

## How it works

- A substitution **cipher** uses the same **alphanumeric** and punctuation **characters** in both **plaintext** and **ciphertext**.
- To **encipher** a message, each plaintext character's alphabetical position is shifted (translated) to the right or left. The character at that new position becomes the ciphertext character.
- To **decipher** a ciphertext, the translation process is reversed.
- The same key must be used to encipher and decipher a message.
- The table below shows the characters' positions in the alphabet with two punctuation marks.

character	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	space	!
position	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	26	27	28

- Example: encipher the character “H” in the plaintext “HELLO” using a key = 3 and a right shift:
  - a. Locate ‘H’; it is the 8<sup>th</sup> character in the table.
  - b. Add the key of 3 to the position;  $8 + 3 = 11$
  - c. Locate position 11; it is the character ‘K.’
  - d. The first letter of the ciphertext is ‘K.’
  - e. If you come to the end of the table, continue counting from the beginning.
  - f. Repeat the previous steps for the remaining characters; the result is the ciphertext ‘KHOOR’.
- A substitution cipher can be cracked performing a **character frequency analysis** of the the ciphertext. In common plaintext writing the characters of the alphabet each have a frequency on average in large text. In the English language a space (**ASCII 32**) is most common character and ‘E’ is most the most common letter. Since ‘E’ is the 5<sup>th</sup> letter in the alphabet, by counting the frequency of each character in the ciphertext, the key can be deduced. For example if the most frequent character in the ciphertext is ‘K’, then there is a good chance, ‘K’ (position 11) is being substituted for ‘E’ (position 5) and the shift key is 6 because  $11 - 5 = 6$ . This type of frequency analysis requires a large ciphertext to assume the most frequent ciphercharacter code for ‘E’.

## What will you do?

1. Practice Caesar ciphering:
  - a. using a right-shift and key = 6 along with the table above. Fill in the blank ciphertext space below.
    - i. plaintext: **HAIL CAESAR!** ciphertext:
  - b. using a right-shift and key = 6 along with the table above. Fill in the blank plaintext space below.
    - i. plaintext:                      ciphertext: **ICHKXEYOEL TF**
  - c. Plug in your micro:bit. Open the “crypt\_3.py” program in the editor, inspect the code, and run the program. Check your encipherment of 1.a. Press the [var] key and select the encipher(“plaintext”, key,chr\_set ) function. Enter the plaintext with quotes and the key as an integer to shift it and the chr\_set. For this action, press the [var] key and select chr\_set\_1. For this activity type encipher(“HAIL CAESAR”, 6, chr\_set\_1). Read the comments in the Python code to help you understand what the code is doing. Do the coding steps match your steps to encipher the plaintext in practice 1a?
  - d. Check your decipherment of 1.b. Press the [var] key and select the decipher(“ciphertext”, key,chr\_set ) function. Enter the ciphertext with quotes, the key as an integer to shift it, and the chr\_set as done in step1.c. For this action, press the [var] key and select chr\_set\_1. Read the comments in the Python code to help you understand what the code is doing. Do the coding

steps match your steps to encipher and decipher the plaintext and ciphertext in practice 1a and 1b?

- e. What happens to the ciphertext if you change the key to a different number and re-run the program?
2. Practice using the character frequency analysis:
  - a. Open the 'prac\_1\_3.py' program in the editor and run the program to count the characters in the first paragraph from the novel "A Tale of Two Cities" by Charles Dickens.
  - b. Exit Python and press the statplot menu, and make a linegraph of L1, the ascii code, vs. L2, the frequency. Select the *zoom* menu and 9. ZoomStat to graph the data. Select the *trace* menu and use the cursor keys to find the most frequent and second most frequent ASCII character codes in the ciphertext. Remember these two codes for the next step.
  - c. Open and run the program 'chr\_ct\_3.py'. Press the [var] key and select the get\_chr(ascii\_code) function. Enter the two codes from the previous step to find their corresponding ASCII characters. For example: >>> get\_chr(72) returns the letter 'H'.
  - d. What is the most frequent letter in the text?
3. Texting an encrypted message:
  - Ensure all group members use the same assigned group number.
  - The **receiver**
    - a. Open the 'recv\_3.py,' this program imports the crypt\_3 program and uses the chr\_set\_2, which allows characters to be lowercase and adds additional punctuation marks. Once the ciphertext is received, the decipher function is use to convert back into plaintext.
    - b. Change the group to the assigned number, then run the program before the sender runs theirs.
  - The **sender**
    - a. Open the program named "send\_3.py" in the editor and run the program to encipher and send the plaintext message to the receiver. This program imports the crypt\_3 program and uses the chr\_set\_2, which allows characters to be lowercase and adds additional punctuation marks.
    - b. Edit the message string, change the group to the assigned number, and then run your program *after* the receiver and hacker have started theirs.
  - The **hacker**
    - a. Open the program named 'hack\_3.py,' in the editor change the group to the assigned number, and then run the program *before* the sender has run theirs.
    - b. After the man-in-the-middle attack steals the ciphertext, repeat the process from the practice above.
      - 1) Press the [var] key and select count\_chrs(), press the [var] key again and select 'stolen\_msg'. It should look like this: >>>count\_chrs(stolen\_msg) then press enter to count the characters in the stolen message.
      - 2) Exit Python and make a statplot of L1(ascii code) and L2 (frequency) and use trace to find most and second most frequent ASCII character codes in the stolen ciphertext.
      - 3) Run the program chr\_ct\_3 and the function get\_chr(ascii\_code) to find the ascii character of from the codes in the scatterplot. Remember, the space character is the most frequent and 'E' second most frequent. Note: the cipher key is the number of letters the plaintext is shifted in the ciphertext.

4) Test your analysis of the key by using `decipher(stolen_msg, key, chr_set_2)`. Did you hack the message?

- After your team runs the activity, the sender should only change the message and key and share the key with the receiver. Don't tell the hacker the new key; *keep it private!* Can the hacker read your message in plaintext as in the 'All Clear' activity?

## Code it

Sender role

```
EDITOR: SEND_3
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from crypt_3 import *
disp_clr()
radio.on()
radio.config(length=250, channel
            =12,power=6,group=1)
key = int(input("key: "))
msg = "The gold coins are hidden
      in the cookie jar!"
cipher = encipher(msg,key)
```

Receiver role

```
EDITOR: RECV_3
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from crypt_3 import *
radio.on()
disp_clr()
radio.config(length=250, channel
            =12,power=6,group=1)
key = int(input("key: "))
print("Waiting for message...")
while not escape():
    ****cipher = radio.receive()
```

Hacker role

```
EDITOR: HACK_3
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from crypt_3 import *
radio.on()
disp_clr()
radio.config(length=250, channel
            =12,power=6,group=1)
key = int(input("key: "))
print("Waiting for message...")
while not escape():
    ****cipher = radio.receive()
```

## Go further

- Try a different role in your team.
- Try changing the key and repeating the activity.

## Check your understanding

- Ciphers are used to **obfuscate** (hide) plaintext messages from hackers.
- A key is required to translate the plaintext characters to the ciphertext characters.
- The sender and receiver must use the same key.
- Frequency analysis of characters is a technique to crack ciphers.

## Help

- Check that everyone on the team is using their assigned group number.
- Ensure the receiver and hacker run their programs and wait before the sender transmits the message.
- Ensure the sender and receiver use the same key and it is private from the hacker.

## Files

- Transfer the activity files below to your calculator using the TI Connect CE Software. The link to download is [here](#). The best practice is to load all files for this cybersecurity activity and then delete them before loading the next set of activity files. This helps keep your calculator organized.

Name	Description
pract_2.py	Practice making a list of channels based on a private key.
send_3.py	Sends text message to receiver using Caesar cipher.
recv_3.py	Receives text message from sender using Caesar cipher.
hack_3.py	Receives obfuscated text messages from the sender.
freq_3.py	Counts the frequency of characters in a string of text. This program is used by the hacker to crack cipher key of intercepted ciphertext.