Data Representation and Cryptography

Data

01001000 01100101 01101100 01101100 01101111, or for those of us that aren't computers, hello! That series of numbers is called binary code and it's the language your computer speaks. In fact, all the information stored on your computer is recorded as a series of 1s and 0s, but how does it work?

Learning Objectives

At the end of this module you will be able to:

1. Explain the process of encoding
2. Define Unicode
3. Define cryptography
4. Explain how math is used for cryptography
5. Demonstrate how to decrypt a message using the affine cypher

Characters

In order to be able to store words, we have to represent the individual characters that make them up.

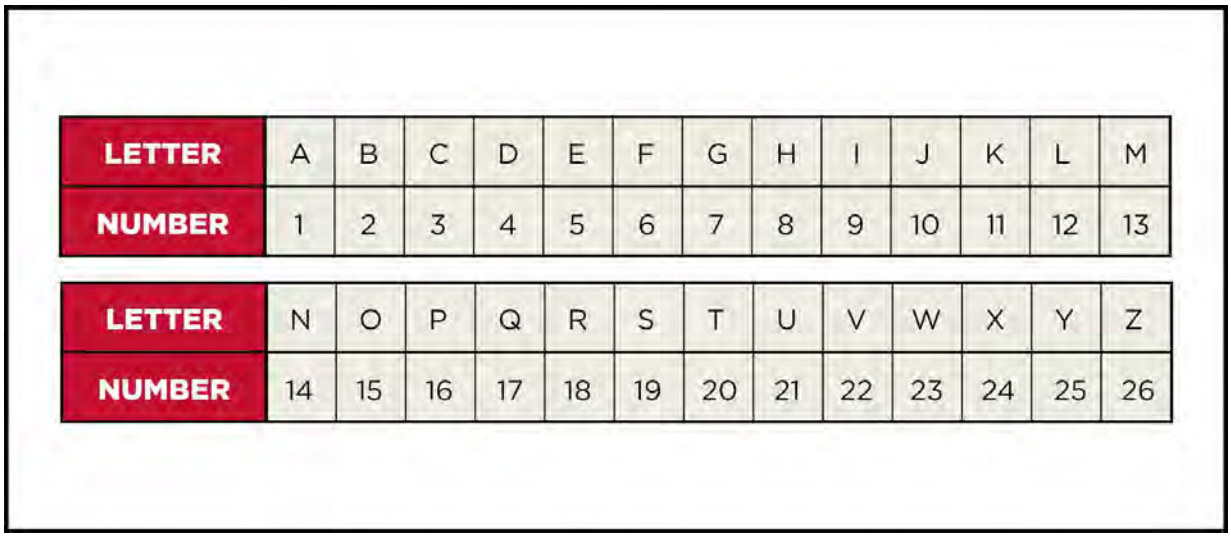
Characters can be:

- Alphabetic: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Numerical: 0123456789
- Punctuation: ,.?:;!'"
- Lots of others: []{}()+-=/%$

So how do we represent them in a computer?

Representing Characters Using Encoding

Since we already have a good system for storing numbers inside a computer (binary code), we can represent each character with a number. One way we could do this would be to represent each letter of the alphabet with a number 1-26, because there's 26 letters. This is called **encoding**.

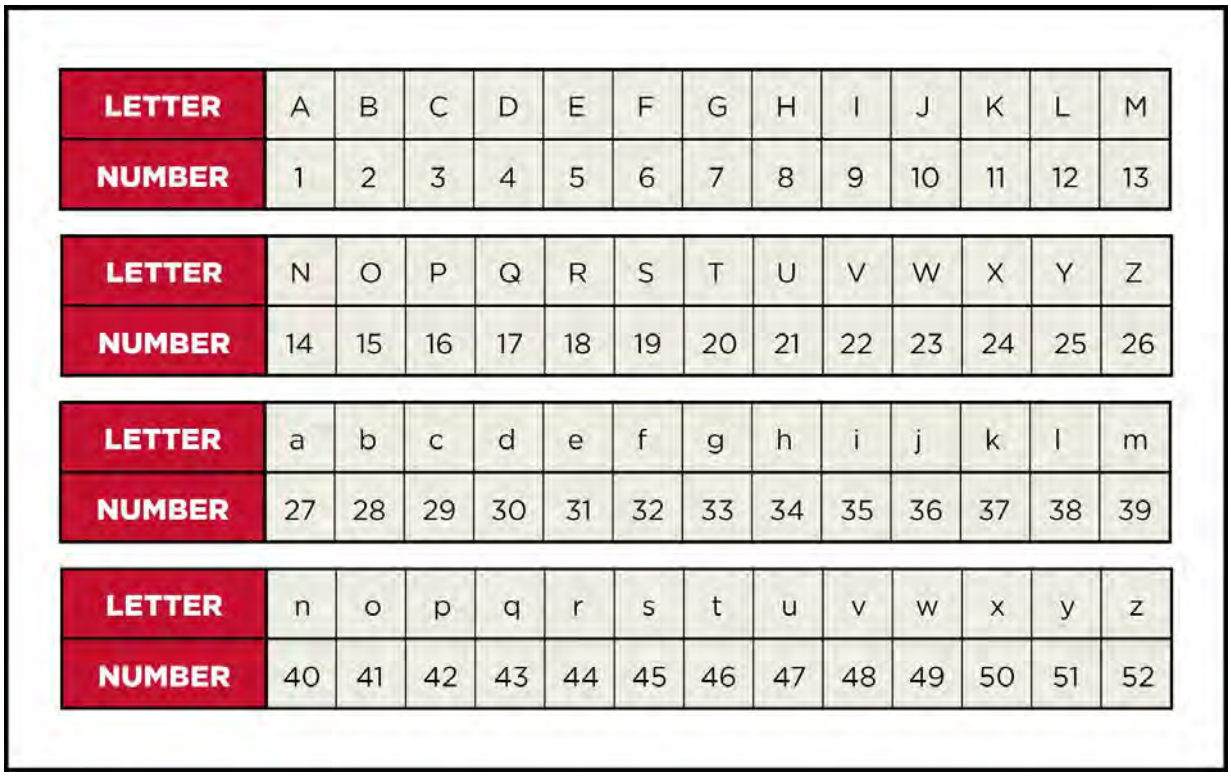


LETTER	A	B	C	D	E	F	G	H	I	J	K	L	M
NUMBER	1	2	3	4	5	6	7	8	9	10	11	12	13

LETTER	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
NUMBER	14	15	16	17	18	19	20	21	22	23	24	25	26

A table displaying all the uppercase letters of the alphabet, and a corresponding number.

While this way of representing numbers works, what if we wanted to represent upper and lowercase letters? We need a unique number for every character, so we would need to keep going past 26...



LETTER	A	B	C	D	E	F	G	H	I	J	K	L	M
NUMBER	1	2	3	4	5	6	7	8	9	10	11	12	13

LETTER	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
NUMBER	14	15	16	17	18	19	20	21	22	23	24	25	26

LETTER	a	b	c	d	e	f	g	h	i	j	k	l	m
NUMBER	27	28	29	30	31	32	33	34	35	36	37	38	39

LETTER	n	o	p	q	r	s	t	u	v	w	x	y	z
NUMBER	40	41	42	43	44	45	46	47	48	49	50	51	52

A table displaying all the uppercase and lowercase letters of the alphabet, and a corresponding number.

How Big Does The Table Need To Be?

If our goal is to be able to store every character needed to write English, we not only need to store uppercase and lowercase letters, we also need numbers, punctuation, and other special characters.

But English isn't the only language in the world. What about:

- Accented characters used in many European languages: ÀÁÂÃÄÅ
- The Greek alphabet: Ελληνικό αλφάβητο
- The Arabic alphabet: عربية أبجدية
- The Devanagari script used for many Indian languages: दवनागरी
- Tens of thousands of Chinese and Japanese characters: 漢字
- And many other languages!

Unicode to the Rescue!

Unicode is essentially a giant table for *all* the world's characters. The Unicode Standard is the result of lots of people working together to find a number for every character we want to use on computers. Right now, the table has 144,697 characters in 159 different alphabets from all over the world, and is still growing! The Unicode Consortium keeps adding new characters and symbols, with a new version of Unicode out about every year.

So what are the *real* numbers of the English alphabet? Not 1 through 26! In fact, Unicode Standard actually uses 65-90 for uppercase letters and 97-122 for lowercase letters. The reason why they have these numbers is because they got their numbers from a previous standard that made some odd choices.

LETTER	A	B	C	D	E	F	G	H	I	J	K	L	M
NUMBER	65	66	67	68	69	70	71	72	73	74	75	76	77

LETTER	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
NUMBER	78	79	80	81	82	83	84	85	86	87	88	89	90

Cryptography

What is cryptography? Simply put, it's the science of secure communications techniques. While cryptography has been around since 1900 BCE, modern cryptography is used to make the internet much more secure.

Think about all the private information you put in your computer when you buy something online. All that information has to be shared with the retailer in order for you to complete your purchase, so what's stopping someone from intercepting that information and stealing it? Encryption! When someone buys something online, the messages they send with their name or credit card number are encrypted so that only the computer they're talking to can read them. That computer decrypts the message, making it readable again, and foiling any would be cyber thieves.

Using Math For Cryptography

Remember when we encoded all those letters as numbers? Well we can use math to transform those numbers. For example, we could add, subtract, or multiply the numbers using special formulas that only we know. This process is known as encryption, and those special formulas are called the key.

Affine Cypher

The affine cypher is a simple kind of cryptography that transforms a number into an encrypted version using a simple algebraic formula.

The input is a number *x*, which is a number that represents a letter. The output *ax+b*, where *a* and *b* are the numbers that make up the key.

Key

The variable A is the first part of the key that will encrypt the input.

Key

The variable **B** is the second part of the key that will encrypt the input.

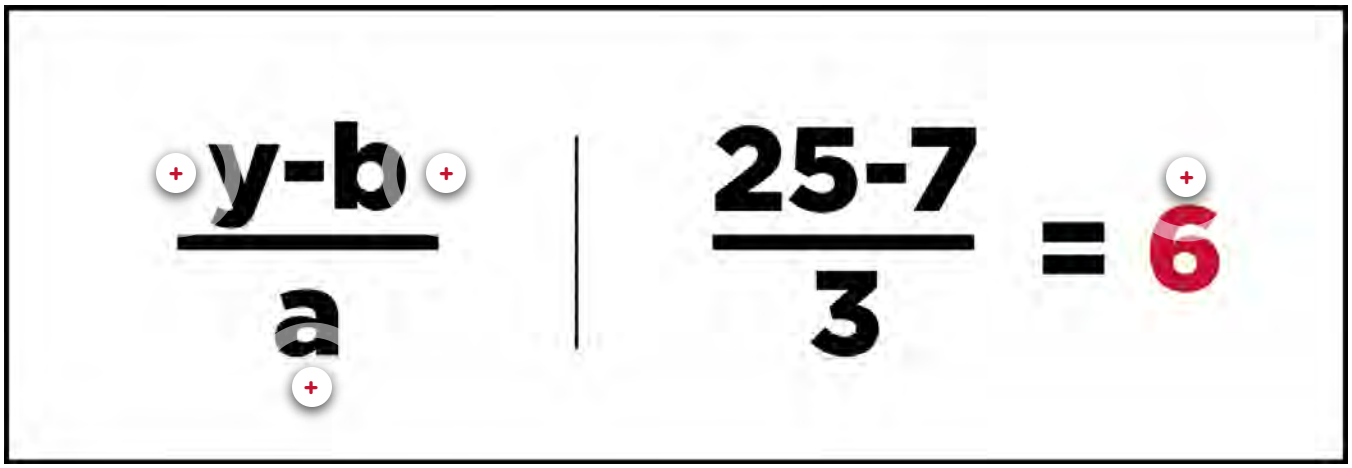
Input

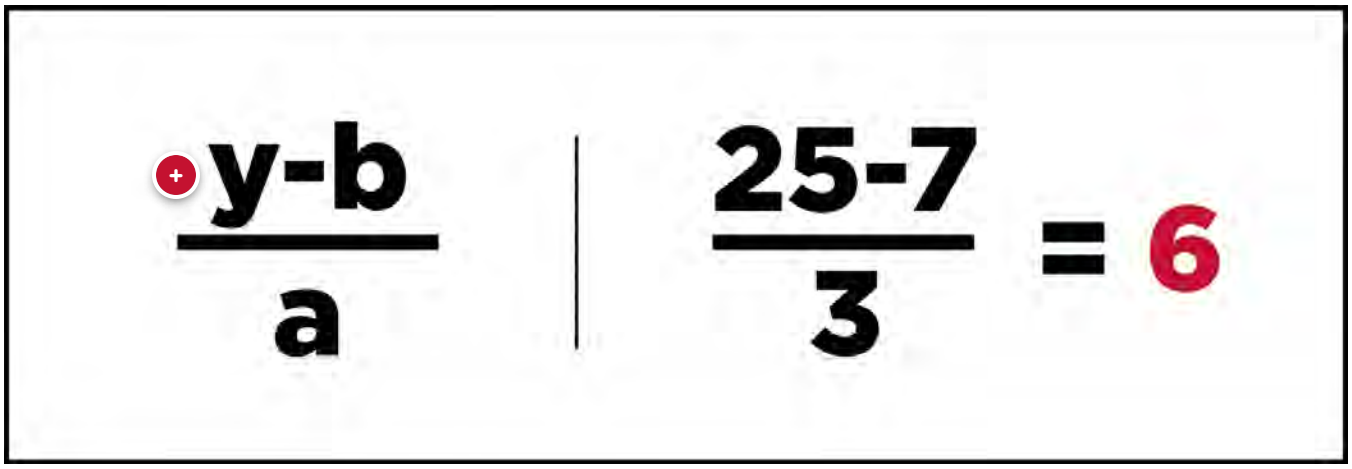
X is the input, or the number you want to encrypt

For example, if a = 3 and b =7, the number 6 would be encrypted as "3 * 6 + 7 = 25"

Decryption With The Affine Cypher

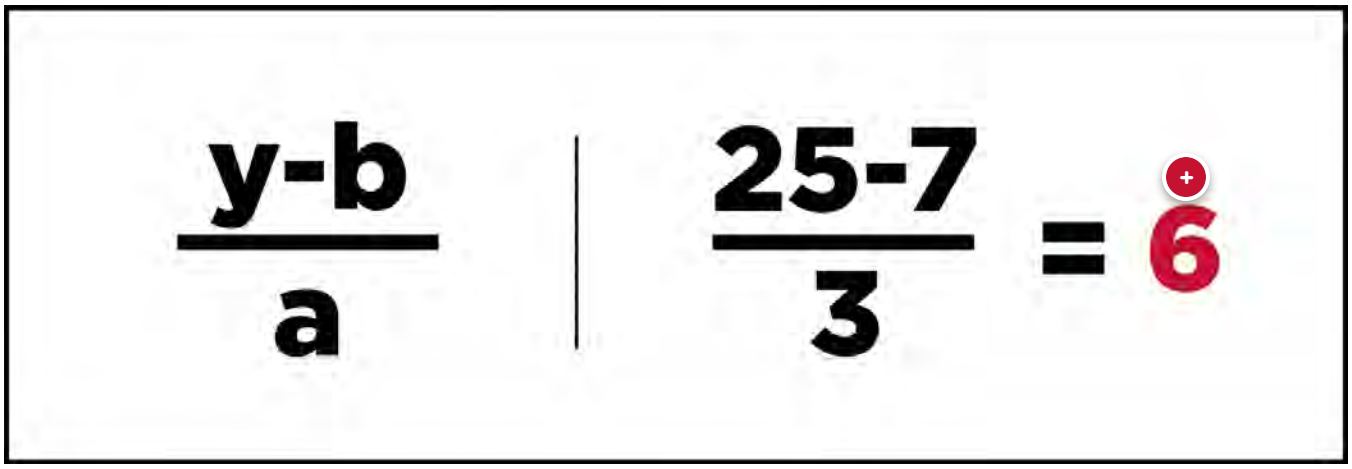
Now that we know how to encrypt, now we need to learn how to decrypt. Decryption is the process of reversing the operation done by encryption using algebra. The formula for decrypting the affine cypher looks like this:





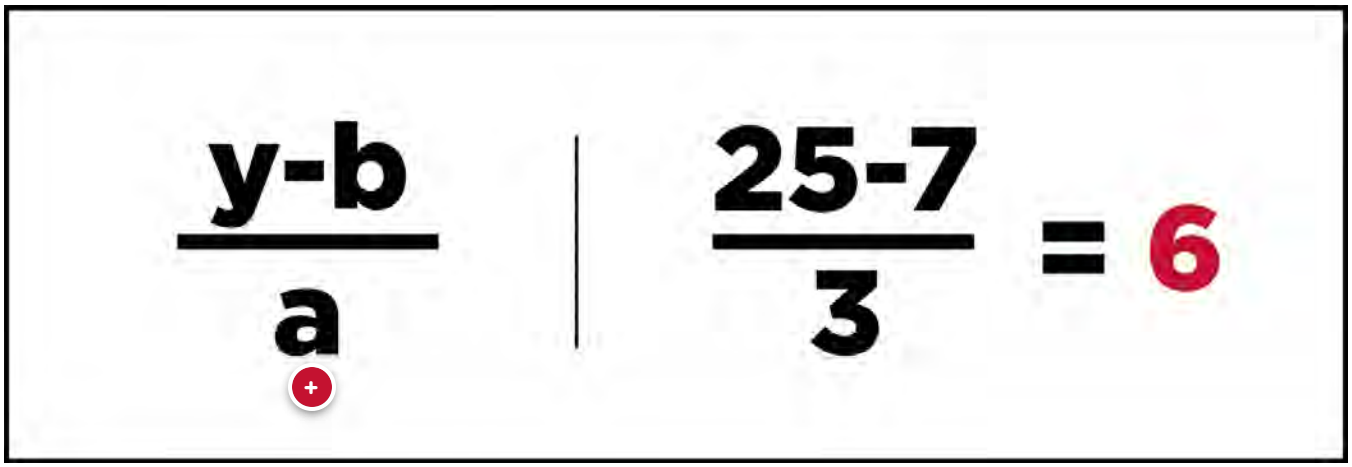
The Encrypted Number

y is the encrypted number we got from following the encryption formula



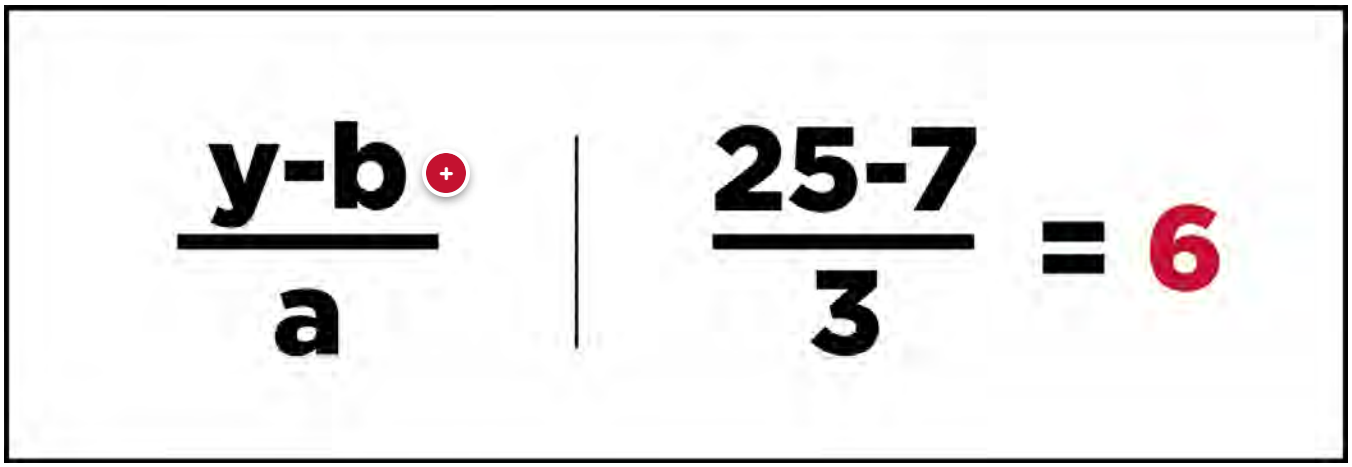
The Original Value

Using our previous example, if $a = 3$ and $b = 7$, we can decrypt the number 25, which will bring us back to our original number, 6!



Part of the Key

a is the other part of the key used to encrypt *y*



Part of the Key

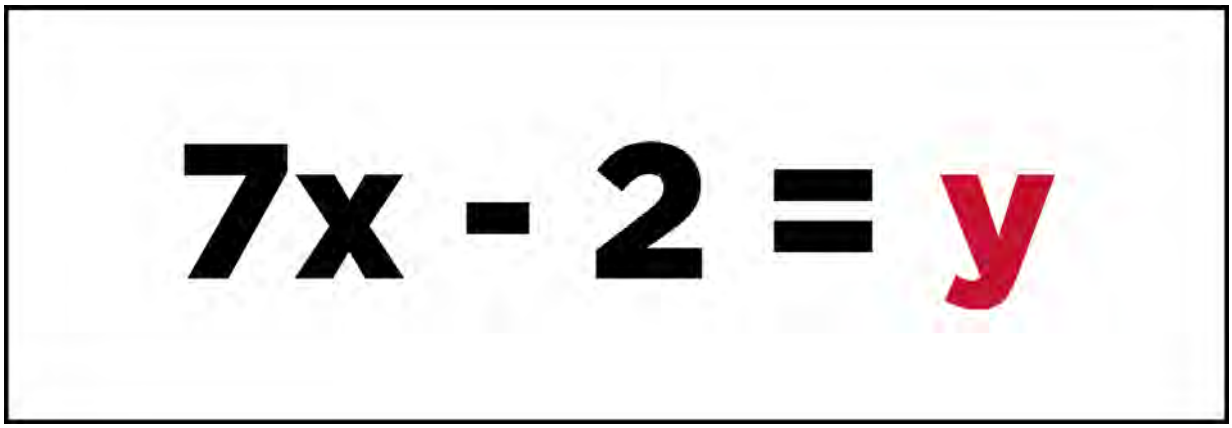
b is part of the key used to encrypt *y*

Affine Cypher Encryption Example

Let's try it out! Consider the message "MATH IS ASTONISHING". If we want to encrypt this message we first need to encode these letters. To keep things simple, let's use the encoding where A through Z map to 1-26. Once the letters are encoded, they will look like this:

13 1 20 8 9 19 1 19 20 15 14 9 19 8 9 14 7

Next, we need to figure out what our key will be. In this example, we will use $a = 7$ and $b = -2$. When we put it all together, our formula will look like this:



7x - 2 = y

In other words, we multiply each number by 7 and then subtract 2. Once we process every number we should end up with:

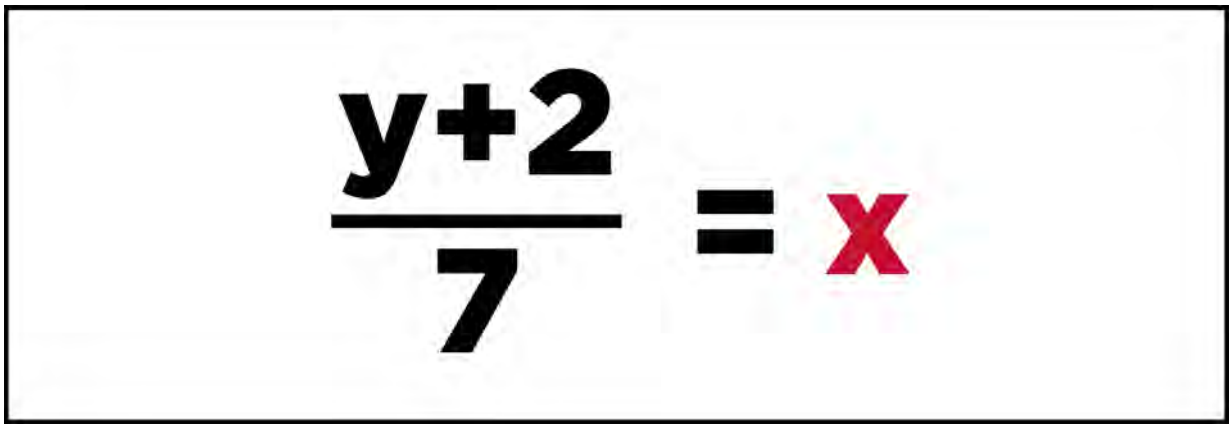
89 5 138 54 61 131 5 131 138 103 96 61 131 54 61 96 47

Affine Cypher Decryption Example

Now that we've tried encrypting numbers, lets try decrypting some. Lets say that we have the following encrypted result:

89 5 138 54 61 131 5 131 138 103 96 61 131 54 61 96 47

In order to decrypt the message, we will need to know the key. Fortunately for us, we know that it's a = 7 and b = -2. If we plug everything in, our decryption formula will look like this:



If everything goes to plan, the result will be the original numbers:

13 1 20 8 9 19 1 19 20 15 14 9 19 8 9 14 7

Knowing that A through Z map to 1-26, the original message can be reconstructed as:

MATHISASTONISHING

Affine Cypher Exercise

Here are some exercises to help you practice using the affine cypher!

1. Pick your own message of all uppercase letters.
2. Change the message into a list of numbers by mapping the letters A through Z to the numbers 1-26.
3. Pick a value for *a* and another value for *b* in order to construct your key. Both numbers should be integers, and can be both negative or positive. Just don't pick 0!

4. Encrypt the message by multiplying each encoded letter by *a* and adding *b*.

5. Trade the results with your neighbor and decrypt each other's messages. Make sure you include the encrypted numbers as well as the values of *a* and *b*, otherwise they'll never be able to solve it!

If you're feeling even more adventurous, you can try the same exercise with the letters encoded in Unicode. The numbers are going to be a lot bigger so a calculator might be useful!

Security

We encrypt things in order to protect information, but how good is the affine cypher? Unfortunately, the affine cypher is old and no longer used due to the fact that it is no longer secure. One fault of the affine cypher is that letters that come later in the alphabet will always have larger values for positive values of *a* and smaller values for negative values of *a.* Another fault is that it is a monoalphabetic substitution cypher. This means that the same letter will always be encrypted as the same number. For example, if I wanted to encrypt the word "see", both "e's" will always be the same number. Lastly, any monoalphabetic substitution cypher is vulnerable to a frequency attack, which uses the fact that some letters in English (like E) occur more often, while other letters (like Q) are rare.

Knowledge Check

A quick knowledge check on data representation and cryptography

Question

01/03

What is the process of assigning letter numbers to store them on a computer

---

- Encrypting
- Encoding
- Decrypting
- Transferring

Question

02/03

Select all options that are true about Unicode

- [ ] Unicode is a giant table that can be used to encode all the world's characters
- [ ] Unicode only stores English characters
- [ ] Unicode no longer adds new characters
- [ ] Unicode contains letters from 159 alphabets

Question

03/03

Which choice best defines cryptography

- The study of secure communications techniques.
- The study of graveyards
- The study of the internet