Eder Mazariegos

08/21/2017

Cryptography

**Cypher Decryption**

1. **Encoded Message**: ozwfoaddlzwjsafklghakzgmdvygtmqsfmetjwdds

**Decoded Message**: when\_will\_the\_rain\_stop\_i\_should\_go\_buy\_an\_umbrella (excluding underscores for spaces)

* I solved this cypher by identifying which letters repeated within the decoded message and then making a list of possible letters that could occur twice within words such as the pp in apple or the ff in off. Once I did that, I found that l, p, f, e, and o were my most common and most practical choices. Afterwards, I then counted the spaces to the left that the letters corresponded to within the decoded message. For example, for l, I counted 8 spaces to the left and got the letter d which corresponded to its actual letter within the cipher. When I was finished with that, I counted the number of spaces left from d to the original letter l and that was exactly 18. I knew this could be a possible match, so I wrote it down and I repeated the process for every double letter like so…… 8 = L, 12 = p, 2 = f, 1 = e, 11 = o. In this case, l was the correct match and the key was 18 because none of the letters I tried with their corresponding keys worked (key p = 14, key f = 24, and key e = 26). Once I finished that, I was given the decrypted message.

1. **Encoded Message**: nenahxwnrwlujbbfruuyaxkjkuhknqdwpahbrwlnrcbwnjaudwlqcrvn

**Decoded Message**: everyone\_in\_class\_will\_probably\_be\_hungry\_since\_its\_near\_lunchtime (excluding underscores for spaces)

* I solved this cypher using the same method as I did for the one above. The only thing I did different this time is that I started with the u and decoded it with the l in mind first instead of starting with the s. Once I started to see the pattern come out and it all was translating into full words I could understand, I kept going until the end.

1. **Encoded Message:** orhqfqjweuuosrgwrfuomovvdolwnjotwuerlfdwvcaefocruchfdcuwnjot

wumovvpwjwbwevwlmdwricqpjwekfdwaondwjfwxfuosobwicqhcjfdoueuu

osrgwrfohicqacgwpigichhoawpwhcjwaveuucrmwlrwuleierlfwvvgwfdw

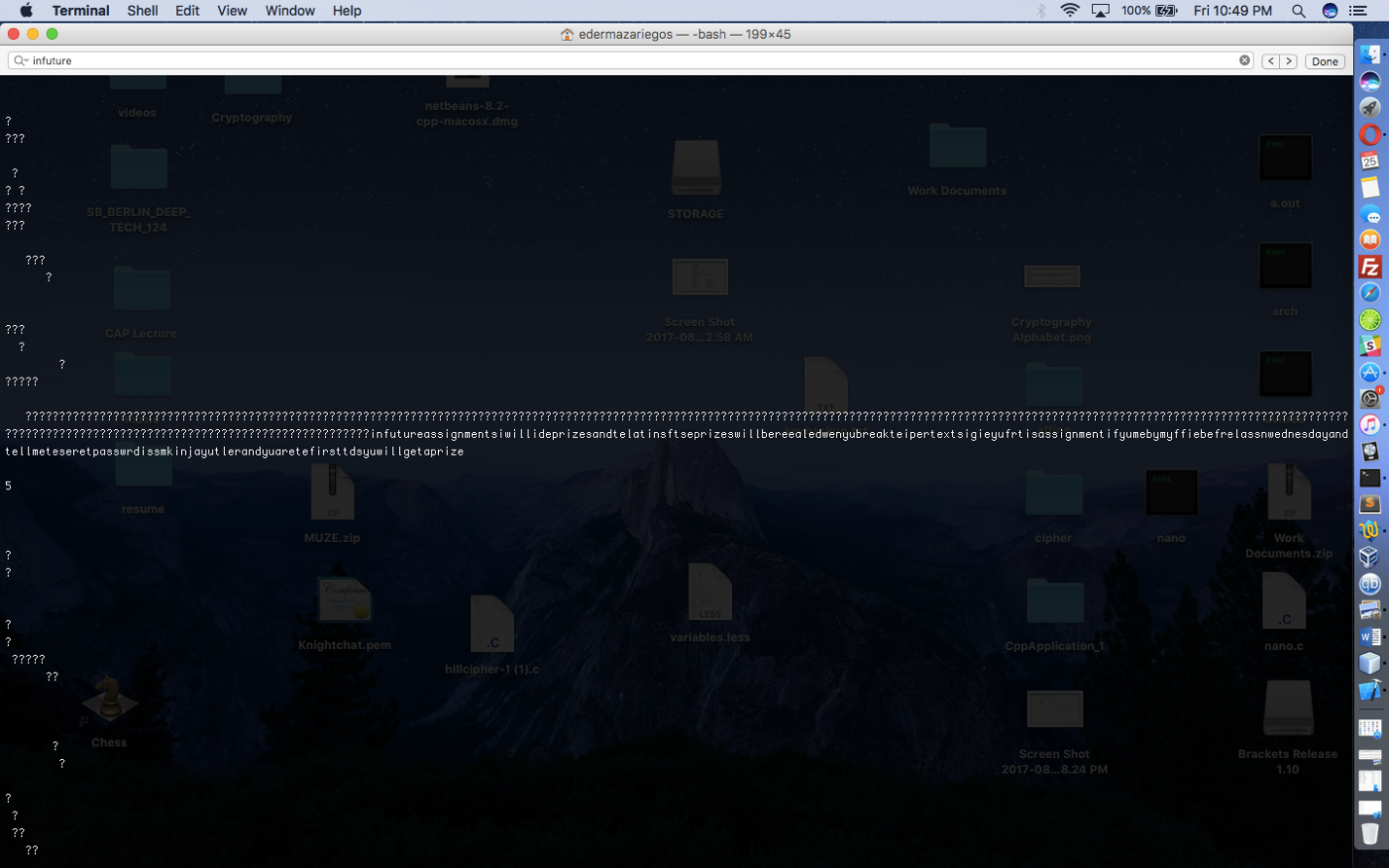
uwajwfneuumcjlouugckorzeiaqfvwjerlicqejwfdwhojuffclcucicqmov

vswfenjotw

**Decoded Message:**

Infutureassignmentsiwillideprizesandtelatinsftseprizeswillbereealedwenyubreakteipertextsigieyufrtisassignmentitellmeteseretpasswrdissmkinjayutlerandyuaretefirsttdsyuwillgetaprize

* This one took the longest out of them all and I had to use a code that I wrote to brute force attack the message. In my original theory, I thought that if I just decrypt the message the way it was encrypted, it would workout perfectly, but that surely was not the case. I ended up writing a code which uses the decryption formula a-1 = (x-b) mod 26. With this formula however, I had a lot of trouble because when I found the inverse, I plugged it into the equation through the code while trying every possible scenario for B and that did not work. So I moved on and made a triple nested for loop that tested every possible value for a up to 26 and b up to 26 while incrementing through a buffer which stored the numeric values of the plaintext. Afterwards, I was given a plethora of script to go through, so I sifted through my terminal output and found something that made since…



which I believe is the message. The source code is below and it took me around 4 hours or so to code because it has been a little while since I have coded from scratch.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: main.c

\* Author: edermazariegos

\*

\* Created on August 25, 2017, 5:18 PM

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define BUFFSIZE 500

/\*

\*

\*/

char charbuff[BUFFSIZE], alpha [26] = {'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'};

int finalbuff[BUFFSIZE], posbuff[BUFFSIZE], x = 0, i = 25, j = 25, a, b;

char ch, dispense;

char \*c;

int main(int argc, char \*argv[]) {

for (int d = 1; d < 2; d++){

char\* fname = argv[d];

FILE\* file = fopen(argv[d], "r");

if (file == NULL) {

fprintf(stderr, "ERROR cannot find file!\n");

}

while((ch = fgetc(file)) != EOF){

if(ch >= 'a' && ch <= 'z'){

charbuff[x] = putchar(ch);

x++;

}

}

}

printf("\n");

int i = 0;

for (c = charbuff; \*c; c++) {

posbuff[i] = (\*c - 'a');

printf ("%d\n", \*c - 'a');

i++;

}

/\*

int a = 3;

int a\_inv = 0;

int flag = 0;

for (int i = 0; i < 26; i++)

{

flag = (a \* i) % 26;

if (flag == 1)

{

a\_inv = i;

}

}

\*/

for (a = 0; a < 26; a++){

for(int b = 0; b < 26; b++){

for (int l = 0; l < BUFFSIZE; l++){

finalbuff[l] = (a \* (posbuff[l] - b)) % 26;

}

printf("\n\n");

printf("%d", b);

printf("\n\n");

for (int g = 0; g < BUFFSIZE; g++){

printf("%c", finalbuff[g]);

}

for (int n = 0; n < BUFFSIZE; n++){

if (finalbuff[n] >= 0){

dispense = alpha[finalbuff[n]];

printf("%c", dispense);

}

}

}

}

return 0;

}

1. **Encoded Message:** puciqmlyveafjbanktydkhzagwyxrwso

**Decoded Message**: packmyboxwithfivedozenliquorjugs

* I encrypted this message with the formula E(x) = (ax + b) mod m. At first, I found that I was moving very slow with the encryption process because there was a lot of repetition within the calculations, which in the end caused me to make some mistakes. After realizing that I may be wrong within the calculations or there could possibly be error, I did the encryption process again and found that I was correct initially.