Network Security (CSE 350)

Assignment-1

Prof. B.N. Jain

Project 1 : Poly-alphabetic substitution

Algorithm:
(1363+1362)%3=1

Submitted by:

Name	Roll No
Vansh	2021363
V. Bharath Krishna	2021362

Brief Overview of Project

Programming Language: C++
Platform: MS Windows

We have implemented a Poly-Alphabetic substitution system using a symmetric Key of known length 4. We implemented 6 functions named check_string, encrypt, Decrypt, hash, recognizable, bruteforce.
The character set consists of lower case English letters, viz. {a, b, …, z}.

We take input for key from key.txt file, 5 sample input for original string from Original_texts.txt.
We check if the key and original string are strings of lowercase English alphabet or not in the Check_string function using the islower() in-built function.

Plaintext is formed by concatenating the original text and the hash value of original text using the hash() Function by means of “+” operator.
The encrypt() and decrypt() and Hash() function results are calculated and visually inspected.
We have also done error handling while reading from the file.

At last bruteforce function is called to find the key and is printed on terminal.

Hash Function

Returns a 16 character string called hash_string which is the hash_value of the original string passed as A parameter. Each character of the original string is placed in a 16 column matrix. The blank elements After the last character in the last row are marked as the character “z”. Then we add the integer ASCII value of every character in a column for all 16 column. Then the 16 integer values are converted into a character by taking its modulo with 26 and adding 97 and assigning the integer to a character. The string of 16 characters in order is returned as hash_value of the original string.

Example:
Original String : thisisthefourthsampleinput

t	h	i	s	i	s	t	h	e	f	o	u	r	t	h	s
a	m	p	l	e	i	n	p	u	t	-	-	-	-	-	-

→

t	h	i	s	i	s	t	h	e	f	o	u	r	t	h	s
a	m	p	l	e	i	n	p	u	t	z	z	z	z	z	z

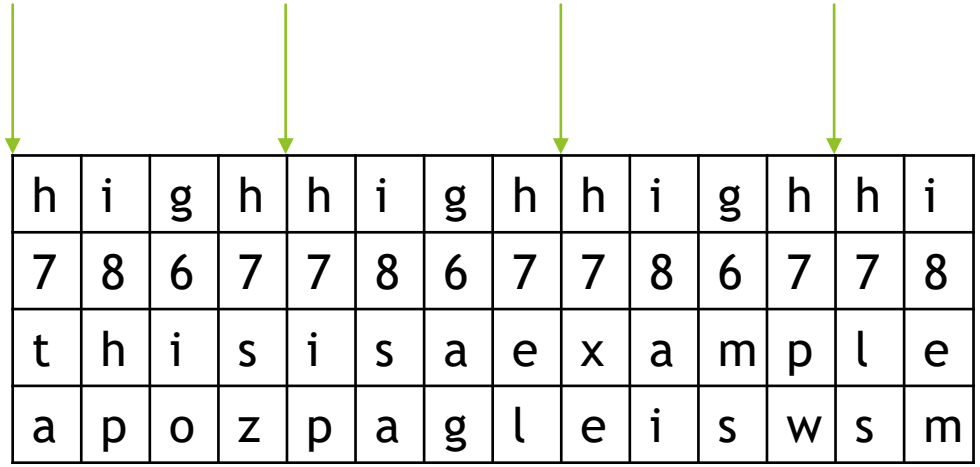
1st row

2nd row

Integer ASCII of 1st row	116	104	105	115	105	115	116	104	101	102	111	117	114	116	104	115
Integer ASCII of 2nd row	97	109	112	108	101	105	110	112	117	116	122	122	122	122	122	122
Addition (+)	213	213	217	223	206	220	226	216	218	218	233	239	236	238	226	237
Addition%26 +97	102	102	106	112	121	109	115	105	107	107	122	102	99	101	115	100

Converting integer ASCII To char	f	f	j	p	y	m	s	i	k	k	z	f	c	e	s	d

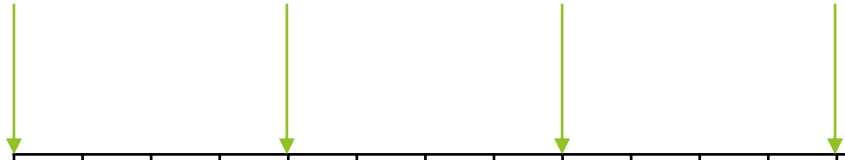
Encryption Function



key	h	i	g	h	h	i	g	h	h	i	g	h	h	i
numeric key	7	8	6	7	7	8	6	7	7	8	6	7	7	8
plaintext	t	h	i	s	i	s	a	e	x	a	m	p	l	e
ciphertext	a	p	o	z	p	a	g	l	e	i	s	w	s	m

In the above example, we are encrypting the plaintext using the key=“high” . We generate the numeric key by subtracting 97 from the integer ASCII value of the respective character from the key in the column (h=104-97=7). For encrypting, we shift each character in the plaintext forward by the respective integer in the numeric key row. In the example above, we can see that the characters of key are repeated till the length of plaintext.

Decryption Function



key	h	i	g	h	h	i	g	h	h	i	g	h	h	i
numeric key	7	8	6	7	7	8	6	7	7	8	6	7	7	8
ciphertext	a	p	o	z	p	a	g	l	e	i	s	w	s	m
plaintext	t	h	i	s	i	s	a	e	x	a	m	p	l	e

In the above example we are decrypting the ciphertext using the key=“high”.
For Decrypting we shift each character in the ciphertext backwards by the respective Integer in the numeric key row. In the example above, we can see that the characters of key are repeated till the length of plaintext.

Recognizable function and Pie Property

Recognizable function takes plaintext as parameter and returns a bool variable whether the plaintext is recognizable or not. Or in simpler words, the plaintext satisfies the pie property or not.

Pie property for a plaintext :

Last 16 characters of plaintext == Hash (Remaining characters of the plaintext)

It is useful to check If the key used to decrypt is correct or not. For example if we can have many combinations of key and each key resulting in a plaintext. How to know which key is correct?

BruteForce

In this function, we try to crack the key by launching a bruteforce attack.
We iterate from “aaaa” to “zzzz” to select a key and for every key, we decrypt the first ciphertext and check if the resulting first plaintext is recognizable or not using the recognizable() function.

If the first plaintext is recognizable, we do the same thing
for all 5 ciphertexts. If the chosen key holds true for all 5 ciphertext, we return the key found. Else we drop this key and repeat this process for a new key. If no key is found, the function returns “ERROR”.

Sample Input/ Output

```
PS C:\Users\vansh\Desktop\NSC-A1> cd "c:\Users\vansh\Desktop\NSC-A1\" ; if ($?) { g++ NSC-A1.cpp -o NSC-A1 } ; if ($?) { .\NSC-A1 }
Known key: high

First OrinialText   : wearediscoveredsaveyourselfover
First PlainText     : wearediscoveredsaveyourselfoverilqbejlwslmeyugd
First CipherText    : dmgyllozjwblymjzhdkfvcxzltlvcmxpsyhlqtczsukfboj
First decryptedText : wearediscoveredsaveyourselfoverilqbejlwslmeyugd

Second OrinialText   : yesthisisthefirstassignmentofnetworksecuritycourse
Second PlainText     : yesthisisthefirstassignmentofnetworksecuritycourseduysepelklqnjgmz
Second CipherText    : fmyaoqypzbnlmqxzaiyzpottlvzvmvkadwxrzmibyqzfjwayzmjbfakwltqsxvpnth
Second decryptedText : yesthisisthefirstassignmentofnetworksecuritycourseduysepelklqnjgmz

Third OrinialText   : cannotfindagoodexampleforanoriginaltextandfinallytheassignmentiscompletedontime
Third PlainText     : cannotfindagoodexampleforanoriginaltextandfinallytheassignmentiscompletedontimedyuffhfzryiqesxf
Third CipherText    : jituvblpulgnvwjleiswsmlvyitvyqmpuiralfzhullpuirsfbnlhaypnvslubozjwswsmzlkwtapukkfclmonfyfqwlzfl
Third decryptedText : cannotfindagoodexampleforanoriginaltextandfinallytheassignmentiscompletedontimedyuffhfzryiqesxf

Fourth OrinialText   : thisisthefourthsampleinput
Fourth PlainText     : thisisthefourthsampleinputffjpymsikkzfcesd
Fourth CipherText    : apozpazolnubybnzhuvslqtwbblmqxetzqqrgnilzl
Fourth decryptedText : thisisthefourthsampleinputffjpymsikkzfcesd

Fifth OrinialText   : thisisgoingtobethebiggestofallthesampleinputsokokokokokokokokokokokokokokokokokokwehaveusedpolyalphabeticsubstitutionalgorithm
Fifth PlainText     : thisisgoingtobethebiggestofallthesampleinputsokokokokokokokokokokokokokokokokokokwehaveusedpolyalphabeticsubstitutionalgorithmzkmayzibmuyueruk
Fifth CipherText    : apozpamvpvmavjkaomhpnokzawlhstzolagtwtkpuxaazwqvrwqvrwqvrwqvrwqvrwqvrwqdlpgclcylkxusfirwoihlaqizbjyapbaapwthsouypbntgsshfhoitceblzar
Fifth decryptedText : thisisgoingtobethebiggestofallthesampleinputsokokokokokokokokokokokokokokokokokokwehaveusedpolyalphabeticsubstitutionalgorithmzkmayzibmuyueruk

LAUNCHING BRUTE FORCE ATTACK...

Found Key: high
PS C:\Users\vansh\Desktop\NSC-A1>
```