CISS451: Cryptography

Dr. Yihsiang Liow (January 6, 2025)

Contents

Chapter 1

Introduction

1.1 Cryptography

Obviously cryptography is about scrambling data to make it meaningless to an eavesdropper and yet it can be descrambled by the person you are sending your data to.

Easy right?

Well yes and no.

(By the way, before I begin, cryptography is not cybersecurity.)

Cryptography is a huge area in computer science, mathematics, and engineering. If you define cryptography as "scrambling and descrambling" of data, then in of itself, cryptography is already an extremely huge area, in the sense that there's a lot of information in this subject. And it's not just an academic subject. It has extremely important consequences in the real world. If there's no cryptography, then there's no e-commerce. More generally, there won't be any electronic communication that requires privacy. The world we know today will be extremely different if cryptography does not exist. For sure millions of jobs will cease to exist if cryptography suddenly disappears or if some very powerful adversary (a power superpower rogue country?) makes cryptography useless.

Therefore cryptography is the cornerstone of information security and a cornerstone of our society.

However you will see that cryptography is even larger than you think. Besides "scrambling and descrambling data" for information security, cryptography is also used for instance in digital signatures which allows Amazon to say "You signed this document to buy the XYZ Complete Kit for Underwater Basket Weaving. Pay up. Don't deny it."

Cryptography can also be used to implement secure voting so that if you were voted to be the next president, the voting system can prove that the votes are counted correctly, and yet ... Tom Smith, the other candidate who did not win the election, cannot possibly find out who voted for you and not him so that Tom can't use his political power (if he has any) for retaliation. So you can ensure the correctness of the voting process *and* you prevent voting fraud. It can even prevent buying votes – if Tom Smith paid Harry Jones to vote for him, after the voting process, Harry cannot prove to Tom that he voted for Tom! If Tom cannot have a guarantee of Harry's vote, he might think twice about paying Harry.

You can also use cryptography to implement digital cash, the electronic equivalent of our physical dollar bills. Except that there's the benefit that digital cash cannot be counterfeited. There's the benefit of privacy too – Amazon, where you bought your "XYZ Complete Kit for Underwater Basket Weaving", does not know that it was you who bought it. Neither can the bank or credit card agency that acts as the intermediate payment gateway. Blockchain-based cryptocurrency such as bitcoin and ether have exploded in value in recent years. Blockchain has applications beyond cryptocurrency. It can be used to protect the integrity of data.

And if you are a spy working in a team that you don't fully trust (is there a double agent or mole in your group?), you can use cryptography to implement secret sharing so that the door to the vault that holds gold bullion is opened only when all the members are present with their part of the own combination to unlock the door to the vault.

You do want to be sure that your self-driving car will not follow an unauthenticated message from your enemy to instructor your car to ram into the nearest police car right? Do you know that Tesla's AI chip has AES cipher built into it at the hardware level so that it can communicate on the car's controller area network securely?

And suppose you have invited a new algorithm/device that you know will be worth billions. But it's at the prototype stage and you need funds from a venture capitalist. How can you convince the venture capitalize that your idea works without revealing it?

And wouldn't it be nice if you send an encrypted search to google, google returns the results – encrypted – which you decrypt. And google does not know what your search was and does not know what the search results were.

And the list of interesting cryptographic problems go on and on.

As the world becomes more and more connected (electronically), there's no doubt that more and more problems like the above will be proposed and solved using cryptography.

You know the "scrambling and descrambling" part of cryptography just from reading novels or watching movies. A "scrambling and descrambling" algorithm is usually called a cipher. Most people equate cryptography with ciphers. Cryptography is more than just about ciphers to achieve message confidentiality. Cryptography is the study of real world problems related to information protection in general using cryptographic tools. Besides ciphers, the other two very important cryptographic tools are cryptographic hash functions and pseudorandom number generators. Cryptographic hash functions is similar to the concept of hash functions in the study of data structures and algorithms (CISS350 and CISS358). Cryptographic hash functions are like hash functions except that they are "stronger". I'll get into that later. In the recipes to solve information protection problems, other mathematical and algorithmic tools are used as well.

Cryptography is a very huge area of study and research in this area is extremely active. Researchers come from computer science, math, and engineering.

1.2 Tools

There are other benefits to studying cryptography besides working in the area of information security. The following are subjects/areas that I hope to touch on, if not all, hopefully most:

- To begin with, cryptography involves algorithms. And you can never get enough of algorithms.
- You will study probability theory which is incredibly important in the real world. Probability theory should be as important as calculus and algebra in college, but for some reason many schools do not emphasize it or do not teach it well. I have no idea why. Taking cryptography and advanced algorithms (CISS451 and 358) hopefully will repair some of that problem.
- You will study information theory which is an area that builds on top of probability theory. Information theory is one of the newest areas of study in computer science, math, and engineering. The concept of

information theory was discovered only recently by Claude Shannon in 1948. Information theory can for instance predict for you how much data (the maximum data rate) that can be sent through a noisy data channel. It can also tell you what is the maximum data compression date. Information theory can also tell you if an encryption-decryption scheme is "absolutely" secure. Although Shannon proposed this concept to study problems in math and engineering, information theory is now also used in physics in an area called quantum information theory.

- You'll be studying number theory. You have seen a bit of number theory in discrete math. Many of the algorithms in crytography are number theoretic in nature. For instance you have seen the Extended Euclidean Algorithm. I bet you have not implemented it. Well ... you'll be studying that in cryptography and seeing it in action. Then you'll see why the Extended Euclidean Algorithm is so important. Therefore the study of cryptography is partially a review of number theoretic concepts from discrete math. Practically speaking, number theory is used in the implementation of the RSA cipher. And lo and behold ... RSA depends on prime factorization, something that you learned in middle school and learned to program in CISS240.
- You'll be studying finite fields. You have actually seen finite fields before. Well ... actually you have only seen one single finite field. In CISS360 the values of a bit, i.e., 0 and 1, forms a field $\{0, 1\}$ called a binary field. But there are many other fields. A field is just a collection of values where you can add, subtraction, multiply, and divide (except you can't divide by 0). You have seen many fields before, fields which are not finite. For instance the set of real numbers $\mathbb{R}$ is a field which is not finite. So are $\mathbb{Q}$ (field of rational numbers) and $\mathbb{C}$ (field of complex numbers). A finite field is just a field with finitely many values. They are very important not just in computer architecture. They appear in cryptography, digital signal processing, data compression, error correction codes, etc.
- You'll be studying groups. You have also seen groups before. If you think about real numbers, you see that you have two operations: addition and multiplication. (Subtraction is related to addition and division is related to multiplication.) A group is basically a set of values together with one operation. For instance the set of real numbers with addition is a group (just forget about the multiplication). While RSA uses number theory, certain cryptographic ciphers use groups. These are called group-based cryptography or discrete log cryptography. The standard group used (right now) is based on elliptic curves. At this point, elliptic curve cryptography (ECC) seems to be more popular in browser and SSL/TLS host. (If your browser is hitting a website that begins with `htts`, then you are using TLS.) ECC is also more suitable for the constrained en-

vironments such as smart cards, smart phones, and IoT. By the way, bitcoin uses ECC too. Besides cryptography, the theory of groups is extremely important and is used in computer science, engineering, physics, and chemistry. For instance group theory is used in computer vision and robotics. In physics and chemistry, group theory appears in quantum mechanics. In general when you study spatial structures, you will frequently find matrices and groups.

- A ring is just a field except that for fields you have addition, subtraction, multiplication, and division (by nonzero), while for rings you have all the four operations except you usually can't divide. The triad of "groups, rings, and fields" forms a very important collection of structures for higher math, computer science, and engineering. The study of "abstract algebra" is the study of groups, rings, and fields (and more). They are becoming more and more important in for instance algorithms, automata theory, and machine learning. By the way don't freak out: the word "abstract" in abstract algebra does not mean more complicated. It means "abstract away the non-essentials so that the theory is more applicable". It's similar to the concept of abstract base class in C++ which is a general class interface applicable to more subclasses.
- Besides the groups-rings-fields triad you will see the structures which are more familiar to you: graphs, matrices etc.
- In the above, I mentioned that you will be studying elliptic curves. Elliptic curves belong to an area of study called algebraic geometry. This is the only class where you can study a tiny bit of algebraic geometry. Elliptic curves are example of abelian varieties which makes them even more special than most curves. And you won't find this concept in any other class here. Furthermore, elliptic curves is a crucial ingredient in the proof of probably the most famous math problem in the world, Fermat's Last Theorem (FLT), which says that it is impossible to find integers $x > 0, y > 0, z > 0$ such that

$$x^n + y^n = z^n$$

if $n > 2$. FLT is so difficult that it remained unsolved for more than 350 years.
- You will be writing python and C++ programs. If you have done C++ (CISS240 and CISS245), you will have no problems picking up python. Python is a super simple programming language and yet it is now the dominant language for scientific computations and seriously heavy data/number crunching computations. Therefore knowing python is crucial. By the way, one of the programming tool for number theory research, SageMath, uses python as its language for accessing the libraries inside SageMath.

The guts of SageMath include libraries and code written in C, C++, Fortran, Lisp, and of course ... python. Here's an example. Suppose you want to compute

$$\int \frac{1}{x^2 + x + 42}\, dx$$

All you need to do is to run SageMath and enter this command:

```
print(integrate(1/(x**2 + x + 42), x))
```

and SageMath will give this to you in a split second:

```
2/167*sqrt(167)*arctan(1/167*sqrt(167)*(2*x+1))
```

i.e.,

$$\int \frac{1}{x^2 + x + 42}\, dx = \frac{2}{167}\sqrt{167}\arctan\left(\frac{1}{167}\sqrt{167}(2x+1)\right)$$

At some point, I hope to incorporate SageMath into this set of notes.

- You will learn some parallel/concurrent programming. This is the only course where you will learn some basic parallel programming.
- You will learn probability through massive data processing programming. Massive data crunching is used a lot in security type programming. This is the only class (for now) that does basic statistical data crunching.

1.3 Pre-requisites

In terms of background, I assume

- You know how to program at the level of CISS240 and CISS245.
- You have some discrete math background at the level of MATH225. In other words, you have seen some proofs, you know a bit of elementary number theory, you have a bit of combinatorics (counting), you know recursion, and you have a basic understanding of algorithms, including big-O.

In terms of computational tools, I assume

- You know how to use one of our fedora virtual machine. I'll be using our Fedora 31 virtual machine. if you did not take CISS245 with me, this is not a deal breaker because you can easily learn how to use a virtual machine in 1 hour. Sams as above. In CISS245 I go over virtual machines and linux commands in one class. I'll just tell you what you need to do.
- You know basic linux commands (from CISS245). See above.

- I will be using python for programming. You do not need to know python. If you have CISS240 and CISS245, learning python will be easy. In fact python is taught in many high schools. Also, most cryptographic programs are computational in nature and so the code does not use a lot of python syntax.

Chapter 2

Classical ciphers

The subtitle: Stuff that you should not use anymore.

However some of these old stuff is important because their ideas are used in modern-day cryptography.

Here we go ...

2.1 Shift cipher debug: classical-cipher-shift-cipher.tex

This is one of the earliest cryptosystems and apparently Julius Caesar used it (high tech, eh?). To encrypt a message, you simply do the following:

$$\alpha \mapsto \delta, \ \ldots$$

or in our alphabet system:

$$\begin{aligned} a &\mapsto d \\ b &\mapsto e \\ c &\mapsto f \\ \vdots & \quad \vdots \end{aligned}$$

i.e. a is replaced by d, b is replaced by e, etc. And of course you "go around in a circle": x is replaced by a, y is replaced by b, z is replaced by c.

$$\begin{aligned} a &\mapsto d \\ b &\mapsto e \\ c &\mapsto f \\ \vdots & \quad \vdots \\ x &\mapsto a \\ y &\mapsto b \\ z &\mapsto c \end{aligned}$$

This is known as the **Caesar cipher**. Caesar cipher

Here's a simple Python code to encrypt a character:

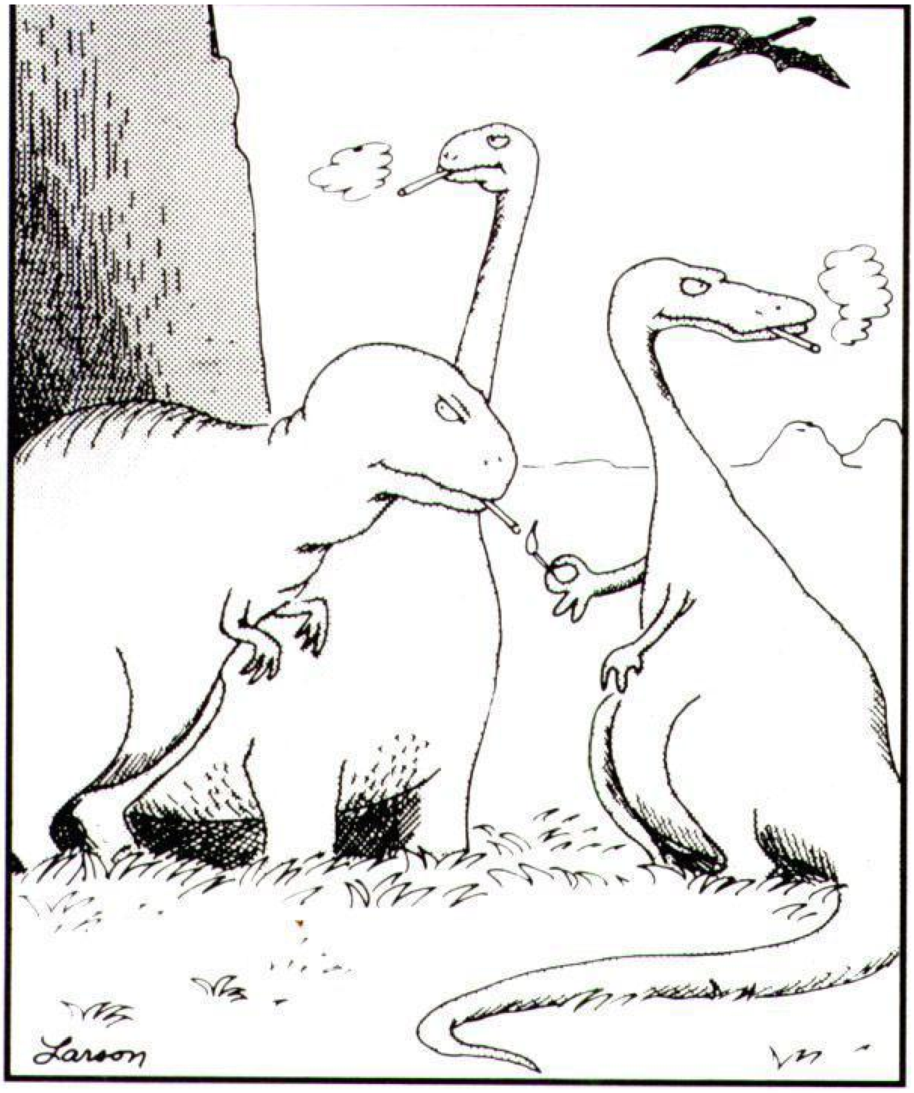
```
def E(x):
    i = ord(x) - ord('a')
    i = (i + 3) % 26
    return chr(ord('a') + i)
```

and for C++:

```
char E(char x)
{
    return (x - 'a' + 3) % 26 + 'a';
}
```

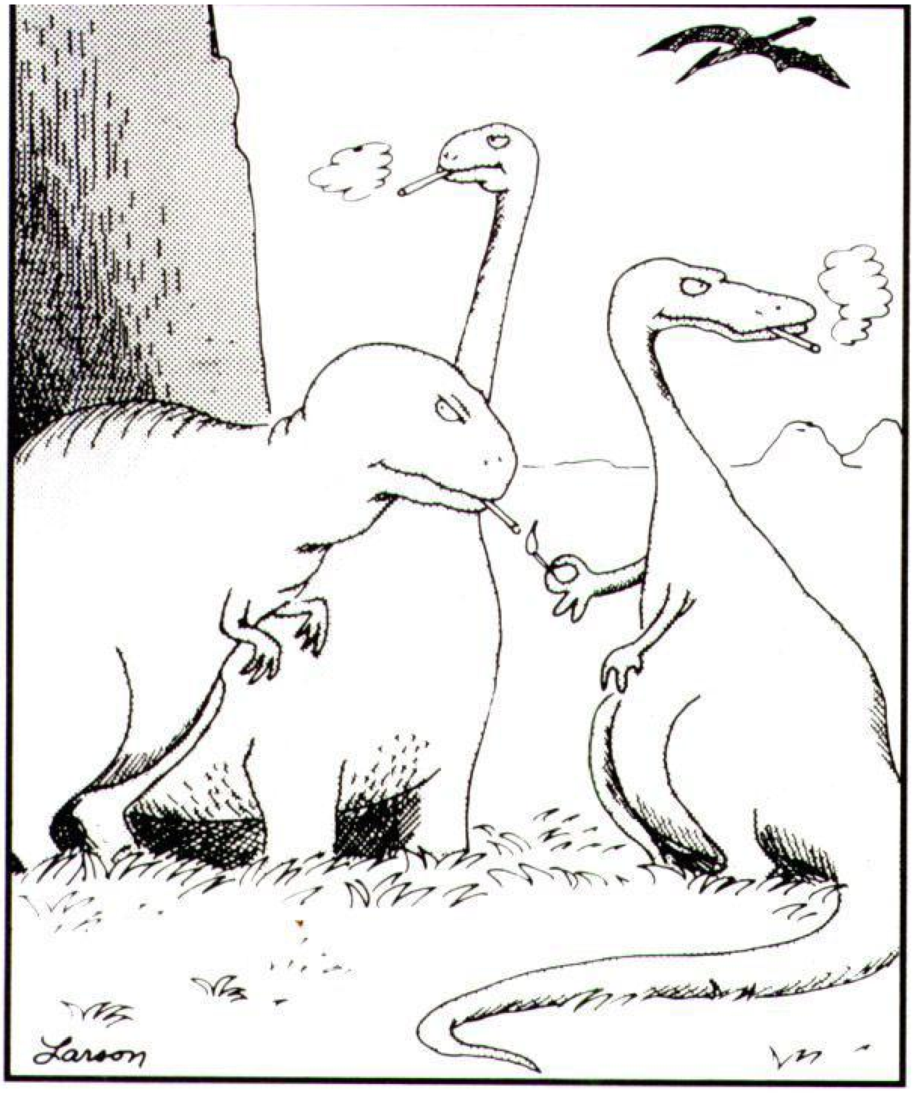
Of course there's no reason why you must "shift by 3". You can also shift by 7. Right? Note that the encryption input is only a single character. Of course for a string, you just encrypt character-by-character. Also, uppercase is replaced by lowercase. Furthermore, anything not *a*-*z* (example: punctuations, spaces) is thrown away.

Easy, right? OK. Break the following the code:



wkhuhdouhdvrqglqrvdxuvehfdphhawlqfw

*** WARNING: SPOILERS ON THE NEXT PAGE ***



The real reason dinosaurs became extinct

Exercise 2.1.1. Do you think Caesar cipher is secure? Do you think Julius Caesar was smart? (Go to solution, page 18) ☐

debug: exercises/shift-0/question.tex

Given P and C be two sets. A **cipher** is just a pair of functions $E : P \to C$ and $D : C \to P$ where E is called the **encryption** and D is called the **decryption** such that

cipher

encryption

decryption

$$D(E(x)) = x \quad \text{for all } x \in P$$

In other words if you decrypt what you have encrypted, you get back the same data. (It'd better be so!) An element of P is called a **plaintext** – it's what you encrypt. An element of C is called a **ciphertext** – it's what you get when you encrypt.

plaintext

ciphertext

Caesar cipher is an example of a cipher. For Caesar cipher $P = C = \{a, b, c, ..., z\}$. Also, although the encryption function maps one character to another, it's understood that if you want to encrypt a string, you simply encrypt each character of the string and join them up into a string.

But if you allow the shift amount in the Caesar cipher to change, then the encryption and decryption depends on the shift amount – the key. A general principle in cryptography is the following concept due to Auguste Kerckhoffs:

Kerckhoffs' principle (1883): A secure cipher should not depend on the secrecy of the encryption and decryption algorithm, but rather on the secrecy of the key.

Kerckhoffs' principle

The opposite and a really bad idea is called **security through obscurity**, i.e., it's the hope that your encrypted messages are safe as long as the encryption and decryption algorithm are kept secret.

security through obscurity

Why is this important? Because it's easy to change the key while changing the encryption and decryption algorithm might not be that easy. If a worker who performs the encryption or decryption process is captured, then he/she can be made (tortured?) to reveal the algorithm. On the other hand, if the key is stolen, then we can simply change the key. So in cryptography, it is always assumed that the algorithms (the cipher) cannot be kept secret for long.

In fact in modern cryptography, once a cipher is designed, the cryptography researcher(s) is expected to publish the cipher so that other researchers can check if the cipher is actually secure.

So we just need to modify the definition of our cipher:

Definition 2.1.1. A **cipher** is a pair of functions $E : K \times P \to C$ and $D : K \times C \to P$ such that if $k \in K$,

cipher

$$D(k, E(k, x)) = x \quad \text{for all } x \in P$$

P is the set of plaintexts, C is the set of ciphertexts, and K is the set of keys. Notice that in the above the key used for encryption k is the same as the key used for decryption.

Instead of writing $E(k, x)$ and $D(k, x)$, it's also common to write $E_k(x)$ and $D_k(x)$. Depending on which book you read, the encryption and decryption functions can also be written e instead of E and d instead of D.

Humans have used ciphers for thousands of years. The early ciphers always use the same key for encryption and decryption. It was only very recently in 1970 that James H. Ellis, asked if it's possible to have a cipher that uses two distinct keys, one for encryption and one for decryption. Ellis was a British cryptographer at the GCHQ (UK Government Communications Headquarters). If this is possible, then only the decyption key has to be kept secret. Why do you want to a use such a cipher?

Well, I can publish the encryption key for such a cipher on a website, you encrypt with the encryption key and send me the ciphertext by email. On receiving the ciphertext, I decrypt it using the decryption key. Note that I can make the encryption key public, but I must keep the decryption key a secret. On the other hand for a symmetric key cipher, we would have to meet secretly and decide on the common key.

Surprisingly such a cipher exists.

So now I have to modify our definition of ciphers ...

Definition 2.1.2. A **symmetric cipher** is a pair of functions $E : K \times P \to C$ and $D : K \times C \to P$ such that if $k \in K$, symmetric cipher

$$D_k(E_k(x)) = x \quad \text{for all } x \in P$$

P is the set of plaintexts, C is the set of ciphertexts, and K be the set of keys.

And of course we also must have

Definition 2.1.3. An **asymmetric cipher** is a cipher where there are two distinct keys, one for encryption and one for decryption. An asymmetric cipher is also called a **public key cipher** because the encryption key can be made public (but the decryption key has to kept secret). In this case, if k, k' are the encryption and decryption keys, then the cipher must satisfy asymmetric cipher public key cipher

$$D_{k'}(E_k(x)) = x$$

for $x \in P$. The encryption key k is called the **public key** (because it can be made public) while the decryption key is called the **private key**. public key private key

Public key ciphers uses quite a bit of math. So we won't see public key ciphers for a while.

Let's go back to our Caesar cipher. You can think of the Caesar cipher as a special case of a symmetric cipher that uses the key 3:

- encryption is "shift forward by 3"
- decryption is "shift backward by 3".

In other words, generalizing the Caesar cipher, we get the shift cipher:

- encryption is "shift forward by k"
- decryption is "shift backward by k".

where k is the key. I hope it's clear that the shift cipher with key 27 is the same as the shift cipher with key 1. Effectively speaking there are only 26 shifts, including the very bad key of 0. Hence for the shift cipher, $K = \{0, 1, 2, ..., 25\}$.

Exercise 2.1.2. Is the Caesar's cipher symmetric or asymmetric? (Duh) (Go to solution, page 19) ☐

debug: exercises/shift-1/question.tex

For classical ciphers, assuming we're only interested in English, the plaintexts are strings involves a-z. I will write $\{a, b, c, ..., z\}^*$ for the set of all strings with characters from $\{a, b, c, ..., z\}$. If n is a positive integer, I will also write $\{a, b, c, ..., z\}^n$ for the set of strings with length n and with characters from the set $\{a, b, c, ..., z\}$. For instance

$$\{a, b, c, ..., z\}^2 = \{aa, ab, ac, ..., zx, zy, zz\}$$

Exercise 2.1.3. How many strings are there in $\{a, b, c, ..., z\}^2$? How many strings are there in $\{a, b, c, ..., z\}^n$? (Go to solution, page 20) ☐

debug: exercises/shift-2/question.tex

In modern day cryptography, we frequently work with bit strings. The set of all bit strings is denoted by $\{0, 1\}^*$. Bit strings of length exactly 8 is denoted by $\{0, 1\}^8$ – these would be bytes. For instance you might have heard of the SHA2 family of hash function. SHA256 takes in bit strings and spits out bit strings of length 256. So SHA256 is a function of type

$$\{0, 1\}^* \to \{0, 1\}^8$$

(Technically speaking SHA256 inputs do have a maximum limit in length, but it's so huge that for practical purposes it's as good as all possible bit strings.)

You know this is coming ... we'll be using *lots* of math to do encryption and decryption. In particular, for this notes, we associate letters a to z with numbers. The encryption and decryption function will work with either numbers of letters. Specifically we have the following correspondence:

In math, "correspondence" is the same as "1-1 correspondence" which is the same as "bijection". Remember bijection? It's time to check your discrete math notes

$$\begin{aligned} a &\leftrightarrow 0 \\ b &\leftrightarrow 1 \\ c &\leftrightarrow 2 \\ \vdots &\quad \vdots \\ z &\leftrightarrow 25 \end{aligned}$$

if E encrypts a to c, I will say either

$$E(a) = c$$

or

$$E(0) = 2$$

Now you might say ... "so what's the big deal? Why rewrite a as 0, b as 1, etc. I can also come up with some secret encoding for instance why can't I rewrite a as a square, b as a triangle, etc.?"

Well ... the reason is because $0, 1, 2, \ldots$ are numbers ... and ... *they have operations (addition, subtraction, multiplication, division).*

With the above in mind, instead of describing the shift cipher as functions on a-z, I'll describe it as function on $\mathbb{Z}/26$:

Definition 2.1.4. The **shift cipher** (E, D) is given by

shift cipher

$$E_k(x) = x + k \pmod{26}$$

and

$$D_k(x) = x - k \pmod{26}$$

It's clear that for the shift cipher (E, D),

$$D_k(E_k(x)) \equiv x \pmod{26}$$

And of course the shift cipher with key $k = 3$ is the **Caesar cipher**.

Caesar cipher

Solutions

Solution to Exercise 2.1.1.

debug: exercises/shift-0/answer.tex

Solution not provided.

Solution to Exercise 2.1.2.

debug: exercises/shift-1/answer.tex

The shift cipher is a symmetric cipher, i.e., it's private key cipher.

Solution to Exercise 2.1.3.

debug: exercises/shift-2/answer.tex

Solution not provided.

2.2 Modular arithmetic debug: classical-cipher-modular-arithmetic.tex

Number theory is basically the study of whole numbers. That however is an over-simplification. The study of number theory involves almost all areas of mathematics. In fact many areas of mathematics were created just to study certain problems in number theory.

Number theory is an extremely huge area of study in Mathematics and Computer Science. It is also extremely fascinating. Many problems is number theory can be stated very simply so that even a high school student can understand the statement of the problem. And yet the techniques used to solve some of these problems require the mathematical tools from almost every area of Math. Gauss once said that "Mathematics is the queen of sciences and number theory is the queen of mathematics".

Although there are many branches within Number Theory, right now we only need to know a little bit about Elementary Number Theory. "Elementary" here does not mean simple (although it will be for us since you're only seeing a small part of Elementary Number Theory). It means we are studying Number Theory using only properties of whole numbers (integers). Research in Number Theory requires real numbers, complex numbers, calculus, geometry, complex analysis, etc.

This will be a very short introduction to the vast area of Number Theory. In fact this is only a tiny fraction of Elementary Number Theory. This is one of the oldest area of Mathematics and one of the fascinating because of its history. If you want to learn more about number theory, just let me know. I can easily find a project for you to work on.

For now, we will look at modular arithmetic. Besides cryptography, modular arithmetic is also used in data compression and error correction codes.

The set of integers $\{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ denoted $\mathbb{Z}$ has two operations $+$ and $\cdot$. In terms of the algebraic structure (i.e. the operations), $\mathbb{Z}$ is known as a **commutative ring**. Basically a commutative ring is a set of "things" with two operations, addition and multiplication, with rules that look like the addition and multiplication rules for $\mathbb{Z}$. For instance one such rule in $\mathbb{Z}$ is

$$a(b + c) = ab + ac$$

This same rule holds true for $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ and polynomials with coefficients in $\mathbb{Z}$.

The reason why mathematicians even bother defining this concept of "commutative ring" is more or less the same reason why we write functions in programs: for re-use. There are *many* naturally occurring rings. So if while developing the theory for $\langle \text{blah}_1 \rangle$ and $\langle \text{blah}_2 \rangle$ and they are both rings, then it's enough to prove a general fact that applies to both and quote the fact. This is also related to the concept of inheritance and abstract base classes. You can think of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ as subclasses of `CommutativeRing`. Therefore if you have a function

```
void f(CommutativeRing & r)
{
  ...
}
```

then `f` can work with `x` if `x` is a object of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$.

The mathematician develop general results while the programmer write general functions working on abstract base class objects. The idea is the same. The reason for generality is efficiency.

Here's a very important advice on studying rings, groups, fields, math, etc. (in fact this applies to any area of study where there is a great deal of generalization): Although the definition and theorems are general, you *always* keep a couple of standard examples in your mind as you read the statements. While reading them, mentally substitute your examples into the facts so that you can associate it to something more concrete. This is not just a learn technique for undergraduate students. Even researchers do that when they read research papers. For the definition of ring below, think of the ring of integers.

I will try to be as informal as possible for now so that you can develop some feel/intuition for what we need for now. Later I'm going to come back to this topic and redo the whole thing rigorously. The focus for now is to understand modulo 26 arithmetic.

Anyway, a **commutatative ring** R is a set of "things" with two operations abstractly denoted by $\oplus$ and $\odot$ ("addition" and "multiplication") Furthermore there are two special "things" in R which we will call 0_R and 1_R. The properties satisfied by $R, 0_R, 1_R, \oplus, \odot$ (of course there must be something satisfied by them!) are as follows. For $\oplus$ the properties are:

Example: the integer $\mathbb{Z}$,...+ and $\cdot$ of $\mathbb{Z}$ ...

0 and 1 of $\mathbb{Z}$ ...

(a) $r \oplus s$ is in R for all r, s in R
(b) $(r \oplus s) \oplus t = r \oplus (s \oplus t)$ for all r, s, t in R
(c) $r \oplus 0_R = r = 0_R \oplus r$ for all r in R
(d) For all r in R there is something in R which we will call r' such that $r \oplus r' = 0_R = r' \oplus r$

$i + j$ is in $\mathbb{Z}$ for integers i, j, ...
$(i+j)+k = i+(j+k)$ for integers i, j, k ...
$i + 0 = i = 0 + i$ for integer i ...
If i is an integer, then $-i$ is also an integer and $i + (-i) = 0 = (-i) + i$.

For $\odot$ the properties are:

(a) $r \odot s$ is in R for all r, s in R
(b) $(r \odot s) \odot t = r \odot (s \odot t)$ for all r, s, t in R
(c) $r \odot 1_R = r = 1_R \odot r$ for all r in R
(d) $r \odot s = s \odot r$ for all r, s in R

$ij = ji$ for integer $i, j, \ldots$
$(ij)k = i(jk)$ for integers i, j, k
$i1 = i = 1i$ for integer i ...
$ij = ji$ for integers i, j

The property involving both $\oplus$ and $\odot$ is

(a) $r \odot (s \oplus t) = r \odot s \oplus r \odot t$

$i(j + k) = ij + ik$ for integers i, j, k. Phew! So $\mathbb{Z}$ is a commutative ring.

Just remember this: A ring is a set of things with addition and multiplication. And when you're lost just think of the set of integers and its operations.

Now we'll be working with the alphabet a,b,...,z. We'll call them by their new identities: $0, 1, ..., 25$. This is not exactly all of $\mathbb{Z}$. The formulas for the encryption and decryption of Caesar's cipher involves addition and substraction. What if we go beyond? What is 3 + 25 (i.e., d + z)? No problem, we will take remainders mod 26. So instead of 3 + 25 we think of (3 + 25) mod 26 instead. Of course the remainders are 0, 1, ..., 25 which is exactly what we want.

In C++-speak, we take %26.

To indicate that we're only interested in remainders or more accurately, we ignore multiples of 26, we write

$$\begin{aligned} 3 + 25 &= 28 \\ &\equiv 2 \pmod{26} \end{aligned}$$

In general, if x and y are integers, we write

$$x \equiv y \pmod{26}$$

if 26 divides $x - y$. It does not mean that x is equal to y. It means that x and y are the same if you ignore additive multiples of 26, i.e.,

$$x \equiv y \pmod{26}$$

is the same as saying

$$x = y + (\text{... some multiple of 26 ...})$$

This is the same as saying the remainder when x is divided by 26 is the same as the remainder when y is divided by 26.

If two numbers differ by a multiple of 26, we say that they are **congruent** mod 26.

congruent

Note that

$$\begin{aligned} 26 &\equiv 0 \pmod{26} \\ 27 &\equiv 1 \pmod{26} \\ 28 &\equiv 2 \pmod{26} \\ &... \end{aligned}$$

and

$$\begin{aligned} -1 &\equiv 25 \pmod{26} \\ -2 &\equiv 24 \pmod{26} \\ -3 &\equiv 23 \pmod{26} \\ &... \end{aligned}$$

So in the mod 26 world, since you are ignoring multiples of 26, *in some sense* there are only 26 numbers:

$$0, 1, 2, 3, ..., 25$$

I'll write $\mathbb{Z}/26$ for this world of 26 values. Remember that in this world, you can write the symbol

$$28$$

but this is the same as 2 in $\mathbb{Z}/26$:

$$28 \equiv 2 \pmod{26}$$

Of course in $\mathbb{Z}$, these symbols, i.e. 28 and 2, are different.

Most of the algebraic rules involving $+, -, *, 0, 1$ applies when working with integers mod 26. For instance suppose

$$x \equiv y \pmod{26}$$

where x and y are integers (i.e., x differs from y by a multiple of 26), then

$$x + z \equiv y + z \pmod{26}$$

where z is an integer. Likewise from

$$x \equiv y \pmod{26}$$

we get

$$xz \equiv yz \pmod{26}$$

It's also true that

$$0 + x \equiv x \pmod{26}$$

and

$$1 \cdot x \equiv x \pmod{26}$$

To be more precise, $\mathbb{Z}/26$ is a commutative ring. It's a finite commutative ring with 26 values. Let me rewrite the axioms for a commutative ring for $\mathbb{Z}/26$.

For $+$ on $\mathbb{Z}/26$, the properties are:

(a) $(r + s) \pmod{26}$ is in $\mathbb{Z}/26$ for all r, s in $\mathbb{Z}/26$
(b) $(r + s) + t \equiv r + (s + t) \pmod{26}$ for all r, s, t in $\mathbb{Z}/26$
(c) $r + 0 \equiv r \equiv 0 + r \pmod{26}$ for all r in $\mathbb{Z}/26$. In fact r' is just $(26 - r) \pmod{26}$. For instance for $r = 2 \pmod{26}$, $r' = 26 - 2 = 24 \pmod{26}$.
(d) For all r in $\mathbb{Z}/26$ there is something in $\mathbb{Z}/26$ which we will call r' such that $r + r' \equiv 0 \equiv r' \oplus r \pmod{26}$

For $\cdot$ the properties are:

(a) $r \cdot s \pmod{26}$ is in $\mathbb{Z}/26$ for all r, s in $\mathbb{Z}/26$
(b) $(r \cdot s) \cdot t \equiv r \cdot (s \cdot t) \pmod{26}$ for all r, s, t in $\mathbb{Z}/26$
(c) $r \cdot 1 \equiv r \equiv 1 \cdot r \pmod{26}$ for all r in $\mathbb{Z}/26$
(d) $r \cdot s \equiv s \cdot r \pmod{26}$ for all r, s in $\mathbb{Z}/26$

The property involving both $+$ and $\cdot$ is

(a) $r \cdot (s + t) \equiv r \cdot s + r \cdot t \pmod{26}$

It's also not too surprising that you can talk about mod n for any positive integer n.

If I don't say so, when I want you write some $x \pmod{26}$, I mean the x such that $0 \leq x < 26$. Of course there's no difference in mod 26 between 2 and 28. But in mod 26, the values in $[0, 26)$ is the preferred range. Also, when I say simplify $27 \pmod{26}$, I mean $1 \pmod{26}$.

Note that since $\mathbb{Z}/26$ is *finite*, you can always solve any equation in mod 26. This is similar to boolean values: there are only two. To solve a boolean equation, you just try all possible boolean values. So to solve a $\mathbb{Z}/26$ equation, you just try all the 26 possible values on all the variables that appear in the equation.

Exercise 2.2.1.

debug: exercises/exercise-0/question.tex

(a) True or false: $100 \equiv 74 \pmod{26}$
(b) True or false: $-3 \equiv 133 \pmod{26}$
(c) True or false: $-20 \equiv 21 \pmod{26}$
(d) True or false: $7 \equiv 3 \pmod{3}$
(e) True or false: $17 \equiv -3 \pmod{5}$
(f) True or false: $21 \equiv 11 \pmod{8}$
(g) True or false: $42 \equiv 0 \pmod{7}$

(Go to solution, page 42) □

Exercise 2.2.2.

debug: exercises/exercise-1/question.tex

(a) Simplify $100 \pmod{26}$.
(b) If you have very simple calculator with $+$, $-$, $*$, $/$ how would simplify $131246845 \pmod{26}$? You have 10 seconds ... the clock is ticking ...
(c) Simplify $33 \pmod{26}$.
(d) Solve $2x + 1 \equiv 0 \pmod{26}$ by brute force. Use Python or C++.
(e) Solve $10x + 20 \equiv 4x + 78 \pmod{26}$ by brute force. Use Python or C++.
(f) Simplify $10x + 20 \equiv 4x + 78 \pmod{26}$ first, and then solve it by brute force. Do you get the same results as in the previous part?
(g) Solve $42x^5 + 10x + 1 \equiv 73 \pmod{3}$.
(h) Solve $(1000000x + 2)^3 \equiv 2 \pmod{3}$.

(Go to solution, page 43) □

Exercise 2.2.3.

debug: exercises/exercise-2/question.tex

(a) If $x \equiv 1 \pmod{7}$ and $x \equiv 5 \pmod{13}$, what can you tell me about x?
(b) If $x \equiv 1 \pmod{7}$ and $x \equiv 5 \pmod{13}$, what can you tell me about $x \pmod{7 \cdot 13}$?
(c) $x \equiv 2 \pmod{7 \cdot 13}$ what can you tell me about $x \pmod{7}$ and $x \pmod{13}$?
(d) True or false: $x^3 \equiv x^0 \pmod{3}$ since the 3 in the exponent can be replaced by 0.
(e) In $\mathbb{Z}$, is it true that $(x + y)^2 = x^2 + y^2$? Try some values for x and y.
(f) In $\mathbb{Z}/2$, is it true that $(x + y)^2 = x^2 + y^2 \pmod{2}$? Try all values for x and y.
(g) In $\mathbb{Z}$, is it true that $(x + y)^3 = x^3 + y^3$? Try some values for x and y.
(h) In $\mathbb{Z}/3$, is it true that $(x + y)^3 \equiv x^3 + y^3 \pmod{3}$? Try all values for x and y.
(i) In $\mathbb{Z}/n$, is it true that $(x + y)^n \equiv x^n + y^n \pmod{n}$? Can you prove your

claim.

(Go to solution, page 44) □

Exercise 2.2.4. Just like for boolean values, you can write down the complete behavior of the boolean and and boolean or and boolean not operators (these are called truth tables), you can also complete specify the complete behavior of addition in mod 26, "negative of" in mod 26, multiplication in mod 26, and also multiplicative inverse mod 26. The multiplicative inverse of x in mod 26 is just the number y mod 26 such that

debug: exercises/exercise-3/question.tex

$$xy \equiv 1 \pmod{26}$$

The multiplicative inverse of $x \pmod{26}$ is written $x^{-1} \pmod{26}$ – this is an integer mod 26!!! It's not a fraction in $\mathbb{R}$!!! Sometimes $x^{-1} \pmod{26}$ might not exist. In that case write None. Write down these 4 tables.

Addition table for $\mathbb{Z}/26$:

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0																										
1																										
2																										
3																										
4																										
5																										
6																										
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18																										
19																										
20																										
21																										
22																										
23																										
24																										
25																										

In the above, when I write 5, I meant of course $5 \pmod{26}$.

Multiplication table for $\mathbb{Z}/26$:

$\times$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0																										
1																										
2																										
3																										
4																										
5																										
6																										
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18																										
19																										
20																										
21																										
22																										
23																										
24																										
25																										

Negative of table for $\mathbb{Z}/26$:

$x \pmod{26}$	$-x \pmod{26}$
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

Multiplicative inverse table for $\mathbb{Z}/26$:

$x \pmod{26}$	$x^{-1} \pmod{26}$
0	None
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

It should be clear that $0 \pmod{26}$ does not have an inverse.

(Go to solution, page 45) □

Exercise 2.2.5. Solve

debug: exercises/exercise-4/question.tex

$$3x = 8$$

in $\mathbb{Z}$. Can you? Now solve

$$3x \equiv 8 \pmod{26}$$

(Go to solution, page 49) □

Exercise 2.2.6.

debug: exercises/exercise-5/question.tex

(a) Solve

$$3x = 1$$

in $\mathbb{Z}$. Of course you can't! Now solve

$$3x = 1$$

in $\mathbb{Q}$. Of course you can! We would say that $\frac{1}{3}$ is the multiplicative inverse of 3 in $\mathbb{Q}$. Also, we would say that, in $\mathbb{Z}$, 3 is not invertible. What about

$$3x \equiv 1 \pmod{26}$$

Can you? If you can then you have found a multiplicative inverse of 3 (mod 26).

(b) Now try to solve

$$ax \equiv 1 \pmod{26}$$

for $a \equiv 0, 1, 2, ..., 25 \pmod{26}$. Which a's have multiplicative inverses? Is there a pattern to a's with multiplicative inverses mod 26 and who which do not?

(c) Now try

$$ax \equiv 1 \pmod{N}$$

where N is a positive integer and $a \equiv 0, 1, 2, ..., N-1 \pmod{N}$; say you try $N = 4, 5, 6, 7, 8$. Notice something? See a pattern? Is there a pattern to a's with multiplicative inverses mod 26 and who which do not?

(Go to solution, page 50) □

Exercise 2.2.7. Solve

debug: exercises/exercise-6/question.tex

$$3x = 13$$

in $\mathbb{Z}$. Of course you can't! Now solve

$$3x = 13$$

in $\mathbb{Q}$. Of course you can! Do you realize that you solved it using the multiplicative inverse of 3 in $\mathbb{Q}$? What about

$$3x \equiv 13 \pmod{26}$$

Are you going to use brute force and try all values (mod 26)? (HINT: You had better not.) (Go to solution, page 51) □

Exercise 2.2.8. Solve

debug: exercises/exercise-7/question.tex

$$3x - 5 \equiv 10x + 20 \pmod{26}$$

by brute force. Next, simplify the above first before solving it. Do you get the same solutions? (Go to solution, page 52) □

Exercise 2.2.9. Solve

debug: exercises/exercise-8/question.tex

$$3x + 12 \equiv 10x + 23 \pmod{26}$$

by brute force. Next, simplify the above first before solving it. (Are you sure it's really simplifed? HINT: Multiplicative inverse.) Do you get the same solutions? (Go to solution, page 53) □

Exercise 2.2.10.

debug: exercises/exercise-9/question.tex

(a) Can you solve

$$x^2 \equiv 1 \pmod{26}$$

Of course one solution is 1 (mod 26). Can you find the other one in 1 second? Are there just two? Or are there more than two solutions? These are (of course) square roots of 1, but in mod 26 and not $\mathbb{R}$.

(b) What about

$$x^2 \equiv 2 \pmod{26}$$

(c) Keep going ... try to solve

$$x^2 \equiv a \pmod{26}$$

for all cases. Draw a table for the square roots of $a \pmod{26}$ for all a's.

(Go to solution, page 54) □

Exercise 2.2.11. Continuing the previous exercise ...

debug: exercises/exercise-10/question.tex

(a) Next, try to solve
$$x^2 \equiv a \pmod{N}$$
for all $a \pmod{N}$ for at least 3 values of N. See any patterns?

(b) The above is probably too tough. What if you try
$$x^2 \equiv 2 \pmod{p}$$
for primes p? Try a few primes (maybe 20-30) and make a table. Notice a pattern?

(Go to solution, page 55) □

Exercise 2.2.12.

debug: exercises/exercise-11/question.tex

(a) What are all the possible values of $x^2 \pmod{4}$?
(Note: The key thing to note is that $x \pmod{4}$ can take all the values 0, 1, 2, 3 mod 4. But the squares $x^2 \pmod{4}$ can only certain values mod 4.)

(b) What are all the possible values of $x^2 \pmod{5}$?

(c) What about squares mod 7?

(d) What about squares mod 15? (Be efficient.)

(e) Prove that there are no integer solutions to
$$5x^2 - 8y^2 = 3$$
(HINT: Prove this by contradiction. Take mod 4 ... and ... Checkmate!) Without modular arithmetic, there's no clear way to solve this problem! Go number theory!)

(Go to solution, page 56) □

Exercise 2.2.13.

debug: exercises/exercise-12/question.tex

(a) Solve
$$5x^2 + y^2 = 3$$
(HINT: You don't really need number theory for this one. Why? But if you want to, imitate the solution for the previous problem.)

(b) Solve
$$11y^2 - 5x^2 = 3$$

(HINT: This is just a slight change from the previous problem. *But* now you need number theory. Try mod 4. If it does not work, try mod 5. Repeat.)

(c) Solve
$$y^2 - 5x^2 = 2$$

(d) What about this one:
$$x^2 - 5y^2 = 1$$

[Aside. Integer solutions to $x^2 - dy^2 = 1$ has been studied since at least 400BC. This equation appear the Cattle Problem of Archimedes:

> If thou art diligent and wise, O stranger, compute the number of cattle of the Sun, who once upon a time grazed on the fields of the Thrinacian isle of Sicily, divided into four herds of different colours, one milk white, another a glossy black, the third yellow and the last dappled. In each herd were bulls, mighty in number according to these proportions: Understand, stranger, that the white bulls were equal to a half and a third of the black together with the whole of the yellow, while the black were equal to the fourth part of the dappled and a fifth, together with, once more, the whole of the yellow. Observe further that the remaining bulls, the dappled, were equal to a sixth part of the white and a seventh, together with all the yellow. These were the proportions of the cows: The white were precisely equal to the third part and a fourth of the whole herd of the black; while the black were equal to the fourth part once more of the dappled and with it a fifth part, when all, including the bulls went to pasture together. Now the dappled in four parts8 were equal in number to a fifth part and a sixth of the yellow herd. Finally the yellow were in number equal to a sixth part and a seventh of the white herd. If thou canst accurately tell, O stranger, the number of cattle of the Sun, giving separately the number of well-fed bulls and again the number of females according to each colour, thou wouldst not be called unskilled or ignorant of numbers, but not yet shall thou be numbered among the wise. But come, understand also all these conditions regarding the cows of the Sun. When the white bulls mingled their number with the black, they stood firm, equal in depth and breadth, and the plains of Thrinacia, stretching far in all ways, were filled with their multitude. Again, when the yellow and the dappled bulls were gathered into one herd they stood in such a manner that their number, beginning from one, grew slowly greater till it completed a triangular figure, there being no bulls of other colours in their midst nor none of them lacking. If thou art able, O stranger, to find out all these things and gather them together in your mind, giving all the relations, thou shalt depart crowned with glory and knowing that thou hast been adjudged perfect in this species of wisdom.

If W, X, Y, Z represents the number of white, black, yellow, dappled bulls, you will get a systems of 7 linear equations, the first two being

$$W = (1/2 + 1/3)X + Z$$
$$X = (1/4 + 1/5)Y + Z$$

together with some constraints such as $W + X$ must be a square. After some manipulations, it can be shown that the equation to solve looks like

$$x^2 - 410286423278424y^2 = 1$$

What was Archimedes thinking? You are find information on the Archimedes Cattle Problem on the web.]

(Go to solution, page 57) □

Exercise 2.2.14. Consider

debug: exercises/exercise-13/question.tex

$$5x^2 - 8y^2 = 3z^2$$

where we are only interested in finding integer solutions. Of course $(0,0,0)$ is a solution, but that's easy. (Right?) In general, an equation like

$$ax^p + by^q = cz^r$$

where $p, q, r > 0$ always has $(0,0,0)$ as a solution. $(0,0,0)$ is sometimes called the trivial solution. So we might as well assume $(x,y,z) \neq (0,0,0)$, i.e., assume they are not all zero. By the way the polynomial

$$5x^2 - 8y^2 - 3z^2$$

is a sum of terms where the number of variables appearing in each term is the same, i.e., 2. Such polynomial are is said to be homogeneous of degree 2.

How many solutions can you find? Is there any at all? Is there a finite number of solutions? If there are infinitely many solutions, can you write them down? If not all of them, maybe an infinite family of them. For instance it would be nice to say: "For any integer n, $x = n, y = 2n, z = n + 5$ is a solution". This would be a 1–parameter family of solutions.

(a) What's the first thing you should do?
(b) Prove that if x, y, z is a solution, then nx, ny, nz is also a solution for any integer n.
(c) In case you can't see a solution right away, do the following. If (x, y, z) are solutions, show that x and z must be even. (HINT: mod 4.) This is helpful since we won't have to check the odd x or odd z caaes. That cuts down a brute force search down by 3/4.
(d) Continuing the above: let $x = 2a$, $y = b$, and $z = 2c$, substitute, and you'll get a new equation in a, b, c.
(e) The equation in a, b, c is easier than the one in x, y, z. Why? Because a solution in the equation in a, b, c for the above question would correspond to a solution x, y, z in the original equatison and a, b, c solution is smaller. Can you now find some solutions for the equation in a, b, c?
(f) I won't go further. But for those who want to solve this probably complete, let me just say that once you have one one solution, you can parametrize all solutions using two integer parameters.

(Go to solution, page 58) □

Exercise 2.2.15. Find all the integer solutions to $3x^2 + 4y^2 = 5z^2$. (Hint: mod 3.) (Go to solution, page 59) □

debug: exercises/exercise-14/question.tex

Exercise 2.2.16. Find all solutions to

debug: exercises/exercise-15/question.tex

$$y^2 = x^3 + x + 1 \pmod{26}$$

(No this is not a random, idle question. It's actually very important.) (Go to solution, page 62) □

Exercise 2.2.17. (This "idle" problem is way more important than you think.)

debug: exercises/exercise-16/question.tex

(a) Compute

$$2^n \pmod{26}$$

for $n = 0, 1, 2, 3, ...$ What do you notice? Can you tell me what is

$$2^{1000} \pmod{26}$$

(b) Now try $3^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $3^{1000} \pmod{26}$

(c) Now try $4^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $4^{1000} \pmod{26}$

(d) Now try $5^n \pmod{26}$ for $n = 0, 1, 2, 3, ...$ Can you tell me what is $5^{1000} \pmod{26}$

(e) Of course since $\mathbb{Z}/26$ is finite, you would expect $a^n \pmod{26}$ to be finite even though $n = 0, 1, 2, ...$ runs through an infinite set. Which a would give you most distinct value of $a^n \pmod{26}$ as you run through all values for n? And do you notice a pattern in the finite collection of all the $a^n \pmod{26}$?

(Go to solution, page 63) □

Exercise 2.2.18. Continuing the previous question ... What about $a^n \pmod{p}$ when p is a prime? How many values do you get as you run through $n = 0, 1, 2, ...$? First try it for $a = 2$: try about 20 to 50 primes p. Do you see a pattern? Once you have spotted the pattern, try $a = 3$, etc. (The above is a very serious research problem for you.) (Go to solution, page 64) □

debug: exercises/exercise-17/question.tex

Exercise 2.2.19.

debug: exercises/exercise-18/question.tex

(a) What is $10 \pmod{3}$? (when simplified).

(b) What is $100 \pmod{3}$? (when simplified).

(c) What is $1000 \pmod{3}$? (when simplified).

(d) What is 5342 (mod 3)? (when simplified).
(e) What is 534603142235187 (mod 3)? (when simplified).
(f) What is the general fact here?

(Go to solution, page 65) □

Exercise 2.2.20.

debug: exercises/exercise-19/question.tex

(a) What is 10 (mod 9)? (when simplified).
(b) What is 100 (mod 9)? (when simplified).
(c) What is 1000 (mod 9)? (when simplified).
(d) What is 9142 (mod 9)? (when simplified).
(e) What is 96331420351887 (mod 9)? (when simplified).
(f) What is the general fact here?

(Go to solution, page 66) □

Exercise 2.2.21.

debug: exercises/exercise-20/question.tex

(a) What is 10 (mod 11)? (when simplified).
(b) What is 100 (mod 11)? (when simplified).
(c) What is 1000 (mod 11)? (when simplified).
(d) What is 9142 (mod 11)? (when simplified).
(e) What is 80432440556787 (mod 11)? (when simplified).
(f) What is the general fact here?

(Go to solution, page 82) □

Exercise 2.2.22. Although the algebraic manipulations on mod N seems to be very similar to the algebraic manipulations in $\mathbb{Z}$ (or even $\mathbb{R}$), certain "bizarre" things do happen. In $\mathbb{Z}$ (in fact in $\mathbb{Q}$ and $\mathbb{R}$ as well), you have the implication

debug: exercises/exercise-21/question.tex

$$xy = 0 \implies x = 0 \text{ or } y = 0$$

Is it true that

$$xy \equiv 0 \pmod{26} \implies x \equiv 0 \pmod{26} \text{ or } y \equiv 0 \pmod{26}$$

For each N, check when

$$xy \equiv 0 \pmod{N} \implies x \equiv 0 \pmod{N} \text{ or } y \equiv 0 \pmod{N}$$

holds. And when it does not, give one counterexample. Do you see a pattern?
(Go to solution, page 83) □

Exercise 2.2.23. (Dr.Liow's magic formula) Suppose I want to simplify 23532 (mod 26). With a calculator or C++ or python, we see quickly that

debug: exercises/exercise-22/question.tex

$$23532 \equiv 2 \pmod{26}$$

I claim that you can use this magic formula: Suppose the digits of a 5–digit number is $edcba$. For instance $edcba = 23532$ where $e = 2, d = 3, c = 5, b = 3, a = 2$. Then

$$edcba \equiv ba + 2(-2c + 6d - 5e) \pmod{26}$$

For instance when $edcba = 23532$, we have

$$edcba \equiv 32 + 2(-2 \cdot 5 + 6 \cdot 3 - 5 \cdot 2) \pmod{26}$$

and

$$32 + 2(-2 \cdot 5 + 6 \cdot 3 - 5 \cdot 2) \equiv 6 + 2(-2) \equiv 2 \pmod{26}$$

which is easier to work with than the much larger 23532. So ... is

$$edcba \equiv ba + 2(-2c + 6d - 5e) \pmod{26}$$

really true for any 5-digit number $edcba$? Write a program to test all cases. Next, write a proof (that does not involve checking all cases like a program).

(Go to solution, page 69) □

Exercise 2.2.24. Continuing the previous question:

debug: exercises/exercise-23/question.tex

(a) I further claim that if the number is an 8–digit number $hgfedcba$, then

$$hgfedcba \equiv ba + 2((-2c + 6d - 5e) + (2f - 6g + 5h)) \pmod{26}$$

This is also the same as saying

$$hgfedcba \equiv ba + 2((2(f - c) + 6(d - g) + 5(h - e)) \pmod{26}$$

Is this true? Write a program to check. If it's true, prove it.

(b) Can you conjecture a general formula? Can you prove it?

(Go to solution, page 70) □

Exercise 2.2.25.

debug: exercises/exercise-24/question.tex

(a) Solve the following

$$\begin{aligned} 9x + 5y &\equiv 3 \pmod{26} \\ 5x + 7y &\equiv 1 \pmod{26} \end{aligned}$$

First solving by writing a program that performs a brute force search for solutions. Next, try to solve it algebraically by hand.

(b) What about this one:

$$\begin{aligned} 3x - y &\equiv 2 \pmod{26} \\ 2x + 19y &\equiv 14 \pmod{26} \end{aligned}$$

(c) And this one:

$$\begin{aligned} 9x + y &\equiv 2 \pmod{26} \\ 19x + 5y &\equiv 7 \pmod{26} \end{aligned}$$

(d) Write a program that solves

$$\begin{aligned} ax + b &\equiv c \pmod{26} \\ dx + ey &\equiv f \pmod{26} \end{aligned}$$

for a, b, c, d, e, f in $\mathbb{Z}/26$ by brute force search. Then write a program that randomly picks a, b, c, d, e, f in $\mathbb{Z}/26$ and ask you to solve it. Print all the cases where a, b, c, d, e, f provides a linear system of two equations or two unknowns where there is no solution. Do you notice a pattern in these degenerate cases?

(Go to solution, page 71) □

Exercise 2.2.26. Notice that earlier on, we wrote down multiplicative inverses of elements in $\mathbb{Z}/26$. Some elements do not have inverses. Can you tell if there's something common among them?

debug: exercises/exercise-25/question.tex

(a) Compute the multiplicative inverses of elements in $\mathbb{Z}/3$.
(b) Compute the multiplicative inverses of elements in $\mathbb{Z}/5$.
(c) Compute the multiplicative inverses of elements in $\mathbb{Z}/7$.
(d) Compute the multiplicative inverses of elements in $\mathbb{Z}/11$.
(e) Do you notice something special about the above cases? How many elements have multiplicative inverse?

(f) Compute the multiplicative inverses of elements in $\mathbb{Z}/6$.
(g) Compute the multiplicative inverses of elements in $\mathbb{Z}/10$.
(h) Compute the multiplicative inverses of elements in $\mathbb{Z}/14$.
(i) Compute the multiplicative inverses of elements in $\mathbb{Z}/15$.
(j) In the above 4 cases is there something special about values which are invertible? Don't see the pattern? In the above, the modulus are all products of two distinct primes.

(Go to solution, page 72) □

Exercise 2.2.27. Suppose you want to write a random number generator. Say the numbers are to be in the range [0,256]. (You can try a bigger range later – don't be too ambitious for now.) Of course use (reasonable) formula.

debug: exercises/exercise-26/question.tex

(a) Go ahead and try this one:

$$h(n) = (2n + 5) \pmod{256}$$

Starting with a seed value of 1, the next value is $h(1) = 7$. And the value after that is $h(7) = 19$. And the next is $h(19) = 43$. What are all the possible values you can obtain using this h?

(b) Now of course you do want to have lots of values. For instance you might want to use this hash function to build a hashtable. Or a cryptographic hash (which we have no covered yet). But in any case, you hope that h covers *all* the values in $[0, 255]$. Was the function above a good function?
(c) What if you use a different seed value? What if you start with 2? Or 3?
(d) Try to find a function that generates as many values in $[0, 255]$ as possible. Can you find one that covers the whole range of $[0, 255]$?

(Go to solution, page 73) □

Solutions

Solution to Exercise 2.2.1.

debug: exercises/exercise-0/answer.tex

(a) True: $26 \mid 100 - 74 = 26$
(b) False: $26 \nmid -3 - 133 = -136 = -130 - 6 = (-6 \cdot 26) + 20$
(c) False: $26 \nmid -20 - 21 = -41 = (-2) * 26 + 11$
(d) False: $3 \nmid 7 - 3 = 4 = 1 cdot 3 + 1$
(e) True: $5 \nmid 17 + 3 = 4 \cdot 5$
(f) False: $8 \mid 21 - 11 = 10 = 1 \cdot 8 + 2$
(g) True: $7 \mid 42 - 0 = 6 \cdot 7$

Solution to Exercise 2.2.2.

debug: exercises/exercise-1/answer.tex

Solution not provided.

Solution to Exercise 2.2.3.

debug: exercises/exercise-2/answer.tex

Solution not provided.

Solution to Exercise 2.2.4.

debug: exercises/exercise-3/answer.tex

Addition table for $\mathbb{Z}/26$:

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1
3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2
4	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3
5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4
6	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5
7	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6
8	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7
9	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8
10	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9
11	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10
12	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11
13	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12
14	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13
15	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
16	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
20	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
21	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
22	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
23	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
24	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
25	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Multiplication table for $\mathbb{Z}/26$:

×	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	0	2	4	6	8	10	12	14	16	18	20	22	24	0	2	4	6	8	10	12	14	16	18	20	22	24
3	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23
4	0	4	8	12	16	20	24	2	6	10	14	18	22	0	4	8	12	16	20	24	2	6	10	14	18	22
5	0	5	10	15	20	25	4	9	14	19	24	3	8	13	18	23	2	7	12	17	22	1	6	11	16	21
6	0	6	12	18	24	4	10	16	22	2	8	14	20	0	6	12	18	24	4	10	16	22	2	8	14	20
7	0	7	14	21	2	9	16	23	4	11	18	25	6	13	20	1	8	15	22	3	10	17	24	5	12	19
8	0	8	16	24	6	14	22	4	12	20	2	10	18	0	8	16	24	6	14	22	4	12	20	2	10	18
9	0	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17
10	0	10	20	4	14	24	8	18	2	12	22	6	16	0	10	20	4	14	24	8	18	2	12	22	6	16
11	0	11	22	7	18	3	14	25	10	21	6	17	2	13	24	9	20	5	16	1	12	23	8	19	4	15
12	0	12	24	10	22	8	20	6	18	4	16	2	14	0	12	24	10	22	8	20	6	18	4	16	2	14
13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13
14	0	14	2	16	4	18	6	20	8	22	10	24	12	0	14	2	16	4	18	6	20	8	22	10	24	12
15	0	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18	7	22	11
16	0	16	6	22	12	2	18	8	24	14	4	20	10	0	16	6	22	12	2	18	8	24	14	4	20	10
17	0	17	8	25	16	7	24	15	6	23	14	5	22	13	4	21	12	3	20	11	2	19	10	1	18	9
18	0	18	10	2	20	12	4	22	14	6	24	16	8	0	18	10	2	20	12	4	22	14	6	24	16	8
19	0	19	12	5	24	17	10	3	22	15	8	1	20	13	6	25	18	11	4	23	16	9	2	21	14	7
20	0	20	14	8	2	22	16	10	4	24	18	12	6	0	20	14	8	2	22	16	10	4	24	18	12	6
21	0	21	16	11	6	1	22	17	12	7	2	23	18	13	8	3	24	19	14	9	4	25	20	15	10	5
22	0	22	18	14	10	6	2	24	20	16	12	8	4	0	22	18	14	10	6	2	24	20	16	12	8	4
23	0	23	20	17	14	11	8	5	2	25	22	19	16	13	10	7	4	1	24	21	18	15	12	9	6	3
24	0	24	22	20	18	16	14	12	10	8	6	4	2	0	24	22	20	18	16	14	12	10	8	6	4	2
25	0	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Negative of table for $\mathbb{Z}/26$:

x	$-x \pmod{26}$
0	0
1	25
2	24
3	23
4	22
5	21
6	20
7	19
8	18
9	17
10	16
11	15
12	14
13	13
14	12
15	11
16	10
17	9
18	8
19	7
20	6
21	5
22	4
23	3
24	2
25	1

Multiplicative inverse table for $\mathbb{Z}/26$:

$x \pmod{26}$	$x^{-1} \pmod{26}$
0	None
1	1
2	None
3	9
4	None
5	21
6	None
7	15
8	None
9	3
10	None
11	19
12	None
13	None
14	None
15	7
16	None
17	23
18	None
19	11
20	None
21	5
22	None
23	17
24	None
25	25

Solution to Exercise 2.2.5.

debug: exercises/exercise-4/answer.tex

Solution not provided.

Solution to Exercise 2.2.6.

debug: exercises/exercise-5/answer.tex

Solution not provided.

Solution to Exercise 2.2.7.

debug: exercises/exercise-6/answer.tex

Solution not provided.

Solution to Exercise 2.2.8.

debug: exercises/exercise-7/answer.tex

Solution not provided.

Solution to Exercise 2.2.9.

debug: exercises/exercise-8/answer.tex

Solution not provided.

Solution to Exercise 2.2.10.

debug: exercises/exercise-9/answer.tex

Solution not provided.

Solution to Exercise 2.2.11.

debug: exercises/exercise-10/answer.tex

Solution not provided.

Solution to Exercise 2.2.12.

debug: exercises/exercise-11/answer.tex

(a) $0^2, 1^2, 2^2, 3^2 \equiv 0, 1, 4, 9 \equiv 0, 1, 0, 1 \pmod 4$ (respectively). Therefore a square mod 4 is either 0 or 1 mod 4.

(b) $0^2, 1^2, 2^2, 3^2, 4^2 \equiv 0, 1, 4, 9, 16 \equiv 0, 1, 4, 4, 1 \pmod 4$ (respectively). Therefore a square mod 5 is either 0 or 1 or 4 mod 5.

(c) $0^2, 1^2, 2^2, 3^2, 4^2, 5^2, 6^2 \equiv 0, 1, 4, 9, 16, 25, 36 \equiv 0, 1, 4, 2, 2, 4, 1 \pmod 4$ (respectively). Therefore a square mod 7 is either 0 or 1 or 2 or 4 mod 7.

(c) Note that $(n-i)^2 \equiv (0-i)^2 \equiv (-i)^2 \equiv i^2 \pmod n$. Therefore $14^2 \equiv 1, 13^2 \equiv 2, ..., 8^2 \equiv 7^2 \pmod{15}$. $0, 1, 4, 9, 16, 25, 36, 49 \equiv 0, 1, 4, 9, 11, 10, 6 \pmod{15}$ (respectively). Therefore a square mod 15 is $0, 1, 4, 6, 9, 10$ or 11 mod 15.

(e) I claim that there are no integer solutions to the given equation. Assume on the contrary that (x, y) is an integer solution to

$$5x^2 - 8y^2 = 3$$

Therefore

$$\begin{aligned} 5x^2 - 8y^2 &\equiv 3 \pmod 4 \\ \therefore \quad 1 \cdot x^2 - 0 \cdot 8y^2 &\equiv 3 \pmod 4 \\ \therefore \quad x^2 &\equiv 3 \pmod 4 \end{aligned}$$

which is a contradiction since squares mod 4 are either 0 or 1 mod 4. Therefore $5x^2 - 8y^2 = 3$ does not have any integer solution.

Solution to Exercise 2.2.13.

debug: exercises/exercise-12/answer.tex

Solution not provided.

Solution to Exercise 2.2.14.

debug: exercises/exercise-13/answer.tex

Solution to Exercise 2.2.15.

debug: exercises/exercise-14/answer.tex

Of course $(x, y, z) = (0, 0, 0)$ is a solution since, for instance, the polynomial is homogeneous. If any two of x, y, z are 0s, then the third must also be 0.

Now consider the case where exactly one of x, y, z is 0.

(a) If $x = 0$, then $4y^2 = 5z^2$. We can assume that all common factors between y, z are removed so that $\gcd(y, z) = 1$. (Suppose $g = \gcd(y, z)$. Let $y' = y/g$ and $z' = z/g$, We again would arrive at $4y'^2 = 5z'^2$.) Since $5 \mid 5z^2$, we have $5 \mid 4y^2$, and hence $5 \mid y$. Therefore $y = 5y'$. Hence $4(5y')^2 = 5z^2$, i.e., $4(5)y'^2 = z^2$, which implies that $5 \mid z$. This contradicts $\gcd(y, z) = 1$.

(b) If $y = 0$, then $3x^2 = 5z^2$. We can assume that all common factors between x, z are removed so that $\gcd(x, z) = 1$. Since $3 \mid 3x^2$, we obtain $3 \mid 5z^2$, which implies that $3 \mid z$. Therefore $z = 3z'$ and hence $3x^2 = 5(3z')^2$. This implies that $x^2 = 5(3)z'^2$. Therefore $3 \mid x^2$, and hence $3 \mid x$. This contradicts $\gcd(x, z) = 1$.

(c) If $z = 0$, then $3x^2 + 4y^2 = 0$. Since $3 \mid 3x^2$, we get $3 \mid 4y^2$, which implies that $3 \mid y$. Let $y = 3y'$. Then $3x^2 + 4(3y')^2 = 0$ and hence $x^2 = 4(3)y'^2$. Therefore $3 \mid x^2$ and hence $3 \mid x$. This contradicts $\gcd(x, y) = 1$.

Now suppose $(x, y, z) \neq (0, 0, 0)$.

Let x, y, z be a solution with $x > 0, y > 0, z > 0$. We may assume $\gcd(x, y, z) = 1$.

METHOD 1. Taking mod 3,

$$y^2 \equiv 2z^2 \pmod 3$$

Squares in mod 3 are 0 or 1 mod 3. If $z^2 \equiv 1 \pmod 3$, then $y^2 \equiv 2 \pmod 3$ which is impossible. Hence $z^2 \equiv 0 \pmod 3$ and therefore $y^2 \equiv 0 \pmod 3$. This implies that $3 \mid z^2$ and $3 \mid y^2$. Therefore

$$3 \mid y, \quad 3 \mid z$$

Let $y = 3y'$ and $z = 3z'$. Then

$$3x^2 + 4(3y')^2 = 5(3z')^2$$

i.e.,

$$x^2 + 12y'^2 = 15z'^2$$

which implies that $3 \mid x$. This is a contradiction since $\gcd(x, y, z) = 1$.

Method 2. Taking mod 4, we get

$$3x^2 \equiv z^2 \pmod 4$$

Then $2 \mid x$ and $2 \mid z$. Let $x = 2x'$ and $z = 2z'$. Then

$$3(2x')^2 + 4y^2 = 5(2z')^2$$

i.e.,

$$3x'^2 + y^2 = 5z'^2$$

Taking mod 4,

$$3x'^2 + y^2 \equiv z'^2 \pmod 4$$

Then

- Assume z' is even. Then either x', y are even or $x'^2 \equiv 1 \equiv y^2 \pmod 4$. If x', y are even and z' is also even, then $\gcd(x, y, z) \neq 1$ which is a contradiction. For the other case

  $$3(4m+1)^2 + (4n+1)^2 = (2z'')^2$$

  i.e.,

  $$3(16m^2 + 8m + 1) + (16n^2 + 8n + 1) = 4z''^2$$

  i.e.,

  $$12m^2 + 6m + 4n^2 + 2n + 1 = z''^2$$

  All we can say is z'' is odd. There's no clear path forward.
- Assume $z'^2 \equiv 1 \pmod 4$. Then $2|x'$ and $y'^2 \equiv 1 \pmod 4$.

There are more cases to consider and this seems to be a bad direction to take.

Method 3. Taking mod 5, we get

$$3x^2 + 4y^2 \equiv 0 \pmod 5$$

i.e.

$$3x^2 \equiv y^2 \pmod 5$$

Squares in mod 5 are $0, 1, 4 \pmod 5$. If $x^2 \equiv 1 \pmod 5$, then $y^2 \equiv 3x^2 \equiv 3 \pmod 5$ which is impossible. If $x^2 \equiv 4 \pmod 5$, then $y^2 \equiv 3x^2 \equiv 12 \equiv 2 \pmod 5$ which is again impossible. Hence $x^2 \equiv 0 \pmod 5$ and hence $y^2 \equiv 0 \pmod 5$. Altogether we have

$$5 \mid x^2, \quad 5 \mid y^2$$

and therefore

$$5 \mid x, \quad 5 \mid y$$

Let $x = 5x'$ and $y = 5y'$. Then we have

$$3(5x')^2 + 4(5y')^2 = 5z^2$$

i.e.,

$$3(5)x'^2 + 4(5)y'^2 = z^2$$

which implies that $5 \mid z^2$. Therefore $5 \mid z$. This contradicts $\gcd(x, y, z) = 1$.
$\square$

Notes.

- What about $9x^2 + 4y^2 = 5z^2$? How far can you go using the method above?
- What about $3x^2 + 4y^2 = 25z^2$? How far can you go using the method above?
- What about $9x^2 + 4y^2 = 25z^2$?
- What about $3x^2 + 2y^2 = 5z^2$?
- What about $3x^3 + 4y^3 = 5z^4$?

Solution to Exercise 2.2.16.

debug: exercises/exercise-15/answer.tex

Solution not provided.

Solution to Exercise 2.2.17.

debug: exercises/exercise-16/answer.tex

Solution not provided.

Solution to Exercise 2.2.18.

debug: exercises/exercise-17/answer.tex

Solution not provided.

Solution to Exercise 2.2.19.

debug: exercises/exercise-18/answer.tex

Solution not provided.

Solution to Exercise 2.2.20.

debug: exercises/exercise-19/answer.tex

Solution not provided.

Solution to Exercise 2.4.1.

debug: exercises/exercise-20/answer.tex

Solution not provided.

Solution to Exercise 2.4.2.

debug: exercises/exercise-21/answer.tex

Solution not provided.

Solution to Exercise 2.2.23.

debug: exercises/exercise-22/answer.tex

Solution not provided.

Solution to Exercise 2.2.24.

debug: exercises/exercise-23/answer.tex

Solution not provided.

Solution to Exercise 2.2.25.

debug: exercises/exercise-24/answer.tex

Solution not provided.

Solution to Exercise 2.2.26.

debug: exercises/exercise-25/answer.tex

Solution not provided.

Solution to Exercise 2.2.27.

debug: exercises/exercise-26/answer.tex

Solution not provided.

2.3 Attacks debug: classical-cipher-attacks.tex

OK. Now it's time to attack some of the classical cryptosystems. Before we do that we want to know exactly what is it we want to achieve.

Recall that a cryptosystems is made up of the encryption and decryption function E and D.

Suppose Alice wants to send a message m to Bob. She encrypts the message to get $E_k(m)$ and delivers $E_k(m)$ to Bob. Bob decrypts $E_k(m)$ by applying D_k and get $D_k(E_k(m)) = m$.

Now let's think about the rogue agent Eve. What does she want?

The most common mode of attack assumes Eve receives a copy of $E_k(m)$ and she wants to derive m. It's even better if she can derive k because in that case she actually has D_k and hence can decrypt all future ciphertexts. (Don't forget we assume that the encryption and decryption algorithm is known. That means if Eve has k, she has D_k and in fact also E_k. Only the key(s) is secret). Not only that. If she has k, she also has E_k and therefore she can actually impersonate Alice!

The various assumptions of what Eve has are called **attack models** or **attack modes** Here are some standard ones.

attack models

attack modes

(a) The **known ciphertext attack** is an attack where the ciphertext of a plaintext is available to the attacker. In other words, this attack involves computing k from $E_k(m)$:

known ciphertext attack

$$E_k(m) \mapsto k$$

This include the case when there is more than one ciphertext.

(b) The **known plaintext attack** is an attack where the Eve has some messages and their ciphertext. So her goal is this:

known plaintext attack

$$m_1, m_2, ..., m_n, E_k(m_1), E_k(m_2), ..., E_k(m_n) \mapsto k$$

This does happen. For instance the breaking of Germany's Enigma ciphers during WWII is based on this assumption.

(c) The **chosen plaintext attack** is an attack where the Eve can actually encrypt messages/plaintexts that she chooses. Make sure you see that this is different from known plaintext.

chosen plaintext attack

(d) The **chosen ciphertext attack** is an attack where the Eve has access to the decryptions of ciphertexts that she chooses. Her goal is to compute

chosen ciphertext attack

the key. Make sure you see that this is different from known plaintext. For instance suppose Eve chooses (a meaningless) ciphertext x and sends it to the decryption machine. Suppose the decryption of x is y. It's very likely that y is totally meaningless. Bob then calls Alice and ask her why she sends the gibberish y message. On listening to their phone conversation, she now knows that

$$D_k(x) = y$$

Eve then attempts to find k.

2.4 Attacking the shift cipher debug: classical-cipher-attacking-shift-cipher.tex

Recall that for the shift cipher, we have

$$E_k(x) \equiv (x + k) \pmod{26}$$
$$D_k(x) \equiv (x - k) \pmod{26}$$

Since $0 \leq k \leq 25$, there are not many possible values for k. So you can try to apply *all* the possible decryptions D_0, D_1, D_2, ..., D_{25}. Assuming the message is something Eve can read, all that is required is that Eve has enough time to read 26 messages. Easy! Since Eve is trying all keys, this is a brute force attack.

But in fact Eve can do better, by using a heuristic approach based on probability. If the message is long enough, based on letter frequencies, heuristically, she can try to tabulate the frequencies of all the characters and then assume the most common occurring character is decrypted as `e` which is statistically the most commonly occurring letter in English. Of course knowing how to decrypt to `e` is sufficient for Eve to decrypt all the letters right?

Too bad if the message is in Russian.

The following is a table of probabilities for each letter used in English.

Letter	Probability
`e`	0.127
`t`	0.091
`a`	0.082
`o`	0.075
`i`	0.070
`n`	0.067
`s`	0.063
`h`	0.061
`r`	0.060
`d`	0.043
`l`	0.040
`c`	0.028
`u`	0.028
`m`	0.024
`w`	0.023
`f`	0.022
`g`	0.020
`y`	0.020
`p`	0.019
`b`	0.015
`v`	0.010
`k`	0.008
`j`	0.002
`x`	0.001
`q`	0.001
`z`	0.001

Of course these are probabilities. It does not mean that the second most frequently occurring letter *must* be `t`! I've divided up the probabilities into groups according to jumps in the values.

It is also useful to know that besides commonly occurring letters, which *pairs* of letters occurring frequently next to each other. These are called **digrams** (or 2–grams). For three, they are called **trigrams** (or 3–grams). The following is a table of commonly occurring digrams and trigrams listed in decreasing order of frequencies:

digrams

trigrams

n	n-grams (in decreasing order)
2	th he in er an re ed on es st en at to nt ha nd ou ea ng as or ti is et it ar te se hi of
3	the ing and her ere ent tha nth was eth for dth

Here are the frequencies of the 2-grams (of course not all 2-grams, just the top few):

2-gram	Probability
th	0.0271
he	0.0233
in	0.0203
er	0.0178
an	0.0161
re	0.0141
es	0.0132
on	0.0132
st	0.0125
nt	0.0117
en	0.0113
at	0.0112
ed	0.0108
nd	0.0107
to	0.0107
or	0.0106
ea	0.0100
ti	0.0099
ar	0.0098
te	0.0098
ng	0.0089
al	0.0088
it	0.0088
as	0.0087
is	0.0086
ha	0.0083
et	0.0076
se	0.0073
ou	0.0072
of	0.0071

and the 3-grams:

3-gram	Probability
the	0.0181
and	0.0073
ing	0.0072
ent	0.0042
ion	0.0042
her	0.0036
for	0.0034
tha	0.0033
nth	0.0033
int	0.0032
ere	0.0031
tio	0.0031
ter	0.0030
est	0.0028
ers	0.0028
ati	0.0026
hat	0.0026
ate	0.0025
all	0.0025
eth	0.0024
hes	0.0024
ver	0.0024
his	0.0024
oft	0.0022
ith	0.0021
fth	0.0021
sth	0.0021
oth	0.0021
res	0.0021
ont	0.0020

and the 4-grams:

4-gram	Probability
`tion`	0.31
`nthe`	0.27
`ther`	0.24
`that`	0.21
`ofth`	0.19
`fthe`	0.19
`thes`	0.18
`with`	0.18
`inth`	0.17
`atio`	0.17
`othe`	0.16
`tthe`	0.16
`dthe`	0.15
`ingt`	0.15
`ethe`	0.15
`sand`	0.14
`sthe`	0.14
`here`	0.13
`thec`	0.13
`ment`	0.12
`them`	0.12
`rthe`	0.12
`thep`	0.11
`from`	0.10
`this`	0.10
`ting`	0.10
`thei`	0.10
`ngth`	0.10
`ions`	0.10
`andt`	0.10

So a slight improve to brute force search of trying $k = 0, 1, 2, \ldots, 25$, is to try encrypt `e` to the most commonly occurring letter, the second, the third, etc.

(Some authors use M for their set of plaintexts instead of P. In that case they might call their plaintexts messages.)

Exercise 2.4.1.

debug: exercises/exercise-20/question.tex

(a) What is 10 (mod 11)? (when simplified).
(b) What is 100 (mod 11)? (when simplified).
(c) What is 1000 (mod 11)? (when simplified).
(d) What is 9142 (mod 11)? (when simplified).
(e) What is 80432440556787 (mod 11)? (when simplified).
(f) What is the general fact here?

(Go to solution, page 82) □

Exercise 2.4.2. Although the algebraic manipulations on mod N seems to be very similar to the algebraic manipulations in $\mathbb{Z}$ (or even $\mathbb{R}$), certain "bizarre" things do happen. In $\mathbb{Z}$ (in fact in $\mathbb{Q}$ and $\mathbb{R}$ as well), you have the implication

debug: exercises/exercise-21/question.tex

$$xy = 0 \implies x = 0 \text{ or } y = 0$$

Is it true that

$$xy \equiv 0 \pmod{26} \implies x \equiv 0 \pmod{26} \text{ or } y \equiv 0 \pmod{26}$$

For each N, check when

$$xy \equiv 0 \pmod{N} \implies x \equiv 0 \pmod{N} \text{ or } y \equiv 0 \pmod{N}$$

holds. And when it does not, give one counterexample. Do you see a pattern?
(Go to solution, page 83) □

Solutions

Solution to Exercise 2.4.1.

debug: exercises/exercise-20/answer.tex

Solution not provided.

Solution to Exercise 2.4.2.

debug: exercises/exercise-21/answer.tex

Solution not provided.

2.5 Affine cipher debug: classical-cipher-affine-cipher.tex

Suppose we use $\mathbb{Z}/26$ instead of `'a'` to `'z'` again. Recall that the shift cipher is

$$E(k, x) = (x + k) \pmod{26}$$

and

$$D(k, x) = (x - k) \pmod{26}$$

The benefit of translating our encryption/decryption "shift up" and "shift down" into mathematical operations in $\mathbb{Z}/26$, is that we can now generalize and use different mathematical formulas!

Here's the affine cipher. The encryption algorithm for the affine cipher looks like this:

$$E((a, b), x) = (ax + b) \pmod{26}$$

Note that the key is not one single number – the key is (a, b) mod 26 which is made up of two numbers. Why is that important?

Because this means that there are more key values! Which means that Eve has to try more key!!!! Get it??

But what is the decryption function? Of course we know that

$$D((a, b), E((a, b), x)) = x \pmod{26}$$

This means that

$$D((a, b), ax + b) = x \pmod{26}$$

Suppose I make a guess ... When I look at the shift cipher, I notice that the decryption function is similar in form to the encryption function. Maybe the decryption function for the affine cipher is similar in form to the encryption function???

Let's try

$$D((a, b), x) = cx + d \pmod{26}$$

In that case, from

$$D((a, b), ax + b) \equiv x \pmod{26}$$

we would get

$$c(ax + b)) + d \equiv x \pmod{26}$$

which gives us

$$cax + cb + d \equiv x \pmod{26}$$

Now what? Well maybe

$$cax \equiv x \pmod{26}$$

and

$$cb + d \equiv 0 \pmod{26}$$

The first condition

$$cax \equiv x \pmod{26}$$

is achieved if we have

$$ca \equiv 1 \pmod{26}$$

Can this be done?

Exercise 2.5.1. Suppose $a = 1$. What c's would make $ca \equiv 1 \pmod{26}$? What if $a = 2$? What about $a = 3$? Etc. (Go to solution, page 89) □

debug: exercises/exercise-30/question.tex

Given an integer a, if c satisfies

$$ca \equiv 1 \pmod{26}$$

we say that c is a **multiplicative inverse** of a mod 26. We usually write c as $a^{-1} \bmod 26$. Note that $a^{-1} \bmod 26$ is NOT a fraction!!! It's a whole number. If a has a multiplicative inverse mod 26, we say that a is **invertible** mod 26.

multiplicative inverse

invertible

Recall from a section section, you already have a table of $a^{-1} \bmod 26$ for all a's in mod 26.

Exercise 2.5.2. How many integers $0, 1, 2, ..., 25$ have multiplicative inverses mod 26? (Go to solution, page 90) □

debug: exercises/exercise-31/question.tex

Therefore to satisfy

$$ca \equiv 1 \pmod{26}$$

we can't just pick any a. We have to pick an a with a multiplicative inverse mod 26.

After we have chosen a good a, what do we do? We then have

$$D((a, b), x) = cx + d \pmod{26}$$

where c is the multiplicative inverse of a mod 26. But what about d???

Remember we still have the condition

$$cb + d \equiv 0 \pmod{26}$$

Writing $a^{-1} \mod 26$ for c, we get

$$a^{-1}b + d \equiv 0 \pmod{26}$$

we get

$$d \equiv -a^{-1}b \pmod{26}$$

Therefore we have the following: The affine cipher is

$$E((a, b), x) = (ax + b) \pmod{26}$$

where a is invertible mod 26 and

$$\begin{aligned} D((a, b), x) &\equiv a^{-1}x - a^{-1}b \pmod{26} \\ &\equiv a^{-1}(x - b) \pmod{26} \end{aligned}$$

Exercise 2.5.3. Write down the (simplified) encryption and decryption function for the affine cipher when the key is $(3, 12)$. Encrypt `gollum` and then decrypt to check that you get back `gollum`. (Go to solution, page 91) □ debug: exercises/exercise-32/question.tex

The above however uses a lot of "iffy" math. For instance we used the fact

$$a(b + c) \equiv ab + ac \pmod{26}$$

(where?) We also use the fact that if

$$a + b \equiv 0 \pmod{26}$$

then

$$a \equiv -b \pmod{26}$$

We seem to be treated math in mod 26 like math in $\mathbb{Z}$ and $\mathbb{R}$!!! Is that justifiable? It turns out that the above algebra is actually correct. I'll have to come back to that later otherwise people will think we are rambling nonsense and making things up.

Exercise 2.5.4. What is the size of the key space for the affine cipher? In the worse case how many tries must Eve attempt before breaking an affine cipher? debug: exercises/exercise-33/question.tex

Compare this with the shift cipher. (Go to solution, page 92) □

Now let's look at attacking the affine cipher.

Recall the encryption and decryption of the affine cipher looks like

$$E_{a,b}(x) \equiv (ax+b) \pmod{26}, \qquad D_{a,b}(x) \equiv a^{-1}(x-b) \pmod{26}$$

Note that the key is (a,b). Note also that a must be invertible mod 26.

Therefore (by the multiplication principle in discrete mathematics), the total numbers of keys is

$$\phi(26) \cdot 26 = 312$$

This is not that big, but it's definitely bigger than the number of keys for the shift cipher (which is 26). This means that to carry out a brute force attack on an affine cipher, assume the attacker has the cipher, he/she must try 312 possible keys.

Exercise 2.5.5. This is easy: Show that $\phi(26) \cdot 26 = 312$. (Go to solution, page 93) □

debug: exercises/affine-0/question.tex

Again you can do a brute force search for a, b. After all there are not that many possibilities for a and b. But we can do better if we use letter (1–gram) frequencies again. Again suppose you have computed the frequencies of the letters of the ciphertext and say that `g` is the most common letter. So you assume `e` is encrypted as `g`. This is the same as saying 4 is encrypted as 6, i.e.,

$$E_{a,b}(4) = 6$$

right? Now using the formula for $E_{a,b}$ we get

$$4a + b = 6$$

To be accurate the equation should be

$$4a + b \equiv 6 \pmod{26}$$

Now suppose the second most common letter in the ciphertext is `y`. So you assume that `t` is encrypted as `y`. This means

$$20a + b \equiv 24 \pmod{26}$$

Right? Yes, no? Think about it. So you can solve for a and b from the linear equations

$$\begin{aligned} a + b &\equiv 6 \pmod{26} \\ 20a + b &\equiv 24 \pmod{26} \end{aligned}$$

Exercise 2.5.6. Solve the above for a and b. (Go to solution, page 94) □

debug: exercises/affine-1/question.tex

Exercise 2.5.7. Now let's abstract the above. Suppose A, B, C, D are numbers and a and b satisfy the equations

debug: exercises/affine-2/question.tex

$$\begin{aligned} Aa + b &= B \\ Ca + b &= D \end{aligned}$$

Solve for a and b in terms for A, B, C, D. If this can be done, then of course you can write a C++ function to return a and b directly. (Go to solution, page 95) □

Exercise 2.5.8. But wait! a is not arbitrary! Remember that a must be invertible in mod 26. Recall that this means there is some integer c such that $ac \equiv 1 \pmod{26}$. Therefore it would be helpful if you have a function that will determine if a is invertible mod 26. How would you do that? (Go to solution, page 96) □

debug: exercises/affine-3/question.tex

Solutions

Solution to Exercise 2.5.1.

debug: exercises/exercise-30/answer.tex

Solution not provided.

Solution to Exercise 2.5.2.

debug: exercises/exercise-31/answer.tex

Solution not provided.

Solution to Exercise 2.5.3.

Solution not provided.

debug: exercises/exercise-32/answer.tex

Solution to Exercise 2.5.4.

debug: exercises/exercise-33/answer.tex

Solution not provided.

Solution to Exercise 2.5.5.

debug: exercises/affine-0/answer.tex

Solution not provided.

Solution to Exercise 2.5.6.

debug: exercises/affine-1/answer.tex

Solution not provided.

Solution to Exercise 2.5.7.

debug: exercises/affine-2/answer.tex

Solution not provided.

Solution to Exercise 2.5.8.

debug: exercises/affine-3/answer.tex

Solution not provided.

2.6 Vigenère cipher debug: classical-cipher-vigenere-cipher.tex

The shift, affine, substitution ciphers are called **monoalphabetic** ciphers – each letter can be mapped to only one letter. monoalphabetic

A **polyalphabetic** ciphers is the opposite of monoalphabetic ciphers. polyalphabetic

The Vigenere cipher is basically a collection of shifts. Here's an example.

My key is going to be the word `fun`. Note that when I translate `fun` to numbers (with `a` $\rightarrow 0$, `b` $\rightarrow 1$, `c` $\rightarrow 2$, ...) I get $5, 20, 13$. So all I need to do is to encrypt the characters by doing

- shift by 5
- shift by 20
- shift by 13
- shift by 5
- shift by 20
- shift by 13
- ...

For instance suppose the plaintext is

```
It's a dangerous business, Frodo, going out your door.
```

I change everything to lowercase and throwing away non a-z and our plaintext `x` is:

```
itsadangerousbusinessfrodogoingoutyourdoor
```

Suppose the encrypted text is

$$y = \mathtt{y}_1\mathtt{y}_2\mathtt{y}_3\mathtt{y}_4\mathtt{y}_5\mathtt{y}_6\cdots$$

Then

- $\mathtt{y}_1$ = shift `i` by 5 = `n`
- $\mathtt{y}_2$ = shift `t` by 20 = `n`
- $\mathtt{y}_3$ = shift `s` by 13 = `f`
- $\mathtt{y}_4$ = shift `a` by 5 = `f`
- $\mathtt{y}_5$ = shift `d` by 20 = `x`
- $\mathtt{y}_6$ = shift `a` by 13 = `n`
- $\mathtt{y}_4$ = shift `n` by 5 = `s`
- $\mathtt{y}_5$ = shift `g` by 20 = `a`
- $\mathtt{y}_6$ = shift `e` by 13 = `r`
- ...

i.e.,

$$y = \texttt{nnffxnsar...}$$

Exercise 2.6.1. What is the size of the set of all possible keys for the Vigenere cipher? Is this better than the shift cipher? (Go to solution, page 111) □

debug: exercises/vigenere-0/question.tex

Now let's attack the Vignere cipher.

Suppose the length of the key is m. Then you can think of your message as being chopped up into m strings and each is encrypted by the shift of a character and then everything is put together. For instance if the ciphertext is

$$y = \texttt{y}_1\texttt{y}_2\texttt{y}_3\texttt{...}$$

and the key is `fun`, i.e. m is 3, then the 3 strings are

$$z_1 = \texttt{y}_1\texttt{y}_4\texttt{y}_7 \cdots$$
$$z_2 = \texttt{y}_2\texttt{y}_5\texttt{y}_8 \cdots$$
$$z_3 = \texttt{y}_3\texttt{y}_6\texttt{y}_9 \cdots$$

Now, the above of course is the encryption process and you know that the key has length 3, therefore you are three different shifts. If you do know the key has length 3, then each of the above z_i is encrypted using the same shift and therefore you can use frequency analysis to decrypt each of them separately.

The problem is when you do *not* have the length of the key. So the first step is always to figure out the length of the key, i.e. m.

Method 1. Here's a strategy. Look at the substrings of the ciphertext of length 3. Think about it. Suppose you see the following in your ciphertext:

$$\ldots\texttt{etu}\ldots\texttt{etu}\ldots$$

What can you do? One might guess that if the distance between the first `e` to the next is d, then m divides d right? Think about it. So you need to scan for all possible substrings of length 3 and look for such distances. Suppose you have a bunch of such distances $d_1, d_2, \ldots, d_k$. Then m must divide all these d_i's. This means that m divides $\gcd(d_1, d_2, \ldots, d_k)$. Obviously the more d_i's you have the better.

Exercise 2.6.2. Write a function (in your favorite programming language). If returns a hashtable of frequencies of strings of length 1. (Go to solution,

debug: exercises/vigenere-1/question.tex page 112) ☐

Exercise 2.6.3. Generalize the above: Write a function (in your favorite programming language) that accepts a string s and returns a hashtable of where the key-value pair (k, v) has a string length length k for key and the value v is the count of occurrences of k in s. (Go to solution, page 113) ☐

debug: exercises/vigenere-2/question.tex

Exercise 2.6.4. Now modify the above: Write a function (in your favorite programming language) that accepts a string s and returns a hashtable of where the key-value pair (k, v) has a string length length k for key and the value v is a list of index positions in s where the k was found. (Go to solution, page 114) ☐

debug: exercises/vigenere-3/question.tex

Exercise 2.6.5. Using the above, write a function that computes a probabilistic guess on the length of the key when given a string encrypted using the Vigenere cipher. (Go to solution, page 115) ☐

debug: exercises/vigenere-4/question.tex

Once you have a probabilistic guess of the length, say m, you break up the encrypted string into m pieces. Each piece is a a shift ciphertext. So you just use whatever we talked about in the sections on shift cipher.

Done!

METHOD 2. Here's another technique. It involves some simple counting from discrete math. Suppose again the length of the key of m and you cut up your string into m pieces so that the characters in each substring is shifted by the same amount. Clearly the probabilities of the letters for each substring is preserved. In other words if `e` is encrypted as `g` for the first string and to `n` for the second, etc., then the probability of `g` in the first string and of `n` in the second string must be approximately the same as the probability of `e`. On the other hand, if we assumed that the length wrongly, say $m + 1$, then the probabilities must be different. As a matter of fact the substrings would appear very random. OK. So let's do some math.

Suppose the frequencies of the letters in a string s are f_0, f_1, ..., f_{25}, i.e., f_0 is the frequency of `a`, etc. Suppose the length of the s is n. Therefore

$$\sum_{i=0}^{25} f_i = n$$

In this string s, the probability p_0 that a randomly chosen character of s is the character `a` is just f_0/n. Etc. In the following I might write $f(0)$ instead of f_0. So in the following discussion if you see p_0 you can think of f_0/n.

In python or C++, for statistical computation of frequencies of characters (or substrings), you can use a hashtable. (Details in assignment.) For instance in python you can do this. A hashtable in python is called a python dictionary. In C++ a hashtable is called an unordered map. Try this for python:

```
s = "tobeornottobethatisthequestionandtheanswerisfortytwo"
f = {}
for c in s:
    if c not in f:
        f[c] = 0
    else:
        f[c] += 1
for c,count in f:
    print(c, count)
```

or

```
s = "tobeornottobethatisthequestionandtheanswerisfortytwo"
f = {}
for c in s:
    f[c] = f.get(c, 0)
for c,count in f:
    print(c, count)
```

Make sure you run the above. (For those of you who have taken CISS350, check my notes on hashtables and review `std::unordered_map`.) A lot more information on python dictionary is found in the assignment. Moving on ...

Now the probability of two randomly chosen characters being the same must be

$$I(s) = \frac{\binom{f_0}{2} + \ldots + \binom{f_{25}}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}$$

$\binom{n}{k}$ is the n–choose–k binomial coefficient, i.e. $n!/((n-k)!k!)$. In particular $\binom{n}{2} = n(n-1)/2$.

One thing nice about the above formula is that if any substitution has been applied to the string, *the above value remains the same.*

Exercise 2.6.6. Approximate $I(s) = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)}$ where $p_i = f_i/n$'s are the probabilities taken from our table of letter probabilities. (Go to solution,

debug: exercises/vigenere-5/question.tex 0.065

page 116) ☐

So $I(s)$ is approximately $\sum_{i=0}^{25} p_i^2$ where p_i is the probability of a character of the string being the i–th letter.

On the other hand if there is no pattern and everything is random, for instance you cut up your ciphertext, which was encrypted by a substitution with a string of length m, into $m+1$ substrings, then the probabilities of the characters would be almost the same since the string is gibberish. In other words, the probabilities would be about $f_i/n = 1/26$.

Exercise 2.6.7. Suppose s is a random string, i.e. $f_i/n = 1/26$. Compute $I(s) = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)}$. (Go to solution, page 117) ☐

debug: exercises/vigenere-6/question.tex 0.038

Exercise 2.6.8. Write a function `I` that accepts a string of lowercase letters or a list of numbers 0..25 and computes the I–value as described above. (Go to solution, page 118) ☐

debug: exercises/vigenere-7/question.tex

So the question is this: Is the I value sufficiently different for a meaningful string and a random string? Is so, then we have an algorithm for determining m:

Test $m = 1$: You get one piece (the complete string) from the ciphertext y. Compute $I(y)$. If $I(y)$ is approximately 0.065, $m = 1$.

Test $m = 2$: Cut up your ciphertext y into two pieces z_1, z_2. Compute $I(z_1)$, $I(z_1)$. If they are approximately 0.065 (you can use the average of $I(z_1)$ and $I(z_1)$), then $m = 2$.

Test $m = 3$: Cut up your ciphertext y into three pieces z_1, z_2, z_3. Compute $I(z_1)$, $I(z_2)$, $I(z_3)$. If they are approximately 0.065 (you can use the average of all three), then $m = 3$.

Etc.

So now we know how to compute the length of the key. Suppose the plaintext is x and the m pieces of x are $x_1, x_2, x_3, ..., x_m$. Recall y is the ciphertext and the m pieces of y are $z_1, z_2, z_3, ..., z_m$. Suppose the shifts are by $k_1, k_2, \ldots k_m$. So z_1 is a shift of x_1 by k_1, z_2 is a shift of x_2 by k_2, etc.

Suppose p_0 is the probability of `a`, p_1 is the probability of `b`, etc. in unencrypted English. Look at y_1. If you look at the probability of `a` in z_1, then you're really looking at the p_α where α is encrypted as `a`. For instance suppose the shift is by 3. In other words x is encrypted as `a`. So the probability of `a` in z_1 is the same as the probability of `x` which is p_{26-3}. For simplicity, from now on all the indices will be considered mod 26. So when I write p_{-3} I really mean p_{26-3}. In general, if the shift is k, then the probability must be p_{-k}. Similarly, the probability of `b` in y_1 must be p_{1-k}. In general, the probability of the i–th character in z_1 must be p_{i-k}.

Now we make the following observation: What is the probability that a randomly chosen character from z_1 and a randomly chosen character from z_2 are both `a`? It is approximately

$$M(y_i, y_j) = \sum_{\ell=0}^{25} p_{\ell-k_1} p_{\ell-k_2}$$

For any z_i and z_j, this is then

$$\sum_{\ell=0}^{25} p_{\ell-k_i} p_{\ell-k_j}$$

You now notice that the above is the same as

$$\sum_{\ell=0}^{25} p_\ell p_{\ell-k_j+k_i}$$

or

$$\sum_{\ell=0}^{25} p_\ell p_{\ell+k_i-k_j}$$

The number $k_i - k_j$ is called the **relative shift** of z_i and z_j. relative shift

Exercise 2.6.9. Using the probabilities from our table of letter probabilities, debug: exercises/vigenere-8/question.tex complete the following table:

Relative shift	M
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

(Go to solution, page 119) □

You notice that when there is no shift, the M value is approximately 0.065. Otherwise it is much smaller, around 0.04. So this what you can do. Take z_1 and z_2 as examples. Compute the M value for z_1 and z_2. Next you shift z_2 by 1 to get $E_1(z_2)$ (E_1 is the encryption for the shift cipher). Now compute the M value for y_1 and $E_1(y_2)$, i.e. $M(y_1, E_1(y_2))$. Next you compute $M(y_1, E_2(y_2))$. Continue until you have the M value of y_1 and $E_{25}(y_1)$. If the M value for y_1 and $E_{17}(y_2)$ is approximately 0.065, then the relative shift is very likely 17. In other words

$$k_1 - k_2 = 17$$

Now perform the same for different pairs of y_i, y_j. You have $\binom{m}{2}$ pairs and you hope to get as many equations as possible.

You should be able to write your shifts in terms of one shift, say the first k_1. For instance the second shift might be $k_1 + 3$, the third is $k_1 + 2$, etc. You still have the unknown k_1.

But that's only one unknown!!!

You can then easily solve it with 26 different values for k_1. Alternatively, for a Vigenere cipher, the shifts come from a sequence of letters and the letters usually make up a meaningful word. You can scan for meaningful words using your sequence of shifts. For instance if the shifts are $(k_1, k_1 + 17, k_1 + 4, k_1 +$

$21, k_1 + 10)$, then when $k_1 = 9$ you get the word `janet`.

Notice that the cryptanalysis as above already hints at the fact that algebra and probability theory are extremely important in this area of study. In particular, there is an area of study called Information Theory that studies the amount of randomness or lack of information in random variables encoded within something called the entropy of random variables. Using information theory one can prove that a substitution ciphertext has a unique decryption if the string has a length of at least 25.

In case the above proof hurts your head too much, let me give you an informal proof.

Suppose I have a sequence of numbers:

$$[0.1, 0.2, 0.3, 0.4]$$

Think of these as the probabilities of a language that involves only 4 letters, say the probabilities of the symbols a, b, c, d. Now I rotate the numbers in a circle by 1 step to get this:

$$[0.4, 0.1, 0.2, 0.3]$$

and by 3 steps to get this:

$$[0.2, 0.3, 0.4, 0.1]$$

These corresponds to a shift cipher of 1 and of 3. The relative shift between the 1st and 2nd is 2 steps: you can see that 0.1 is 2 steps away between the two sequences. One way to compute the relative shift (i.e. 2) is to compute the sum of products of the corresponding terms:

```
Relative shift the second by 0
[0.4, 0.1, 0.2, 0.3]
[0.2, 0.3, 0.4, 0.1]
--------------------
0.4*0.2 + 0.1*0.3 + 0.2*0.4 + 0.3*0.1 = 0.2200...

Relative shift the second by -1
[0.4, 0.1, 0.2, 0.3]
[0.3, 0.4, 0.1, 0.2]
--------------------
0.4*0.3 + 0.1*0.4 + 0.2*0.1 + 0.3*0.2 = 0.2399...
```

```
Relative shift the second by -2
[0.4, 0.1, 0.2, 0.3]
[0.4, 0.1, 0.2, 0.3]
--------------------
0.4*0.4 + 0.1*0.1 + 0.2*0.2 + 0.3*0.2 = 0.3000...

Relative shift the second by -3
[0.4, 0.1, 0.2, 0.3]
[0.1, 0.2, 0.3, 0.4]
--------------------
0.4*0.1 + 0.1*0.2 + 0.2*0.3 + 0.3*0.4 = 0.2399...
```

You see that the largest value gives you the relative shift.

Of course if the probabilities were from ciphertexts, the numbers will only be *close* to $0.1, 0.2, 0.3, 0.4$ if the lengths are large enough.

[By the way, the reason why you get the largest value when the numbers line up nicely is because the above sum of product of corresponding term is called the inner product and the inner product is largest when the two sequence of number, think vectors, are pointing in the same direction.]

Exercise 2.6.10. Here's another example. Say this is the plaintext:

debug: exercises/vigenere-9/question.tex

```
This is a test. This tutorial is aimed at getting
familiar with the bare bones of LaTeX. First, ensure
that you have LaTeX installed on your computer (see
Installation for instructions of what you will
need). We will begin with creating the actual source
LaTeX, and then take you through how to feed this
through the LaTeX system to produce quality output,
such as postscript or PDF.
```

We use the key of `fun` to obtain the ciphertext

```
ybvxcffnrxngmcfyogtlvffvxuvryqfntjngnhtkuznfvfljnnuy
brguejvbsyftzyfnrczvwmgjhfzlrybnysbzbnayyfnrccaxnnqf
riiadihwwbrjhyyexyrnhfyuyqugniakienhfylhhnvthftzjmug
dihbcyqhrjxjjqvqfojavsqvybpwynycalnujupyonqmbzlpjfny
ykfhqybrsnnpyltogmlbzaumijyisjyqybvxnuwihlbgmyyfnrcm
```

```
lxnrrnbulbiopjkhffvysbzncznfzwufmctmgxwenjgtlciz
```

The key `fun` gives us the shifts $(5, 20, 13)$. Let's try to re-discover the shifts.

The (sorted) incidence values I for up to length 6 are

```
Average I-values     keylength
-------------------- ---------
0.06467025914470374  3
0.06262443438914027  6
0.04968484284445197  1
0.04938482570061517  4
0.04908514013749339  5
0.048510313216195575 2
```

Therefore we conjecture that the length is 3. (It shouldn't be surprising that multiples of 3 will also give high I-values, right?) In more details, for the case of testing a keylength of 3, the 3 strings are:

```
yxfxmytfxrfjnknfnygjstfcwjzyyzafcxqidwryxnyqnknyhttm
dbqjjqjsywyljyqzjyfysptmzmyjyxwlmfcxruijfyzzzftxnti

bcnncolfuynnhuflnbuvyznzmhlbsbyncnfiiwjyyhuuiihlnhzu
ichxqfaqbycnuomlfyhbnyolaiiybnibynmnnlokfsnnwmmwjlz

vfrgfgvvvqtgtzvjurebfyrvgfrnbnyranrahbherfygaefhvfjg
hyrjvovvpnaupnbpnkqrnlgbujsqvuhgyrlrbbphvbcfucgegc
```

with respective I-values

```
0.0651056539121
0.0687226346849
0.0601824888371
```

The I–values are computed with

$$\frac{\sum_{i=0}^{25} f(i)f(i-1)}{n(n-1)}$$

where f is the frequency function for the string (for instance $f(0)$ is the frequency of `a`) and n is the length of the string. The average of the above 3

I-values for keylength 3 is 0.0646702591447.

Next we compute the M-values. This means that for each distinct pair of strings (from the 3), let f_1 and f_2 be their frequencies and n, n' be their length, we compute

$$\frac{\sum_{i=0}^{25} f_1(i) f_2((i-g) \bmod 26)}{nn'}$$

with all possible relative shifts $g = 0, 1, 2, ..., 25$. ($f(0)$ means the frequency of `a` in the first string, etc.)

```
M-values        Index of 1st string Index of 2nd string Relative shift g
--------------- ------------------- ------------------- ----------------
0.0498633235932                   1                   2                0
0.0344990102743                   1                   2                1
0.0310114054105                   1                   2                2
0.0330851164106                   1                   2                3
0.0452446036384                   1                   2                4
0.0342162315016                   1                   2                5
0.0333678951833                   1                   2                6
0.0332736355924                   1                   2                7
0.0331793760015                   1                   2                8
0.0291262135922                   1                   2                9
0.0398718069564                   1                   2               10
0.0707889527759                   1                   2               11
 0.044584786502                   1                   2               12
0.0329908568197                   1                   2               13
0.0328023376379                   1                   2               14
0.0415684795928                   1                   2               15
0.0334621547742                   1                   2               16
0.0401545857291                   1                   2               17
0.0454331228202                   1                   2               18
 0.031576962956                   1                   2               19
0.0286549156377                   1                   2               20
0.0325195588651                   1                   2               21
0.0441134885475                   1                   2               22
0.0446790460929                   1                   2               23
0.0419455179565                   1                   2               24
0.0379866151381                   1                   2               25
0.0355035217971                   1                   3                0
0.0272225395012                   1                   3                1
0.0371216447744                   1                   3                2
0.0509232819341                   1                   3                3
 0.046925566343                   1                   3                4
0.0324576432515                   1                   3                5
0.0341709499334                   1                   3                6
0.0503521797068                   1                   3                7
0.0397867885018                   1                   3                8
0.0293165810013                   1                   3                9
0.0390253188654                   1                   3               10
0.0426422996383                   1                   3               11
0.0329335617742                   1                   3               12
0.0278888254331                   1                   3               13
0.0438796877974                   1                   3               14
0.0348372358652                   1                   3               15
 0.028745478774                   1                   3               16
0.0399771559109                   1                   3               17
0.0667237768894                   1                   3               18
0.0402627070246                   1                   3               19
 0.029602132115                   1                   3               20
```

0.0323624595469	1	3	21
0.0442604226156	1	3	22
0.0345516847516	1	3	23
0.0413097277746	1	3	24
0.037216828479	1	3	25
0.0413097277746	2	3	0
0.035313154388	2	3	1
0.0286502950695	2	3	2
0.0434989529792	2	3	3
0.0369312773653	2	3	4
0.0317913573196	2	3	5
0.0440700552065	2	3	6
0.0655815724348	2	3	7
0.0368360936608	2	3	8
0.031505806206	2	3	9
0.0355987055016	2	3	10
0.0369312773653	2	3	11
0.037216828479	2	3	12
0.0395012373882	2	3	13
0.0356938892062	2	3	14
0.0318865410242	2	3	15
0.0336950314106	2	3	16
0.0343613173425	2	3	17
0.055396916048	2	3	18
0.0403578907291	2	3	19
0.04016752332	2	3	20
0.0357890729107	2	3	21
0.0484485056158	2	3	22
0.0339805825243	2	3	23
0.0326480106606	2	3	24
0.0328383780697	2	3	25

We sort and list 6 rows with the highest M-values:

M-values	Index of 1st string	Index of 2nd string	Relative shift g
0.0707889527759	1	2	11
0.0667237768894	1	3	18
0.0655815724348	2	3	7
0.055396916048	2	3	18
0.0509232819341	1	3	3
0.0503521797068	1	3	7

For the first row, if the M-value using index 1 and index 2 string (i.e., z_1, z_2) with a relative shift of 11. The computed M–value is 0.0707.... In other words if f_1 and f_2 denote the frequency data of z_1 and z_2 respectively, then the M–value shown in the first row above is

$$\frac{\sum_{i=0}^{25} f(i) f'((i-11) \bmod 26)}{n_1 n_2}$$

where n_1 and n_2 are the lengths of z_1, z_2 respectively and $f_1(0)$ is the frequency of a in z_1, $f_1(1)$ is the frequency of b in z_1, etc. From the top 2 entries of M-values:

M-values	Index of 1st string	Index of 2nd string	Relative shift g

```
--------------- ------------------- ------------------- ----------------
0.0707889527759                  1                   2               11
0.0667237768894                  1                   3               18
```

we get

$$k_1 - k_2 \equiv 11 \pmod{26}$$
$$k_1 - k_3 \equiv 18 \pmod{26}$$

or

$$k_2 \equiv k_1 - 11 \equiv k_1 + 15 \pmod{26}$$
$$k_3 \equiv k_1 - 18 \equiv k_1 + 8 \pmod{26}$$

At this point, even though we don't have the 3 shifts yet, we at least know that the shifts is probably of the form

$$(k_1, \quad (k_1 + 15) \bmod 26, \quad (k_1 + 8) \bmod 26)$$

There is now *one* unknown. Now trying different values of k_1, we see that when $k_1 = 5$, we get

$$k_1 = 5, \quad k_2 = 20, \quad k_3 = 13$$

which corresponds to the word `fun`. Vóila! ... we have discovered the key. (Go to solution, page 120) □

Exercise 2.6.11. Here are things you need to do right away.

debug: exercises/vigenere-10/question.tex

1. Given a string y (a ciphertext) and an integer m (say 3) How to you form z_1, z_2, z_3 in python?
2. Given a string z write a function (call if `freq`) so that `freq(z)` returns the frequency data of z as a dictionary.
3. Write a function `avg_I` such that `avg_I(y, m)` computes the average I–value of the m pieces $z_1, z_2, ..., z_m$ of y.
4. Write a function `estimate_m` such that `estimate_m(y)` gives as estimate for m the length of key by doing a for-loop on $m = 1, 2, 3, ...$ until a large enough I value is attained and at that point the m is returned. You might want to write it with two other paramters, i.e., `estimate_m(y, m_cutoff=20, I_cutoff=0.06)` so that when the average I is ≥ 0.06, the m is returned. The function returns `None` if m reaches `m_cutoff`.
5. Write a function `M` such that `M(freq1, freq2, g)` computes the M–value for the obvious data.

6. Write a function `relative_shifts` such that `relative_shifts(freqs)` return a tuple of $m-1$ value for the relative shifts. (For instance in the example above, when $m=3$, the relative shifts are 15 and 8.
7. Finally once you call `relative_shifts` you have the $m-1$ relative shifts. So a for-loop for k_1 over a..z and print all the 26 words using the relative shifts and for the shifts, print the keys (as a word) and print the decryption of the first (say) 40 characters. The output with the meaning key and the meaningful decryption will give you the key. (This last part can be automated if you have a dictionary of English words and you know how to perform word segment – see CISS358.)

(Go to solution, page 121) □

Solutions

Solution to Exercise 2.6.1.

debug: exercises/vigenere-0/answer.tex

Solution not provided.

Solution to Exercise 2.6.2.

debug: exercises/vigenere-1/answer.tex

Solution not provided.

Solution to Exercise 2.6.3.

debug: exercises/vigenere-2/answer.tex

Solution not provided.

Solution to Exercise 2.6.4.

debug: exercises/vigenere-3/answer.tex

Solution not provided.

Solution to Exercise 2.6.5.

debug: exercises/vigenere-4/answer.tex

Solution not provided.

Solution to Exercise 2.6.6.

debug: exercises/vigenere-5/answer.tex

Solution not provided.

Solution to Exercise 2.6.7.

debug: exercises/vigenere-6/answer.tex

Solution not provided.

Solution to Exercise 2.6.8.

debug: exercises/vigenere-7/answer.tex

Solution not provided.

Solution to Exercise 2.6.9.

debug: exercises/vigenere-8/answer.tex

Solution not provided.

Solution to Exercise 2.6.10.

debug: exercises/vigenere-9/answer.tex

Solution not provided.

Solution to Exercise 2.6.11.

debug: exercises/vigenere-10/answer.tex

Solution not provided.

2.7 Substitution cipher debug: classical-cipher-substitution-cipher.tex

OK folks, things are getting more fun (or worse, depending on your point of view). If you've read your notes and following the readings you should know why the substitution cipher is not that easy to break. Right?

Because of the gap in the probability between `e` and the next letter, you can usually figure out what `e` encrypts to. The problem is that next group of letters have frequencies which are rather close together.

Here's a suggestion. If you know (or think you do) that `e` is encrypted to `r`, then you should use the digrams. Basically the idea is the same as solving crossword puzzles. You want to decrypt the letters near to those letters which are already decrypted, just like in crossword puzzles you want to work near to whose rows and columns that are already solved. So look at all the places where `r` are decrypted as `e`. Look at the letter before and after it and hope that you can now decrypt by looking at digrams in the ciphertext of the form `?r` and `r?`

[USE THIS AS EXAMPLE `a/a03-substitution-example/answers/main.pdf`]

Example 2.7.1. The goal is to decrypt the following ciphertext (i.e., you need to compute the plaintext) and also to discover the key used. The substitution cipher is used.

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewm
hmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehq
ermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtx
lajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatlj
mwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqq
evjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlh
jggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamt
liammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqt
xltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmloji
akexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
```

```
mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvat
kkmxnmh
```

Solution. The top few 1–gram frequencies of the ciphertext are

```
1gram: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30 ...
```

The gap between the frequency of `m` and `t` is extremely large. Therefore we suspect that part of the encryption is `e->m`. The rest, at least up to `x` are most probably from `t, a, o, i, n, s, h, r`:

$$\{\texttt{t, a, o, i, n, s, h, r}\} \rightarrow \{\texttt{t, q, h, j, e, i, a, x}\}$$

We can try different possible assignments on the above 8 letters to 8 letters, but that's $8! = 40320$ which is too big. At this point we have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
-e--------------------e--------------e----------e-e--------------------------e--e-e-----e--------

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
--------e-e------e-----e---e-e----------e---e-------e-----------e-----------e-----e------e---------

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
-----------------e-----------e---e-------------e------------------e-e--e----e-----e---------e----e----

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
--ee-e--ee------------e---------e-----------------e-e--e----e----------------e-----e---e--ee----e---

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
---e--e-----------e---e-----e---e----e-------------------e--------------------e-----e---------e-----e--

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
------------------e--------------e----e----e-------------------e------e----e--------e-----------e-e-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
-------------------------------e----------e---e-e----ee-e--e-e---e-------e---e--------------e----e---

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
e--------------e------e--------e---e----------e----------e--------e-----e--e-e--e---e-------------------e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
--e-----e--e----e---e------e----------------e-e-------------------------e------------------------------

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e---e-------------e---------e-------------e---------e--e-

ciphertext
2-grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
         tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3-grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
         feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
plaintext
1-grams: e t a o i n s h r
2-grams: th he in er an re ed on es st en at to nt ha nd
         ou ea ng as or ti is et it ar te se hi of
3-grams: the ing and her ere ent tha nth was eth for dth
```

We now look at 2-grams and 3-grams.

The common 2-grams are `th, he, in, er, an, re, ed, on, es, st`. Since we are assuming `e->m`, `mh` is either from `er` or `ed` or `es`. Note that `er` and `re` are common 2-grams. We also note that `mh` and `hm` are high frequency 2-grams in the ciphertext. Note further that `ere` is a common 3-gram and `mhm` is also a common 3-gram in the ciphertext. We suspect that `h->r`. Therefore we now have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
-e------r-----------r-e------r-------e-----------re-----------------r------e--ere-----e------r-

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
--------e-e------e-r---e---e-e---------re--re---r---e-------r---e--------r----er---er-----er--------

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
-----------------e-----------e--re-------------e-------------------re-e--e----e-----e--r------e--r-e----

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
--ee-er-ee------------e--------re-----------------e-er-e---re----------------e-----e---er-ee---re---

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
--re--e-----------er--e-----e---e----e-r---------------e----------r---------e--r-er---r----e-----er--

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
------------------e--------------e----e----e----r-------------er-----e----e---r----e-----r-----ere-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
-------r------------r----------er---------e--re-e----ee-e--ere---e-r-----er-re-------------er---e-r-

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--r----------er-----er-------e--re----------er-------e-----e---rre--ere--e---er--r---------------e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
-re-----e--e----e---e--r---e--r------------ere-----------------r---------e-------r--r---r----------

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e---e------r-----re--------e---r-----r--e--r------e--er

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h
```

Now we look for `the`. We have `the -> ??m`. The most commonly occurring ciphertext of this form is `iam`, `ewm`, `mhm`, `twm`. `mhm` is from `ere` which we already know so this is useless. So we are left with `iam`, `ewm`, `twm`. The frequency between `iam` and `ewm` is a huge 30% drop. So hopefully `the -> iam`. This means `t->i` and `h->a`. We have to take note of this since here we are creating two substitutions and the confidence is not as high. If we end up in a deadend, we will have to backtrack to this point. With the above two new substitutions, we have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
-e--t---r-t-t-------r-e------r---t---e----------here---t---------------------r------e-tere-----e-t----r-

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
---t----e-e------e-r---e---e-e---------re--re---r---et-----tr---e--t--t--r---her---er-----er------t-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
--t----t---------e-----t-----e--re-t-----------e-------------------re-e--e----e-----e--r----t-e--r-e-t--

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
-heeher-ee------------e--------ret--t---t----t----eherhe-t-re-t-----t-------he-----e--her-ee-t-re---

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
--re-te-----------er--e---t-e---e----e-r---------------te--t-------r-------t-e--r-er---r--t-e--tter--

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
------------tt---he--------------e----e----e----r-------------her-----e-tt-e---r----e-----r----tere-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
-------r--t---------rh--------her--------he--re-e--t-ee-e--ere---e-r-----er-ret------t--t-her---e-rh
```

```
mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--r-t-------her----ter-------e--re----------er-------e-t---e---rre--eret-e-t-er--rt-------------e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
-re----het-et---e--he--rt--e--r-----------here---t------------rt--------e-------r--r---r-------h---

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e---e------r-t---re-------t-e--t-r-----r--e---r-----e--er
ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a
```

`ent` is also a common plaintext trigram. This is encrypted as `m?i`. The only one that fits is `mqi` but the frequency of this is only 4 – so this is probably wrong.

Another high frequency plaintext 3-gram is `tha`. This would encrypt as `ia?`. We notice that `iam` has a high frequency. So perhaps `a->m`.

Now let's look at pairs of digrams.

`es,st` is a high frequency digram. This is encrypted as `m?,?i`. The only possibility is `mq,qi`. So we suspect `s->q`. This is what we have now:

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hes-t---r-t-t----h--r-eh-----r---t---e-h--h--s---ere---th----s---s-------r--s-s-e-tere-h-sse-ts---r-

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
---t----e-eh-s---e-r---e---e-e--s----s-re--re---rs--eth---str---e--t--t--rs---er---er-----ers-----t-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
-sth-s-t--sh-ss--e-h--ht-----es-re-th----h-ss--e---h-s----s----rese--eh-she----se--r---sthe--r-est-h

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
--ee-ersee------h-----e------s-reth-th-st--h-th---e-er-e-t-re-t-----t-----h--es----e---er-ee-t-re---

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
--reste-----sh----ersheh--the---e----e-r--h--h--ss----te--th---ss-r-------the--r-er---r--the--tters-

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
------s----stt-h--e-s-s-----------es---esh--e----r-------------erh-s-hesttheh--r--s-e-----r---stere-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
-------r--t---------r------s---erh-s--ss--e--rehe--thee-es-ere---e-r-----er-re-t-----t--ts-er---e-r-

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--r-t--------er---ster----h--es-re----sh----ers-----hest---e---rre--erethe-ther--rts--h---h--h---e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
-re-----ethet---es--e--rt--e--r---sh---s---ere---th---------h--rth-s-----e-------r--r---r-----------

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e---e------rst---ress-----the--t-r---s-r--ess-r-h---e--er

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
        plaintext
1grams: e t a o i n s h r
```

```
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a, s->q
```

Note that `ti,is` is a common plaintext 2-gram. When encrypted, this is `i?,?q`. Unfortunately we can't find this pattern.

We now have enough substitutions to consider multiple cases of pairs of digrams.

Consider the common plaintext digram `aj,jq`. With what we have at this point, the encryption is `h?,?s -> aj,jq`. The possibilities for `h?,?s` are

- `he,es`: Therefore `e->j`, but `e` is already encrypted as `m`.
- `ha,as`: Therefore `a->j`.
- `hi,is`: Therefore `i->j`.

So we have `a->j` or `i->j`. Before we make a choice, let's consider more digrams.

Consider `h?,e?`. `h?,e?` might be encrypted as `at,mt`. Possibilities for `h?,e?`

- `hi,ei`: But `ei` is not common.
- `ha,ea`: Therefore `a->t`.

Therefore `a->t`.

Consider `h?,?r`. `h?,?r` might be encrypted to `at,th`. The only possibilities for `h?,?r` are

- `ha,ar`: Therefore `a->t`.
- `hi,ir`: But `ir` is not common.

Therefore `a->t`.

Consider `?s,e?`. `?s,e?` might be encrypted as `tq,mt`. The only possibility for `?s,e?` is `as,ea` which implies `a->t`.

`?s,e?` might be encrypted as `tq,th`. The only possibility for `?s,e?` is `as,ea` which implies `a->t`.

`e?,?r` might be encrypted to `mt,th`. The possibilities for `e?,?r` are

- `ed,rd`: But `rd` is not common.
- `es,sr`: But `sr` and `os` not common.
- `en,nr`: But `nr` is not common.
- `ea,ar`: Therefore `a->t`.
- `et,tr`: But `tr` is not common.

All in all, this case implies `a->t`.

From all the above cases, it seems that `a->t` and `i->j`. (The argument for `a->t` is stronger.) We now have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesati-ar-tati---hair-ehi--a-r-a-ta--e-hi-h-as---ere--ith----s-a-sa---ia-ra-sasie-tere-hisseats---r-

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
---t--a-e-ehisa--eara--e-a-e-e-as-i-as-re-are---rs--ethi--stra--e--t--t--rs---er---eri--a-ers--a-it-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisit-ashissi-e-hi-ht-----es-reatha-a-hissi-ea--hisi---si---rese--ehishea--ase--r---sthe-ar-estih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
a-ee-ersee-----ah--a--ei--ia-s-rethathist--hatha-ie-er-e-t-re-t----it-----ha-es-i--e---er-ee-tire--a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
--reste-----sh----ersheha-the-a-ea---ear--hi-hiass--iate-itha-ass-ria-----the--r-er---ri-the-atters-

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
--a--asa---stt-ha-eas-s-i-i-------es-a-esha-e-a--ri---i--------erhis-hestthehair-as-e---iar--astere-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
----i--r--ti-a------r-i---is---erhis-assi-e--rehea-thee-es-ere---e-ra----er-reat--a--t--ts-er---ear-

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--riti-a-a---er--aster---ah--es-rea---sh----ersa--a-hest-i-ea-arre--erethe-ther-arts--hi--hi-ha--e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
are-a---etheta--esa-e--rt--e--r---sha--s---ere--ith------a--hairthisa--a-e----i--r-ari--r----i----i-

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e-a-e-----irsti--ressi----the--t-ri--s-r--ess-r-ha--e--er

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
        plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a, s->q, a->t, i->j
```

At this point we can already see (possibly) "`he sat`" at line one and "`that his`" at line 4 and "`his -hest the hair`" at line 4 – perhaps "`chest`" is the second word?

Next, we try `tx,jx`. `a?,i?` might be encrypted as `tx,jx`. The possibilities for `a?,i?` are

- `an,in`: Therefore `n->x`.
- `ar,ir`: But `ir` is not common.

We get

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinar-tatin--hair-ehin-a-r-a-ta--e-hi-h-as---ere--ith----s-a-san--ia-ra-sasientere-hisseats--nr-

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
-n-t--a-e-ehisa--earan-e-a-e-e-as-i-as-re-are---rs--ethin-stran-e--tn-t--rs---er---erin-a-ers-na-it-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisit-ashissi-e-hi-ht----nes-reatha-a-hissi-ean-hisi---sin--resen-ehishea--asen-r---sthe-ar-estih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
a-ee-erseen---nah--an-ein-ia-s-rethathist--hatha-ie-er-ent-re-t---nit-----ha-es-i--e---er-eentire--a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
n-reste--n--sh----ersheha-the-a-ean--ear--hi-hiass--iate-ithanass-rian----the--r-er---ri-the-atters-
```

```
uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
--a--asa---stt-ha-eas-s-i-i-n-----es-a-esha-e-an-ri---in----n--erhis-hestthehair-as-e---iar--astere-

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
---nin-r-ntina--n---r-in--is---erhis-assi-e--rehea-thee-es-ere---e-ra--n-er-reat--a--t--ts-er---ear-

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--riti-a-an--er--aster---ah--es-rea---sh----ersan-a-hest-i-ea-arre--erethe-ther-arts--hi--hi-ha--e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
are-a---etheta--esa-e--rt--en-r---shan-s---ere--ith--n---a--hairthisan-a-e----in-r-arin-r----in---i-

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e-a-e-----irsti--ressi-n--then-t-ri--s-r--ess-r-ha--en-er

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
        plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x
```

The beginning of the plaintext now reads "`he sat in a`." We now look at `xl`,`ml` and `ex`,`eh`.

`n?`,`e?` might be encrypted as `xl`,`ml`. The possibilities for `n?`,`e?` are

- `nt`,`et`: But `t` is already assigned.
- `nd`,`ed`: Therefore implies `d->l`.
- `ng`,`eg`: But `eg` is not common.

`?n`,`?r` might be encrypted as `ex`,`eh`. The possibilities for `?n`,`?r` are

- `in`,`ir`: But `i` is already assigned.
- `an`,`ar`: But `a` is already assigned.
- `on`,`or`: This implies `o->e`.
- `en`,`er`: But `e` is already assigned.

Adding `d->l` and `o->e`, we get

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinarotatin--hair-ehinda-roadta--e-hi-h-as-o-ered-ith-oo-s-a-sanddia-ra-sasienteredhisseats--nro

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
-ndto-a-e-ehisa--earan-e-ade-e-as-i-as-re-ared-orso-ethin-stran-e--tnot-orsoo-er-o-erin-a-ersona-it-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisit-ashissi-e-hi-htoo-ones-reatha-a-hissi-eandhisi--osin--resen-ehishead-asenor-o-sthe-ar-estih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
a-ee-erseen--onah--an-ein-ia-s-rethathisto-hathadie-er-ent-redtodonit-o--dha-es-i--edo-er-eentire--a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
ndrestedon--sho--dershehadthe-a-eand-eard-hi-hiasso-iate-ithanass-rian----the-or-er--oridthe-atterso

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
--a--asa--osttoha-eas-s-i-iono----es-adesha-edandri---in-do-no-erhis-hestthehair-as-e---iar--astered

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
do-nin-rontina-on---r-in--is-o-erhis-assi-e-oreheadthee-es-ere---e-ra--nder-reat--a--t--ts-er---ear-
```

```
mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er--riti-a-and-er--aster---ah--es-reado-sho--dersanda-hest-i-ea-arre--eretheother-artso-hi--hi-ha--e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
areda-o-etheta--esa-e-ort-oenor-o-shands-o-ered-ith-on---a--hairthisanda-e--o-in-roarin-r----in--oi-

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e-ade-----irsti--ressiono-thenotorio-s-ro-essor-ha--en-er

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
        plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x, d->l, o->e
```

The beginning reads "`he sat in a rotatin--hair-ehinda-road...`" which is very likely "`he sat in a rotating chair-ehinda-road...`", giving us `g->n` and `c->v`. This gives us

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinarotatingchair-ehinda-roadta--e-hich-asco-ered-ith-oo-s-a-sanddiagra-sasienteredhisseats--nro

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
-ndto-ace-ehisa--earance-ade-egas-i-as-re-ared-orso-ethingstrange--tnot-orsoo-er-o-eringa-ersona-it-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisit-ashissi-e-hichtoo-ones-reatha-a-hissi-eandhisi--osing-resencehishead-asenor-o-sthe-argestih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
a-ee-erseen--onah--an-eingia-s-rethathisto-hathadie-er-ent-redtodonit-o--dha-es-i--edo-er-eentire--a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
ndrestedon--sho--dershehadthe-aceand-eard-hichiassociate-ithanass-rian----the-or-er--oridthe-atterso

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
--ac-asa--osttoha-eas-s-iciono----es-adesha-edandri---ingdo-no-erhischestthehair-as-ec--iar--astered

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
do-nin-rontina-ongc-r-ing-is-o-erhis-assi-e-oreheadthee-es-ere---egra--ndergreat--ac-t--ts-er-c-ear-

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er-critica-and-er--aster---ah-ges-reado-sho--dersandachest-i-ea-arre--eretheother-artso-hi--hicha--e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
areda-o-etheta--esa-e-ort-oenor-o-shandsco-ered-ith-ong--ac-hairthisanda-e--o-ingroaringr----ing-oic

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
e-ade-----irsti--ressiono-thenotorio-s-ro-essorcha--enger

ciphertext
1grams: m:116 t:95 q:75 h:74 j:68 e:65 i:59 a:59 x:53 l:40 k:34 g:30
2grams: mh:25 ia:20 hm:20 aj:19 wm:18 at:16 am:16 tq:15 mq:15 xn:14
        tx:14 mt:14 jx:14 xl:12 qi:12 jq:12 th:11 ml:10 ex:10 eh:10
3grams: wmh:11 ajq:11 jxn:10 iam:10 txl:9 hml:7 ewm:7 mhm:6 otq:5 mhq:5
        feh: 5 twm: 4 qqj: 4 oaj: 4 mth:4 mqi:4 mia:4 jva:4 imh:4 iat:4
        plaintext
1grams: e t a o i n s h r
2grams: th he in er an re ed on es st en at to nt ha nd
        ou ea ng as or ti is et it ar te se hi of
3grams: the ing and her ere ent tha nth was eth for dth

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x, d->l, o->e, g->n, c->v
```

Near the middle of the first line, "`-hich-asco-ered-ith`" is probably "`which-asco-eredwith`" giving us `w->o`.

On the second line "`orso-ethingstrange-`" is probably "`or something strange-`", giving us `m->r`.

On the third line "`sthe-argestiha-ee-erseen`" is probably "`s the largest i have ever seen`." This gives us `l->k` and `v->w`.

At this point we have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinarotatingchair-ehinda-roadta-lewhichwascoveredwith-oo-sma-sanddiagramsasienteredhisseats--nro

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
-ndto-acemehisa--earancemademegas-iwas-re-ared-orsomethingstrange--tnot-orsoover-oweringa-ersonalit-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisitwashissi-ewhichtoo-ones-reathawa-hissi-eandhisim-osing-resencehisheadwasenormo-sthelargestih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
aveeverseen--onah-man-eingiams-rethathisto-hathadievervent-redtodonitwo-ldhavesli--edovermeentirel-a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
ndrestedonm-sho-ldershehadthe-aceand-eardwhichiassociatewithanass-rian--llthe-ormer-loridthelatterso

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
-lac-asalmosttohaveas-s-iciono--l-es-adesha-edandri--lingdownoverhischestthehairwas-ec-liar-lastered

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
downin-rontinalongc-rvingwis-overhismassive-oreheadthee-eswere-l-egra--ndergreat-lac-t--tsver-clearv

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er-criticalandver-master--lah-ges-reado-sho-ldersandachestli-ea-arrelweretheother-artso-himwhicha--e

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
areda-ovetheta-lesave-ortwoenormo-shandscoveredwithlong-lac-hairthisanda-ellowingroaringr-m-lingvoic

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
emade--m--irstim-ressiono-thenotorio-s-ro-essorchallenger

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x, d->l, o->e, g->n, c->v, w->o, m->r,
l->k, v->w
```

At the third line "`hisim-osing-resencehisheadwasenormo-sthelargestihaveeverseen`" is probably "`his imposing presence his head was enormous the largest i have ever seen`" giving us `p->g` and `u->b`.

At line 7, "`hismassive-orehead`" is "`his massive forehead`" giving us `f->f`.

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinarotatingchair-ehinda-roadta-lewhichwascoveredwith-oo-smapsanddiagramsasienteredhisseatspunro

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
undtofacemehisappearancemademegaspiwaspreparedforsomethingstrange-utnotforsooverpoweringapersonalit-

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisitwashissi-ewhichtoo-ones-reathawa-hissi-eandhisimposingpresencehisheadwasenormousthelargestih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
aveeverseenuponahuman-eingiamsurethathistophathadieverventuredtodonitwouldhaveslippedovermeentirel-a

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
ndrestedonm-shouldershehadthefaceand-eardwhichiassociatewithanass-rian-ulltheformerfloridthelatterso

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
-lac-asalmosttohaveasuspicionof-luespadeshapedandripplingdownoverhischestthehairwaspeculiarplastered

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
downinfrontinalongcurvingwispoverhismassiveforeheadthee-eswere-luegra-undergreat-lac-tuftsver-clearv

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
er-criticalandver-masterfulahugespreadofshouldersandachestli-ea-arrelweretheotherpartsofhimwhichappe

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
```

```
areda-ovetheta-lesavefortwoenormoushandscoveredwithlong-lac-hairthisanda-ellowingroaringrum-lingvoic

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
emadeupm-firstimpressionofthenotoriousprofessorchallenger

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x, d->l, o->e, g->n, c->v, w->o, m->r,
l->k, v->w, p->g, u->b, f->f
```

The beginning "`hesatinarotatingchair-ehinda-roadta-le`" is "`he sat in a rotating chair behind abroad table`" giving us `b->u`.

At line 5, "`restedonm-shoulder`" is "`rested on my shoulder`" giving us `y->s`.

At line 8, "`shouldersandachestli-ea-arrel`" is "`shoulders and a chest like a barrel`, giving us `k->c`.

At line 3, "`itwashissi-ewhichtookonesbreathaway`" is "`it was his size which took ones breath away`" giving us `z->p`.

We now have

```
amqtijxtheitijxnvatjhumajxltuhetlitukmoajvaotqvewmhmlojiaueecqrtgqtxlljtnhtrqtqjmximhmlajqqmtiqgbxhe
hesatinarotatingchairbehindabroadtablewhichwascoveredwithbooksmapsanddiagramsasienteredhisseatspunro

bxlieftvmrmajqtggmthtxvmrtlmrmntqgjotqghmgthmlfehqermiajxnqihtxnmubixeifehqeewmhgeomhjxntgmhqextkjis
undtofacemehisappearancemademegaspiwaspreparedforsomethingstrangebutnotforsooverpoweringapersonality

tqiajqjiotqajqqjpmoajvaieecexmquhmtiatotsajqqjpmtxlajqjrgeqjxnghmqmxvmajqamtlotqmxehrebqiamkthnmqija
asthisitwashissizewhichtookonesbreathawayhissizeandhisimposingpresencehisheadwasenormousthelargestih

twmmwmhqmmxbgextabrtxumjxnjtrqbhmiatiajqiegatiatljmwmhwmxibhmlielexjioebklatwmqkjggmlewmhrmmxijhmkst
aveeverseenuponahumanbeingiamsurethathistophathadieverventuredtodonitwouldhaveslippedovermeentirelya

xlhmqimlexrsqaebklmhqamatliamftvmtxlumthloajvajtqqevjtimojiatxtqqshjtxubkkiamfehrmhfkehjliamktiimhqe
ndrestedonmyshouldershehadthefaceandbeardwhichiassociatewithanassyrianbulltheformerfloridthelatterso

uktvctqtkreqiieatwmtqbqgjvjexefukbmqgtlmqatgmltxlhjggkjxnleoxewmhajqvamqiiamatjhotqgmvbkjthgktqimhml
blackasalmosttohaveasuspicionofbluespadeshapedandripplingdownoverhischestthehairwaspeculiarplastered

leoxjxfhexijxtkexnvbhwjxnojqgewmhajqrtqqjwmfehmamtliammsmqomhmukbmnhtsbxlmhnhmtiuktvcibfiqwmhsvkmthw
downinfrontinalongcurvingwispoverhismassiveforeheadtheeyeswerebluegrayundergreatblacktuftsveryclearv

mhsvhjijvtktxlwmhsrtqimhfbktabnmqghmtlefqaebklmhqtxltvamqikjcmtuthhmkomhmiameiamhgthiqefajroajvatggm
erycriticalandverymasterfulahugespreadofshouldersandachestlikeabarrelweretheotherpartsofhimwhichappe

thmltuewmiamitukmqtwmfehioemxehrebqatxlqvewmhmlojiakexnuktvcatjhiajqtxltumkkeojxnhethjxnhbrukjxnwejv
aredabovethetablesavefortwoenormoushandscoveredwithlongblackhairthisandabellowingroaringrumblingvoic

mrtlmbgrsfjhqijrghmqqjexefiamxeiehjebqghefmqqehvatkkmxnmh
emadeupmyfirstimpressionofthenotoriousprofessorchallenger

e->m, r->h, t->i, h->a, s->q, a->t, i->j, n->x, d->l, o->e, g->n, c->v, w->o, m->r,
l->k, v->w, p->g, u->b, f->f, b->b, y->s, k->c, z->p
q->q, x->x, j->j
```

Note that `q,x,j` were not used in the plaintext. We have added `q->q, x->x, j->j` to the substitution key. The following is the plaintext with spaces inserted (puncuations not restored):

> he sat in a rotating chair behind a broad table which was covered with books maps and diagrams as i entered his seat spun round to face me his appearance made me gasp i was prepared for something strange but not

for so overpowering a personality as this it was his size which took ones breath away his size and his imposing presence his head was enormous the largest i have ever seen upon a human being i am sure that his tophat had i ever ventured to don it would have slipped over me entirely and rested on my shoulders he had the face and beard which i associate with an assyrian bull the former florid the latter so black as almost to have a suspicion of blue spade shaped and rippling down over his chest the hair was peculiar plastered down in front in a long curving wisp over his massive forehead the eyes were blue gray under great black tufts very clear very critical and very masterful a huge spread of shoulders and a chest like a barrel were the other parts of him which appeared above the table save for two enormous hands covered with long black hair this and a bellowing roaring rumbling voice made up my first impression of the notorious professor challenger

□

The following programs are helpful (so go ahead and write them):

(a) Code to print the top 1-grams, 2-grams, 3-grams. The 1-grams will help determine the character that `e` is encrypted to. The trigrams might help determine what `the` is encrypted to.
(b) Given a character `c`, code that computes character(s) `d` such that `cd` and `dc` occurs most frequently. So if you suspect `e` is encrypted to `r`, your code will print all `x` such that `rx` and `xr` is common will be helpful in decoding `x`.
(c) Given a collection of common 2-grams (in plaintext), a partially specified substitution, compute pairs of commonly ocurring digrams of the form `xy,xz` or `xy,zx` or `xy,yz` or `xy,zy` (i.e., there are three distinct characters in the pairs of digrams) where two of the characters have already been decrypted and the remaining one has not and has not been assigned to a plaintext character.
(d) Instead of the above where there are two digrams, listing 4-grams where the decryption of 3 are known and one is unknown is also useful.

Here are some exercises for you:

Exercise 2.7.1. Find the key and the plaintext of the following ciphertext encrypted using the substitution cipher:

debug: exercises/substitution-cipher-0/question.tex

```
psxdltuxtcwauuxvifgtzwacsppstpcvxqtdgcvxqtdgrvxbxd
pwzxdgxgadtdipsxvtvlspstpwiijxgihpinxvtblxdxbiwajx
```

```
tralphvxihpifteiijteihpaptwzpstpxnxvzidxbevxtpsqtb
ptjxdtqtztdgpsxzbaurwzqtwjxgfivqtvgbawxdptdgbptvad
ctbsivptnxdhxiflzrvxbbxbwxgqagxdadctbapqxdppitutve
wxpxvvtlxpstpwtzevitgtdgqsapxadpsxbhdwacsppsxlsawg
vxdewadjadcwxtdxgpsxavtvubidpsxevitgfwtpetwhbpvtgx
tdgctmxgauuxgatpxwzexwiqpsxuqtbtwtjxyhbpwajxtwtjxa
dpsxexthpaxbifaptwztwtjxqapsbqtdbtdgtdabwtdgtdgqxx
radcqawwiqbexzidgapqxvxcvxxdbwirxbgippxgqapscvinxb
ifpvxxbtdgtuagpsxpvxxbcwxtuxgpsxqsapxwauebifbptphx
btctadbptwappwxsawwpipsxwxfpqtbtvihdgqsapxehawgadc
qapsrawwtvbtdgpipsxvacsptqtpxvftwwltuxphuewadcgiqd
tuidcuibbzbpidxbpibrwtbsadpipsxwtjxbpxrbfxgfviupsx
pxvvtlxpipsxqtpxvtdgipsxvbpxrbpipsxcvxxdwtqdbexbag
xaptqtztlvibbpsxcvtbbzbwirxbgxxvqxvxfxxgadctdgadps
xgabptdlxqsxvxpsxcvinxbifpvxxbpsaljxdxgadpiqstpwii
jxgtwuibptfivxbpqxvxxdivuihbbstrxbifcvxzbpidxwajxd
ipsadcpstppsxlsawgvxdstgxnxvbxxdexfivx
```

(Go to solution, page 135) □

Exercise 2.7.2. Here's another one that is harder:

debug: exercises/substitution-cipher-1/question.tex

```
oznftyomrrtqlnlzftqlqlmxemftlnyoozjtzyvqlzfzvgmnzm
drsjmrrlxtqlxyjtynmvxqlhlmnfjmkmvxpyhvlglvhqlvdznj
qzvpemrlomjtynfhqzjqqlxylfzvtlnezttlvtrstqnyipqyit
tqlxmsynmttlvxzvpmjnzjaltemtjqoynmrrhlavyhqlhlmnft
qlezvdlx
```

(Go to solution, page 136) □

Exercise 2.7.3. And another:

debug: exercises/substitution-cipher-2/question.tex

```
xsdwoddnuskapbgxaayrpcuevbsxjjuskyaqddpdbxbvrolrhb
qsyqmsdnvxumxsnxsxgxfuskbdahjrbbdbbursrhgxssdmnxme
dnxmrqsnevdedsexsnjxqbdnusevdumgunbebrgdcvxevdxedn
cuevvubvxbed
```

(Go to solution, page 137) □

Once you have solved a few substitution ciphers by hand, you are ready to write a program to automate your process. Instead of a perfect solver, your program should aim to print a list of possible decryptions, ordered by likelihood (of being the original plaintext).

You can learn more about breaking substitution ciphers by doing a google search. Clearly the process of breaking a substitution cipher involves trying substitutions. This leads to search algorithms. Starting with the traditional backtracking search, you will be led to other heuristic search algorithms such as local seach algorithms, genetic algorithms, etc. Studying substitution ciphers will also lead to studies of probability theory and specifically markov chains. A search on google will reveal many research papers (many of which are very recent) on substitution cipher, AI search algorithms, markov chains, etc.

Solutions

Solution to Exercise 2.7.1.

debug: exercises/substitution-cipher-0/answer.tex

ANSWER: Then came a glimmer of daylight that grew and grew, and presently ended in another arch that looked out over a scene so like a picture out of a book about Italy that everyone's breath was taken away, and they simply walked forward silent and staring. A short avenue of cypresses led, widening as it went, to a marble terrace that lay broad and white in the sunlight. The children, blinking, leaned their arms on the broad, flat balustrade and gazed. Immediately below them was a lake just like a lake in "The Beauties of Italy" a lake with swans and an island and weeping willows; beyond it were green slopes dotted with groves of trees, and amid the trees gleamed the white limbs of statues. Against a little hill to the left was a round white building with pillars, and to the right a waterfall came tumbling down among mossy stones to splash into the lake. Steps fed from the terrace to the water, and other steps to the green lawns beside it. Away across the grassy slopes deer were feeding, and in the distance where the groves of trees thickened into what looked almost a forest were enormous shapes of grey stone, like nothing that the children had ever seen before.

Solution to Exercise 2.7.2.

debug: exercises/substitution-cipher-1/answer.tex

ANSWER: First of all there is the Headmaster of Fiction. He is invariably called "The Doctor," and he wears cap and gown even when birching malefactors–which he does intermittently throughout the day–or attending a cricket match. For all we know he wears them in bed.

Solution to Exercise 2.7.3.

debug: exercises/substitution-cipher-2/answer.tex

ANSWER: An exceedingly small boy with snapping blue eyes, a shock of sunburned hair and an amazing self-possession of manner, darted around the tent and paused in their midst, somewhat heated with his haste. "'

2.8 Permutation cipher debug: classical-cipher-permutation-cipher.tex

In the above ciphers, a character is replaced by another. The **permutation cipher** is different: each character is *moved* to a different *location* in the plaintext to form the ciphertext. permutation cipher

Here's an example. Look at this:

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

It's just the function where a value in the top row maps to the corresponding value at the bottom row:

$$\begin{aligned} \pi(1) &= 3 \\ \pi(2) &= 1 \\ \pi(3) &= 2 \\ \vdots \ & \ \ \vdots \\ \pi(6) &= 6 \end{aligned}$$

Let π is called a permutation of $\{1, 2, 3, 4, 5, 6\}$. A **permutation** is simply a bijection (1–1 and onto) function from $\{1, 2, 3, ..., n\}$ to $\{1, 2, 3, ..., n\}$. As a shorthand, the above permutation permutation

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

is also written as

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

This is called the **cycle notation** of π. In the above example, we say that the permutation π has **length** 6. cycle notation length

Exercise 2.8.1. What is the cycle notation of debug: exercises/permutation-0/question.tex

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 1 & 7 & 2 & 5 & 8 & 3 & 6 \end{pmatrix}$$

(Go to solution, page 148) □

How does a permutation give you an encryption? For the permutation $\pi = (1 \ \ 3 \ \ 2)(4 \ \ 5)(6)$, you do this:

1. character at position 1 goes to position 3,
2. character at position 2 goes to position 1,
3. etc.

In the example above, the n is 6. For instance for the string `marvin` and the permutation $\pi = (1 \ \ 3 \ \ 2)(4 \ \ 5)(6)$, the since in the permutation $1 \mapsto 3$, the character at position 1 goes to position 3:

```
1 2 3 4 5 6
m a r v i n
    m
```

since $3 \mapsto 2$, the character at position 3 goes to position 2:

```
1 2 3 4 5 6
m a r v i n
  r m
```

Etc. Altogether I get

$$E(\pi, \texttt{marvin}) = \texttt{armivn}$$

This assumes your string has a length which is a multiple of 6. So you might need to add some dummy data at the end of your string.

Note that in the above example, you have to encrypt substrings of length 5. Therefore if the original plaintext does not have a length that is a multiple of 5, you would need to pad it with some characters until the length is a multiple of 5.

Exercise 2.8.2. How would you choose to pad the plaintext? You might want to finish this section before you answer this question. (I.e., you want to know what Eve will do and design your padding to make it difficult for her.) (Go to solution, page 149) □

debug: exercises/permutation-1/question.tex

The decryption is the same algorithm as the encryption except that the permutation is read in the "opposite direction". If

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

then the inverse of π, denoted by π^{-1} is the "opposite" of the above:

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

(do you see the row switch?) and to make things look nice, you arrange the top row:

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

Using the cycle notation,

$$\pi^{-1} = (1\ \ 2\ \ 3)(4\ \ 5)$$

So

$$D(\pi, y)$$

is the same as

$$E(\pi^{-1}, y)$$

So

$$\begin{aligned} D((1\ \ 3\ \ 2)(4\ \ 5), \texttt{armivn}) &= E((1\ \ 3\ \ 2)(4\ \ 5)^{-1}, \texttt{armivn}) \\ &= E((1\ \ 2\ \ 3)(4\ \ 5), \texttt{armivn}) \\ &= \texttt{marvin} \end{aligned}$$

Exercise 2.8.3. debug: exercises/permutation-2/question.tex

1. Encryption `"Where there's life there's hope ... and need of vittles."` using the permutation cipher with key $\pi = (1\ \ 5)(2\ \ 6\ \ 3\ \ 4)$. Pad with `z` if necessary.
2. What is the inverse of π?
3. Decrypt your ciphertext from the first part.

(Go to solution, page 150) □

Exercise 2.8.4. What is the size of the key space of the permutation cipher if you know that the permutation is a permutation on 1, 2, 3, 4, 5? What if it's a permutation on 1, 2, 3, 4, 5, 6, 7, 8, 9, 10? (Go to solution, page 151) □ debug: exercises/permutation-3/question.tex

Exercise 2.8.5. Why is the permutation cipher a special case of the Hill cipher? To try it out, write down an example of Hill cipher and re-interpret debug: exercises/permutation-4/question.tex it as a permutation cipher. You'll see that the matrix has a special form – it has only 0s and 1s arranged in a certain way. Such a matrix is called a **permutation matrix**. (Go to solution, page 152) □ permutation matrix

Breaking the permutation cipher

Frequency analysis obviously won't help: each letter is not replaced by another. It's the position of a letter that is changed.

Now suppose the permutation is

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

Suppose you know the length of the permutation is 5. After breaking your ciphertext into substrings of length 5, suppose one of these substrings is

```
hatoe
```

We know that `th` is a commonly occurring digram. Then it's natural to suspect that this comes from the encryption of 5 letters containing `th`, either

```
plaintext : ..... th... ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... .th.. ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... ..th. ...
ciphertext: ..... hatoe ...
```

or

```
plaintext : ..... ...th ...
ciphertext: ..... hatoe ...
```

Whereas for a substring of length 5, there are $5! = 120$ possible permutations, once two is fixed, there are $3! = 6$ possible permutations. This might cut down on the search for the key and help break the ciphertext. If `th` is not found or the above analysis involving `th` does not work, you go on to the next commonly occurring digram.

Of course you do not know if the length of the permutation is length 5. You therefore have to do a loop over all possible permutation lengths. Of course the key length cannot be 1 since that would be doing nothing! For key of size two, the permutation must be $(1,2)$. For a key of size three, there are $3! = 6$ permutations: There are 6 possible permutations: $(1)(2)(3)$, $(1)(2,3)$, $(2)(1,3)$, $(3)(1,2)$, $(1,2,3)$, $(1,3,2)$. Since there are not many permutations of length 3, for this case, you might want to simply try all permutations. In general, you try key lengths dividing the length of the ciphertext.

Note that it's also possible that the `th` in the plaintext is broken up with the `t` is one substring of length 5 and `h` in the following substring. For instance, here is a case:

```
plaintext : canth efish
ciphertext: acnht feihs
```

Of course it's also OK if you do not wish to consider cases where the digram is split across two substrings, since in the above case, you would have picked up the `he` when you consider a key of length (say) 10.

By looking at lots of digrams and trigrams, if the ciphertext is long enough, you should be able to break the code.

Example 2.8.1. Let me break the following permutation ciphertext

```
imahsnrwnesotanfusefvitieedoslwswma
```

SOLUTION.

The length is 35. Since the length of the permutation must divide 35, it must be 1, 5, 7, 35.

LENGTH 1. Of course if the length is 1, then the ciphertext is the same as the plaintext. But the plaintext is meaningless. Therefore the length cannot be 1.

LENGTH 5. Assume the key length is 5. I break up the string above into substrings of length 5 to make it easier to read:

```
imahs nrwne sotan fusef vitie edosl wswma
```

LENGTH 5. DIGRAM TH. I don't see any `th` in each substrings or consecutive substrings.

LENGTH 5. DIGRAM HE. I do see `he`:

```
imahs nrwne sotan fusef vitie edosl wswma
```

If you assume these two characters come from `he`, then it has to come from this:

```
plaintext:  ....h e.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which means the permutation contains $1 \mapsto 5, 5 \mapsto 4$. If that's the case, we have

```
plaintext:  s...h e...n n...a f...e e...i l...s a...m
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

Look at just the first group of 5 characters the `s...h` can only be `simah`, `siamh`, `saimh`, `samih`, `smiah`, or `smaih`. The only promising one is `siamh` (maybe "Siam has cats ...?"). Using this we get

```
plaintext:  siamh enwrn nstoa ffsue evtii leods awwsm
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless.

[force newpage]

LENGTH 5. DIGRAM `in`. The next digram to try is `in`:

```
plaintext:  ....i n.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

or

```
plaintext:  ....i n.... ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

The first is not possible. (Why?) The second gives us

```
plaintext:  h...i n...n a...s e...f i...v s...e m...w
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

Looking at just the first 5 characters, the only possible decryptions are `hmasi`, `hmsai`, `hamsi`, `hasmi`, `hsami`, `hsmai`. Only `hamsi`, `hasmi`, are promising. From `hamsi`,

```
plaintext:  hamsi nwren atons esuff itiev sodle mwsaw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless. From `hasmi`,

```
plaintext:  hasmi nwern atnos esfuf iteiv solde mwasw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```

which is meaningless.

LENGTH 5. DIGRAM `er`. The next digram to try is `er`:

- From
  ```
  plaintext:  ..... er... ..... ..... ..... ..... .....
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```
  we get
  ```
  plaintext:  sm... er... no... fu... ei... ld... as...
  ciphertext: imahs nrwne sotan fusef vitie edosl wswma
  ```
  The first 5 characters can only be `smahs`, `smash`, `smhas`, `smhsa`, `smsah`, `smsha`. Only `smash` seems to form a word. (`smhas` might be for instance `a chasm has separated us`, but the `smhas` is the *beginning* of the plaintext.) Using that, we get
  ```
  plaintext:  smash erwen notna fusfe eitei ldols aswam
  ```

```
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
which is not meaningful.

- From
```
plaintext:  ..... .er.. ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
we get
```
plaintext:  .sm.. .er.. .no.. .fu.. .ei.. .ld.. .as..
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
The first 5 characters can only be `ismah`, `ismha`, `asmih`, `asmhi`, `hsmia`, `hsmai`, none of which is meaningful, except possibly for `ismah` (example: `is mahi mahi a fish?`), `ismha` (example: `a schism happened overnight`), `asmih` (example: `protoplasm i have`), `asmhi` (example: `protoplasm have i`) but you can check that these do not lead to anything meaningful.

- From
```
plaintext:  ..... ..er. ..... ..... ..... ..... .....
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
we get
```
plaintext:  ..sm. ..er. ..no. ..fu. ..ei. ..ld. ..as.
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
The first 5 characters can only be `iasmh`, `ihsma`, `aismh`, `ahsmi`, `hisma`, `hasmi`. Only `ahsmi`, `hisma` and `hasmi` seem promising. For `ahsmi`, we get
```
plaintext:  ahsmi wnern tanos sefuf tieiv oslde wmasw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
which is meaningless. For `hisma`, we get
```
plaintext:  hisma nnerw asnot effus iveit seldo mwasw
ciphertext: imahs nrwne sotan fusef vitie edosl wswma
```
which finally gives us the plaintext

`hismannerwasnoteffusiveitseldomwasw`

This gives us

`His manner was not effusive. It seldom was.`

The last `w` is redundant (it's a padding).

The key (i.e., permutation) is $(1, 4, 2)(3, 5)$ with a key length of 5. □

Exercise 2.8.6. Of course the process is tedious and error prone. So you know what to do:

debug: exercises/permutation-5/question.tex

- Write a program to encrypt and decrypt messages for the permutation cipher. For instance you can try this permutation

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

  by entering `2,3,5,4,1`. Note that (mathematically speaking) permutations are written as bijection between sets $\{1, 2, 3, ..., m\}$. You might want to start with 0 instead. In that case, you should tell your user.
- From the above experience of computation-by-hand, we see that for breaking a permutation cipher, the program accepts a ciphertext and ask you for a permutation. Furthermore your program should allow you to enter a permutation partially. For instance to enter this permutation

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

  the user enters `?,3,?,?,1` in your program. You also want to allow the user to modify a permutation.
- You also want the program to list commonly occurring digrams that appears in the ciphertext, especially including those that appear within substrings of length m (the length of the permutation).
- With all the above, you can then automate the process to imitate what I did in the previous example.

(Go to solution, page 153) □

Exercise 2.8.7. Decrypt this permutation ciphertext:

debug: exercises/permutation-6/question.tex

`rhetibbalhotnweerstthigalontakeinuntolfemsorywae`

and write down the key. It's advisable to do some of these decryption by hand.

(Go to solution, page 154) □

Exercise 2.8.8. Decrypt this permutation ciphertext:

debug: exercises/permutation-7/question.tex

`uyashonlpotlsasdaskeathrsshcdp`

and write down the key. It's advisable to do some of these decryption by hand.
(Go to solution, page 155) □

Exercise 2.8.9. Decrypt this permutation cipher:

debug: exercises/permutation-8/question.tex

```
konusyowairduathsiatirttleistmkewihsihenatprdmpevn
oanigoliorwackatmhfianboemlrteseeudganubtodatofeao
hispaxtinyioenedpipstcheiaatlelairywdhlitsisdntehe
owytmmhaotorlteedmnhieswwanogutyyf
```

and write down the key. It's advisable to do some of these decryption by hand. (Go to solution, page 156) □

Exercise 2.8.10. Decrypt this permutation ciphertext:

debug: exercises/permutation-9/question.tex

```
wshiattesotbtfeistamiweseostwrhttmsoiefd
```

and write down the key. It's advisable to do some of these decryption by hand. (Go to solution, page 157) □

Exercise 2.8.11. You have been intercepting ciphertexts sent between Alice and Bob. You know that they use the permutation cipher. You notice this fact: All the 1000 ciphertexts have length 720. Why is that? Or is that just coincidence? (Go to solution, page 158) □

debug: exercises/permutation-10/question.tex

Solutions

Solution to Exercise 2.8.1.

debug: exercises/permutation-0/answer.tex

Solution not provided.

Solution to Exercise 2.8.2.

debug: exercises/permutation-1/answer.tex

Solution not provided.

Solution to Exercise 2.8.3.

debug: exercises/permutation-2/answer.tex

Solution not provided.

Solution to Exercise 2.8.4.

debug: exercises/permutation-3/answer.tex

Solution not provided.

Solution to Exercise 2.8.5.

debug: exercises/permutation-4/answer.tex

Solution not provided.

Solution to Exercise 2.8.6.

debug: exercises/permutation-5/answer.tex

Solution not provided.

Solution to Exercise 2.8.7.

debug: exercises/permutation-6/answer.tex

plaintext: therabbitholewentstraightonlikeatunnelforsomeway

the rabbit-hole went straight on like a tunnel for some way

(3,1,2,0)

Solution to Exercise 2.8.8.

debug: exercises/permutation-7/answer.tex

youshallnotpassdaserkthaspscdh

yousha llnotp assdas erktha spscdh

(1,5,0,3,4,2)

Solution to Exercise 2.8.9.

debug: exercises/permutation-8/answer.tex

youknowsaidarthuritsattimeslikethiswhenimtrappedinavogonairlockwithama
nfrombetelgeuseandabouttodieofasphyxiationindeepspacethatireallywishid
listenedtowhatmymothertoldmewheniwasyoung

(5,1,3,0,2,6,7,4)

Solution to Exercise 2.8.10.

debug: exercises/permutation-9/answer.tex

It was the best of times, it was the worst of times

itwasthebestoftimesitwastheworstoftimesd

itwasthe bestofti mesitwas theworst oftimesd

(3,6,0,4,1,5,2,7)

ehtitsaw itfbeots sawmetis tsrthows dseofmit

Solution to Exercise 2.8.11.

debug: exercises/permutation-10/answer.tex

Solution not provided.

2.9 Hill cipher debug: classical-cipher-hill-cipher.tex

Hill's cipher was announced in 1929. This cipher works with a block of n characters at a time. For both encryption and decryption, you treat each block of n characters (or mod 26 integers) as a column vector and we multiply it with an n–by–n matrix. Of course everything is done in mod 26. Here are more details. Hill's cipher

Recall how to multiple a n–by–n matrix with a column vector of size n. I'll do this for $n = 2$.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

(If you have not seen this before take the computer graphics class CISS380. You can't call yourself a serious computer scienctist if you can't do matrix multiplication. Most top CS programs actually has a linear algebra requirement.) That's matrix multiplication where all numbers are in $\mathbb{R}$. The (multiplicative) **inverse matrix** of inverse matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The expression $ad - bc$ is called the **determinant** of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and is written determinant

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

If a matrix has an inverse, we say that the matrix is **invertible**. invertible

Not only can you multiply a 2-by-2 matrix with a 2-by-1, you can actually also multiply a 2-by-2 matrix with a 2-by-2. Here's how you do it:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB & aC + bD \\ cA + dB & cC + dD \end{pmatrix}$$

Exercise 2.9.1. The following refers to matrices with $\mathbb{R}$ values. Do all these exercises: debug: exercises/hill-0/question.tex

1. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$
2. Compute
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}$$
3. If M, N are 2-by-2 matrices, is it always true that $MN = NM$?
4. Prove the following theorem: Let A, B, C be three 2-by-2 matrices. Prove that
$$(AB)C = A(BC)$$

□

(Go to solution, page 168) □

Define the following:
$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$
This is called the (2-by-2) **identity matrix**.

identity matrix

Exercise 2.9.2. The following refers to matrices with $\mathbb{R}$ values.

debug: exercises/hill-1/question.tex

1. Show that
$$I \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} I$$
2. Show that
$$I \begin{pmatrix} 2 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$
3. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}^{-1}$$
Then show that
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$

□

(Go to solution, page 169) □

Exercise 2.9.3. The following refers to matrices with $\mathbb{R}$ values. Let M be a

debug: exercises/hill-2/question.tex

2-by-2 matrix.

1. Is every matrix invertible?
2. Prove that if M is a 2-by-2 matrix,
$$MM^{-1} = I = M^{-1}M$$
3. Prove that if M is a 2-by-2 matrix, Prove that
$$IM = M = MI$$
4. Prove that if v is a 2-by-1 matrix,
$$Iv = v$$

□

(Go to solution, page 170) □

The set of n–by–n matrices with $\mathbb{R}$ values is usually denoted by $\mathrm{M}_n(\mathbb{R})$. The set of invertible matrices is usually denoted by $\mathrm{GL}_n(\mathbb{R})$. This is called the **general linear group** of n-by-n matrices over $\mathbb{R}$.

general linear group

It turns out that everything works just as fine if the numbers are in mod 26. You just need to be careful with the determinant of the inverse computation. Recall that
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$
for values in $\mathbb{R}$. This is the same for values in $\mathbb{Z}/26$ as long as you think of
$$\frac{1}{ad - bc}$$
as the multiplicative inverse of $ad - bd$ in $\mathbb{Z}/26$.

Exercise 2.9.4. The following refers to matrices with $\mathbb{Z}/26$ values. All the values in the matrices should be simplied to a value in the range 0..25. Do all these exercises:

debug: exercises/hill-3/question.tex

1. Compute
$$\begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$

2. Compute
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 4 & 12 \end{pmatrix}$$
3. If M, N are 2-by-2 matrices, is it always true that $MN = NM$?
4. Prove the following theorem: Let A, B, C be three 2-by-2 matrices. Prove that
$$(AB)C = A(BC)$$
(Look at the facts below on breaking the Hill's cipher. Why did I ask you to prove this theorem? Where is this fact used below?)
5. Does
$$\begin{pmatrix} 0 & 2 \\ 23 & 7 \end{pmatrix}$$
have an inverse? If it does compute it. If not explain why.
6. Does
$$\begin{pmatrix} 1 & 2 \\ 23 & 7 \end{pmatrix}$$
have an inverse? If it does compute it. If not explain why.
7. Compute
$$\begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1}$$
and show that
$$\begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1} \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix} = I = \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix} \begin{pmatrix} 5 & 2 \\ 23 & 7 \end{pmatrix}^{-1}$$

□

(Go to solution, page 171) □

Exercise 2.9.5. Say the key is

debug: exercises/hill-4/question.tex

$$M = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$$

Here $n = 2$, i.e., the block size is 2.

1. What is the encryption of $x =$ `fortytwo`? Call the ciphertext y.
2. What is the inverse matrix M^{-1} of M? (Note: we are in mod 26.) Check that $MM^{-1} = I = M^{-1}M$.
3. Decrypt your y and make sure you get x.
4. Look for the probability of 2-grams. Take a long enough string, encrypt

it using Hill cipher and with the above key. Look at the probability of 2-grams in the ciphertext. Compare the two probabilities of the 2-grams from the plaintext and from teh cipherptext.

(Go to solution, page 172) □

This should not be surprising: The set of n–by–n matrices with mod 26 values is usually denoted by $\mathrm{M}_n(\mathbb{Z}/26)$. The set of invertible matrices is usually denoted by $\mathrm{GL}_n(\mathbb{Z}/26)$.

Exercise 2.9.6. Let

debug: exercises/hill-5/question.tex

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

a mod 26 matrix. Prove that

$$MI = M = IM$$

for any 2-by-2 mod 26 matrix M. □ (Go to solution, page 173) □

Note that in the above, since the block size is 2, you would need to encrypt a string with a string length that is a multiple of 2. If it's not, you can (for instance) pad with some character.

The important thing to observe is that if

$$E(M, (x, y)) = (x', y')$$

(M is a matrix key) note that x takes part in the value x' *and* y'. Same thing for y'. This means that if I change one character in my plaintext, *two* character in the ciphertext is changed. Informally this means that the statistical behavior of x is spread out across multiple characters. Note only that x' also depends on several values of the key (the matrix). This makes it much hard to analyze the key one part at a time when studying ciphertexts. This makes it a lot harder to break Hill's cipher, especially when the matrice size is larger, say the matrix is 32-by-32.

The above two important observations were later formalized and studied by Claude Shannon in 1945.

1. The fact that a character of the ciphertext depends on several parts of the key is the concept of **confusion** and
   confusion
2. the fact that one character in the plaintext influences multiple characters in the ciphertext is called the concept of **diffusion**
   diffusion

More on that in a much later chapter.

Exercise 2.9.7. Write a function `matmult` so that if `M` is a 2D array of size 2-by-2 and `v` is a column vector, i.e., a 2D array of size 2-by-1, then `matmult(M, v)` will return the product `Mv` as described above. Remember to mod by 26 all the values. (If you know your linear algebra – either from computer graphics CISS380 or linear algebra math class, you can figure out how to complete this function for `M` and `v` of sizes n-by-n and n-by-1 respectively) (Go to solution, page 174) □

debug: exercises/hill-6/question.tex

Exercise 2.9.8. Use a 2D array for 2-by-2 matrices. Write a function `matdet` so that `matdet(M)` return the determinant of `M`. Remember that we are in mod 26.

(Go to solution, page 175) □

debug: exercises/hill-7/question.tex

Exercise 2.9.9. Use a 2D array for 2-by-2 matrices. Write a function `matinv` so that `matinv(M)` return the inverse matrix of `M`. Return `None` if `M` is not invertible. Remember that we are in mod 26. (Go to solution, page 176) □

debug: exercises/hill-8/question.tex

Exercise 2.9.10. Let

debug: exercises/hill-9/question.tex

$$M = \begin{pmatrix} 3 & 7 \\ 5 & 2 \end{pmatrix}$$

1. Encrypt $x =$ `solongandthanksforthefish` using Hill's cipher with key M. Call the resulting string `y`.
2. Compute M^{-1}.
3. Decrypt y. You should get x.

(Go to solution, page 177) □

Exercise 2.9.11.

debug: exercises/hill-10/question.tex

1. What is the size of the key space assuming the matrix is 2-by-2? (I mean for mod 26 of course.) This means: how many invertible mod 26 matrices are there? Or: What is $|\mathrm{GL}_2(\mathbb{Z}/26)|$? You can write down a few invertible mod 26 matrices. But after a while you might want to write a program.
2. See if you can find a plausible formula for the case of $n = 2$. The formula

involves 2 and 13 because $26 = 2 \times 13$ and because $n = 2$. (Yes, there's a formula.) It also works if mod 26 is replaced by mod N for any positive integer $N \geq 1$. You can try to work with matrices with values in $\mathbb{Z}/5$ then $\mathbb{Z}/8$, etc. and see if you can find a formula for $|\,\mathrm{GL}_2(\mathbb{Z}/N)|$. (The formula works when n is any integer ≥ 1. But you would need to know about matrices of size larger than 2-by-2.)

□

(Go to solution, page 178) □

Collecting up all the information above, formally, the Hill's cipher of block size n (just think of $n = 2$) is defined as follows: Let

$$P = C = \{0, ..., 25\}^n = (\mathbb{Z}/26)^n$$

and

$$K = \mathrm{GL}_n(\mathbb{Z}/26)$$

The encryption function E

$$E : K \times P \to C$$

is defined to be

$$E(M, x) = Mx$$

and

$$D : K \times C \to P$$

is defined to be

$$D(M, x) = M^{-1}x$$

And (E, D) is a cipher because

$$D(M, E(M, x)) = D(M, Mx) = M^{-1}Mx = Ix = x$$

Exercise 2.9.12. What about 3-by-3 matrices? How do you compute the inverse? What is the size of the key space for 3-by-3? What about n–by–n? debug: exercises/hill-11/question.tex
(Go to solution, page 179) □

Breaking Hill's cipher

Now how do you break the Hill's cipher?

Let's stick to a 2-by-2 key. Suppose you have the following: you have the two plaintexts $x_1 = (A, B)$ and $x_2 = (C, D)$ and their encryptions $y_1 = (A', B')$ and $y_2 = (C', D')$. Therefore

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} aA + bB \\ cA + dB \end{pmatrix} = \begin{pmatrix} A' \\ B' \end{pmatrix}$$

and

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} C \\ D \end{pmatrix} = \begin{pmatrix} aC + bD \\ cC + dD \end{pmatrix} = \begin{pmatrix} A' \\ B' \end{pmatrix}$$

In the above, the unknowns are a, b, c, d, i.e., the key. These two equations can be combined to get an equation involving the product of 2-by-2 matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} aA + bB & aC + bD \\ cA + dB & cC + dD \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix}$$

You now have this matrix equation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix}$$

Note that you have the values of $A, B, C, D, A', B', C', D'$. The unknowns are a, b, c, d. What do you do? You multiply on the right by the inverse matrix of $\begin{pmatrix} A & C \\ B & D \end{pmatrix}$ to get

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

which gives you

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} I = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

which finally gives you the key

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} A' & C' \\ B' & D' \end{pmatrix} \begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1}$$

There's an assumption above: we have to assume that $\begin{pmatrix} A & C \\ B & D \end{pmatrix}$ is invertible. It's possible the first 4 characters of the original message does not form an invertible matrix. But if you have enough pairs of plaintext and corresponding ciphertext blocks, you hope to be able to get an invertible matrix and perform the above computation. Therefore we are assuming the known plaintext attack model. Or we are assuming the chosen plaintext attack model.

Solutions

Solution to Exercise 2.9.1.

debug: exercises/hill-0/answer.tex

Solution not provided.

Solution to Exercise 2.9.2.

debug: exercises/hill-1/answer.tex

Solution not provided.

Solution to Exercise 2.9.3.

debug: exercises/hill-2/answer.tex

Solution not provided.

Solution to Exercise 2.9.4.

debug: exercises/hill-3/answer.tex

Solution not provided.

Solution to Exercise 2.9.5.

debug: exercises/hill-4/answer.tex

Solution not provided.

Solution to Exercise 2.9.6.

debug: exercises/hill-5/answer.tex

Solution not provided.

Solution to Exercise 2.9.7.

debug: exercises/hill-6/answer.tex

Solution not provided.

Solution to Exercise 2.9.8.

debug: exercises/hill-7/answer.tex

Solution not provided.

Solution to Exercise 2.9.9.

debug: exercises/hill-8/answer.tex

Solution not provided.

Solution to Exercise 2.9.10.

debug: exercises/hill-9/answer.tex

Solution not provided.

Solution to Exercise 2.9.11.

debug: exercises/hill-10/answer.tex

Solution not provided.

Solution to Exercise 2.9.12.

debug: exercises/hill-11/answer.tex

Solution not provided.

2.10 One-time pad cipher debug: classical-cipher-one-time-pad-cipher.tex

The one time pad is easy: Suppose Bob wants to send a message. Bob will need to translate this into bits. (For instance use the ASCII code or some other agreed upon format.) Say the plaintext (in bits) is 00101011.

The key is a very long sequence of random bits that Bob and Alice has agreed upon. Suppose the sequence is 0101101011101110001010101100110101001.

Bob takes his message 00101011 and exclusive-or with the first eight bits of the key 0101101011101110001:

```
00101011
01011010 11101110001010101100110101001
01110001
```

He then removes the 8 bits used:

~~01011010~~11101110001010101100110101001

and sends the ciphertex (in bits) 01110001 to Alice.

Once Alice received 01110001, she exclusive-or with her key:

```
01110001
01011010 11101110001010101100110101001
00101011
```

which is the plaintext. She also removes the bits in her key used in the decryption: ~~01011010~~11101110001010101100110101001.

The next time the communicate, they will start with the remaining bits of their key.

That's it.

It's a vry simple cipher. We don't have enough math (yet) to prove it, but you can sense the this cipher is extremely secure. Why?

Exercise 2.10.1. Suppose the ciphertext is zsdeasheir. (Of course you have to convert this to bits, say using ASCII code. Find a key so the the above is decrypted to killatfour. Find another key so that the above is decrypted to anapplepie. (Go to solution, page 182) □

debug: exercises/otp-0/question.tex

There's a rumor that during the Cold War, Washington D.C. communicates with Moscow using one time pad.

To be secure the key must be a random sequence. Furthermore, the key cannot be reused. The other problem is that the key is really long.

(Using the concept of entropy of information theory, Claude Shannon can prove that the ciphertext contains no information about the plaintext, other than the length.)

Solutions

Solution to Exercise 2.10.1.

debug: exercises/otp-0/answer.tex

Solution not provided.

2.11 Block and stream ciphers debug:

classical-cipher-block-and-stream-cipher.tex

Recall that in Caesar cipher, you only define the encryption and decryption for single characters.

The encryption/decryption is then extended to a whole string by encrypting/decrypting character-by-character.

So if you're encrypting the string `cat` with the encryption function E_K, you just do

$$E_K(\texttt{cat}) = E_K(\texttt{c})E_K(\texttt{a})E_K(\texttt{t})$$

Another thing you should know is that, instead of writing

$$\texttt{cat}$$

might write

$$\texttt{c}||\texttt{a}||\texttt{t}$$

i.e., it's common to use || to denote concatenation of data.

A **block** cipher is a cipher system where more than one character (a block) is encrypted together at the time same.

But why encrypt a block of characters at a time?

In the case of shift and affine cipher, the frequency of the plaintext character is the same as the frequency of the corresponding ciphertext character. Sure, an `a` might become a `t` which looks different from `a`. But guess what? `a`'s fingerprint – its frequency – follows it to the ciphertext.

The same idea is the same for Vigenere as long as you have the length of the key.

The next cipher, Hill cipher, is difference. We're going to encrypt several characters at the same time. FOr instance Hill cipher (depending on the key) can encrypt 2 characters at the same time to produce 2 new characters.

Exercise 2.11.1. Is Caesar's cipher a block or stream cipher?

2.12 Linear feedback shift register debug: classical-cipher-lfsr.tex

Recall that the one time pad uses exclusive-or to operator on bit sequences. Note that the exclusive-or is the same as addition mod 2!!! In other words addition *in mod 2*

$$\begin{aligned} 0 + 0 &\equiv 0 \pmod 2 \\ 0 + 1 &\equiv 1 \pmod 2 \\ 1 + 0 &\equiv 1 \pmod 2 \\ 1 + 0 &\equiv 0 \pmod 2 \end{aligned}$$

is the same as exclusive-or operation *on bits*:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 0 &= 0 \end{aligned}$$

(I'm using $\oplus$ for exclusive-or bit operator – that's pretty standard.)

Now I'm going to this: first I define the following bit sequence of length 5:

$$x_1 x_2 x_3 x_4 x_5 = 10110$$

which is the same as defining integer $x_1, ..., x_5$ in $\mathbb{Z}/2$. Then I define

$$x_{n+6} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod 2$$

for $n \geq 0$. For instance

$$x_6 \equiv x_1 + x_2 + x_4 \equiv 1 + 0 + 1 \equiv 0 \pmod 2$$

Here are next about 15:

$$\begin{aligned}
x_6 &\equiv x_1 + x_2 + x_4 \equiv 1 + 0 + 1 \equiv 0 && \pmod 2 \\
x_7 &\equiv x_2 + x_3 + x_5 \equiv 0 + 1 + 0 \equiv 1 && \pmod 2 \\
x_8 &\equiv x_3 + x_4 + x_6 \equiv 1 + 1 + 0 \equiv 0 && \pmod 2 \\
x_9 &\equiv x_4 + x_5 + x_7 \equiv 1 + 0 + 1 \equiv 0 && \pmod 2 \\
x_{10} &\equiv x_5 + x_6 + x_8 \equiv 0 + 0 + 0 \equiv 0 && \pmod 2 \\
x_{11} &\equiv x_6 + x_7 + x_9 \equiv 0 + 1 + 0 \equiv 1 && \pmod 2 \\
x_{12} &\equiv x_7 + x_8 + x_{10} \equiv 1 + 0 + 1 \equiv 1 && \pmod 2 \\
x_{13} &\equiv x_8 + x_9 + x_{11} \equiv 0 + 0 + 1 \equiv 1 && \pmod 2 \\
x_{14} &\equiv x_9 + x_{10} + x_{12} \equiv 0 + 0 + 1 \equiv 1 && \pmod 2 \\
x_{15} &\equiv x_{10} + x_{11} + x_{13} \equiv 0 + 1 + 1 \equiv 0 && \pmod 2 \\
x_{16} &\equiv x_{11} + x_{12} + x_{14} \equiv 1 + 1 + 1 \equiv 1 && \pmod 2 \\
x_{17} &\equiv x_{12} + x_{13} + x_{15} \equiv 1 + 1 + 0 \equiv 0 && \pmod 2 \\
x_{18} &\equiv x_{13} + x_{14} + x_{16} \equiv 1 + 1 + 1 \equiv 1 && \pmod 2 \\
x_{19} &\equiv x_{14} + x_{15} + x_{17} \equiv 1 + 0 + 0 \equiv 1 && \pmod 2 \\
x_{20} &\equiv x_{15} + x_{16} + x_{18} \equiv 0 + 1 + 1 \equiv 0 && \pmod 2
\end{aligned}$$

More generally, after defining $x_1, ..., x_5$ (the initial conditions) you can generated the sequence x_i using

$$x_{n+6} \equiv c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} + c_4 x_{n+4} + c_5 x_{n+5} \pmod 2$$

for $n \geq 0$ for constants $c_1, ..., c_5$ in $\mathbb{Z}/2$. I will say that this is linear relation has **degree 5**. Even more generally, you can have any number of bits for the initial condition. Say you begin with $x_1, ..., x_k$ (the initial condition) and the relation is

degree 5

$$x_{n+k+1} \equiv c_1 x_{n+1} + c_2 x_{n+2} + c_3 x_{n+3} + \cdots + c_k x_{n+k} \pmod 2$$

LSRFs are very easy to implement in both hardware and software and they are extremely fast (they just access bits and XOR them). The LSRF generator itself need to remember the k bits $c_1, ..., c_k$ (which is fixed) and the k bits of the sequence so far $x_{n+1}, ..., x_{n+k}$ in order to generate the next bit x_{n+k+1}.

In the above example, the sequence from x_1 to x_{20} is

1011001000111101011001000111101011 0...

Notice that the pattern repeats itself:

$$101100100011110 \;\; 101100100011110 \;\; 10110...$$

The period is 15.

The problem with the one-time pad is that you need to generate a random sequence of 0s and 1s. You can see that with 5 bits

$$x_1x_2x_3x_4x_5 = 10110$$

and the relation

$$x_{n+6} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod 2$$

which involves (1,1,0,1,0) (5 bits), a total of 10 bits) we can generate

$$101100100011110$$

which has length 15. The 15 bits is somewhat random – we say that the 15 bits are **pseudorandom**. Therefore LFSR can be used to generate a pseudorandom bit sequence, which can used, for instance, as a key for the one-time pad. Of course you want to find a LSFR with extremely long periods. pseudorandom

Exercise 2.12.1. What if you define debug: exercises/lfsr-0/question.tex

$$x_1x_2x_3x_4 = 1011$$

and use

$$x_{n+5} \equiv x_{n+1} + x_{n+2} + x_{n+4} \pmod 2$$

for $n \geq 0$ instead? (Compare with the LSRF above). What sequence do you get? What is the period? (Go to solution, page 189) □

Exercise 2.12.2. Compute all the possible degree 1 LSRF bit sequences. How many possibilities are there? What is the period for each of them? (Go to solution, page 190) □ debug: exercises/lfsr-1/question.tex

Exercise 2.12.3. Compute all the possible degree 2 LSRF bit sequences. How many possibilities are there? What is the period for each of them? (Go to solution, page 191) □ debug: exercises/lfsr-2/question.tex

Exercise 2.12.4. Compute all the possible degree 3 LSRF bit sequences. How many possibilities are there? What is the period for each of them? (Go to solution, page 192) ☐

debug: exercises/lfsr-3/question.tex

Exercise 2.12.5. Write a program that generated a sequence of integers with values or 0 and 1 using the LFSR method. You want to have a `c` array as a parameter. Your generator also need to remember bits of the sequence $x_1, \ldots$ needed to generate the next bits. In general, for a fixed k, you want to have an array c of size k and an array x also of size k. When you call your function, the next bit is placed in the array x, shifting the bits of x so that one bit is lost. So if you want the full sequence for analysis, you need a very long array to keep the bits before it's removed from x. For instance the main program might look like this:

debug: exercises/lfsr-4/question.tex

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
bits = [x1, x2, x3, x4, x5] # the full sequence
LSRF(c, x) # append rightmost value of x to the right of bits
LSRF(c, x) # append rightmost value of x to the right of bits
etc.
```

Besides putting the new bit into `x`, it's also a good idea to return that bit as well. Then the above becomes

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
bits = [x1, x2, x3, x4, x5] # the full sequence
b = LSRF(c, x); append b to the right side of bits
b = LSRF(c, x); append b to the right side of bits
etc.
```

If you like you can also write an LFSR class. Then the above becomes

```
x = [x1, x2, x3, x4, x5] # the initial bits
c = [c1, c2, c3, c4, c5] # the coefs of the linear relation
LFRS = LRFSClass(c, x)
bits = [x1, x2, x3, x4, x5] # the full sequence
b = LSRF.run(); append b to the right side of bits
b = LSRF.run(); append b to the right side of bits
etc.
```

In the above, the code works with integers 0 and 1. For scenarios where there is a huge number of bits, the bits are packed into a register (say of size 64 bits). (Go to solution, page 193) □

Exercise 2.12.6. For a given initial sequence of bits and the sequence `c`, write a function that attempts to compute the length of the period. Try lots of examples and see if you can produce cases of extremely long periods. (Go to solution, page 194) □

debug: exercises/lfsr-5/question.tex

Solutions

Solution to Exercise 2.12.1.

debug: exercises/lfsr-0/answer.tex

Solution not provided.

Solution to Exercise 2.12.2.

debug: exercises/lfsr-1/answer.tex

Solution not provided.

Solution to Exercise 2.12.3.

debug: exercises/lfsr-2/answer.tex

Solution not provided.

Solution to Exercise 2.12.4.

debug: exercises/lfsr-3/answer.tex

Solution not provided.

Solution to Exercise 2.12.5.

debug: exercises/lfsr-4/answer.tex

Solution not provided.

Solution to Exercise 2.12.6.

debug: exercises/lfsr-5/answer.tex

Solution not provided.

Index

Bibliography

[1] Leslie Lamport, *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994. (EXAMPLE)