Contents

1 Classical Cryptography **2**

1.1 Introduction to Cryptography 2

1.1.1 Language 2

1.1.2 Caesar Ciphers 4

1.1.3 Monoalphabetic substitution 7

1.1.4 Frequency Analysis of Monoalphabetic Ciphers 7

1.2 Polyalphabetic Ciphers 13

1.2.1 The Alberti Cipher 14

1.2.2 The Vigenère Cipher 14

1.2.3 Autokey ciphers 15

1.2.4 Modern Stream Ciphers 16

1.2.5 Binary encryption 16

1.3 Statistical Models and Cryptanalysis 17

1.3.1 The Index of Coincidence 17

1.3.2 Breaking the Vigenère cipher 19

1.3.3 The Autokey cipher and cribs 25

1 Classical Cryptography

1.1 Introduction to Cryptography

1.1.1 Language

Cryptology is the study of sending (hopefully) secure messages over non-secure channels. We sometimes use the term *cryptography* for the study of designing such systems; *cryptanalysis* is the study of breaking the security of such systems.

There are three basic types of cryptography.

Steganography Stegaography is the concealing of your message so that evesdroppers can't tell you're sending a message. When this works it is incredibly effective; if no one knows you're sending a message at all, then no one is even trying to read it.

On the other hand, steganography doesn't work very well at all if people know what you're doing. Once they know where to look, your steganography has failed, so it is very fragile.

Famously, the ancient Greek tyrant Histiaeus wanted to send secret message plotting a revolt against the Persians to his nephew Aristagoras. He shaved the head of one of his slaves, tattooed a message on his head, then waited for the hair to grow back. Aristagoras was instructed to shave the slave's head and read the message.

This worked very effectively, but would not have succeeded at all if slave-head-tattooing had been a common method of sending secret messages. It also didn't have very good performance: it took months to send a short message.

Modern security practice rarely involves the tattooing of slaves, but we do have more sophisticated steganographic techniques. A common one involves hiding a message in an image file by adjusting the least significant digit of the color intensity of each pixel. This will change the image imperceptibly but allows the encoding of about one character for every three pixels of image. (If you've read the Orson Scott Card novel *Shadow of the Hegemon*, this sort of technique is an important plot point there).

More prominently, a Russian hacking group called "Turla" recently used the comments in Britney Spears's instagram account to secretly send instructions and updates to infected computers. Messages like `#2hot make loved to her, uupss #Hot #X` were secretly encoded addresses for new control servers for the malware.

We won't spend a lot of time in this course discussing steganography, because the actaul steganographic part of the message tends to involve abstractly clever hiding places more

than the use of interesting mathematics.

Codes Codes are prearranged set of signals and representations for specific meanings.

Codes make substitutions at the level of words or phrases; the stereotypical spy-talk "The eagle flies at midnight" is a code phrase because it sends a specific and complex prearranged thought.

A code is effectively a parallel language, and in fact uncommon and poorly-known languages have been used as codes. In World War II, the US Army often used Native Americans speaking to each other in their native languages ("code talkers") to transmit information that their opponents could not decipher through any sort of mathematical analysis.

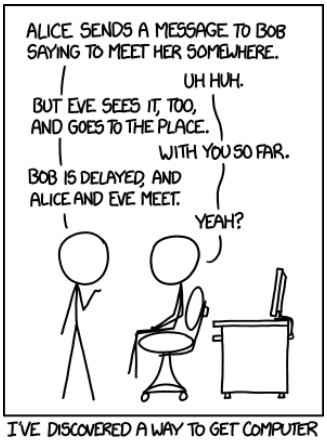
We will again mostly not discuss codes, because they primarily aren't mathematically interesting; they are abstract enough that in effect, you either know the correspondences or you don't. (The primary disadvantage of codes is that you have to communicate and protect very extensive codebooks for them to be practicable). However, we may discuss the idea of *encoding* and *coding theory*, which are related but distinct.

Ciphers Ciphers are the primary subject of this course.

In contrast to codes, which make substitutions at the level of meanings and words and phrases, ciphers replace individual characters (or bits) of data.

Ciphers are very flexible and can encrypt arbitrary messages, since they replace character-by-character.

The basic setup we wish to study features two parties (usually named *Alice* and *Bob*) who wish to communicate with each other, but wish to prevent a third party "evesdropper" or "assailant" (usually named *Eve*) from understanding their messages.



I'VE DISCOVERED A WAY TO GET COMPUTER SCIENTISTS TO LISTEN TO ANY BORING STORY.
https://xkcd.com/1323

The message Alice wishes to send to Bob is called the *plaintext*. She will use some prearranged encryption method to convert it into a *ciphertext*, which she sends to Bob. Bob uses his knowledge of the encryption method to convert the ciphertext back into the plaintext; even if Eve acquires the ciphertext, she shouldn't be able to perform the same conversion to acquire (or, in more malicious scenarios, replace) the plaintext.

1.1.2 Caesar Ciphers

One of the earliest known ciphers is what's known as the *Caesar cipher* or *Shift cipher*. (Julius Caesar was hardly the first to use them; there are examples dating back to at least Sparta, and of variants used by the Hebrews before 500 BCE).

The Caesar cipher works by picking a number n between 1 and 25, called the *shift*, and replacing each letter with the letter n places to the left. (Note: some sources shift to the right instead; the direction doesn't really matter as long as you're consistent). The traditional story has Caesar using a shift of three to the left:

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

> If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.
>
> — Suetonius, Life of Julius Caesar 56

Example 1.1. Suppose we have the ciphertext "QEFP JBPPXDB EXP YBBK BKZFMEBOBA YV X ZXBPXO ZFMEBO TFQE X PEFCQ LC QEOBB", and we know it has been enciphered by a Caesar cipher with a shift of three. We can decipher it by shifting each letter forward by three places. we get "THIS MESSAGE HAS BEEN ENCIPHERED BY A CAESAR CIPHER WITH A SHIFT OF THREE."

Remark 1.2. The most commonly used Caesar cipher today is ROT13, for "rotation 13", which is a shift by thirteen. Notice that in this case the algorithms for encryption and decryption are actually identical.

ROT13 was in common use on the early internet for spoilers: it's easy to decrypt so it doesn't prevent anyone who wants to read it from reading it, but it keeps you from accidentally reading the message without intending to. Thus ROT13 is built into a number of text editors and other computing tools.

People sometimes joke about the encryption algorithm double-ROT13, which is "obviously" twice as secure.

There are 26 possible shift ciphers. So if you know you're dealing with a shift cipher it's actually pretty easy to decode.

Example 1.3. Suppose we have the message IWXH RXEWTG XH HWXUITS QN TATKTC, and we know it has been encoded by some Caesar cipher. Since there are only twenty-six possibilities we can just try each one. In fact, we can apply each shift to just the first word and see which shifts make sense:

0	IWXH	7	PDEO	14	WKLV	21	DRSC
1	JXYI	8	QEFP	15	XLMW	22	ESTD
2	KYZJ	9	RFGQ	16	YMNX	23	FTUE
3	LZAK	10	SGHR	17	ZNOY	24	GUVF
4	MABL	11	THIS	18	AOPZ	25	HVWG
5	NBCM	12	UIJT	19	BPQA		
6	OCDN	13	VJKU	20	CQRB		

Looking down this table, we see the only shift that produces a recognizable word is a shift of eleven, giving "THIS" as output. If we shift the entire message by eleven, we get the message "THIS CIPHER IS SHIFTED BY ELEVEN".

In order to make this process slightly harder, most enciphered text messages will be sent without spaces between words. (This was also easier on early transmission technology). Consider the following example:

Example 1.4. CWUHLYUXNBCMWUHSIO

We could try every shift on the entire message. In this case, the message is short enough that that's reasonable, but for longer messages we might not want to convert every letter with every possible shift.

Instead, we can test the first few letters with each shift, and see which ones look like they might contain or be part of words. Taking the first three letters "CWU" we get the following table:

0	CWU	7	JDB	14	QKI	21	XRP
1	DXV	8	KEC	15	RLJ	22	YSQ
2	EYW	9	LFD	16	SMK	23	ZTR
3	FZX	10	MGE	17	TNL	24	AUS
4	GAY	11	NHF	18	UOM	25	BVT
5	HBZ	12	OIG	19	VPN		
6	ICA	13	PJH	20	WQO		

Based on just these letters, it's hard to tell which options are good; 4 gives us "gay" which is a word, and 6, 12, and 24 all give us collections of letters that at least look like they could be English. But we can be pretty confident at throwing out strings like "FZX" and "JDB" if we expect the message to be words.

Using a few more letters gives us better data. If we used the first six instead of the first three, we would get the table

0	CWUHLY	7	JDBOSF	14	QKIVZM	21	XRPCGT
1	DXVIMZ	8	KECPTG	15	RLJWAN	22	YSQDHU
2	EYWJNA	9	LFDQUH	16	SMKXBO	23	ZTREIV
3	FZXKOB	10	MGERVI	17	TNLYCP	24	AUSFJW
4	GAYLPC	11	NHFSWJ	18	UOMZDQ	25	BVTGKX
5	HBZMQD	12	OIGTXK	19	VPNAER		
6	ICANRE	13	PJHUYL	20	WQOBFS		

and the only terribly useful-looking shift is 6, which gives "I can re"as plausible English. So using a shift of 6, we get "ICANREADTHISCANYOU". After squinting, we see the message: "I can read this. Can you?"

Modular arithmetic Recall from Math 2971 that we say two integers a, b are equivalent modulo m, and write $a \equiv b \mod m$, if $m|b-a$. This is an equivalence relation, and is preserved by addition, subtraction, and multiplication.

The Caesar cipher is easy to describe in terms of modular arithmetic. We can identify each letter with a number modulo 26. There are different conventions here, but we'll follow Hoffstein, Pipher, and Silverman:

Definition 1.5. There is a bijection between the letters of the alphabet and numbers modulo 26, sending A to 0, B to 1, and so on, up to sending Z to 25.

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

Then we can see a Caesar cipher as choosing some fixed number n, and subtracting that number from each letter in your message. For the rest of the class we'll probably *add* the number instead, because it's easier to think about.

From this perspective, a key is just a number from 0 to 25 inclusive.

Example 1.6. If the message is "MEET ME AT MIDNIGHT", we can render this as the list of numbers 12 4 4 19 12 4 0 19 12 8 3 13 8 6 7 19.

If we want to encrypt this with a shift of plus 15, we get 1 19 19 8 1 19 15 8 1 23 18 7 23 21 22 8, which translates back to "BTTIBTPIBXSHXVWI" which is our ciphertext.

To decrypt, we simply subtract 15 from each letter and get back "MEETMEATMIDNIGHT".

1.1.3 Monoalphabetic substitution

The shift ciphers above are limited by the fact that you have to keep the letters in order—you only change which one comes first. But there's no reason we can't reorder them. A *monoalphabetic substitution cipher* does away with that constrant, and simply pairs each plaintext letter up with a distinct ciphertext letter.

Example 1.7. An example monoalphabetic substitution cipher would be

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

To encipher a message, we find each letter of the plaintext in the first row, and replace it with the corresponding letter in the second row. So the message "THIS IS A SIMPLE SUBSTITUTION CIPHER" becomes "OBHM HM G MHSTNE MVIMOHOVOHFD LHTBEU". We can decrypt the ciphertext by looking up each letter of the ciphertext in the second row and finding the corresponding character in the first row. It can be helpful to have a table sorted by the ciphertext characters instead of the plaintext characters:

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

Thus a choice of monoalphabetic substitution cipher is equivalent to a choice of a permutation of the alphabet (or to an element of the group S_{26}). Therefore there are $26! \approx 4 \times 10^{26}$ possible monoalphabetic substitution ciphers, so *unlike* in the case of a Caesar cipher, you can't realistically try all the possibilities until you find one that works. We need to be a bit cleverer.determine the substitution.

1.1.4 Frequency Analysis of Monoalphabetic Ciphers

Fortunately, language has a structure to it: not all strings of letters are equally common. (If I tell you my message is either "brown" or "yoltk", you can make a pretty good guess as to which it is).

The first and simplest tool we have is the relative frequency of letters in English text. This approach is usually credited to the Arab philosopher Abu Yusuf Ya'qub ibn Ishaq Al-Kindi in the ninth century CE. The basic idea is that not all letters occur equally often; if your ciphertext has one letter appearing ten times in fifty total letters, it's probably not a "q" or "z".

There are also more sophisticated approaches we can take to frequency analysis, because English does not have its letters distributed in a random order. (That is, our model of the plaintext has more information than just the frequencies of individual letters). That is, if

we see the same pair of letters appearing in the same order many times, we might guess that this pair is "th" or "he" or "an". If we see the same trio of letters appearing many times, we might guess that it is "the" or "and".

Below are tables showing the frequency with which each letter appears in English texts (Figure 1.1), and the frequencies of the most common English bigrams (Figure 1.2), drawn from Hoffstein, Pipher, and Silverman. (Note that different sources will have slightly different numbers due to using different corpuses).

E	13.11%	M	2.54%	A	8.15%	N	7.10%
T	10.47%	U	2.46%	B	1.44%	O	8.00%
A	8.15%	G	1.99%	C	2.76%	P	1.98%
O	8.00%	Y	1.98%	D	3.79%	Q	0.12%
N	7.10%	P	1.98%	E	13.11%	R	6.83%
R	6.83%	W	1.54%	F	2.92%	S	6.10%
I	6.35%	B	1.44%	G	1.99%	T	10.47%
S	6.10%	V	0.92%	H	5.26%	U	2.46%
H	5.26%	K	0.42%	I	6.35%	V	0.92%
D	3.79%	X	0.17%	J	0.13%	W	1.54%
L	3.39%	J	0.13%	K	0.42%	X	0.17%
F	2.92%	Q	0.12%	L	3.39%	Y	1.98%
C	2.76%	Z	0.08%	M	2.54%	Z	0.08%

Figure 1.1: English Letter Frequencies

th	he	an	re	er	in	on	at	nd	st	es	en	of	te	ed
168	132	92	91	88	86	71	68	62	53	52	51	49	46	46

Figure 1.2: Most common English bigrams (frequency per 1000 words)

Example 1.8. Let's try to decrypt the ciphertext:

```
JNRZR BNIGI BJRGZ IZLQR OTDNJ GRIHT USDKR ZZWLG OIBTM NRGJN IJTZJ LZISJ NRSBL
QVRSI ORIQT QDEKJ JNRQW GLOFN IJTZX QLFQL WBIMJ ITQXT HHTBL KUHQL JZKMM LZRNT
OBIMI EURLW BLQZJ GKBJT QDIQS LWJNR OLGRI EZJGK ZRBGS MJLDG IMNZT OIHRK MOSOT
QHIJL QBRJN IJJNT ZFIZL WIZTO MURZM RBTRZ ZKBNN LFRVR GIZFL KUHIM MRIGJ LJNRB
GKHRT QJRUU RBJLW JNRZI TULGI EZLUK JRUST QZLUK EURFT JNLKJ JNRXR S
```

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Frequency	0	15	0	5	5	6	15	8	27	30	13	25	12	19	10	0	16	33	9	20	12	2	7	3	0	24

Letter	R	J	I	L	Z	T	N	Q	B	G	K	U	M	O	S	H	W	F	E	D	X	V
Frequency	33	30	27	35	24	20	19	16	15	15	13	12	12	10	9	8	7	6	5	5	3	2

Bigram	JN	NR	TQ	LW	RB	RZ	JL
Frequency	11	8	6	5	5	5	5

Figure 1.3: Frequency count for example 1.8

We begin by counting the frequency of each letter, and put our results in figure 1.3

The most common letter is "R" so we'll guess that "R" is encrypting "e". We notice that the most common bigrams in the ciphertext are "JN" and "NR", and the most common bigrams in English are "th" and "he"; this leads us to guess that "JNR" is "the". (This is also reassuring since the second most common English letter is "t" and the second-most-common letter in the ciphertext is "J", to which we've just assigned the letter "t").

```
theZe BhIGI BteGZ IZLQe OTDht GeIHT USDKe ZZWLG OIBTM heGth ItTZt LZISt heSBL
QVeSI OeIQT QDEKt theQW GLOFh ItTZX QLFQL WBIMt ITQXT HHTBL KUHQL tZKMM LZehT
OBIMI EUeLW BLQZt GKBtT QDIQS LWthe OLGeI EZtGK ZeBGS MtLDG IMhZT OIHeK MOSOT
QHItL QBeth ItthT ZFIZL WIZTO MUeZM eBTeZ ZKBhh LFeVe GIZFL KUHIM MeIGt LtheB
GKHeT QteUU eBtLW theZI TULGI EZLUK teUST QZLUK EUeFT thLKt theXe S
```

There are a few things we could do now. We can look at our other list of common bigrams. We see that "JL" is common in the ciphertext, which means we need a common English bigram whose first letter is "t"; "te" is the most common, but is ruled out since we know that "e" is "R". No others appear on our list.

We also have "RB" and "RZ" as common bigrams, and we know that "R" is "e". Looking at our list, it looks like these bigrams are probably "er" and "es". It's not entirely clear which should be which; one would make the first word of our text "there" and the second would give "these", which are both perfectly reasonable. It's not clear what to do here.

We can also just go ahead and guess that our next-most-common ciphertext letters "I" and "L" are our next-most-common plaintext letters "a" and "o". (This also makes "JL" into "to", which seems quite plausible!) That would give us

```
theZe BhaGa BteGZ aZoQe OTDht GeaHT USDKe ZZWoG OaBTM heGth atTZt oZaSt heSBo
QVeSa OeaQT QDEKt theQW GoOFh atTZX QoFQo WBaMt aTQXT HHTBo KUHQo tZKMM oZehT
OBaMa EUeoW BoQZt GKBtT QDaQS oWthe OoGea EZtGK ZeBGS MtoDG aMhZT OaHeK MOSOT
QHato QBeth atthT ZFaZo WaZTO MUeZM eBTeZ ZKBhh oFeVe GaZFo KUHaM MeaGt otheB
```

```
GKHeT QteUU eBtoW theZa TUoGa EZoUK teUST QZoUK EUeFT thoKt theXe S
```

At this point we might want to go looking through the text for guessable words. We see "at" several times, and might start noticing some patterns. What sticks out to me is the string "eth atth", which looks like it ends in "something-eth at th-". Almost certainly, the next letter should be a vowel; since we have "e,a,o" already spoken for, it should be "i" or "u". The ciphertext letter "T" is quite common; since "i" is a common English letter and "u" is not, we'll guess that "T" becomes "i".

```
theZe BhaGa BteGZ aZoQe OiDht GeaHi USDKe ZZWoG OaBiM heGth atiZt oZaSt heSBo
QVeSa OeaQi QDEKt theQW GoOFh atiZX QoFQo WBaMt aiQXi HHiBo KUHQo tZKMM oZehi
OBaMa EUeoW BoQZt GKBti QDaQS oWthe OoGea EZtGK ZeBGS MtoDG aMhZi OaHeK MOSOi
QHato QBeth atthi ZFaZo WaZiO MUeZM eBieZ ZKBhh oFeVe GaZFo KUHaM MeaGt otheB
GKHei QteUU eBtoW theZa iUoGa EZoUK teUSi QZoUK EUeFi thoKt theXe S
```

Looking back at our list of bigrams, we see that "TQ" is common. An English bigram whose first letter is "i" is probably "in", so we might guess that "Z" is "n"; but this gives some unlikely strings in our message like "thene BhaGa" or "that in tonaSt" or "-eth at thinFano" or "toW thenaiUoGa". While any one of these is possible, they don't seem likely. Unfortunately we don't have any other common i-initial bigrams to look at.

So let's go back to our idea that maybe "Z" is "s" or "r". "r" is the more common English letter, so we might try that first. "there BhaGa" seems plausible; but "that irtora" is improbable, as is "-eth at thirFaro" or "toW theraiUoGa".

So we try "s", and we get "these BhaGa"; "that is to sa", "-eth at this Faso" and "toW thesaiUoGa". The first three seem extremely likely, and the fouth possible, so we guess that "Z" is "s".

We can now look to sort out words, or just go back to our frequency charts. The next most common bigram is "TQ" which is "iQ", and the most common English bigram beginning with "i" is "in. Also, the next most common letter is "Q", and the next most common English letter is "n", so we might guess from both of these things that "Q" is "n".

```
these BhaGa BteGs asone OiDht GeaHi USDKe ssWoG OaBiM heGth atist osaSt heSBo
nVeSa Oeani nDEKt thenW GoOFh atisX noFno WBaMt ainXi HHiBo KUHno tsKMM osehi
OBaMa EUeoW Bonst GKBti nDanS oWthe OoGea EstGK seBGS MtoDG aMhsi OaHeK MOSOi
nHato nBeth atthi sFaso WasiO MUesM eBies sKBhh oFeVe GasFo KUHaM MeaGt otheB
GKHei nteUU eBtoW thesa iUoGa EsoUK teUSi nsoUK EUeFi thoKt theXe S
```

The next most common letter after this is "B". Our first thought is that we can take the next most common English letter of "r", but then we get "rhaGarte" which really doesn't

look likely. On the other hand, if we replace our "G" with "r" we get "BharaBters" which suggests "B" for "c".

```
these chara cters asone OiDht reaHi USDKe ssWor OaciM herth atist osaSt heSco
nVeSa Oeani nDEKt thenW roOFh atisX noFno WcaMt ainXi HHico KUHno tsKMM osehi
OcaMa EUeoW const rKcti nDanS oWthe Oorea EstrK secrS MtoDr aMhsi OaHeK MOSOi
nHato nceth atthi sFaso WasiO MUesM ecies sKchh oFeVe rasFo KUHaM Meart othec
rKHei nteUU ectoW thesa iUora EsoUK teUSi nsoUK EUeFi thoKt theXe S
```

At this point we really should be looking to recognize words in teh text. The phrase "that is to saS theS" strongly suggests that "S" is "y". We see the string "constrKct" which tells us "K" has to be a vowel; the only one left is "u", and "construct" is a reasonable word.

```
these chara cters asone OiDht reaHi UyDue ssWor OaciM herth atist osayt heyco
nVeya Oeani nDEut thenW roOFh atisX noFno WcaMt ainXi HHico uUHno tsuMM osehi
OcaMa EUeoW const ructi nDany oWthe Oorea Estru secry MtoDr aMhsi OaHeu MOyOi
nHato nceth atthi sFaso WasiO MUesM ecies suchh oFeVe rasFo uUHaM Meart othec
ruHei nteUU ectoW thesa iUora EsoUu teUyi nsoUu EUeFi thout theXe y
```

"sMecies" tells us that "M" is probably "p". "Fithout" suggests that "F" is "w", and then "without the Xey" seriously limits what "X" can stand for; we guess "X" is "k". "the saiUor" is probably "the sailor", so "U" is probably "l".

```
these chara cters asone OiDht reaHi lyDue ssWor Oacip herth atist osayt heyco
nVeya Oeani nDEut thenW roOwh atisk nowno Wcapt ainki HHico ulHno tsupp osehi
Ocapa EleoW const ructi nDany oWthe Oorea Estru secry ptoDr aphsi OaHeu pOyOi
nHato nceth atthi swaso WasiO plesp ecies suchh oweVe raswo ulHap peart othec
ruHei ntell ectoW thesa ilora Esolu telyi nsolu Elewi thout theke y
```

"aEsolutely" implies that "E" is "b". The most common leftover letter is "O"; we see it in "these characters, as one OiDht reaHily..." and in "not suppose hiO capable" and "the Oore abstruse cryptoDraph". From these, we guess "O" is "m" (especially to fill out "him") and then guess that "D" is "g", giving us "might" and "cryptograph".

```
these chara cters asone might reaHi lygue ssWor macip herth atist osayt heyco
nVeya meani ngbut thenW romwh atisk nowno Wcapt ainki HHico ulHno tsupp osehi
mcapa bleoW const ructi ngany oWthe morea bstru secry ptogr aphsi maHeu pmymi
nHato nceth atthi swaso Wasim plesp ecies suchh oweVe raswo ulHap peart othec
ruHei ntell ectoW thesa ilora bsolu telyi nsolu blewi thout theke y
```

Finally, "reaHily guess" implies that "H" is "d". We're running out of letters now; we look at the "V" and see it twice, in "that is to say they conVey a meaning" and "a simple

species such howeVer as would", and it looks like "V" is actually "v"! We just need to translate the "W", and "but then Wrom" tells us that "W" is "f".

```
these chara cters asone might readi lygue ssfor macip herth atist osayt heyco
nveya meani ngbut thenf romwh atisk nowno fcapt ainki ddico uldno tsupp osehi
mcapa bleof const ructi ngany ofthe morea bstru secry ptogr aphsi madeu pmymi
ndato nceth atthi swaso fasim plesp ecies suchh oweve raswo uldap peart othec
rudei ntell ectof thesa ilora bsolu telyi nsolu blewi thout theke y
```

Going through and respacing, we get:

These characters, as any one might readily guess, form a cipher—that is to say, they convey a meaning; but then from what is known of Kidd, I could not suppose him capable of constructing any of the more abstruse cryptographs. I made up my mind, at once, that this was of a simple species—such, however, as would appear to the crude intellect of the sailor, absolutely insoluble without the key.

And the key to the cipher is

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

A few things to notice: by chance, "V" becomes "v", and "F" and "W" are interchanged. Nothing requires that sort of thing to happen, but nothing prohibits it either.

Also notice that there are some ciphertext and plaintext letters that we don't have correspondences for. The plaintext simply never used an "x" or a "z", so we don't know what rule it would have used for them, had it needed one. But if we got a future message in the same cipher, it would be quite easy to determine the meanings of the "A" and "C" in the message.

Note that this process requires experimentation and can take a number of wrong turns; I personally spent quite a while convinced that "L" was "i" in the preceeding cipher. If something isn't work, revisit one of your earlier guesses and try something different.

This sort of approach can fail in a few ways:

1. The message could be too short. If the message is ten letters long we can't possibly do any useful statistical analysis on it. (In fact a ten-letter message is generally impossible

to decipher even in principle; an English message typically must be at least 27.6 letters to be amenable to frequency analysis. We'll discuss this general topic later, in **??**).

2. The text could be atypical. The pangram "A quick brown fox jumped over the lazy dog" is often used as a test sentence in many applications. But this message has four "o"s and only two "e"s, so statistical approaches will be somewhat misleading.

   The likelihood of this happening by accident, and the difficulty of it happening on purpose, decrease as the messages get longer. But it's completely possible that even a long message is highly atypical in this way; Ernest Vincent Wright wrote a full novel, called "Gadsby", without using the letter "e".

3. The text might not even be English. For instance, in Portuguese the most common letter is "a", not "e", and the letter "t" barely cracks the top ten. If you do statistical analysis assuming the encrypted message is English, but it's actually Portuguese, you may never even make enough progress to realize your error.

4. The message might not be enciphered with a monoalphabetic substitution cipher at all.

The first problem is a problem of not enough data; the third and fourth problems are problems of having a flawed model. (The second problem is a mix of the two). Both of these are important problems that come up any time we do statistical analysis. If our modelling assumptions are wrong, all the statistics in the world won't help us.

However, this sort of statistical analysis is powerful enough and well-enough understood that monoalphabetic ciphers are considered pretty thoroughly insecure; despite the number of possible keys, anyone who knows what they're doing can break these ciphers easily, and this has been true for over a millennium.

1.2 Polyalphabetic Ciphers

The next major advance in cryptography began the 1500s with the invention of the polyalphabetic cipher.

The basic idea here is simple. A monoalphabetic substitution cipher is vulnerable to attacks based on letter frequency, exploiting the fact that one letter in the ciphertext always represents the same letter in the plaintext. We can (partially) defeat this attack by not always using the same ciphertext letter to represent the same plaintext letter.

1.2.1 The Alberti Cipher

The first (known) implementation of this idea was straightforward. Leon Battista Alberti in 1467 would write in a Caesar cipher, but from time to time he would change to a different shift, and signal this shift with the use of a capital letter.

This method of encryption by itself isn't much better than a single Caesar cipher, since you can easily tell when the cipher is shifting, and simply break each portion on its own.

Example 1.9.

However, this idea of using more than one substitution to encrypt a message is very powerful. Basically all strong encryption through the end of World War II is based on this principle.

1.2.2 The Vigenère Cipher

The most common and simplest example of polyalphabetic cipher is usually attributed to Blaise de Vigenère, though it's probably more accurately attributed to Giovan Battista Bellasco in 1553; Vigenère improved on it in 1586 and then got the credit.

To use the Vigenère Cipher we first need a *key word*, which can be any word (or any string of letters). We treat each letter as determining a specific Caesar cipher. There are a number of ways to determine this correspondence, but we will use the numerical correspondence we established earlier, adding the letter of our keystream to the letter of our ciphertext.

Algorithm 1.1 (Vigenère)**.** Start with a key word. Encrypt the first letter of your plaintext using the Caesar cipher corresponding to the first letter of your key word. Encrypt the second letter of your plaintext according to the second letter of your key word, and so on, repeating your key word when you reach the end.

To decrypt the cipher, we do the same thing. We use hte same keyword, but we subtract the keystream instead of adding it.

Example 1.10. Suppose we want to encrypt the plaintext "I LOVE CRYPTOLOGY" using the key word "MATH". We start by writing the message out, with the key word repeated under it (this repeated keyword is called the *keystream*):

Plaintext	I L O V E C R Y P T O L O G Y
Keystream	M A T H M A T H M A T H M A T

It's probably helpful at this point to replace the letters with their corresponding numbers, which gives

Plaintext	8	11	14	21	4	2	17	24	15	19	14	11	14	6	24
Keystream	12	0	19	7	12	0	19	7	12	0	19	7	12	0	19
Ciphertext	20	11	7	2	16	2	10	3	1	19	7	18	0	6	17
Ciphertext	U	L	H	C	Q	C	K	D	B	T	H	S	A	G	R

To decrypt we do the same thing in reverse, subtracting the keystream from the ciphertext to get the plaintext.

This cipher is highly resistant to the sort of frequency analysis-based attacks we will study in section 1.1.4. Common letters like "e" and "a" will be be substituted by multiple different letters, and each letter in the ciphertext representes more than one plaintext letter, so knowing which letter appears most frequently in the ciphertext doesn't convey much information. The Vigenère Cipher effectively smooths out the frequencies so that frequency analysis does not work.

In fact, for over two hundred years people considered the Vigenère cipher effectively unbreakable. However in 1854 Charles Babbage successfully completed a public challenge to decrypt a Vigenère ciphertext; however, he never published his techniques. In 1863 Friedrich Kasiski published a method that is effective at breaking the Vigenère cipher, which we will study in section 1.3.2.

1.2.3 Autokey ciphers

Any cipher that works by generating a keystream and then adding it to the plaintext is called a *stream cipher*. There are a number of variants we can discuss.

Blaise de Vigenère actually developed an autokey cipher, which uses the plaintext to generate the keystream. There are a number of variants to this idea.

One simple algorithm is to form a keystream by using a keyword and then following it with the plaintext. So under this algorithm, if the keyword is "MATH" and the message is "I LOVE CRYPTOLOGY", we get the keystream "MATHILOVECRYPTO". Then we encrypt:

Plaintext	I	L	O	V	E	C	R	Y	P	T	O	L	O	G	Y
Keystream	M	A	T	H	I	L	O	V	E	C	R	Y	P	T	O

Plaintext	8	11	14	21	4	2	17	24	15	19	14	11	14	6	24
Keystream	12	0	19	7	8	11	14	21	4	2	17	24	15	19	14
Ciphertext	20	11	7	2	12	13	5	19	19	21	5	9	3	25	12
Ciphertext	U	L	H	C	M	N	F	T	T	V	F	J	D	Z	M

Notice that we got the same first four letters as with the Vigenère cipher (since the key is the same and four letters long), but things change after that.

In order to decrypt a message encrypted in this way, we decrypt in chunks. Knowing the keyword lets us decrypt the first four letters; knowing the first four letters lets us decrypt the second set of four letters; and so on.

1.2.4 Modern Stream Ciphers

There are a number of modern usable stream cipher algorithms.

These usually involve plugging key data into a pseudorandom number generator to generate a keystream.

There are two big weaknesses that limit the use of stream ciphers. In order to maintain security, a stream cipher needs to

- Use a different key for every message; and
- Produce a keystream that has a long period before repeating itself.

Most cryptography in use today uses other principles, which we will discuss later on in the course.

1.2.5 Binary encryption

We should conclude by discussing how this applies to encrypting computer data, which does not consist of Roman letters.

Computers encode data in *binary* strings of ones and zeroes. A *bit* is a single zero-or-one character; a *byte* is a string of eight bits and is the unit computers use to represent a single character of text.

Thus we can treat computers as working with an alphabet of two symbols. In this context monoalphabetic substitution is quite useless: either you change nothing, or you simply switch the ones and zeroes with each other. With only two possible keys, brute forcing is easy.

However, a polyalphabetic cipher like the Vigenère cipher works quite well. We can take as our key some binary string and add it (modulo 2) to our plaintext to get our ciphertext.

Remark 1.11. Adding bits modulo 2 is the same as the XOR operation, which returns 1 if its inputs are different, and 0 if they are the same.

Example 1.12. Suppose our key is 10010011 and we wish to encrypt the plaintext

01010000 01001111 01001011 01000101 00100000 00110101 00111001 00110100 00110101 00111000 00101100 00110110 00110010.

Adding our key to each byte, we get the output

11000011 11011100 11011000 11010110 10110011 10100110 10101010 10100111 10100110 10101011 10111111 10100101 10100001.

To decrypt, we simply subtract (or add, since they're the same thing modulo 2) our key back from our ciphertext to get the plaintext.

1.3 Statistical Models and Cryptanalysis

Last week we saw that we can break a monoalphabetic substitution cipher with statistical analysis and a little bit of hard work. But there are other ciphers, like the Vigenère cipher, that we can't break easily. But how can we tell which situation we're in? How can we tell if we're looking at a monoalphabetic cipher or a polyalphabetic one, without something already knowing? One answer comes from the index of coincidence.

1.3.1 The Index of Coincidence

Definition 1.13. Let $\mathbf{s} = c_1 c_2 \dots c_n$ be a string of n letters. The *index of coincidence* of $mathbf{s}$ is denoted $\text{IndCo}(\mathbf{x})$ and is defined to be the probability that two randomly chosen characters in the string $\mathbf{s}$ are identical.

Proposition 1.14. *Let $\mathbf{s} = c_1 c_2 \dots c_n$ be a string of n, and let F_i be the frequency with which the letter i appears in the string $\mathbf{s}$. Then*

$$\text{IndCo}(\mathbf{s}) = \frac{1}{n(n-1)} \sum_{i=0}^{25} F_i(F_i - 1). \tag{1}$$

Proof. There are $\binom{n}{2} = \frac{n(n-1)}{2}$ different ways to select two letters at random from $\sim$.

Each letter i appears F_i times, so there are $\binom{F_i}{2} = \frac{F_i(F_i)-1}{2}$ ways to choose the letter i twice. Adding these ways for each letter, there are

$$\sum_{i=0}^{25} \frac{F_i(F_i-1)}{2} \tag{2}$$

ways to choose two of the same letter from the string.

The chance of two randomly chosen letters being identical is equal to the number of ways to choose two identical letters, divided by the total number of ways to choose letters. Thus we divide equation (2) by $\frac{n(n-1)}{2}$ and get the formula in equation (1). □

For any given string we can calculate the index of coincidence from the frequency table.

Example 1.15. In your homework, we encrypted the string $\mathbf{s}$ ="Rats live on no evil star". This message has twenty letters total; it has ten distinct letters, and each appears twice. Thus we have

$$\text{IndCo}(\mathbf{s}) = \frac{1}{20 \cdot 19} \cdot 10(2 \cdot 1) = \frac{1}{19} \approx 0.53.$$

We also looked at the string $\mathbf{t}$ = "A man a plan a canal panama". This message has 21 total letters, with 10 "a"s, 2 "m"s, 4 "n"s, 2 "p"s, 2 "l"s, and 1 "c". Thus we compute

$$\text{IndCo}(\mathbf{t}) = \frac{1}{21 \cdot 20}(10 \cdot 9 + 2 \cdot 1 + 4 \cdot 3 + 2 \cdot 1 + 2 \cdot 1 + 1 \cdot 0) = \frac{108}{420} \approx .257.$$

As we'll see, this index of coincidence is really unusually high. But this isn't really surprising; you probably noticed already that this text has a ridiculous number of "a"s in it.

(Notice also that the term from the "c" is $1 \cdot 0 = 0$. This makes sense because the odds of the "c" matching another letter in the text are in fact zero, since there's only one of them').

We can also calculate two very important and common values for the index of coincidence:

Proposition 1.16. *1. If* $\mathbf{s}$ *is a string of letters generated uniformly at random, then* $\text{IndCo}(\mathbf{s}) \approx .038$.

2. If $\mathbf{s}$ *is a string of letters with the frequencies common in written English, then* $\text{IndCo}(\mathbf{s}) \approx .068$.

Proof. 1. The letters are generated uniformly at random, so $F_i \approx \frac{n}{26}$ for each i. Thus we have

$$\begin{aligned}\text{IndCo}(\mathbf{s}) &\approx \frac{1}{n(n-1)} \sum_{i=0}^{25} \frac{n}{26}\left(\frac{n}{26} - 1\right) = \frac{1}{n(n-1)} \sum_{i=1}^{25} \frac{n^2}{26^2} - \frac{n}{26} \\ &= \frac{1}{n(n-1)}\left(\frac{n^2}{26} - n\right) = \frac{n/26 - 1}{n-1} \approx \frac{1}{26} \approx .038.\end{aligned}$$

We probably could have guessed this without working through the computation—if the letters are chosen randomly, the chances of two of them matching should indeed be about 1/26.

2. This is a straightforward if tedious calculation from the letter frequencies given in figure 1.1. I might write it up for here later.

□

Corollary 1.17. *If* $\mathbf{s}$ *is a string of letters corresponding to an English text encrypted by a simple (monalphabetic) substitution cipher, then we should expect* $\text{IndCo}(\mathbf{s}) \approx .068$.

Proof. A monoalphabetic substitution cipher is just a permutation of the alphabet; so while the frequency of individual letters is altered by applying a substitution cipher, the index of coincidence is not □

This gives us a way to test whether a string of letters was likely enciphered by a simple substitution cipher or not. We compute the index of coincidence of the string. If the index is close to .068 then we likely have a string encrypted with a monoalphabetic cipher. If the index is close to .038 then it is likely that our string is not encrypted monoalphabetically.

This test is especially useful when we proceed to break the Vigenère cipher in the next section.

1.3.2 Breaking the Vigenère cipher

Monoalphabetic ciphers are fairly simple to implement, but also quite easy to break. The Vigenère cipher is much harder to break; for three centuries it was known as "The Undecipherable Cipher". In 1854 Charles Babbage successfully broke it, and in 1863 Friedrich Kasiski published a method for breaking the Vigenère cipher.

Finding the Key Length Cryptanalysis of the Vigenère cipher proceeds in two steps. The first (and more difficult) step is to determine the length of the key. There are two basic approaches to this, but both use the same basic idea.

The low-tech way is to look for repeated strings in the ciphertext. If the same string appears in more than one place, it's likely (but not definite!) that it's the same plaintext string encrypted by the same part of the keyword, so the distances between these reoccurrences gives us information about possible keyword lengths.

A simple version of this is to displace the ciphertext by $2, 3, 4, \ldots$ places, and see which displacement generates the most coincidences—points where the displaced ciphertext is identical to the original.

Another variant is to look for places where the same trigram is repeated more than once in the ciphertext, and measure the offset or distance between them. Then find a number that is a factor of most (but not necessarily all) of thse offset numbers; that's probably the length of the keyword.

This method is called the *Kasiski Method* or the *Kasiski Test*, since it was first developed by Charles Babbage.

Example 1.18. Consider the ciphertext :

```
zpgdl rjlaj kpylx zpyyg lrjgd lrzhz qyjzq repvm swrzy rigzh
zvreg kwivs saolt nliuw oldie aqewf iiykh bjowr hdogc qhkwa
jyagg emisr zqoqh oavlk bjofr ylvps rtgiu avmsw lzgms evwpc
dmjsv jqbrn klpcf iowhv kxjbj pmfkr qthtk ozrgq ihbmq sbivd
ardym qmpbu nivxm tzwqv gefjh ucbor vwpcd xuwft qmoow jipds
fluqm oeavl jgqea lrkti wvext vkrrg xani
```

First, we can compute that the index of coincidences for this ciphertext is about .039. This suggests it wasn't encrypted with a monoalphabetic substitution cipher.

We think that it was encrypted with a Vigenère cipher, and we want to find the key length. The first method is to look for coincidences. To do that we can lay out the lines of ciphertext shifted. So the first line would be:

```
0:zpgdl rjlaj kpylx zpyYg lrjgd lrzhz qyjzq repvm swrzy rigzh
1: zpgd lrjla jkpyl xzpYy glrjg dlrzh zqyjz qrepv mswrz yrigz h
```

1 coincidence

```
0:zpgdl rjlaj kpylx zpyyg lrjgd lrzhZ qyjzq repvm swrzy rigzh
2:  zpg dlrjl ajkpy lxzpy yglrj gdlrZ hzqyj zqrep vmswr zyrig zh
```

1 coincidence

```
0:zpgdl rjLaJ kpylx zpyyg lrjgd lrzhz qyjzq repvm swrzy Rigzh
3:   zp gdLrJ lajkp ylxzp yyglr jgdlr zhzqy jzqre pvmsw Rzyri gzh
```

3 coincidences

```
0:zpgdl rjlaj kpylx zpyyg lrjGd lrzhz qyjZQ repvm swrzy rigzh
4:    z pgdlr jlajk pylxz pyyGl rjgdl rzhZQ yjzqr epvms wrzyr igzh
```

3 coincidences

```
0:zpgdl rjlaj kpylx zPYyg lrjgd LRzhz qyjzq repvm swrzy rigZh
5:       zpgdl rjlaj kPYlx zpyyg LRjgd lrzhz qyjzq repvm swrZy rigzh
```

5 coincidences

```
0:zpgdl rjlaj kpyLx zpyYg lrjgd lrzhz qyjZq repvm swrzy rigzh
2:        zpgd lrjLa jkpYl xzpyy glrjg dlrZh zqyjz qrepv mswrz yrigz h
```

3 coincidences

```
0:zpgdl rjlaj kpylx zpyyg lrjgd lrzhz qyjzq repvm swrzy rigzh
2:         zpg dlrjl ajkpy lxzpy yglrj gdlrz hzqyj zqrep vmswr zyrig zh
```

2 coincidences

From this we might guess that the keyword has length five; but this is a small sample size.

If we do this count for the entire ciphertext (preferably but not necessarily by computer), we get:

Shift	1	2	3	4	5	6	7	8	9
Coincidences	6	6	9	5	8	13	15	11	11

This suggests but does not prove that the keyword has length 7.

If we look at repeated trigrams instead, we get the following results:

Trigram	Places	Offset	Trigram	Places	Offset
`avl`	117 and 258	$141 = 3 \cdot 47$	`bjo`	86 and 121	$35 = 5 \cdot 7$
`dlr`	4 and 25	$21 = 3 \cdot 7$	`gdl`	3 and 24	$16 = 2^4$
`lrj`	5 and 21	$98 = 2 \cdot 7^2$	`msw`	40 and 138	$84 = 2^2 \cdot 3 \cdot 7$
`pcd`	149 and 233	$13 = 13$	`qmo`	241 and 254	$98 = 2 \cdot 7^2$
`vms`	39 and 137	$84 = 2^2 \cdot 3 \cdot 7$	`vwp`	147 and 231	$84 = 2^2 3 \cdot 7$
`wpc`	148 and 232	$21 = 3 \cdot 7$	`zhz`	28 and 49	$21 = 3 \cdot 7$

We see that the factor 7 appears often in the offset column; thus the keyword is probably length 7. This matches our tentative conclusion from earlier.

The higher-tech version of this is to use the index of coincidence again. The index of coincidence only pays attention to *letter* distributions, and doesn't care about their order. So if we take all the letters that are encrypted by the same letter of the keyword, and calculate the index of coincidence, we should get a number close to .068; if not, we should get an index closer to .038.

To run this test, we break our ciphertext into k pieces: the first has letters $1, k+1, 2k+1, \ldots$, the second has letters $2, k+2, 2k+2, \ldots$, and so on. We compute the index of coincidence for each of these strings. If most of them are close to .068 then the keyword is probably of length k; if most of them are close to .038 then the keyword is probably not of length k.

Example 1.19. Continuing the look at our previous ciphertext, we can compute the indices of coincidence for each substring, for each possible shift.

Shift	indices							
2	.038	0.40						
3	0.39	0.42	0.38					
4	0.34	0.42	0.39	0.35				
5	0.38	0.39	0.43	0.28	0.36			
6	0.38	0.40	0.39	0.38	0.32	0.33		
7	0.62	0.57	0.65	0.60	0.60	0.64	0.64	
8	0.37	0.29	0.38	0.33	0.34	0.57	0.40	0.39

One of these rows looks very unlike the others; this again gives evidence that the key length is 7.

Finding the Key Once we have found the key length, there are two basic approaches we can use to finding the key.

First, we can simply do a frequency analysis on subsets of the ciphertext. If we believe that the key has length 5, then the first, sixth, eleventh, etc. letters are all shifted by the same amount. So we can do a frequency analysis on this set to decipher the shift.

Note that the frequency analysis is made much easier by the fact that we know we're working with a Caesar cipher, so there are only 26 possibilities. Thus we're trying to select a shift that makes *all* the high-frequency ciphertext letters into high-frequency plaintext letters.

Example 1.20. In the previous example, we've now seen that the key length is seven. So we can look at the substring $\mathbf{s}_1$ consisting of the 1st, 8th, 15th, ... letters, which is `zlxrh rrhwl oehdw eokli lwvlh phqby nwhwf julrx x` which has frequency counts

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Frequency	0	1	0	1	2	1	0	6	1	1	1	6	0	1	2	1	1	4	0	0	1	1	5	3	1	1

We see the frequent letters here are H, L, and less so W, R, and X in that order. We might guess that either H or L corresponds to a plaintext e. If H corresponds to e, that sends L to i, R to o, W to t, and X to u, which isn't unreasonable; if L corresponds to e, then that sends H to a, R to k, W to p, and X to q, which seems somewhat less likely, since we don't expect k and q to be among the most common letters in an English string.

This isn't dispositive, but it is highly suggestive that the first letter of the keyword corresponds to a shift three to the right; thus the first letter might be D. We can repeat this process for each substring.

This works, but involves a lot of guesswork and intuition and puzzle-solving. There is a second method that requires a bit more calculation, but is also rather more robust and automatic.

Definition 1.21. Let $\mathbf{s} = c_1c_2\dots c_n, \mathbf{t} = d_1d_2\dots d_m$ be two strings of letters. Then we define the *mutual index of coincidence* to be $\text{MutIndCo}(\mathbf{s}, \mathbf{t})$, the chance that a randomly selected letter of $\mathbf{s}$ is the same as a randomly selected letter of $\mathbf{t}$.

Proposition 1.22. *Let $\mathbf{s} = c_1c_2\dots c_n, \mathbf{t} = d_1d_2\dots d_m$ be two strings of letters, and let $F_i(\mathbf{s})$ be the number of times the ith letter appears in the string $\mathbf{s}$. Then:*

1.

$$\text{MutIndCo}(\mathbf{s},\mathbf{t}) = \frac{1}{nm}\sum_{i=0}^{25} F_i(\mathbf{s})F_i(\mathbf{t}).$$

2. *If the letters of* $\mathbf{s}$ *and* $\mathbf{t}$ *are drawn from the same distribution, given by taking English frequencies and permuting the letters, then* $\text{MutIndCo}(\mathbf{s},\mathbf{t}) \approx .068$.

3. *If the letters of* $\mathbf{s}$ *and* $\mathbf{t}$ *are drawn from different such distributions, then* $\text{MutIndCo}(\mathbf{s},\mathbf{t}) \approx .038$.

We can use the mutual index of coincidence to test if two strings are drawn from the same substitution. To decrypt a Vigenere cipher, we need to figure out the shift of each substring. We can use the mutual index of coincidence to compute the *relative* shifts.

Let $\mathbf{s_i} = c_i, c_{i+k}, c_{i+2k}, \dots$, and define $\mathbf{s}_i + \sigma$ to be the string $\mathbf{s}_i$ with each letter shifted by σ. Then we can compute $\text{MutIndCo}(\mathbf{s}_i, \mathbf{s}_j + \sigma)$ for $0 \leq \sigma \leq 25$. We expect 25 of these computations to be low like .038, and the last to be high like .068; this last one gives us the relative shift between the ith and jth letter of the keyword.

Thus if β_i is the ith letter of the keyword, this does not and cannot tell us what β_i is; but it can give us relationships among the β_i.

After computing $k-1$ of these, with luck we should know the relative shifts of all the letters of the keyword; this means there are only 26 possible keys, and we can try all of them. Of course, sometimes the mutual index of coincidence happens, randomly to not give enough information; this is a problem we can solve by simply computing more possible mutual indices, possibly up to all $\frac{k(k-1)}{2}$ possibilities.

Example 1.23. If we compute all the possible mutual indices of coincidence in our ciphertext, for a key length of seven, then we get the following "large" indices that indicate a true collision:

i	j	σ	MutIndCo$(i, j+\sigma)$	Relative shift equation
1	3	1	.067	$\beta_1 - \beta_3 = 1$
3	7	10	.069	$\beta_3 - \beta_7 = 10$
1	4	19	.071	$\beta_1 - \beta_4 = 19$
1	6	16	.071	$\beta_1 - \beta_6 = 16$
3	4	18	.073	$\beta_3 - \beta_4 = 18$
3	5	24	.067	$\beta_3 - \beta_5 = 24$
3	6	15	.074	$\beta_3 - \beta_6 = 15$
4	6	23	.066	$\beta_4 - \beta_6 = 23$
4	7	18	.071	$\beta_4 - \beta_7 = 18$
6	7	21	.069	$\beta_6 - \beta_7 = 21$

Our goal is to solve the equations in the right-hand column—or at least as many as possible! It's entirely possible that one of the high indices will be an accident; when this happens, you can try dropping one constraint or another and see what you get.

But in this case, the system is fairly straightforward to solve. We get:

$$\beta_3 = \beta_1 + 25 \qquad \beta_4 = \beta_1 + 7$$
$$\beta_6 = \beta_1 + 10 \qquad \beta_7 = \beta_3 + 16 = \beta_1 + 15$$
$$\beta_5 = \beta_3 + 2 = \beta_1 + 1$$

and we can check that this doesn't generate any inconsistencies. Notice that we don't have a solution for β_2 in here, because we didn't get any high indices of coincidence involving $\mathbf{s}_2$. One option is to take the best ones we have; the highest is MutIndCo$(2, 4+24) = .061$, which suggests $\beta_2 = \beta_4 + 24 = \beta_1 + 5$. The other is to look at the results not involving β_2 and basically guess.

So what do our results give us? Well, they tell us that if we know β_1, we know the entire keyword. For instance, if $\beta_1 = 0 = A$, then the keyword is AFZHBKP. If we try decrypting our ciphertext with this word (by *subtracting* it from the ciphertext), we get `zkhwkhulvkdoowxuq...` which isn't very promising.

But recall that there are now only 26 possible keys, so we can simply try each of them. If we do that, we get a table something like this:

β_1	Keyword	Potential plaintext
0	AFZHBKP	`zkhwkhulvkdoowxuq`
1	BGAICLQ	`yjgvjgtkujcnnvwtp`
2	CHBJDMR	`xifuifsjtibmmuvso`
3	DICKENS	`whetherishallturn`
4	EJDLFOT	`vgdsgdqhrgzkkstqm`
5	FKEMGPU	`ufcrfcpgqfyjjrspl`
6	GLFNHQV	`tebqebofpexiiqrok`

We see that when $\beta_1 = 3$, the keyword is "DICKENS", and we get recognizable English out of the ciphertext: we get

```
wheth erish alltu rnout tobet heher oofmy ownli feorw hethe rthat stati onwil
lbehe ldbya nybod yelse these pages musts howto begin mylif ewith thebe ginni
ngofm ylife ireco rdtha tiwas borna sihav ebeen infor medan dbeli eveon afrid
ayatt welve ocloc katni ghtit wasre marke dthat thecl ockbe ganto strik eandi
began tocry simul taneo usly
```

and after replacing spacing and punctuation, we get:

"Whether I shall turn out to be the hero of my own life, or whether that station will be held by anybody else, these pages must show. To begin my life with the beginning of my life, I record that I was born (as I have been informed and believe) on a Friday, at twelve o'clock at night. It was remarked that the clock began to strike, and I began to cry, simultaneously."

Remark 1.24. These sorts of attacks can totally work on a Vigenère cipher applied to binary messages. But you have to rely entirely on things more like trigram coincidences and less like single-letter coincidences, since there are only two "letters".

1.3.3 The Autokey cipher and cribs

Using a crib One common tool in cryptanalysis is a *crib*, which is a known or guessed bit of plaintext corresponding to a ciphertext. (The term comes from the phrase "to crib notes" or "to crib an answer", meaning to copy or cheat on an assignment).

Often a crib can be used to dramatically simplify cryptanalysis. (In fact, frequency analysis is essentially an attempt to imitate a crib).

Cribs were famously used in Bletchley Park during World War II (where the term was coined). Many German Enigma operators used standardized terminology, including the regular use of the word *Wetter* ("weather") in weather reports, and one operator who repeatedly transmitted the message "Nothing to report".

Enigma operators were required to spell out all numbers, so Turing determined that the single most common word in messages was *eins*, meaning "one". Turing precomputed a catalog of what *eins* would look like encrypted in every possible position with various keys, which dramatically sped up decryption processes by seeing which of those were possible and judging them most likely.

You will notice that this is basically the same idea as frequency analysis: instead of taking common letters, we instead look for common words. *eins* was not enough to break messages on its own, but it could give substantial speedups and hints for other encryption messages.

Breaking the Autokey cipher Cribs are an especially powerful tool in breaking the Autokey cipher, since the plaintext is *also* most of the keystream.

The basic idea is that we guess a word or phrase we expect to see in the plaintext. Since that word would also have to appear in the keystream, we try using that word as the key at every possible point, and see when the results are plausible English strings. We can extend this out to guess most or all of the message.

Example 1.25. Suppose we intercept the ciphertext:

`OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW`

We guess that it contains the word "the" somewhere. So we simply see what happens if we use "the" as the key at every possible position.

First we decrypt everything with an offset of zero:

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

A couple of these strings look like they might be English; we might notice "tim" or "aso". Let's see what happens if we assume "tim" is actually part of the plaintext. We now need to see what happens if we assume the original keyword is any of various lengths.

If the keyword has length four, we get

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- --- --- --- --f bn- the -as o-- --- --- --- --- --- --- --- --- ---

--- --- --- --- --- --t he- aso -gu s-- --- --- --- --- --- --- --- --- ---
```

and while "gus" is possible, "fbn" probably is not.

If the keyword has length five, we get:

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- --- --- --- -er e-- the --a so- --- --- --- --- --- --- --- --- ---

--- --- --- --- --- -th e-- aso --m ob- --- --- --- --- --- --- --- --- ---
```

This looks more promising, but if we continue building out we get:

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- ono --m di- -er e-- the --a so- -mo b-- auo --- --- --- --- --- ---

--- --- mdi --e re- -th e-- aso --m ob- -au o-- ncj --- --- --- --- --- ---
```

which looks unlikely. With a key of length six we get

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- --- --- --- gqu --- the --- aso --- --- --- --- --- --- --- --- ---

--- --- --- --- --- the --- aso --- gxw --- --- --- --- --- --- --- --- ---
```

Which again doesn't look like English. We can try longer keywords but nothing useful will show up. Similarly, we can try working with the "tim" but we won't really get anywhere.

So is this all the possibilities? No: the table we made started with "the" with an offset of zero from the ciphertext. We need to try again with offsets of one and two. If we build the table with an offset of two, we get

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

We don't see anything terribly promising until we see past the wraparound; then we notice that the putative plaintext has "est" in it. So let's try assuming that's correct. With a keyword length of four we get

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- --- -wj q-t he- est --- --- --- --- --- --- --- --- --- --- --- ---

--- --- --- -th e-e st- ezr --- --- --- --- --- --- --- --- --- --- --- ---
```

which doesn't look like English. So we try a key length of five:

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --- --- tim --t he- -es t-- --- --- --- --- --- --- --- --- --- --- ---

--- --- --- the --e st- -ns a-- --- --- --- --- --- --- --- --- --- --- ---
```

which looks like it could work. So we expand out another step, keeping in mind that we're guessing the keyword is length five so our keystream doesn't actually go earlier than the sixth character:

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
--- --s o-- tim --t he- -es t-- nsa --- --- --- --- --- --- --- --- --- ---

so- --i m-- the --e st- -ns a-- com --- --- --- --- --- --- --- --- --- ---
```

This still looks good, so we fill out the rest:

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
wa- --s o-- tim --t he- -es t-- nsa --c om- -ic a-- dan --h ea- -we r-- res

so- --i m-- the --e st- -ns a-- com --i ca- -da n-- hea --w er- -re s-- ple
```

At this point we just need to find any part of the message where we can make a guess to fill in the blanks, and we're done. We can try a few things, but perhaps we notice that the last part is "s– ple", which might be "sample" or "simple". Guessing "sample" gives us

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
wa- --s o-- tim --t he- -es t-- nsa --c om- -ic a-- dan --h ea- -we raa res
so- --i m-- the --e st- -ns a-- com --i ca- -da n-- hea fsw era are sam ple
```

which doesn't work very well; but guessing "simple" and working backwards gives us

```
OOF IKA AQW MPQ UMX ZXY IRK TZS PGM GPK QMI PLC NWX KEN QLD IRF SNI JAM GPW
wat ers ome tim est heq ues tio nsa rec omp lic ate dan dth ean swe rsa res

som eti mes the que sti ons are com pli cat eda ndt hea nsw ers are sim ple
```

"Sometimes the questions are complicated, and the answers are simple."