

Problem 2.4.25 (10 points)**.** If a running key ciphertext begins with the letter `T`, what are the most likely and least likely plaintext / key stream pairs that resulted in that ciphertext?

Problem 2.4.26 (10 points)**.** If a running key ciphertext begins with the letters `TU`, what are some likely and unlikely plaintext / key stream pairs that resulted in that ciphertext?

Problem 2.4.27 (10 points)**.** If a running key ciphertext begins with the letters `MOI`, what are some likely and unlikely plaintext / key stream pairs that resulted in that ciphertext?

Problem 2.4.28 (20 points)**.** Recover the plaintext from the following ciphertext, which was encrypted using the running key cipher.
`EVSYO IAQZH DNHOS WTEKI QWJSD KTQDG RUTTD YTCNS FFQUT LLCBJ`

Problem 2.4.29 (15 points)**.** Recover the plaintext from a peer's ciphertext, which was encrypted using the Running Key Cipher in Problem 2.4.21.

Discussion What are some strengths and weaknesses of the Running Key Cipher? What could make it stronger?

One-Time Pad / Vernam Cipher In order to circumvent the known attacks on Vigenère's cipher, Frank Miller, a banker from Sacramento, CA, invented what later became called the **one-time pad** encryption method in 1882 [2]. In 1917, Gilbert Vernam, an engineer at AT&T Bell Labs, applied a similar procedure, with two major differences being that encryption and decryption were performed in binary and that the key was repeated. Vernam received a patent for his techniques in 1919. Around 1918, United States Army officer Joseph Mauborgne, realized that if Vernam's key was perfectly random and was never repeated, that one could in fact have perfect security. This fact was formally proven independently by Claude Shannon, known as the father of information theory, by 1945 and by Shannon's Soviet counterpart Vladimir Kotelnikov in 1941. To summarize, we formally define the one-time pad.

Definition 2.4.3. Let $1 < m \in \mathbb{N}$ be a modulus and $n \in \mathbb{N}$ the length of a plaintext message M. Thus, $M \in \mathbb{Z}_m^n$. Let $K \in \mathbb{Z}_m^n$ be a keystream such that each element of K is chosen randomly. K is called a **one-time pad** if it is never repeated or reused. The component-wise addition of K and M, modulo m to produce a ciphertext $C \in \mathbb{Z}_m^n$ is called the **one-time pad encryption method**. In other words, $C = M \oplus K$.

Thus, the two critical components of this method are that:

1. the keystream is perfectly random and
2. the keystream is never repeated for a subsequent encryption.

In fact, the one-time pad encryption method is the only known encryption algorithm with perfect security.

Theorem 2.4.1 (Kotelnikov and Shannon)**.** The one-time pad encryption method allows for perfect security. That is, it cannot be cracked.

Discussion What are some of the advantages and disadvantages, possibilities and difficulties of using the one-time pad?

Originally, one-time pads were lists of random letters, but in the digital age, we often use a **stream** of **bits** or **bytes**. A **bit** is an integer in the set $\{0, 1\}$ and a stream of bits is often denoted $\{0, 1\}^*$. We will denote the set of bits interchangeably as $\mathbb{Z}_2$ and $\mathbb{F}_2$ and a bit stream of length n by $\mathbb{Z}_2^n$ or $\mathbb{F}_2^n$. A **byte** is a string of 8 bits. A byte can be represented as an element of $\mathbb{Z}_{256}$ or as a string of two **hexadecimal** (i.e. base 16) digits.

Problem 2.4.30 (10 points)**.** Let $M_1, M_2 \in \{0, 1\}^*$ be two plaintext messages and let $K \in \{0, 1\}^*$ be a keystream of bits. Let $C_1 = M_1 \oplus K$ and $C_2 = M_2 \oplus K$. Explain why security is compromised when the key K is used for both ciphertexts.

Problem 2.4.31 (10 points)**.** Explain in your own words why you think that the one-time pad cannot be cracked.

Discussion What are some sources of random data (letters, bits, bytes, numbers, etc.)?

Problem 2.4.32 (15 points)**.** Describe the Solitaire (or Pontifex) cryptosystem. What can be said about its security?

Problem 2.4.33 (15 points)**.** Can you extend or improve upon the concept of the Solitaire (or Pontifex) cryptosystem?

Problem 2.4.34 (10 points)**.** Explore a topic that came out of a class discussion or solve a problem inspired by the content of this chapter.

Chapter 3

Modern Symmetric-Key Cryptography

In this section, we transition from classical cryptography, which was largely done by hand, to modern cryptography, an age which is marked by the use of machines and computers. While simple cipher devices, such as the scytale and cipher disks, were used for many hundreds of years, the first truly mechanical cipher machines were first designed, patented, and built beginning in the late 1910s, around the end of World War I [4, 7]. Specific independent inventors included the American Edward Hebern, the Swede Arvid Damm, the Dutchman Alexander Koch, and the German Arthur Scherbius. Scherbius is the most famous of this group as the inventor of the Enigma machine, used by the Germans in World War II. Use of cipher machines reached its peak during World War II, but various machines were used into the 1970s and 1980s.

It became increasingly painstaking to cryptanalyze these cipher machines by hand, so Allied cryptanalysts, notably the British cryptanalysts Alan Turing and Max Newman invented machines to crack messages that were intercepted from various Axis nations [7]. Turing first developed the concept of a programmable computer, though in 1823 Babbage was the first to design a machanical calculator. Newman took Turing's ideas and made a blueprint for the first programmable computer, Colossus, which was built in 1943. After the war, Colossus and its blueprints were destroyed. However, in 1945, the first (unclassified) programmable computer, ENIAC, was built by J. Presper Eckert and John Mauchly of the University of Pennsylvania [7]. With the birth of the computer age, all text was internally represented in binary. Since then, virtually all modern symmetric ciphers are described in binary, so aside from Section 3.1.1, we will assume that all plaintexts, keys, and ciphertexts are in binary.

This switch from written text to binary resulted in the codification of various standards for translating text to binary. Some common methods are ASCII (American Standard Code for Information Interchange), Extended ASCII, and UTF-8, the latter of which is the most common standard in use today. ASCII represents letters and other symbols as a string of 7 bits, one bit shy of a byte, so Extended ASCII uses an eighth bit to represent 128 additional characters. UTF-8 contains ASCII as a subset and allows for the digitization of virtually all international characters and symbols. The following is an extended ASCII table. The labels of the rows and columns represent the hexadecimal digits of the given row or column. So the letter `A` is encoded as the hexadecimal number `41`, which in binary would be `01000110`. The boxed entries are various control commands, such as `20` (SPACE) and `7F` (DELETE).

ISO-8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

With this historical and technical background established, we will explore the two main types of modern substitution ciphers: stream ciphers and block ciphers.

3.1 Stream Ciphers

All of the cryptosystems that we have considered so far fall under the category of **stream ciphers**. Here is a formal definition.

Definition 3.1.1. A **stream cipher** is a substitution cipher in which the plaintext is encrypted one character (e.g., a letter, bit, or byte) at a time.

The following are examples of stream ciphers used since 1900. We won't spend time studying mechanical (e.g. rotor) machines in depth, but are included since they played an important role in cryptographic history. Stream ciphers are used frequently to encrypt streaming content, such as cell phone calls and internet traffic, because of its speed, simplicity, security, and the flexibility to be used when the length of the plaintext is unknown.

3.1.1 Mechanical Cipher Machines

The following are examples of some mechanical cipher machines that were in use from the late 1910s into the 1980s.

- A-21 (Arvid Damm)
- Enigma (Arthur Scherbius)
- Hebern Machine (Edward Hebern)
- KL-51 (NSA)
- Lorenz (Germany)

- M209-A (Sold to the United States by Boris Hagelin)
- PURPLE (97-shiki O-bun in-ji-ki)

Read §2.12 of [8]

3.1.2 Random and Pseudorandom Number Generators

So if a one-time pad offers perfect security, an obvious solution would be to somehow generate a random or pseudorandom **keystream**, a stream of bits to be used as the key, and use a bit-wise addition, modulo 2, also called a bit-wise **XOR**, or $\oplus$, to combine it with the plaintext. In this case, the sender and receiver would have to agree on a method to generate a random or pseudorandom sequence of bits and perhaps agree on an additional key (called a **seed**) for the pseudorandom number generator. Before we proceed, it will help to define some terms. (See Chapter 5 of [4].)

Definition 3.1.2. A **random number generator** is a device or algorithm which outputs a sequence of statistically independent and unbiased numbers [4].

Typically, the numbers we will generate are binary digits, or bits, so we may alternatively call such a generator a **random bit generator**.

Definition 3.1.3. A **pseudorandom number generator** (PRNG) is a deterministic algorithm which inputs a truly random digit sequence and outputs an expanded digit sequence which "appears" to be random. The input to the PRNG is called the **seed**, while the output of the PRNG is called a **pseudorandom digit sequence** [4].

Again, we will typically be generating a sequence of pseudorandom bits.

The one word in the definition of a PRNG that needs further elaboration in order for this definition to be rigorous is the word *appear*. What does it mean to *appear* random? There are various statistical tests to measure randomness. (See §5.4.4 of [4] or the Statistical Test Suite developed by NIST [5].) If a sequence of numbers passes all the tests for randomness in a given test suite, then we say the the sequence **appears** random.

Now just because a number sequence is pseudorandom, doesn't mean that it's secure enough for cryptographic purposes. To this end, we introduce another term.

Definition 3.1.4. A PRNG is **cryptographically secure** if

1. its seed is sufficiently long to make it infeasible to perform a brute-force attack on the PRNG by testing all possible seeds and
2. given any sequence of digits generated by the PRNG, there is no polynomial-time algorithm that can predict the next digit with any degree of certainty (e.g., with probability greater than 1/2 for a pseudorandom bit string).

We may soften this definition somewhat by allowing the sequence of digits in point #2 to be sufficiently short, specifically, shorter than the length of any intended plaintext or short enough to be stored digitally. It is also common to weaken point #2 to say that predicting the next bit or digit must be at least as hard as a brute-force attack. We may also add in an assumption that there is sufficient security because the PRNG is based on a mathematical problem that is believed to be solvable in polynomial time.

Read §2.10 of [8].

As a note on chronology, the **Blum-Blum-Shub** PRNG (BBS) was published in 1986 and it was motivated by developments in public-key cryptography, notably the RSA cryptosystem, which we will discuss in Section 5. Its claim of cryptographic security is based on the belief that the **quadratic residuosity problem** is computationally difficult when the modulus n is a large composite number. In other words, given $n = pq$, it is believed to be hard to determine if for some $a \in \mathbb{Z}_n$, whether the congruence $x^2 \equiv a \pmod{n}$ has a solution. This problem is easy if the modulus is prime, so this PRNG relies on the belief that the **integer factorization problem** is difficult for large composite integers.

Problem 3.1.1. [10 points] Let $x_n = 7x_{n-1} + 1 \pmod{10}$ be a linear congruential generator. What are all the possible period lengths of this recurrence?

Problem 3.1.2. [10 points] Let $x_n = 7x_{n-1} + 1 \pmod{11}$ be a linear congruential generator. What are all the possible period lengths of this recurrence?

Problem 3.1.3. [10 points] Based on the results of Problems 3.1.1 and 3.1.2, make a conjecture about how the modulus relates to the possible period lengths of a linear congruential generator. (You may want to experiment further.)

Problem 3.1.4. [10 points] Suppose that you intercepted a message that used the output of a linear congruential generator as the key. Suppose that you determined that the modulus is 17 and that the first three elements of the keystream were 7, 13, and 9. Based on this information, determine the linear congruential generator that was used and the next three elements of the keystream.

Problem 3.1.5. [15 points] Suppose that you intercepted a message that used the output of a linear congruential generator as the key. Suppose that you determined that the modulus is 997 and that the first three elements of the keystream were 6, 795, and 978. Based on this information, determine the linear congruential generator that was used and the next three elements of the keystream.

Problem 3.1.6. [20 points] Suppose that you intercepted a message that used the output of a linear congruential generator as the key and determined the first ten elements of the keystream: 51, 93, 68, 39, 17, 7, 73, 6, 2, and 9. Based on this information, determine the linear congruential generator that was used and the next three elements of the keystream.

Problem 3.1.7 (10 points)**.** Let $p = 263$ and $q = 283$. If we wanted to use these primes to use BBS as a cryptographically secure PRNG, how many bits could we use with each iteration so generate a cryptographically secure pseudorandom bit stream? Use any seed of your choice to generate 12 pseudorandom bits.

Problem 3.1.8. [10 points] If you wanted to use the Blum-Blum-Shub PRNG as a cryptographically secure PRNG and wanted to use 4 bits with each iteration, how many bits would the modulus n have to have? How many bits would p and q have to have?

Problem 3.1.9. [15 points] As a follow-up to Problem 3.1.8, find primes p and q that would allow you to extract four pseudorandom bits with each iteration. Use the resulting BBS PRNG to generate 32 pseudorandom bits (8 bytes) and encrypt "`MAT 337!`" using extended-ASCII to encode the eight characters.

Problem 3.1.10. [10 points] Let $p = 7$ and $q = 11$. What are the possible period lengths of the BBS PRNG?

Problem 3.1.11. [10 points] If $n = pq$ is the modulus of a BBS PRNG, what are some seeds that you would obviously want to avoid?

Problem 3.1.12. [15 points] If $n = pq$, and $x_{k+1} = x_k^2 \pmod{n}$, then

$$x_k = \left(x_0^{2^k} \pmod{\lambda(n)} \right) \pmod{n} \ ,$$

where $\lambda(n) = \text{lcm}((p-1)(q-1))$.

3.1.3 Other Cryptographically Secure PRNGs

The following is a list of other PRNGs that are considered to be cryptographically secure.

- ANSI X9.17 Standard
- Blum-Micali
- CryptGenRandom

- Fortuna
- ISAAC
- NIST SP 800-90A Standards
- Yarrow

Problem 3.1.13 (15 points)**.** Do some basic research on one of the PRNGs in the list above. Describe the method, state where and/or how the PRNG is used in practice, and make some brief comments on its security. If it is feasible, generate a cryptographically secure pseudorandom bit sequence by hand or write a program to generate such a sequence.

3.1.4 Linear Feedback Shift Registers

Read §2.11 of [8].

Problem 3.1.14 (10 points)**.** §2.13 Exercise #19

Problem 3.1.15 (10 points)**.** §2.13 Exercise #20

Problem 3.1.16 (10 points)**.** §2.13 Exercise #21

Problem 3.1.17 (10 points)**.** §2.13 Exercise #22

Problem 3.1.18 (15 points)**.** §2.14 Computer Problem #11

Problem 3.1.19 (15 points)**.** §2.14 Computer Problem #12

Problem 3.1.20 (15 points)**.** §2.14 Computer Problem #13

3.1.5 Examples of Modern Stream Ciphers

- A5/1
- ChaCha (Dan Bernstein)
- ISAAC (Robert Jenkins, Jr.)
- RC4 (Ron Rivest)
- Salsa20 (Dan Bernstein)
- SEAL (Phillip Rogaway and Don Coppersmith)
- Speck (NSA)
- Spritz (Ron Rivest)

Problem 3.1.21 (15 points)**.** Explore one of the stream ciphers listed above, or any other stream cipher. Include:

1. A mathematical description of the cipher, with pictures if it helps to elucidate the description.
2. Encrypt your initials using the equivalent non-extended ASCII encoding with the stream cipher and a key of your choice.
3. Give some examples of applications that use the stream cipher in practice.
4. Briefly describe the security of the stream cipher. Are there any attacks on the stream cipher that are better than a brute-force attack of the key? If so, are these attacks on the cipher practical or impractical? (This is not intended to be overly technical and should be understood by anyone else in the class.)

The disadvantage of just using a stream cipher to encrypt information is that the ciphertext is prone to a **bit-flipping attack**, , a form of a man-in-the-middle attack, in which an eavesdropper intercepts the message and changes some of the ciphertext. For example, if Eve, the eavesdropper intercepted a message from Alice to Bob that read "Alice is sending Bob 4 Bitcoins" and she knew the structure of the message, then she could XOR the binary equivalent of "Bob $\oplus$ Eve" in the appropriate location of the message thus changing it to "Alice is sending Eve 4 Bitcoins" and Eve is suddenly 4 Bitcoins richer! To avoid such an attack, ciphertexts that were encrypted with a stream cipher should include some kind of authentication, such as a message authentication code (MAC) , so that any tampering of the ciphertext would be detected. Man-in-the-middle attacks will be discussed more completely in Section 3.3.5 and MACs will be discussed in Section 6.2.1.

3.2 Block Ciphers

In contrast to a stream cipher, **block ciphers** encrypt multiple characters, bits, or bytes, called **blocks**, at a time. Some classical cryptosystems encrypted blocks of two letters at a time, such as the Playfair Cipher, while modern block ciphers will encrypt plaintext in blocks of 64 or 128 bits, for example. We include two classical cryptosystems here as motivation for modern block ciphers, so with the classical ciphers, the Playfair cipher (§3.2.1) and the Hill cipher (§3.2.2), plaintext and ciphertext will use elements of $\mathcal{L}$, while subsequent sections will use binary.

3.2.1 The Playfair Cipher

Read §2.6 of [8].

Problem 3.2.1. [10 points] Encrypt a plaintext of your choice with at least 100 characters with the Playfair cipher using a key your choice. The rest of the class will be challenged later to cryptanalyze the ciphertext.

Problem 3.2.2 (15 points)**.** Recover the plaintext from a peer's ciphertext, which was encrypted using the Playfair cipher in Probem 3.2.1.

3.2.2 The Hill Cipher

Read §2.7 of [8].

As noted in [8], the Hill cipher was not often used, but is notable for being the first to utilize algebraic techniques. The cipher uses distinct encryption and decryption functions (namely, linear transformations) which are inverses of each other. Its downside is that the decryption function is easy to determine from the encryption function because finding the inverse of an invertible matrix is computationally easy.

Problem 3.2.3 (10 points)**.** §2.13 Exercise #13

Problem 3.2.4 (10 points)**.** §2.13 Exercise #14

Problem 3.2.5 (15 points)**.** §2.13 Exercise #15

Problem 3.2.6 (15 points)**.** §2.13 Exercise #16

Problem 3.2.7 (15 points)**.** §2.13 Exercise #17

Problem 3.2.8 (20 points)**.** §2.13 Exercise #18

Problem 3.2.9 (10 points)**.** §2.14 Computer Problem #10

Problem 3.2.10. [10 points] Give an example of an invertible 3×3 matrix with entries in $\mathbb{Z}_{26}$ and find the inverse of the matrix.

Problem 3.2.11 (10 points)**.** How many invertible 2×2 matrices with entries in $\mathbb{Z}_{26}$ are there?

Problem 3.2.12 (10 points)**.** How many invertible 2×2 matrices with entries in $\mathbb{Z}_{37}$ are there?

Problem 3.2.13 (10 points)**.** How many invertible 3×3 matrices with entries in $\mathbb{Z}_{26}$ are there?

Problem 3.2.14 (10 points)**.** How many invertible 3×3 matrices with entries in $\mathbb{Z}_{37}$ are there?

Problem 3.2.15 (15 points)**.** Let $n \in \mathbb{N}$ and let p be a prime. How many invertible $n \times n$ matrices with entries in $\mathbb{Z}_p$ are there?

Problem 3.2.16. [10 points] Encrypt a plaintext of your choice with at least 100 characters with the Hill cipher using a key your choice. (This key could be any matrix found in Problem 3.2.10. The rest of the class will be challenged later to cryptanalyze the ciphertext.

Problem 3.2.17 (10 points if the key is known; 20 points if the key is unknown)**.** Recover the plaintext from a peer's ciphertext, which was encrypted using the Hill cipher in Problem 3.2.16.

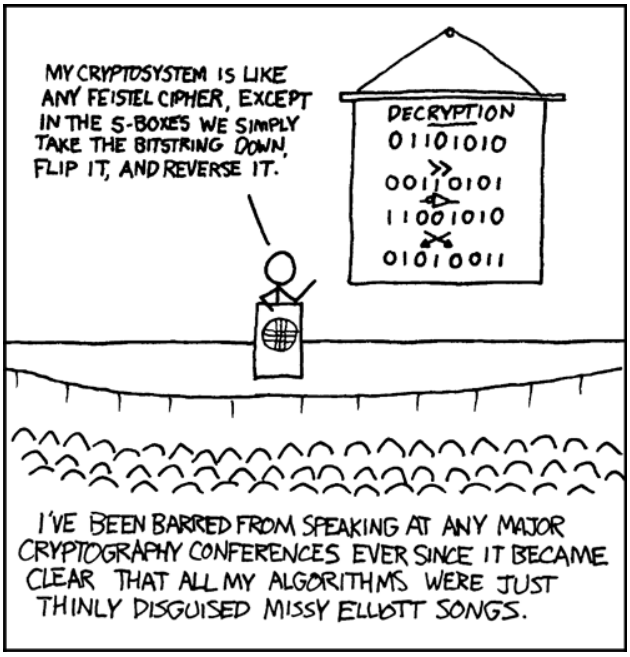
3.2.3 Data Encryption Standard (DES)

Read §4.1, 2, 4, and 5 of [8].

Discussion What are some of the big ideas in the development of DES? What is a Feistel system?

Definition 3.2.1. A function f is called a **pseudorandom function** if f is an efficient deterministic function whose output cannot be efficiently distinguished from random data.

Discussion Explain why the designers of DES needed a function f, as denoted in the description of DES and a simplified DES-type algorithm, that is a pseudorandom function.



Problem 3.2.18 (20 points)**.** Briefly describe the Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes of operation for block ciphers. What are any strengths and/or weaknesses of each?

3.2.4 Other Feistel-based Block Ciphers

- Blowfish
- Camellia

- CAST-128 and CAST-256
- RC5
- RC6
- Skipjack
- TEA (Tiny Encryption Algorithm) and variants
- Twofish
- 3DES (Triple DES) (§4.6 of [8])

Problem 3.2.19 (20 points)**.** Although many block ciphers are proprietary, the description of many block ciphers have been published. Do some basic research on one of the Feistel-based block ciphers above. Answer as many of the following questions as you can.

1. Who created the cipher, what is (are) their affiliation(s), and when was it developed?
2. What applications use the block cipher?
3. What size keys does the cipher use and what is the block size that the cipher uses?
4. Does the cipher make any modifications to the pure Feistel structure?
5. Describe the pseudorandom function f and any design considerations of the creators.
6. Briefly state the best known attacks on the cipher. (You will likely come across terminology that we haven't defined yet. That's OK, much of it is quite technical, and later we'll only get a very rough idea of some of the concepts. You're certainly not expected to understand how differential cryptanalysis works, for example.)

3.2.5 Other Block Cipher Designs and Components

- Key Schedule
- S-box (substitution box)
- P-box (permutation box)
- Lai-Massey Scheme
  - IDEA

3.2.6 Advanced Encryption Standard (AES)

Read Chapter 5 of [8].

3.3 Cryptanalysis of Stream and Block Ciphers

3.3.1 Brute Force Attack

Definition 3.3.1. A **brute force attack** on a cryptogram is a cryptanalytic technique in which keys are systematically used to decrypt until the correct key is found.

Problem 3.3.1 (10 points)**.** If a message is encrypted using a 64-bit key, what is an upper bound on the amount of time it would take 1000 computers to crack the ciphertext if each computer can test a million keys per second?

Problem 3.3.2 (10 points)**.** If a message is encrypted using a 128-bit key, what is an upper bound on the amount of time it would take a billion computers to crack the ciphertext if each computer can test a billion keys per second?

3.3.2 Known-Plaintext Attacks

Read §1.1.1 #2 of [8].

3.3.3 Chosen-Plaintext Attacks

Read §1.1.1 #3 of [8].

3.3.4 Differential Cryptanalysis

Read §4.3 of [8].

3.3.5 Meet-in-the-Middle Attacks

Read §4.7 of [8].

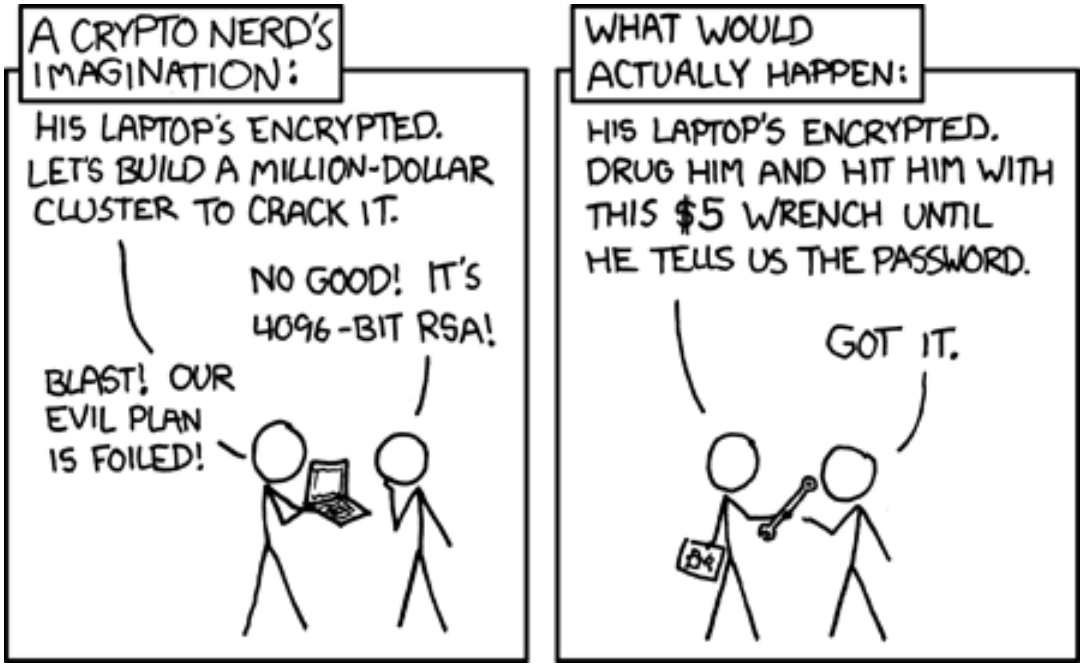
3.3.6 Black-bag cryptanalysis

This isn't mathematical cryptanalysis, but **black-bag cryptanalysis** refers to a generally more practical form of cryptanalysis in which encryption and/or decryption keys or passwords are stolen by any number of means. This may include the following.

- Installation of Trojan horse software on a computer to send passwords to an attacker.
- Installation of a keystroke logger to record typing.
- Finding a password written on a piece of paper.
- Installation of bugs (video and/or audio surveillance) in a room.
- Reading electromagnetic transmissions from an electronic device.
- Social engineering

Despite the nontechnical aspects of black-bag methods, it is important to be aware of these forms of cryptanalysis since otherwise unbreakable encryption can be broken if the key is revealed on account of carelessness.

3.3.7 Rubber-hose cryptanalysis



As with black-bag methods, **rubber-hose cryptanalysis** is not a mathematical technique, but rather any method of recovering encrypted content, a password, or encryption/decryption keys via blackmail, coercion, threats, or torture.

Problem 3.3.3 (15 points)**.** §4.9 Exercise #1.a.

Problem 3.3.4 (15 points)**.** §4.9 Exercise #1.b.

Problem 3.3.5 (15 points)**.** §4.9 Exercise #1.c.

Problem 3.3.6 (10 points)**.** §4.9 Exercise #2.

Problem 3.3.7 (15 points)**.** §4.9 Exercise #3.

Problem 3.3.8 (15 points)**.** §4.9 Exercise #4.

Problem 3.3.9 (15 points)**.** §4.9 Exercise #5.

Problem 3.3.10 (15 points)**.** §4.9 Exercise #6.

Problem 3.3.11 (15 points)**.** §4.9 Exercise #7.

Problem 3.3.12 (10 points)**.** §4.9 Exercise #8.

Problem 3.3.13 (15 points)**.** §4.9 Exercise #9.

Problem 3.3.14 (10 points)**.** §4.9 Exercise #10.

Problem 3.3.15 (10 points)**.** §4.9 Exercise #11.

Problem 3.3.16 (15 points)**.** §4.10 Computer Problem #1.a and 1.b.

Problem 3.3.17 (10 points)**.** §4.10 Computer Problem #1.c.

Problem 3.3.18 (10 points)**.** §4.10 Computer Problem #1.d.

Problem 3.3.19 (10 points)**.** §4.10 Computer Problem #2.

Problem 3.3.20 (10 points)**.** Explore a topic that came out of a class discussion or solve a problem inspired by the content of this chapter.

Chapter 4

Mathematical Interlude

4.1 Hash Functions

Definition 4.1.1. A **hash function** $h : \{0,1\}^* \to \mathbb{N}$ is a function that maps an arbitrarily long input to an integer of some fixed size.

Hash functions are commonly used in computer science primarily to store data in large arrays in a way that makes the data quick and easy to store and retrieve. Hash functions have different uses in cryptography, but require two additional properties in order to make them useful for cryptographic purposes.

Definition 4.1.2. A **cryptographically secure hash function** is a hash function h such that

1. h is **one-way**, that is, given some element $r \in \mathbb{N}$ in the range of h, it is computationally infeasible to compute an input m such that $h(m) = r$; and
2. it is computationally infeasible to find a **collision**, that is, two distinct elements m_1 and m_2 in the domain of h such that $h(m_1) = h(m_2)$.

The following are examples of hash functions that have once been or are considered to be cryptographically secure.

- MD5 (Ron Rivest)
- MD6 (Ron Rivest et al.)
- SHA-1 (NSA)
- SHA-2 (NSA)
- SHA-3 / Keccak (Guido Bertoni, Joan Daemen, Gilles Van Assche, and Michaël Peeters

Read §8.1-8.3, 8.7 of [8].

4.2 The Birthday Problem

The **birthday problem** is a standard cryptanalytic technique that is used to attack hash functions. It is also at the heart of an algorithm called Pollard's Rho algorithm for factoring integers and computing discrete logarithms. (§8.4.1 of [8])

Read §8.4 of [8].

Problem 4.2.1. (10 points) §8.8 Exercise #1

Problem 4.2.2. (10 points) §8.8 Exercise #3

Problem 4.2.3. (20 points) §8.8 Exercise #5

Problem 4.2.4. (10 points) §8.8 Exercise #6

Problem 4.2.5. (5 points) §8.8 Exercise #8

Problem 4.2.6. (10 points) §8.8 Exercise #9

Problem 4.2.7. (15 points) §8.8 Exercise #10

Problem 4.2.8. (20 points) §8.8 Exercise #11

Problem 4.2.9. (10 points) §8.9 Computer Problem #1

Problem 4.2.10. (10 points) §8.9 Computer Problem #2

4.3 Group Theory

Definition 4.3.1. A **group** is a nonempty set G, together with a binary operation, $*$, such that the following four axioms hold.

1. G is **closed** under the operation $*$. That is, for all $a, b \in G$, $a * b \in G$.
2. The operation $*$ is **associative** in G. That is, for all $a, b, c \in G$, $(a * b) * c = a * (b * c)$.
3. G possesses an **identity element**, e. That is, for all $a \in G$, $e * a = a * e = a$.
4. Every element $a \in G$ has an **inverse** in G. That is, for all $a \in G$, there is some element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$. (If * is an additive operation, then the inverse of a is typically denoted $-a$.

Notice that groups need not be commutative. Such groups have a special name.

Definition 4.3.2. If any two elements $a, b \in G$ commute under the operation $*$, that is, if $a * b = b * a$, for all $a, b \in G$, then G is called an **abelian** group.

Problem 4.3.1 (10 points)**.** Show that the set $\mathbb{Z}_n$ is an abelian group under the operation of addition modulo n.

Problem 4.3.2 (10 points)**.** Show that the set $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$ is an abelian group under the operation of multiplication modulo p.

Problem 4.3.3 (10 points)**.** Show that the set $\mathbb{Z}_n^* = \{1 \leq a < n \mid \gcd(a, n) = 1\}$ is an abelian group under the operation of multiplication modulo n.

Definition 4.3.3. The number of elements of a group G is called its **order** and is denoted $|G|$.

Definition 4.3.4. Euler's phi (or totient) function is defined to be $\varphi(n) = |\mathbb{Z}_n^*| = |\{1 \leq a < n \mid \gcd(a, n) = 1\}|$.

Definition 4.3.5. If G is a group, $H \subseteq G$, and H is a group, then H is called a **subgroup** of G and we write $H \leq G$.

Definition 4.3.6. The subgroup of G **generated** by $g \in G$ is the set $\langle g \rangle = \{g^n \mid n \in \mathbb{Z}\}$. If there is some $g \in G$ such that $\langle g \rangle = G$, then G is called a **cyclic** group and g is called a **generator** of G.

Definition 4.3.7. If p is a prime and $r \in \mathbb{Z}_p^*$ such that $\langle r \rangle = \mathbb{Z}_p^*$, then r is called a **primitive root** modulo p.

Problem 4.3.4 (15 points)**.** Prove that all cyclic groups are abelian.

Problem 4.3.5 (10 points)**.** Find a generator of $\mathbb{Z}_{97}^*$.

Theorem 4.3.1 (Lagrange)**.** If $H \leq G$, then $|H| \mid |G|$.

Problem 4.3.6 (15 points)**.** Prove Lagrange's Theorem (Theorem 4.3.1).

4.4 Modular Exponentiation

Read §3.5 of [8].

In many situations, we must perform some exponentiation, say $a^x \pmod m$, for some $m \in \mathbb{N}$, $a \in \mathbb{Z}_m$, and $x \in \mathbb{Z}$. If $|x|$ is very large, then we certainly do not want to multiply a (or its inverse) by itself $|x|$ times. A common approach to compute this number is **binary exponentiation**. It is best illustrated with an example.

Example. Compute: $2^{42} \pmod{97}$.

We compute this using the binary representation of the exponent and make successive squarings.

$2^{42} = 2^{32+8+2} = (2^{32})(2^8)(2^2)$.

k	2^k	$2^{2^k} \pmod{97}$
0	1	2
1	2	$2^2 = 4$
2	4	$4^2 = 16$
3	8	$16^2 \equiv 62 \pmod{97}$
4	16	$62^2 \equiv 61 \pmod{97}$
5	32	$61^2 \equiv 35 \pmod{97}$

So

$$2^{42} = (2^{32})(2^8)(2^2) \equiv 35 \cdot 62 \cdot 4 \equiv 47 \pmod{97} \ .$$

Problem 4.4.1 (10 points)**.** Compute $3^{409} \pmod{997}$.

Problem 4.4.2 (10 points)**.** Compute $3^{-251} \pmod{997}$.

Problem 4.4.3 (15 points)**.** What is the **non-adjacent form** (NAF) of an integer and how can it be used to improve binary exponentiation?

Fermat's Little Theorem

Theorem 4.4.1 (Fermat's Little Theorem)**.** If p is a prime and $a \in \mathbb{Z}$ such that $p \nmid a$, then $a^{p-1} \equiv 1 \pmod p$.

Problem 4.4.4 (10 points)**.** Prove Fermat's Little Theorem.

Problem 4.4.5 (15 points)**.** In Section 4.3, we defined a primitive root of a prime p as an integer r such that $\langle r \rangle = \mathbb{Z}_p^*$. Describe a method to determine whether or not a given $r \in \mathbb{Z}$ is a primitive root modulo p that is more efficient than a brute-force search.

Euler's Theorem

Theorem 4.4.2 (Euler)**.** If $a, n \in \mathbb{Z}$ such that $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod n$.

Problem 4.4.6 (10 points)**.** Prove Euler's Theorem.

Problem 4.4.7 (10 points)**.** §3.13 #2 of [8].

Problem 4.4.8 (10 points)**.** §3.13 #11 of [8].

Problem 4.4.9 (10 points)**.** §3.13 #12 of [8].

Problem 4.4.10 (10 points)**.** §3.13 #13 of [8].

Problem 4.4.11 (10 points)**.** §3.13 #15 of [8].

Problem 4.4.12 (10 points)**.** §3.13 #16 of [8].

Problem 4.4.13 (10 points)**.** §3.13 #17 of [8].

Problem 4.4.14 (20 points)**.** §3.13 #20 of [8].

Problem 4.4.15 (20 points)**.** §3.13 #21 of [8].

Problem 4.4.16 (10 points)**.** §3.14 #3 of [8].

Problem 4.4.17 (10 points)**.** §3.14 #7 of [8].

Problem 4.4.18 (10 points)**.** §3.14 #9 of [8].

Carmichael Function

Definition 4.4.1. The **Carmichael function**, $\lambda : \mathbb{N} \to \mathbb{N}$, on input n is the smallest positive integer $\lambda(n)$ such that $a^{\lambda(n)} \equiv 1 \pmod{n}$ for every a that is relatively prime to n.

Quadratic Residues and Modular Square Roots Stretching the concept of modular exponentiation, we can consider modular square roots. This has some interesting cryptographic applications that we will encounter later. Let's establish some terminology.

Definition 4.4.2. Let $n \in \mathbb{Z}$. An integer a is said to be a **quadratic residue** modulo n if there is some $x \in \mathbb{Z}$ such that $x^2 \equiv a \pmod{n}$. If a is a quadratic residue modulo n and $x^2 \equiv a \pmod{n}$, then x is a **square root** of a modulo n and we can write $x \equiv \sqrt{a} \pmod{n}$.

For the case that $n = p$ is an odd prime, the **Legendre symbol** is defined:

$$\left(\frac{a}{p}\right) = \begin{cases} -1 & \text{if } a \text{ is not a quadratic residue modulo } p \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p \\ 0 & \text{if } p \mid a. \end{cases}$$

Theorem 4.4.3 (Euler's Criterion)**.** If p is an odd prime and $a \in \mathbb{Z}$ such that $\left(\frac{a}{p}\right) = 1$, then

$$a^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right) \pmod{p}.$$

Problem 4.4.19 (15 points)**.** Prove Euler's Criterion.

Problem 4.4.20 (15 points)**.** Prove that if p is an odd prime and $p \nmid ab$, then

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right).$$

Problem 4.4.21 (10 points)**.** If p is an odd prime, determine $\left(\frac{-1}{p}\right)$.

Problem 4.4.22 (10 points)**.** If p is an odd prime and $a \equiv b \pmod{p}$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.

A much more efficient way to determine whether a given integer is a quadratic residue modulo an odd prime is to apply the Law of Quadratic Reciprocity.

Theorem 4.4.4 (Quadratic Reciprocity)**.** Let p and q be odd primes.

$$\left(\frac{2}{p}\right) = 1 \text{ if and only if } p \equiv 1, 7 \pmod{8}$$

$$\left(\frac{q}{p}\right) = \begin{cases} -\left(\frac{p}{q}\right) & \text{if } p, q \equiv 3 \pmod{4} \\ \left(\frac{p}{q}\right) & \text{otherwise.} \end{cases}$$

Problem 4.4.23 (10 points)**.** Find all quadratic residues modulo 19.

Now we turn to actually computing modular square roots. For primes congruent to 1 modulo 4, the process can be a little involved. Fortunately, it is straightforward for all other primes, and these are the kinds of primes that cryptographers are most interested in.

Theorem 4.4.5. If $p \equiv 3 \pmod{4}$ is prime and $\left(\frac{a}{p}\right) = 1$, then $\sqrt{a} \equiv \pm a^{(p+1)/4} \pmod{p}$.

Problem 4.4.24 (10 points)**.** Determine $\pm\sqrt{31} \pmod{43}$.

Problem 4.4.25 (15 points)**.** Prove Theorem 4.4.5.

Problem 4.4.26 (10 points)**.** Show that if p is an odd prime and $a \in \mathbb{Z}$ such that $\left(\frac{a}{p}\right) = 1$, then $x^2 \equiv a \pmod{p}$ has two distinct solutions in $\mathbb{Z}_p^*$. (In other words, a has two square roots modulo p.)

Now if $n = pq$ is a product of primes, then we can compute square roots modulo n.

Theorem 4.4.6. Let $n = pq$ be a product of distinct primes and let $a \in \mathbb{Z}$ such that $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$. Then a is a quadratic residue modulo n and

$$\sqrt{a} \equiv r_1 dq + r_2 cp \pmod{n} \ ,$$

where $r_1 \equiv \sqrt{a} \pmod{p}$, $r_2 \equiv \sqrt{a} \pmod{q}$, and $c, d \in \mathbb{Z}$ such that $cp + dq = 1$.

Notice that since $p = q$, $\gcd(p, q) = 1$, so there do exist $c, d \in \mathbb{Z}$ such that $cp + dq = 1$.

Problem 4.4.27 (15 points)**.** Prove Theorem 4.4.6.

Problem 4.4.28 (10 points)**.** Find a quadratic residue modulo 77 and compute its square root modulo 77.

Problem 4.4.29 (15 points)**.** Show that if p and q are distinct odd primes, $n = pq$ and $a \in \mathbb{Z}$ such that $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$, then $x^2 \equiv a \pmod{n}$ has four distinct solutions in $\mathbb{Z}_p^*$. (In other words, a has four square roots modulo n.)

4.5 Discrete Logarithms

Definition 4.5.1. Let G be a group, $g \in G$, and $h \in \langle g \rangle$. That is, there is some $x \in \mathbb{N}_0$ such that $g^x = h$. The integer x is called the **discrete logarithm** of h in $\langle g \rangle$. If p is prime and $g \in \mathbb{Z}_p^*$ such that $\langle g \rangle = \mathbb{Z}_p^*$, then for some $h \in \mathbb{Z}_p^*$, the integer $x \in \mathbb{N}_0$ such that $g^x \equiv h \pmod{p}$ is the discrete logarithm of h to base g modulo p.

Read §7.1 and 7.2 in [8].

Problem 4.5.1 (10 points)**.** Compute $\log_2(3) \pmod{13}$. Compute $\log_2(11) \pmod{13}$.

Problem 4.5.2 (10 points)**.** Use Pohlig-Hellman to compute $\log_2(14) \pmod{19}$.

Problem 4.5.3 (15 points)**.** §7.6 #5 of [8]

Problem 4.5.4 (15 points)**.** §7.6 #6 of [8]

Problem 4.5.5 (15 points)**.** §7.6 #7 of [8]

Problem 4.5.6 (10 points)**.** §7.6 #8 of [8]

Problem 4.5.7 (10 points)**.** §7.7 #2 of [8]

Problem 4.5.8 (5 points)**.** §7.7 #3 of [8]

Problem 4.5.9 (10 points)**.** §7.7 #4 of [8]

4.6 Primality Testing

Read §6.3 of [8].

Problem 4.6.1 (10 points)**.** §6.9 #13 of [8]

Problem 4.6.2 (15 points)**.** Describe a method to test whether a given $n \in \mathbb{N}$ is composite or possibly prime.

4.7 Integer Factorization

Read §6.4 of [8].

Problem 4.7.1 (10 points)**.** §6.8 #12 of [8]

Problem 4.7.2 (10 points)**.** §6.8 #13 of [8]

Problem 4.7.3 (10 points)**.** §6.9 #4 of [8]

Problem 4.7.4 (10 points)**.** §6.9 #5 of [8]

Problem 4.7.5 (10 points)**.** §6.9 #6 of [8]

Problem 4.7.6 (10 points)**.** §6.9 #8 of [8]

Problem 4.7.7 (10 points)**.** §6.9 #9 of [8]

Problem 4.7.8 (10 points)**.** §6.9 #10 of [8]

Problem 4.7.9 (10 points)**.** §6.9 #12 of [8]

4.8 Finite Fields

We learned about Linear Feedback Shift Registers earlier. To get a deeper understanding into why they function the way they do, we turn to finite fields. This section is also applied to many areas of cryptography and coding theory.

Read §3.11 of [8].

Definition 4.8.1. A **field** is a nonempty set K, together with two commutative binary operations, addition and multiplication, having the following properties.

- K is an abelian group with identity 0 under addition.
- $K^* = K \setminus \{0\}$ is an abelian group with identity 1 under multiplication.
- $a(b+c) = ab + ac$ for all $a, b, c \in K$.

Problem 4.8.1 (10 points)**.** Give three examples of fields of infinite order (cardinality).

Problem 4.8.2 (10 points)**.** Show that $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ is a field if and only if p is prime. ($\mathbb{Z}_p$ will often be denoted $\mathbb{F}_p$ or $GF(p)$.)

Theorem 4.8.1. If K is a finite field, then $|K| = p^n$, for some prime p and $n \in \mathbb{N}$. $K \cong \mathbb{F}_p[x]/\langle f(x)\rangle$, where $f(x) \in \mathbb{F}_p[x]$ is irreducible.

Problem 4.8.3 (20 points)**.** Show that if p is a prime and $f(x) \in \mathbb{F}_p[x]$ is irreducible, then $\mathbb{F}_p[x]/\langle f(x)\rangle$ is a field of order p^n.

Problem 4.8.4 (15 points)**.** §3.13 #33 of [8].

Problem 4.8.5 (10 points)**.** §3.13 #34 of [8].

Problem 4.8.6 (15 points)**.** Let $p = 2$. Find an irreducible cubic polynomial over $\mathbb{F}_2$. Describe the elements of $\mathbb{F}_{2^3}$.

Chapter 5

Public-Key Cryptography

5.1 The Concept

Read §6.7 of [8].

The concept of Diffie-Hellman Key Exchange can be understood even by elementary school children with an activity called "Chocolate key cryptography," using M&Ms [1].

5.2 Diffie-Hellman-(Merkle) Key Exchange

Read §7.4 of [8].

Problem 5.2.1 (15 points)**.** Describe how to extend the Diffie-Hellman Key Exchange for three people to share a common secret key.

Problem 5.2.2 (15 points)**.** Describe how to extend the Diffie-Hellman Key Exchange for n people to share a common secret key.

Problem 5.2.3 (15 points)**.** §7.6 #10 of [8]

5.3 Kid Krypto

Kid Krypto is a warm-up to the RSA public-key cryptosystem. Many of the concepts are identical, but the mathematics of Kid Krypto is a little more basic.

If Bobby would like to receive a secret message from his friends, he first chooses any four positive integers that only he will know: a, b, A, and B. Then he computes

$$\begin{aligned} M &= ab - 1 \\ e &= AM + a \\ d &= BM + b \\ n &= \frac{ed - 1}{M} \end{aligned}$$

Bobby tells everyone the numbers e (for encryption) and n. To send a plaintext message $m \in \mathbb{Z}_n$, the sender computes

$$c = em \pmod{n} \ .$$

To decipher the ciphertext c, Bobby computes

$$dc \pmod{n} \ .$$

Problem 5.3.1 (10 points)**.** Let $a = 47$, $b = 22$, $A = 11$, and $B = 5$. Compute M, e, d, and n.

1. Encode the plaintext message $m = 2020$ using e and n above. (Check that $dc \pmod n = m$.)
2. Decode the encrypted message $c = 43155$. (Check that $em \pmod n = c$.)

Problem 5.3.2. [10 points] Bobby announces his public key: $n = 17239722505$, $e = 25540219$. Alice sends him the encrypted message $c = 7218695996$. What is Alice's message?

Problem 5.3.3 (10 points)**.** Explain why Kid Krypto works; that is, how do you know that $m = dc \pmod n$?

Problem 5.3.4 (10 points)**.** Is Kid Krypto secure? Discuss its strengths and weaknesses.

Problem 5.3.5 (10 points)**.** Using the same values of n and e from Problem 5.3.2, decode the value of m from the intercepted message $c = 6599969821$. The number you've found is not just a randomly chosen number. It corresponds to an English word or phrase. What is the word and how is it derived from m?

5.4 RSA

Read §6.1 (6.2 is optional) of [8].

Problem 5.4.1 (10 points)**.** §6.8 #1 of [8]

Problem 5.4.2 (10 points)**.** §6.8 #2 of [8]

Problem 5.4.3 (10 points)**.** §6.8 #3 of [8]

Problem 5.4.4 (10 points)**.** §6.8 #4 of [8]

Problem 5.4.5 (10 points)**.** §6.8 #5 of [8]

Problem 5.4.6 (10 points)**.** §6.8 #6 of [8]

Problem 5.4.7 (15 points)**.** §6.8 #7 of [8]

Problem 5.4.8 (10 points)**.** §6.8 #8 of [8]

Problem 5.4.9 (20 points)**.** §6.8 #9 of [8]

Problem 5.4.10 (15 points)**.** §6.8 #11 of [8]

Problem 5.4.11 (15 points)**.** §6.8 #16 of [8]

Problem 5.4.12 (10 points)**.** §6.8 #17 of [8]

Problem 5.4.13 (15 points)**.** §6.8 #19 of [8]

Problem 5.4.14 (10 points)**.** §6.8 #22 of [8]

Problem 5.4.15 (5 points)**.** §6.9 #1 of [8]

Problem 5.4.16 (10 points)**.** §6.9 #2 of [8]

Problem 5.4.17 (10 points)**.** §6.9 #3 of [8]

Problem 5.4.18 (20 points)**.** §7.6 #12 of [8]

5.5 ElGamal

Read §7.5 of [8].

Problem 5.5.1 (10 points)**.** §7.6 #11 of [8]

5.6 Elliptic Curve Cryptography

There are special groups arising from curves called **elliptic curves** that are ubiquitous in cryptography today. An understanding of group theory and finite fields will facilitate understanding these objects more fully and will allow for problems to be solved much more easily.

Read Chapter 16 of [8].

Problem 5.6.1 (15 points)**.** §16.7 #2 of [8]

Problem 5.6.2 (10 points)**.** §16.7 #3 of [8]

Problem 5.6.3 (10 points)**.** §16.7 #4 of [8]

Problem 5.6.4 (10 points)**.** §16.7 #5 of [8]

Problem 5.6.5 (15 points)**.** §16.7 #6 of [8]

Problem 5.6.6 (15 points)**.** §16.7 #7 of [8]

Problem 5.6.7 (10 points)**.** §16.7 #8 of [8]

Problem 5.6.8 (20 points)**.** §16.7 #9 of [8]

Problem 5.6.9 (10 points)**.** §16.7 #10 of [8]

Problem 5.6.10 (15 points)**.** §16.7 #11 of [8]

Problem 5.6.11 (15 points)**.** §16.7 #12 of [8]

Problem 5.6.12 (15 points)**.** §16.7 #13 of [8]

Problem 5.6.13 (20 points)**.** §16.7 #14 of [8]

Problem 5.6.14 (20 points)**.** §16.7 #15 of [8]

Problem 5.6.15 (15 points)**.** §16.7 #16 of [8]

Problem 5.6.16 (15 points)**.** §16.7 #17 of [8]

Problem 5.6.17 (15 points)**.** §16.7 #18 of [8]

Problem 5.6.18 (15 points)**.** §16.7 #19 of [8]

Problem 5.6.19 (20 points)**.** §16.7 #20 of [8]

Problem 5.6.20 (15 points)**.** §16.8 #1 of [8]

Problem 5.6.21 (10 points)**.** §16.8 #2 of [8]

Problem 5.6.22 (20 points)**.** §16.8 #3 of [8]

Problem 5.6.23 (15 points)**.** §16.8 #4 of [8]

Problem 5.6.24 (10 points)**.** §16.8 #5 of [8]

Problem 5.6.25 (20 points)**.** Give the mathematical details of Curve25519 and explain the difference in the representation of this curve from the standard Weierstrass form of an elliptic curve. Who created the curve, what were the design considerations, and why is this curve used in so many applications today?

Chapter 6

Cryptographic Protocols

6.1 Zero-Knowledge Proofs

Read §14.1 of [8].

Problem 6.1.1 (10 points). §14.3 #1 of [8]

Problem 6.1.2 (15 points). §14.3 #2[a,b] of [8]

Problem 6.1.3 (10 points). §14.3 #2[c] of [8]

Problem 6.1.4 (10 points). §14.3 #3[a] of [8]

Problem 6.1.5 (15 points). §14.3 #3[b, c] of [8]

Problem 6.1.6 (15 points). §14.3 #4[a, b] of [8]

Problem 6.1.7 (15 points). §14.3 #4[c, d] of [8]

Problem 6.1.8 (15 points). §14.3 #5 of [8]

Problem 6.1.9 (15 points). §14.3 #6 of [8]

6.2 Identification Schemes and Authentication

Read §14.2 of [8].

6.2.1 Message Authentication Codes

Definition 6.2.1. A **message authentication code (MAC)** is data transmitted with a message that verifies that the message came from the stated sender (authentication) and that the data has not been altered during the transmission (integrity).

The following are examples of MAC algorithms and frameworks

- CBC-MAC
- Checksum (using a hash function), e.g. md5sum, sha1sum
- HMAC
- Poly1305 and Poly1305-AES (Dan Bernstein)

Problem 6.2.1 (15 points). 1. Who created the MAC, what is (are) their affiliation(s), and when was it developed?

2. How is the MAC or MAC framework used?

3. Discuss the details of the MAC or framework. Pictures help.

4. Discuss the security of the MAC or MAC framework.

6.3 Digital Signatures

Read Chapter 9 of [8].

Another signature scheme is Rabin's Signature Algorithm. Suppose that Alice sends a message M to Bob. She wants to sign the message to prove to Bob that she alone sent the message.

1. Alice chooses two distinct primes $p, q \equiv 3 \pmod 4$ and keeps these two primes private. (Ideally, p and q are **safe** primes, i.e. primes such that $p = 2r + 1$ and $q = 2s + 1$, where r and s are prime.)

2. Alice and Bob agree on a cryptographically secure hash function, h.

3. Alice chooses a random padding u and concatenates $m = M||u$.

4. If $h(m)$ is not a quadratic residue modulo p and modulo q, go back to Step 3.

5. Alice computes $s = \sqrt{h(m)}$ and sends s and m to Bob.

6. Bob verifies that $s^2 \equiv h(m) \pmod n$.

Problem 6.3.1 (10 points)**.** Why is Rabin's Signature Algorithm secure?

Problem 6.3.2 (10 points)**.** Choose two 3-digit safe primes and sign the message $M = 1618$. Use a 1-digit pad and skip the use of a hash function.

Problem 6.3.3 (15 points)**.** §9.6 #1 of [8]

Problem 6.3.4 (15 points)**.** §9.6 #2 of [8]

Problem 6.3.5 (10 points)**.** §9.6 #3 of [8]

Problem 6.3.6 (10 points)**.** §9.6 #4[a] of [8]

Problem 6.3.7 (10 points)**.** §9.6 #4[b] of [8]

Problem 6.3.8 (10 points)**.** §9.6 #4[c] of [8]

Problem 6.3.9 (15 points)**.** §9.6 #5 of [8]

Problem 6.3.10 (10 points)**.** §9.6 #6 of [8]

Problem 6.3.11 (10 points)**.** §9.6 #7 of [8]

Problem 6.3.12 (15 points)**.** §9.6 #8 of [8]

Problem 6.3.13 (10 points)**.** §9.7 #1 of [8]

Problem 6.3.14 (10 points)**.** §9.7 #2 of [8]

Problem 6.3.15 (10 points)**.** §9.7 #3 of [8]

6.4 Secret Sharing and Threshold Schemes

Read Chapter 12 of [8].

Problem 6.4.1 (10 points)**.** Let p be the smallest prime greater than 2^{20} and let $q(x) = ax^2+bx+c \pmod p$ be a quadratic polynomial modulo p. A PIN has been encoded as the constant term, c, of $q(x)$. Alice, Bob, and Charlie have been given the points of the curve $y = q(x)$: $(984544, 900317)$, $(248781, 118521)$, and $741200, 198787)$, respectively. Recover the PIN.

Problem 6.4.2 (10 points)**.** §12.3 #1 of [8]

Problem 6.4.3 (10 points)**.** §12.3 #2 of [8]

Problem 6.4.4 (5 points)**.** §12.3 #3 of [8]

Problem 6.4.5 (10 points)**.** §12.3 #4 of [8]

Problem 6.4.6 (10 points)**.** §12.3 #5 of [8]

Problem 6.4.7 (15 points)**.** §12.3 #6 of [8]

Problem 6.4.8 (15 points)**.** §12.3 #7 of [8]

Problem 6.4.9 (10 points)**.** §12.3 #8 of [8]

Problem 6.4.10 (15 points)**.** §12.3 #9 of [8]

Problem 6.4.11 (15 points)**.** §12.3 #10 of [8]

Problem 6.4.12 (10 points)**.** §12.3 #11 of [8]

Problem 6.4.13 (10 points)**.** §12.4 #2 of [8]

Problem 6.4.14 (10 points)**.** §12.3 #3 of [8]

6.5 Digital Coin-Flipping

Read §13.1 of [8].

Problem 6.5.1 (10 points)**.** §13.3 #1[a] of [8]

Problem 6.5.2 (10 points)**.** §13.3 #1[b,c] of [8]

Problem 6.5.3 (10 points)**.** §13.3 #1[d] of [8]

Problem 6.5.4 (10 points)**.** §13.3 #2[a] of [8]

Problem 6.5.5 (15 points)**.** §13.3 #2[b,c] of [8]

Problem 6.5.6 (10 points)**.** §13.3 #2[d] of [8]

Problem 6.5.7 (15 points)**.** §13.3 #3[a, b] of [8]

Problem 6.5.8 (10 points)**.** §13.3 #3[c] of [8]

6.6 Data Integrity

Read §10.8 of [8].

Problem 6.6.1 (10 points)**.** §10.9 #4 of [8]

Problem 6.6.2 (10 points)**.** §10.9 #6 of [8]

Chapter 7

Blockchain and Cryptocurrencies

- Digital cash and cryptocurrencies
  Before Bitcoin, digital cash systems relied on a central certifying authority (a bank) to regulate any transaction and generally did not discuss the creation of money. Bitcoin changed the digital cash paradigm by uniting the creation of digital money and the decentralization of transaction authentication. Since Bitcoin, hundreds of other so-called *cryptocurrencies* have been created based on Bitcoin concepts. For background on digital cash, read §11.1 in [8].

- The blockchain concept
  Bitcoin introduced the concept of the blockchain, but numerous other applications of this digital technology have been considered. Describe the mathematics and algorithms of a blockchain and its potential uses besides cryptocurrencies.

- KutzCoins
  Make progress towards a system in which students are compensated with a digital currency (which Dr. L. calls *KutzCoins*) that could be redeemable for Bear Bucks by donating computing time on their computers for student and faculty research.

Chapter 8

Personal Encryption Software

- Data encryption: Encrypt individual files, directories, and in some cases, whole drives.
  - **PGP** (Pretty Good Privacy) `pgp.com` This is the original software package for personal data encryption, email encryption, and digital signatures, etc. It was created by Phil Zimmerman in 1991. Currently, Symantec distributes the original PGP software as part of their encryption software suite. PGP was the inspiration for the OpenPGP standard for data encryption.
  - **GPG, Gnu PG** (Gnu Privacy Gard) `https://gnupg.org/` GPG is an open source alternative of PGP and is developed based on the OpenPGP standard to be interoperable with PGP and other OpenPGP-compliant systems.
  - **Gpg4win** `http://www.gpg4win.org/` Gpg4win is an open-source email and file encryption package for Windows built on GPG.
- Full Disk Encryption: Encrypt an entire drive on your computer. There are several proprietary and open-source packages available for any major operating system. See `https://en.wikipedia.org/wiki/Comparison_of_` for a comparison.
  - **BestCrypt** `www.jettico.com` works with Windows, Mac, Linux, and Android and is one of the oldest disk encryption software packages available.
  - **Bitlocker** `http://windows.microsoft.com/en-US/windows7/products/features/bitlocker` Bitlocker comes standard with certain versions of Windows Vista and later versions.
  - **FileVault** and **FileVault 2** comes standard with Mac OS X Panther (10.3) and later and Lion (10.7) and later, respectively.
  - **VeraCrypt** `https://veracrypt.codeplex.com/` is a freeware fork of TrueCrypt and was developed by the French Cryptography group IDRIX.
- Email encryption packages, servers, and plugins, with digital signatures:
  - **Enigmail** `https://enigmail.net/home/index.php` Plugin for Thunderbird based on OpenPGP.
  - **GPG, GPGMail, Gpg4win** `https://gpgtools.org/`
  - **Mailvelope** `https://www.mailvelope.com/` Mailvelope is a user-friendly and open-source webmail email encryption package that can also be downloaded as an add-on for Chrome and Firefox.
  - **PGP**
  - **ProtonMail** `https://protonmail.ch/` Swiss-based (out of US and EU jurisdictions) encrypted webmail developed by CERN researchers.
  - **RetroShare**
- Anonymous and secure internet browsing, networking, file transfers, and VPNs.

  - **Astoria** `http://nrg.cs.stonybrook.edu/astoria-as-aware-relay-selection-for-tor/` Astoria is a Tor client that circumvents attacks on anonymity on the Tor internet browsing network. Source code is forthcoming.
  - **I2P** `https://geti2p.net/en/` I2P (Invisible Internet Project) is similar to Tor and RetroShare (See below), but is based on the garlic routing concept to improve on the onion routing concept employed by Tor.
  - **Let's Encrypt** `https://letsencrypt.org/` From a Slashdot article, dated June 17, 2015, "Let's Encrypt will provide free-of-charge SSL and TSL certificates to any webmaster interested in implementing HTTPS for their products."
  - **OpenSSH** `http://www.openssh.com` OpenSSH is an open-source alternative to SSH and allows users to log into a remote server securely and transfer files securely (via sftp).
  - **OpenVPN** `openvpn.net` OpenVPN is an open-source virtual private network (VPN)
  - **RetroShare** `http://retroshare.sourceforge.net/` This is an all-encompassing open-source package allowing for secure chatting, calling, video, mail, file-sharing, social media, and internet browsing
  - **SecureDrop** `https://securedrop.org` SecureDrop is an anonymous-tip / whistle-blower submission system for people to send information to news outlets without revealing their identity.
  - **Tor, Tor Browser** (The Onion Router) `https://www.torproject.org/` From their website, "Tor prevents people from learning your location or browsing habits. Tor is for web browsers, instant messaging clients, and more. Tor is free and open source for Windows, Mac, Linux/Unix, and Android." and "Tor Browser contains everything you need to safely browse the Internet."

- Instant Messaging and Off-the-Record Messaging `https://otr.cypherpunks.ca/` Off-the-Record Messaging, or OTR, is a protocol for secure instant messaging applications. There are several other chat programs that utilize the OTR protocol. See (`https://en.wikipedia.org/wiki/Off-the-Record_Messaging`) for a list of most such programs. Here are a couple.
  - **Pidgin** (`http://pidgin.im/`) allows OTR as a plug-in.
  - **Kopete** (`https://userbase.kde.org/Kopete`) has OTR built-in.
  - **RetroShare**
  - **Tox** `https://tox.im` Tox is an open-source and secure instant-messaging and video-calling software package. Technically, Tox is the core functionality that another graphical user interface (GUI) will interact with. There are several GUIs that allow Tox to be run on any major operating system.
  - **What's App?"** What's App is a texting application with end-to-end encryption.

- Intenet phones and video phones.
  - **RetroShare**
  - **Tox**
  - **What's App?**

- Operating systems and cryptographic file systems
  - **eCryptfs** (Linux)
  - **Ext4** (Linux)
  - **NTFS with EFS** (Windows)
  - **Tails** (The Amnesic Incognito Live System) `https://tails.boum.org/` From their website: "Tails is a live operating system, that you can start on almost any computer from a DVD, USB stick, or SD card. It aims at preserving your privacy and anonymity" by forcing you to use encryption for all relevant applications. It is built upon Debian Linux.

  - **Qubes** `https://www.qubes-os.org` "A reasonably secure operating system."

- Cryptocurrencies. The enigmatic group or individual known as Satoshi Nakamoto revolutionized the digital currency paradigm in 2009 by creating Bitcoin, the first decentralized cryptocurrency. Bitcoin was perhaps inspired by ideas posted by Wei Dai (b-money) and Nick Szabo (bit gold) on their websites in 1998 and 2005, respectively. The following is a list of some of the more prominent cryptocurrencies (out of many hundreds that have been created). Most cryptocurrencies are based on Bitcoin's "block chain" concept to "mine" new coins and verify transactions. In each case, cryptographic protocols secure transactions, mining, anonymity (at least partially), digital wallets, and decentralization.

  - **Bitcoin** `https://bitcoin.org/en/` (BTC, 2009)
  - **Dash** `https://www.dashpay.io/` (DASH, 2014) Dash was originally named XCoin, then its name was changed to Darkcoin, and finally Dash. Dash was created to add anonymity to Bitcoin.
  - **Dogecoin** `http://dogecoin.com/` (DOGE, 2013) This is one of the more popular cryptocurrencies solely because it is named after an internet meme.
  - **Litecoin** `https://litecoin.org/` (LTC, 2011) Litecoin was an early fork of Bitcoin.
  - **Namecoin** `http://namecoin.info/` (NMC, 2011) Namecoin was the first fork of Bitcoin and functions as an alternative decentralized DNS (domain name system).
  - **Nxt** `http://nxt.org/` (NXT, 2013) Nxt was designed to have financial applications and services built upon it (via coin "coloring"), for example, allowing one to prove possession of intellectual property or copyrights, transfer stocks, bonds, property, commodities, etc.
  - **Primecoin** `http://primecoin.io/` (XPM, 2013) Primecoin is based on the mathematical difficulty of finding chains of prime numbers, the first cryptocurrency to be based on a problem in scientific computing.
  - **Ripple** `https://ripple.com/` (XRP, 2013) Ripple began in 2004 with the development of a global network that allowed users to create a decentralized currency and make secure transactions. With the creation of Bitcoin, this concept was extended.
  - **RSCoin** I knew it was bound to happen. The central bank of the U.K., the Bank of England, commissioned two researchers at University College London to devise a centralized cryptocurrency framework based on the blockchain concept.
  - **Zerocoin** `http://zerocoin.org/` Zerocoin is a proposed cryptocurrency that allows for more anonymity in a transaction.

- Educational software

  - CrypTool `https://www.cryptool.org/en/`

Problem 8.0.1 (20 points)**.** Do some basic research on one or more of the personal encryption software products listed above, or any other personal encryption product. You may choose any software that we haven't already seen how to install and use in class.

1. Describe the main use(s) of the product.
2. State what operating system(s) or device(s) the software runs on.
3. Describe the underlying cryptographic protocols and algorithms that the product uses.
4. If possible, make some comments on the claimed or proven security of the product.
5. Describe how to install and use the software. (This may require you to actually install and use the product.)

Chapter 9

Final Report Topics

In this final chapter, we have included some topics for a final paper, if the instructor so desires to use this for a final assessment. The purpose of a final paper is to apply and extend topics that were covered in the course. Here are some suggested criteria for the final report.

Criteria

- Apply and extend at least one topic from the course.
- Research at least one cryptographic topic that was not covered in the course.
- Provide at least one meaningful example.
- Use at least three references, at least two of which are books or peer-reviewed scholarly articles.

Timeline

1. Choose a topic.
2. Write and submit a tentative outline (sections, subsections, etc.) with a (possibly partial) list of references.
3. Write and submit a rough draft.
4. Submit the final draft.

The following is a list of suggested topics and is by no means exhaustive. Some points below contain multiple topics, but are grouped together based on thematic similarities. Some of the names may look scary, but a brief description follows each topic to alleviate any fear. If you have an idea for a topic not on this list, you are encouraged to discuss the topic with your instructor as early as possible in order to refine and focus the idea and to get some good references to get started. Some topics will require somewhat more extensive background knowledge in number theory, abstract algebra, probability, computer science, or linear algebra.

Topics

- Digital cash and cryptocurrencies
  Before Bitcoin, digital cash systems relied on a central certifying authority (a bank) to regulate any transaction and generally did not discuss the creation of money. Bitcoin changed the digital cash paradigm by uniting the creation of digital money and the decentralization of transaction authentication. Since Bitcoin, hundreds of other so-called *cryptocurrencies* have been created based on Bitcoin concepts. For background on digital cash, read §11.1 in [8].
- The blockchain concept
  Bitcoin introduced the concept of the blockchain, but numerous other applications of this digital technology have been considered. Describe the mathematics and algorithms of a blockchain and its potential uses besides cryptocurrencies.

- Homomorphic encryption
  How does one perform arithmetic on encrypted data without first decrypting the data? Homomorphic encryption provides solutions. Applications range from voting to banking. Question: Can the concepts of homomorphic encryption be applied to the Bitcoin concept to further anonymize transactions? This topic will require knowledge in abstract algebra, since it applies the concept of a homomorphism.

- Hyperelliptic curves
  Hyperelliptic curves are a generalization of elliptic curves. They provide security equivalent to elliptic curves with some trade-offs: one can work with smaller numbers, which speeds up arithmetic, but the arithmetic rules are more complex than that of elliptic curves.

- Identity-based encryption
  Alice wants to send Bob an encrypted message, but how does Alice know that Bob's publicly available public key wasn't changed by an eavesdropper? Wouldn't it be nice if Bob's public key was his email address or phone number? This is the concept of identity-based encryption. Read §16.6 in [8] to get started.

- Cryptanalysis of block and stream ciphers (e.g. Linear and Differential Cryptanalysis)
  What are some of the tricks of the trade for cracking block and stream ciphers? By cracking, we mean getting information about the key and/or plaintext, in whole or in part. Read §4.3 of [8] to get started on differential cryptanalysis.

- Integer factorization (e.g. Dixons Random Squares, Quadratic Sieve)
  The security of RSA rests on the presumed difficulty of factoring a large composite number. The best general-purpose factoring algorithm today, the (General) Number Field Sieve (GNFS), builds a difference of squares. While the GNFS is quite technical, some of the foundational concepts that are applied in Dixon's Random Squares and Pomerance's Quadratic Sieve are rather accessible. Read §6.4.1 and try Problem #28 in §6.9 in [8] to get started on the Quadratic Sieve.

- Computing group orders and discrete logarithms (e.g. index calculus, Rho, Kangaroo)
  The security of the Diffie-Hellman Key Exchange and other discrete logarithm-based cryptosystems relies on the difficulty to compute discrete logarithms and group orders. Methods include the Pohlig-Hellman Algorithm, index calculus, Shanks' Baby-Step Giant-Step Algorithm, Pollard's Rho Method, and Pollard's Kangaroo Method. Read §7.2.1, 7.2.2, or 7.2.3 of [8] to get started on the Pohlig-Hellman Algorithm, Baby-Step Giant-Step, or index calculus, respectively.

- Quantum cryptography
  Most cryptosystems today are designed to be secure when using a classical computer. However, properties of quantum particles give rise to a whole new paradigm of computing, called *quantum computing*. Techniques have been developed for key exchange and commitment, among others. Read §19.1 and §19.2 of [8] to get started.

- Shor's Algorithm
  If a sufficiently large quantum computer can be (has been) built, then integers will be able to be factored and discrete logarithms can be computed in polynomial time using Shor's Algorithm, rendering RSA insecure. Read §19.3 of [8] to get started.

- Post-quantum cryptography
  If a sufficiently large quantum computer can be (has been) built, then we will need to use cryptosystems that do not rely on integer factorization or discrete logarithms in order to guarantee security. Examples include lattice-based cryptosystems, multivariate cryptography, and hash-based cryptography.

- Coding Theory (Error-correcting codes)
  How can we tell if data has been altered, either intentionally or because of transmission over a noisy channel? If data has been altered, can the original data be recovered? There are a number of techniques of introducing *redundancy* to detect and eve correct errors. These redundancy systems are called *codes*. To get started on coding theory, read Chapter 18 of [8].

- Lattices, LLL, and NTRU
  A lattice is a remarkably straightforward geometric concept, yet is the source of a number of hard problems that form the foundation for some cryptosystems, and provide techniques for cracking cases of RSA. Read Chapter 17 of [8] to get started with lattices.

- Onion and Garlic Routing / Tor and I2P
  How do you keep your online activities anonymous? Various protocols have been developed to protect your online privacy and to prevent internet service providers, advertising agencies, data miners, and government agencies from spying on you. What are the underlying cryptographic protocols and concepts that go into all this?

- Digital watermarking
  Digital watermarking is like digital steganography and authentication combined. A digital watermark is a message embedded in a file that identifies the owner of the file. It is used to identify copyright violations.

- Steganalysis
  Describe at least one method of performing steganalysis (detection of steganography) that uses advanced mathematical methods. If you detect any steganography "in the wild," this would be a major component of the project.

- Broadcast encryption
  How do you send an encrypted message to a set of people in which each person has their own unique decryption key? Broadcast encryption technology is used with cable TV and DVDs.

- Other cryptosystems: Look under the hood
  Find an example of a cryptosystem (public-key or private-key), application, or protocol that we have not studied, or only superficially, and give a deeper description and analysis.

- Be creative
  Devise your own cryptosystem, protocol, or application and analyze it.

Bibliography

[1] Dale Bachman, Ezra Brown, and Anderson Norton. Chocolate key cryptography. *Mathematics Teacher*, 104(2):100–4014, 2010.

[2] Stephen Bellovin. Frank Miller: Inventor of the one-time pad. *Cryptologia*, 35(3):203–222, 2011.

[3] David Kahn. *The Codebreakers: The Story of Secret Writing*. Macmillan, New York, 1967.

[4] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Chapman & Hall/CRC, Boca Raton, FL, 1996. Online at http://cacr.uwaterloo.ca/hac/.

[5] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Special Publication 800-22, NIST, Gaithersburg, MD, 2010. Online at http://csrc.nist.gov/groups/ST/toolkit/random_number.html/.

[6] Bruce Schneier. *Applied Cryptography*. Wiley, New York, 2nd edition, 1996.

[7] Simon Singh. *The Code Book*. Doubleday, New York, 1999.

[8] Wade Trappe and Lawrence Washington. *Introduction to Cryptography with Coding Theory*. Pearson, Upper Saddle River, NJ, 2nd edition, 2006.

Index

Tabula Recta

This is a standard table for using some substitution ciphers, such as Vigenère's Cipher and the Running Key Cipher.

The left column is the plaintext, the top row is the key, and the contents of the table are the ciphertext.

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