  
  
**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Subject Code: CSCI361** |  |
| **Subject Name: Cryptography** |  |
| **Submission Type: online** |  |
| **Assignment Title: assignment 1** |  |
| **Student Name:wael abdella** |  |
| **Student Number:5215018** |  |
| **Student Phone/Mobile No.** |  |
| **Student E-mail:waa744** | @uowmail.edu.au |
| **Lecturer Name:** **Dr. Halim Khelalfa** |  |
| **Due Date:** |  |
| **Date Submitted:** |  |

|  |  |
| --- | --- |
| **PLAGIARISM:** The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:**STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student:** |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Optional Marks:** | | **Comments:** | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Lecturer Assignment Receipt**(To be filled in by student and retained by Lecturer upon return of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Student:** | | |

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Lecturer** | | |

**Containers and data structures:**

vector <int> CipherTextvec;

This vector is to hold the text values in decimals

vector <int> CipherTextvecCopy

This vector will contain a copy of the ciphertext which will be used in the final deciphering phase

vector <int> KnownKeyLengthVec

This vector holds the decimal values after knowing the key length

vector <int> KeyvaluesforCipher

This vector holds the key values that have been deciphered

vector <int> Positions

vector that holds the positions of the most reoccurring decimals in the letter frequency map

vector <char> letterCorrectionvector

This is a vector that will hold key size amount of deciphered letters

vector <string> Dictionary

This vector will hold the values that are read from the dictionary

vector <string> FinalizedWords

The set will hold the correct values for the key length size number of words

vector <char> mostOccuringLetters

vector that hold the most occurring letters in the english language

map<int,double> LetterFrequencymap; //

displays the most reoccuring letters in the English alphabet

map<int,double>::iterator itr;

iterator for the Letter Frequency map

**Reading from file:**

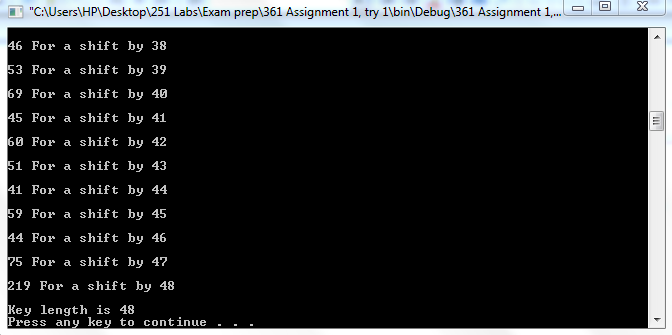
The file contains hexadecimal values, every two characters are equal to one hexadecimal value.

Two char variables are used to read the first two values, then they are converted into one string, that is then converted into a decimal.

The decimal value of the hexadecimal read is then pushed into the vector called CipherTextvec that holds integers. This repeats until all the values in the text are read.

The reason for converting the hexadecimals to decimals is to keep a consistent data type, ASCII values are in decimals. Since the ciphertext values are in the CipherTextvec vector i can close the file and only use that vector to continue.

**Finding The key Length**

The values saved in the CipherTextvec are copied into a new vector called TempIntsvec, the values of CipherTextvec are shifted by one position and then compared, counting the number of matching values. Then the shift incremented one more time and the number of matching letters are counted again. When the key size is reached there will be a spike in the number of matches, this is because all the letters will be shifted by the same key thus increasing the probability of matching letters. The spike is the key length. 

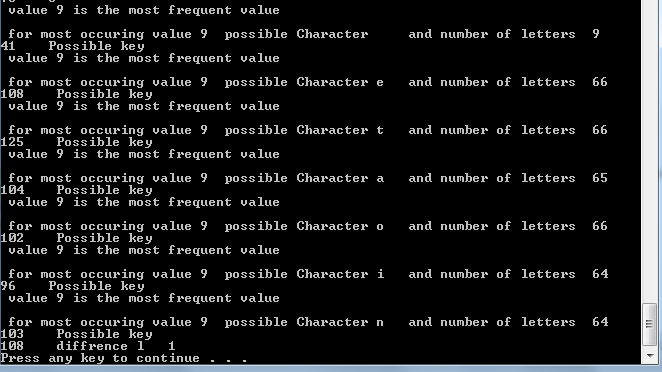
**Using the key length and letter frequency**

Now That the key length is known (48), we use that key length and get every value key length apart, those values will be ciphered using the same character of the key and put inside a vector (KnownKeyLengthVec). After that all the values are moved to a map called LetterFrequencymap. The value is put in the first position and the number of times the value occurs is second. Using our knowledge on the English language we know that the most occurring characters are

1. (Space)
2. E
3. T
4. A
5. O
6. I
7. N

Using this we can assume that the most occurring value in the map is the space character. so, the most occurring value in the map is XOR’d with space to get the result which is equivalent to the key. However, there are two problems with this method, first is that the most reoccurring character may not actually be the space character and the second is the fact that there could be more than one value that appears the most in the map.

To resolve this issue I saved the peak value/values and XOR’d it/them with the most reoccurring value and counted the number of small letter results it gives. I did this with all peaks with all the most reoccurring letter in the English language. This gives you a very high chance of getting the correct key.



As you can see here if I had assumed the most frequent character was a space, I would’ve only had 9 letters as a result. That would have been wrong instead my program give me the letter that gives the first instance of the most small letters. Which in this case is 108 or the letter l.

(XOR’d with e)

The keys are saved in a vector called KeyvaluesforCipher.

**Deciphering**

Now that the key is available it is XOR’d with the CipherTextvec, and displayed

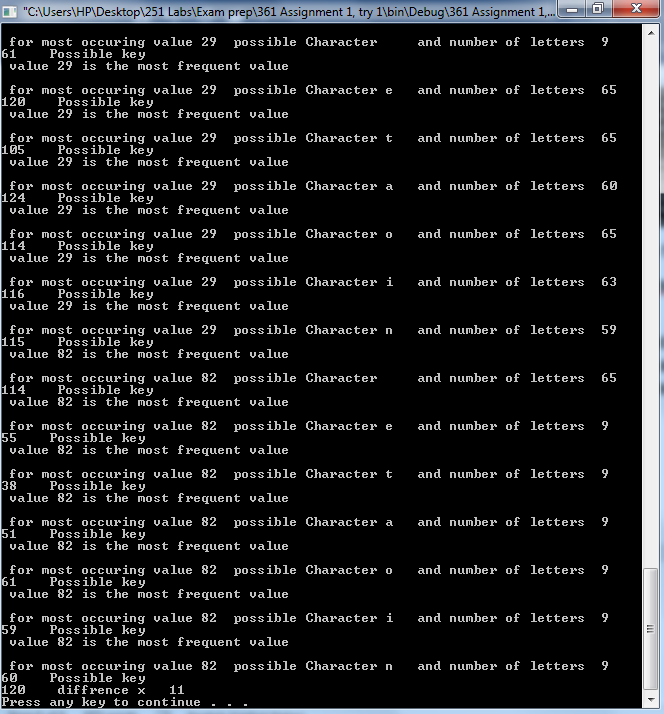


As you can see in the first sentence

In Ethereug, smara contracts are written in the programminm langutge Solidity

At positions 11 and 18 we have the wrong result, this is because of a rare case where there are multiple most occurring letters yield the same result for the number of small letters but the first result isn’t the correct one

For position 11



Most common character e is the first letter to have the highest number of small letters thus, it is chosen however we saw that it was wrong. This is one of the rare cases when multiple letters have the same highest peak and the first most frequent letter to have the peak is the wrong character.

**Final step**

To solve this issue I used a text file containing words from the English language, then I read the first keyword length size letters (48):

“In Ethereug, smara contracts are written in the”

I then take the first word and look for all the words in the dictionary with the same size, after that I start counting how many matching letters they have between them, the word with the most matches in the dictionary replaces the current word in the sentence. So “In” is replaced by “In” however “Ethereug” is replaced by “Ethereum” this is done to the whole string.

Now that the string is corrected all that is left is for it to be XOR’d with the first key size(48) hexadecimals and the correct key will appear.

The key is: les#$onxqertTYuoprasthn@@gboqZmk lmswiaaoojyfpl34