1 Introduction to Cryptography

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.

1.1.1 Steganography

Steganography 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.

1.1.2 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.

1.1.3 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.

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

ALICE SENDS A MESSAGE TO BOB SAYING TO MEET HER SOMEWHERE.
UH HUH.
BUT EVE SEES IT, TOO, AND GOES TO THE PLACE.
WITH YOU SO FAR.
BOB IS DELAYED, AND ALICE AND EVE MEET.
YEAH?
I'VE DISCOVERED A WAY TO GET COMPUTER SCIENTISTS TO LISTEN TO ANY BORING STORY.

https://xkcd.com/1323

shouldn't be able to perform the same conversion to acquire (or, in more malicious scenarios, replace) the plaintext.

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?"

1.2.1 Modular arithmetic

Recall from Discrete Math 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".

1.3 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, and use statistical properties of the ciphertext to determine the substitution. We'll discuss the details of this next week.

1.4 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.

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 1.8.

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.

1.4.1 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 1.9. 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 2.1. 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 2.2.

1.4.2 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 1.10. 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 1.11. 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.

1.5 Other Stream 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.

1.5.1 Autokey ciphers

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.