MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG438 Information and Network Security: Practical Assignment 1

Date handed out: **See ODTUClass**
Date submission due: **See ODTUClass**

Simple Cipher

Purpose

The main purpose of this programming assignment is to revise and gain hands-on experience on the classical ciphers.

Introduction

Cryptography is the science and study of disguising messages so that only certain people can see through the disguise. Before beginning, let's define some terms. An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. As seen in Fig. 1, the process of converting from plaintext to ciphertext is known as **encryption**; restoring the plaintext from the ciphertext is **decryption**.

In the encryption process, the algorithm, known as **cipher**, takes the plaintext and a **key** (aka password) as input, applies the encryption algorithm and outputs the ciphertext. Similarly, the decryption process takes a ciphertext and the key, applies the decryption algorithm, and outputs the plaintext.

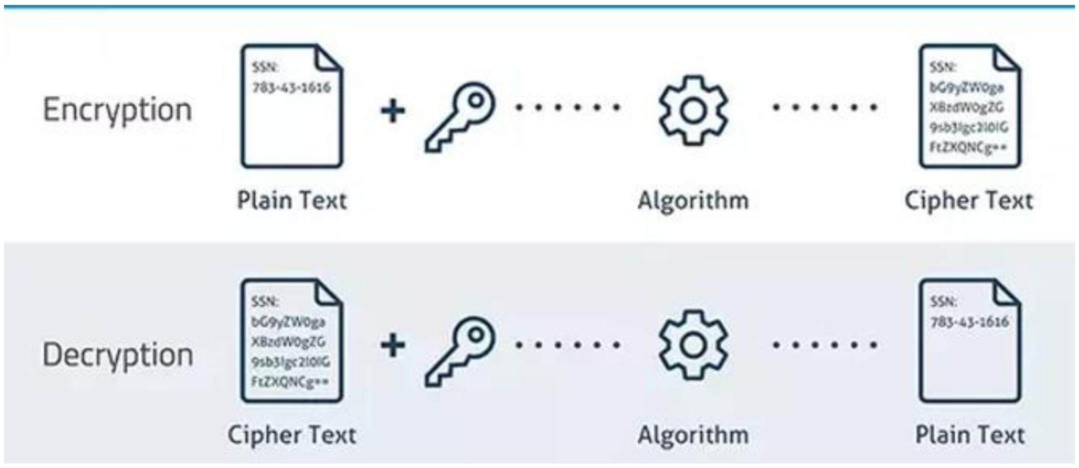


Fig. 1. Encryption and Decryption Process

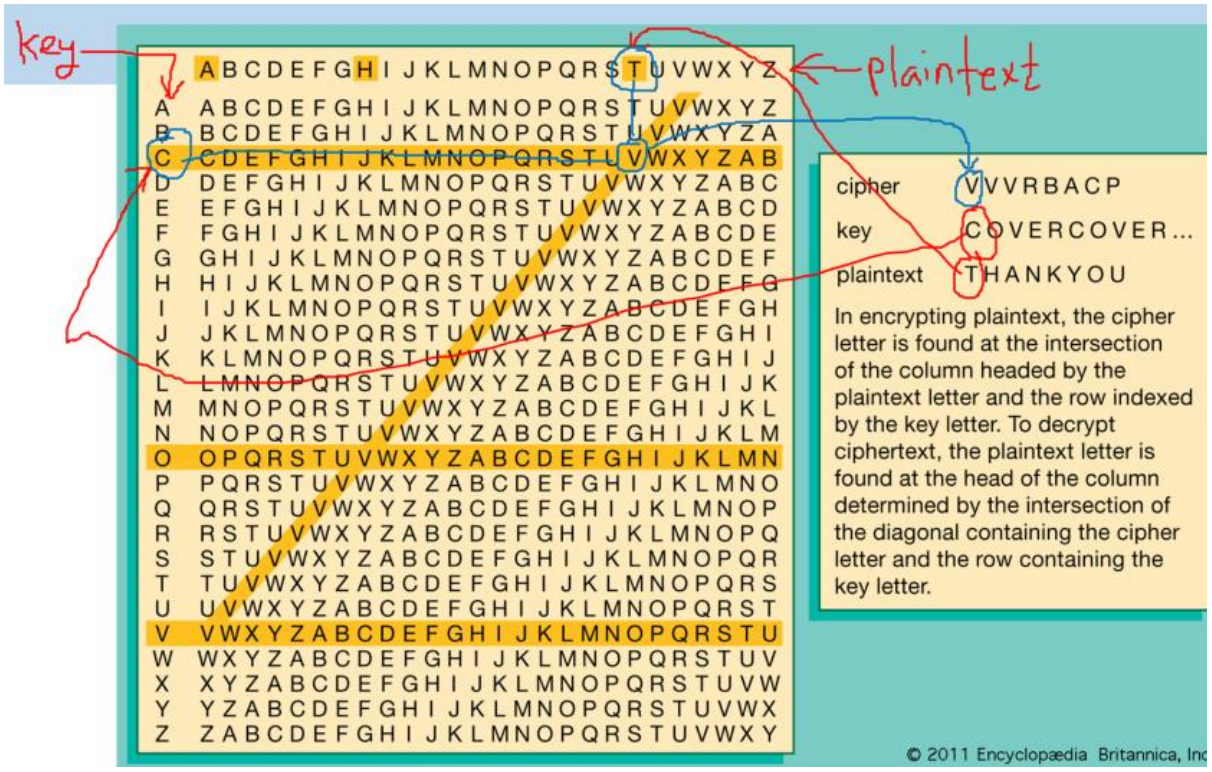
The encryption and decryption algorithm are explained below in detail.

Encryption Algorithm

Encryption will be carried out in two phases.

First Phase:
First begin with a polyalphabetic substitution. Assume a sequence of plaintext letters $P = p_0, p_1, p_2, \dots, p_{n-1}$ and a key consisting of the sequence of letters $K = k_0, k_1, k_2, \dots, k_{m-1}$, where typically $m < n$. The sequence of ciphertext letters $C = c_0, c_1, c_2, \dots, c_{n-1}$ is calculated as follows:

You will use a conversion table (see **Fig. 2** for an illustration) to encrypt the given plaintext. The table will be given to you in form of a two-dimensional array (see the attachment). The rules are very simple. In encrypting plaintext, the cipher letter (c_0) is found at the intersection of the column headed by the plaintext (i.e. p_0) and the row indexed by the key letter (i.e. k_0). And then you will continue with the next letter in the plaintext and in the key until all the letters in the plaintext is converted to the ciphertext letter. As the length of the key is shorter than the length of the plaintext, the key will be repeated.


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

cipher	VVVRBACP
key	COVERCOVER...
plaintext	THANKYOU

In encrypting plaintext, the cipher letter is found at the intersection of the column headed by the plaintext letter and the row indexed by the key letter. To decrypt ciphertext, the plaintext letter is found at the head of the column determined by the intersection of the diagonal containing the cipher letter and the row containing the key letter.

Fig. 2. Encryption and Decryption Conversion Table (taken from Britannica) – This is just for illustration of the algorithm. The conversion table you will use in this assignment is given in the attachment.

For example:
First get the text from the user.
Input: *C IS FUN*
Trim the spaces from the input and obtain the plaintext.
Plaintext: *CISFUN*
Get the key from the user
Key: *PEACE*

First take the first character of plaintext (i.e. *C*) and take the first character of the key (i.e. *P*) and find the intersected character (i.e. *R*) in the table. And repeat this step until you convert all the letters in the plaintext. Notice that the key is repeated if necessary.

Plaintext	C	I	S	F	U	N
Key	P	E	A	C	E	P
Generated Ciphertext	R	M	S	H	Y	C

After the conversion process, you obtain the output (i.e. ciphertext for the first phase).
Ciphertext: *RMSHYC*

Second Phase:
In this phase. You will use the output of the first phase (e.g. *RMSHYC* for the example) as input. And then apply the following steps:

1. Divide the input into two groups from the half, then
2. Each half is written on a line,
3. Obtain the ciphertext by selecting the corresponding letters from left to right

If the length of the input text is odd, then add '0' (zero) as padding character.

For example;
Input text: *RMSHYC*

Divide it to the half

First group:	*RMS*
Second group:	*HYC*

Obtain the ciphertext by selecting the columns left to right (as depicted in **Fig. 3**).

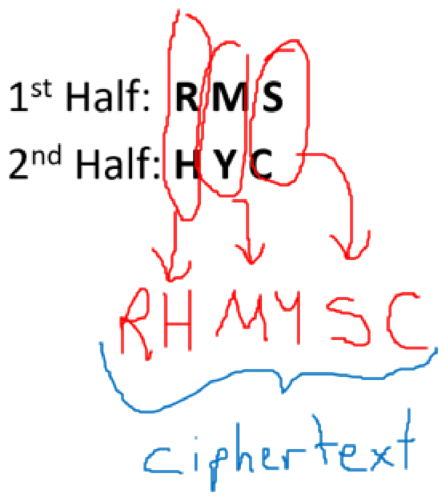


Fig. 3. Simple Transposition

And, you obtain the ciphertext. This is the final output of the encryption.
The generated ciphertext: *RHMYSC*

Decryption Algorithm

Decryption is the reverse operation of encryption. Similarly, there will be two phases corresponding to each encryption phase in reverse order (see **Fig. 5**).

First Phase:
This is the reverse operation of encryption phase 2. First, find the halves as seen in **Fig. 4** and then combine them. For example, assuming the ciphertext is *RHMYSC*, then the decrypted text is *RMSHYC*. This will be input for the second phase of the decryption. If you added any padding character in encryption phase 2, remove it.

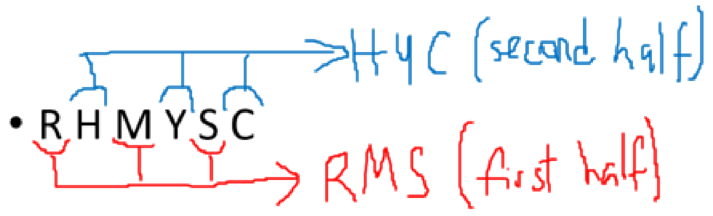


Fig. 4. Decryption

Second Phase:
This is the reverse operation of encryption phase 1. To decrypt the text, the plaintext letter is found at the head of the column determined by the intersection of the diagonal containing the cipher letter and the row containing the key letter using the conversion table provided (see **Fig. 2**).

Assumptions

1. Consider only English letters (i.e. 26 letters) in plaintext, in ciphertext, and in key, no need to consider other characters (e.g. digits, punctuation characters, etc.).
2. Erase any space found in the input.
3. Convert the text uppercase first.

Algorithm Explained

The program flow is depicted in **Fig. 5**.

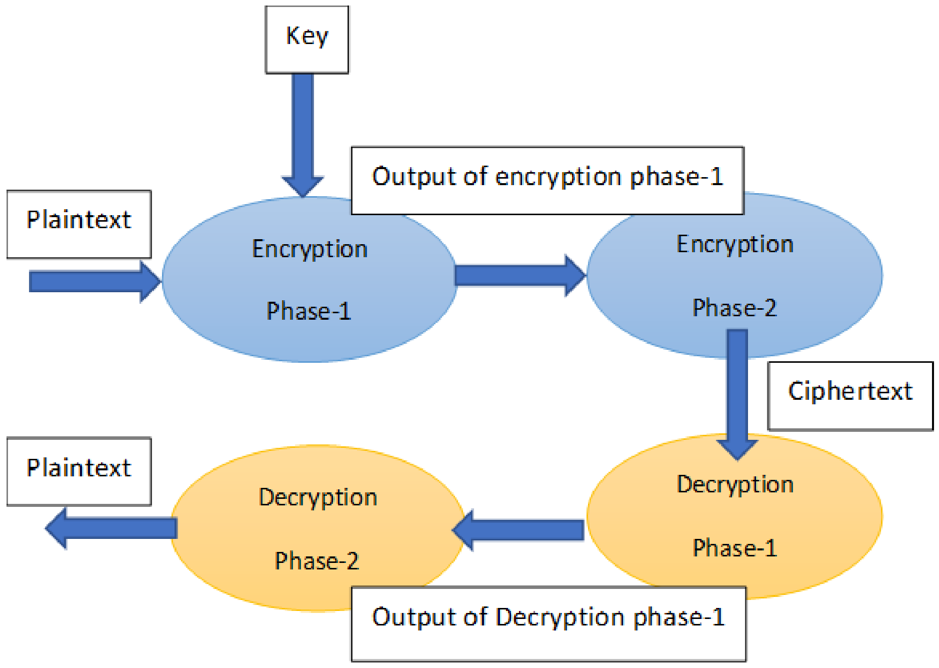


Fig. 5. Program Flow

Prepare the input string (i.e. plaintext for encryption or ciphertext for decryption):

1. Read the input from the user.
2. Trim whitespace (e.g., space).

3. Convert all the letters to uppercase.

Prepare the key:

1. Read the key from the user (both for encryption and decryption).
2. If the key length is less than the length of the input string, then repeat the keyword to match the length of the input string.

Encryption:

1. Apply the first phase for the encryption. **Hint:** The columns and rows are shown with letters in the conversion table. Therefore, you need to find the index (the order of the letter in English alphabet) of the letter. In order to match the letter and its corresponding order in the alphabet, you may use ASCII table as a trick. As all our letters in the uppercase form, you may obtain the order of the letter in English alphabet by subtracting 65, whereas the ASCII code of 'A' is 65. For example, if you subtract 'A' - 65, you obtain 0, which is the index for the character 'A' to determine the column or row index in the conversion table.
2. Apply the second phase for the encryption. Use the output of phase 1 as the input for the phase 2. If the length of the input text is odd, add '0' (zero) as the padding character.
3. Print the ciphertext.

Decryption:

1. Apply the first phase of the decryption algorithm (, which is the reverse operation of phase-2 of encryption). First, if you added padding character in the encryption phase 2, do not forget to remove it. To apply the decryption, you may think to implement an additional function, which returns the column index of the conversion table, when the cyphertext character is found in the corresponding key letter row.
2. Apply the second phase of the decryption algorithm (, which is the reverse operation of phase-1 of encryption).
3. Print the plaintext.

Programming Requirements

- You may use any programming language to implement. The interface will be similar as in the given Sample Run subsection below.

- The conversion table for the encryption phase-1 and the decryption phase-2 is provided as a 2-D array. See the attachment.

- The following functions must be implemented:

    - **Menu**: This will print the menu as seen in the sample run below. It will take the user selection. The menu will repeat until the user selects exit option.

    - **Encrypt**: This will encrypt the plaintext using the key provided by the user and it will return the ciphertext using the algorithm defined above.

    - **Decrypt**: This will decrypt the ciphertext using the same key in the encryption and it will return the plaintext using the algorithm defined above.

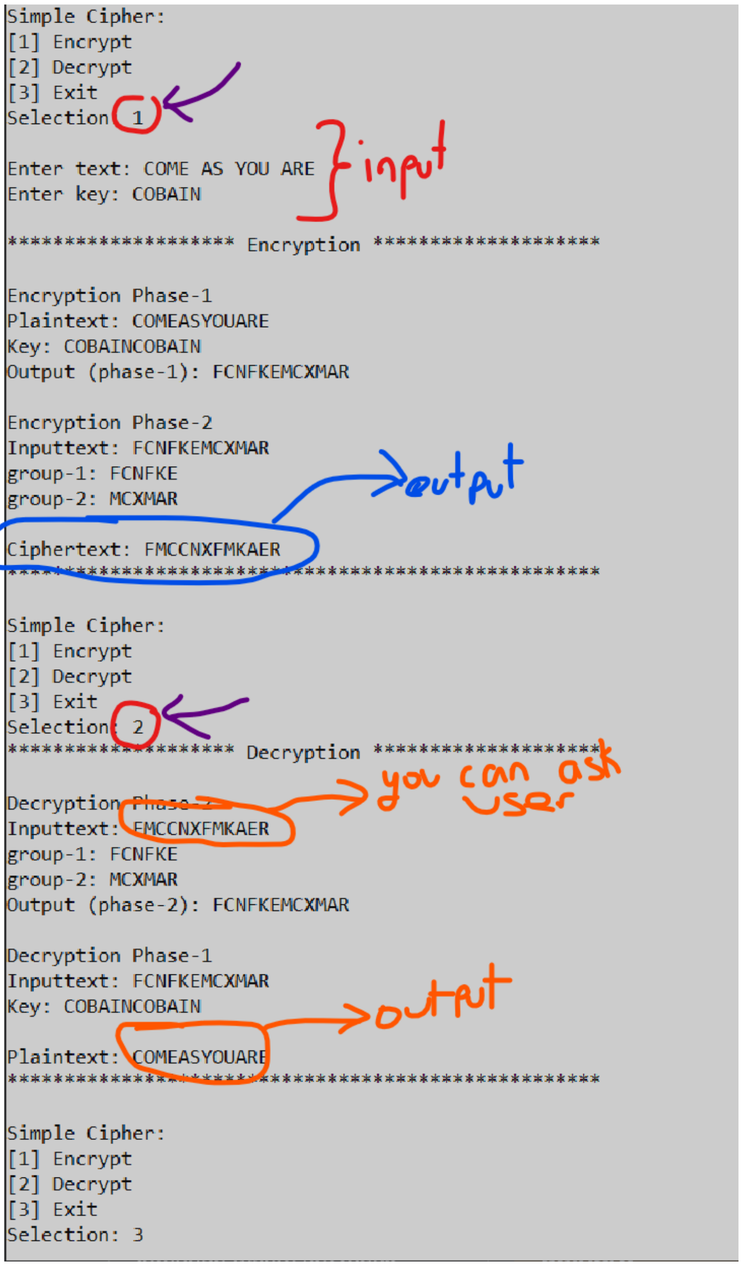
- And you may implement any other helper functions such as trimming the spaces in a string, getting the length of a string etc. Some useful functions you may consider implementing:

    - A function that returns the length of a given string. Just count the letters in a loop until the null character at the end of the string.

- A function that repeats the key. You will repeat key as long as the length of the input text. You may keep two indices in a loop as one goes through the length of the input text and the other resets in each repetition of the key.
- A function to trim the space from the given string.

Important note: Go through the sample run carefully. It's very helpful to understand about this practical assignment. Take a look at the grading distribution to get idea which parts of your codes will be evaluated strictly.

Sample Run



```
Simple Cipher:
[1] Encrypt
[2] Decrypt
[3] Exit
Selection: 1

Enter text: COME AS YOU ARE
Enter key: COBAIN

******************** Encryption ********************

Encryption Phase-1
Plaintext: COMEASYOUARE
Key: COBAINCOBAIN
Output (phase-1): FCNFKEMCXMAR

Encryption Phase-2
Inputtext: FCNFKEMCXMAR
group-1: FCNFKE
group-2: MCXMAR

Ciphertext: FMCCNXFMKAER
****************************************************

Simple Cipher:
[1] Encrypt
[2] Decrypt
[3] Exit
Selection: 2
******************** Decryption ********************

Decryption Phase-2
Inputtext: FMCCNXFMKAER
group-1: FCNFKE
group-2: MCXMAR
Output (phase-2): FCNFKEMCXMAR

Decryption Phase-1
Inputtext: FCNFKEMCXMAR
Key: COBAINCOBAIN

Plaintext: COMEASYOUARE
****************************************************

Simple Cipher:
[1] Encrypt
[2] Decrypt
[3] Exit
Selection: 3
```

Grading

Your program will be graded as follows:

Grading Distribution	**Mark (100)**	**Remarks**
Menu function	5	
Encryption operation Phase-1	20	
Encryption operation Phase-2	20	
Decryption operation Phase-1	20	
Decryption operation Phase-2	20	
Other useful functions (trim, padding, uppercase, etc.)	10	
Readme.txt and demo video	5	

Rules

Please make sure that you follow the restrictions for the assignment as follows:

- Follow the mentioned programming requirements.
- If your program does not compile, then you will get 0.
- Strictly obey the input/output format. Do not print extra things.

Submission

You need to submit a ZIP file (firstname_lastname.zip, e.g., okan_topcu.zip) including the following:

1. Your source codes.
2. Your executable if needed.
3. *Readme.txt* file for your source code. This should include a short description of your program and **it should explain how to compile and run your code**, please include your name, surname, and student id at the top.
4. A short video showing the execution demo. You may provide a link to an outside site (e.g. Youtube)

To submit your assignment, simply select the appropriate assignment link from the ODTUCLASS page. Upload your zip file and click submit (clicking send is not enough). Please make sure all source files are included in your zip file when submitted. A program that does not compile as submitted will be given 0 points. Only your final submission will be graded. Remember there is no late submission for this assignment.

Academic Honesty

You need to submit your own solution. Your source code will be investigated for neatness and cheating. All the programming assignments will be required to be implemented individually and any code sharing will be considered as cheating. Please note that if you are caught cheating, you can get zero from this assignment and also from all the others. See the course syllabus for further details and clarification.

Attachments

Table.txt: Two-Dimensional Array Definition