Q1 Team Name
0 Points

Cryptophilic

Q2 Commands
5 Points

List the commands used in the game to reach the ciphertext.

go -> wave -> dive -> go -> read -> password -> c

Q3 Analysis
50 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Use Latex wherever required. If your solution is not readable, you will lose marks. If necessary the file upload option in this question must be used TO SHARE IMAGES ONLY.)

As the spirit said, we needed to apply the given transformations in the sequence EAEAE on the input block to obtain the output block, where:
> A was an 8x8 key matrix with elements from F_{128}
> E was an 8x1 vector whose elements are numbers between 1 and 126. E was applied on a block by taking the i^{th} element of the block and raising it to the power given by i^{th} element in E.

When we whispered **password** near the screen, the coded password received was **gsiiipillogpftijmomkfqmrfqmmmhfk**.

PART 1 : .

When we entered several inputs and observed their output ciphertexts, we found out that the ciphertexts had letters from 'f' to 'u' only. Therefore, 16 characters represented by 4 bits were being used (f - 0000, g - 0001, ..., u - 1111). But, it was given that the size of the field was 128, so each byte should be interpreted as an element from F_{128}. So, each byte can take values from **ff (0)** to **mu (127)**.

To verify, we used our code **analyze.cpp**. The output was:

a: 0 ; n: 515
b: 0 ; o: 511
c: 0 ; p: 504
d: 0 ; q: 513
e: 0 ; r: 502
f: 1506 ; s: 471
g: 1528 ; t: 496
h: 1493 ; u: 499
i: 1487 ; v: 0
j: 1469 ; w: 0
k: 1513 ; x: 0
l: 1473 ; y: 0
m: 1504 ; z: 0

So our observation was verified.

PART 2 :

> Another observation that we made was that when we gave inputs in which the prefix of bytes were 0, the corresponding prefix were also 0.
> When we changed i^{th} byte in the input, only the bytes after it were changed in the output and the bytes before i^{th} byte were the same in output.

This made us conclude that a should be a **Lower Triangular Matrix**.

PART 3 : Using our code **inp-outp-gen.cpp** -

We used input format $C^{i-1}PC^{8-i}$ and generated plaintexts (C represents the string ff and P represents a string in $(ff....mu)$. The inputs were stored in **gen-inp.txt**. For the plaintexts, we generated the corresponding ciphertext outputs, which were stored in **gen-outp.txt**.

PART 4 : Using our code **diag-elem.cpp** -

Since A is a lower triangular matrix, when we gave inputs in which only i^{th} byte was non-zero, the i^{th} byte of output would be: $(A_{ii}(A_{ii}X_i^{E_i})^{E_i})^{E_i}$; where A_{ii} is the i^{th} element in the diagonal of matrix A, X_i is the i^{th} byte (the only non-zero byte) of the input and E_i is the i^{th} byte of E. We then passed all possible input values (0 - 127) as X_i and got the outputs. Then we used brute force for the values of A_{ii} (0 - 127) and E_i (1-126) and got the values which mapped the inputs to their corresponding outputs - for all $1 <= i <= 8$. We got a set of possible pairs of (A_{ii}, E_i) for each i using our code -

For $i = 1$: (27, 1) ; (84, 19) ; (84, 107) ;
For $i = 2$: (35, 73) ; (70, 117) ; (84, 64) ;

Assignment 5
GRADED

2 DAYS, 22 HOURS LATE

GROUP
Tanishq Rajesh Chourishi
Shubhi Kesarwani
Abhinav Maheshwari
View or edit group

TOTAL POINTS
60 / 60 pts

QUESTION 1
Team Name **0** / 0 pts

QUESTION 2
Commands **5** / 5 pts

QUESTION 3
Analysis R **50** / 50 pts

QUESTION 4
Password **5** / 5 pts

QUESTION 5
Codes **0** / 0 pts

For $i = 3$: (14, 89) ; (43, 40) ; (72, 125) ;
For $i = 4$: (12, 75) ; (24, 28) ; (30, 24) ;
For $i = 5$: (67, 105) ; (109, 59) ; (112, 90) ;
For $i = 6$: (11, 50) ; (86, 16) ; (97, 61) ;
For $i = 7$: (27, 21) ; (63, 88) ; (123, 18) ;
For $i = 8$: (9, 101) ; (38, 14) ; (108, 12) ;

PART 5 : Using our code **A-E-Password-finder.cpp** -

Since we had obtained the diagonal elements of matrix A and the elements of vector E, we could use them to find the remaining non-diagonal entries of A. Since A was a lower triangular matrix, we needed to find A_{ij} for $i > j$. To do so, we used the values of A_{ii} and A_{jj} and iterated over them in a triangular manner. This gave us the exact value of every diagonal element, and value of the element just below it, using the possible values we had already found. To get the remaining values in A, we used iteration for all values from 0-127 and used EAEAE over matrix A and vector E, and used the formula : $(A_{ii}(A_{ii}X_i^{E_i})^{E_i})^{E_i}$ for plaintext and ciphertext pairs. This checked if the equation was held true. This analysis gave us the matrix A as follows:

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 115 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 15 & 28 & 43 & 0 & 0 & 0 & 0 & 0 \\ 96 & 22 & 26 & 12 & 0 & 0 & 0 & 0 \\ 96 & 42 & 3 & 110 & 112 & 0 & 0 & 0 \\ 28 & 43 & 19 & 51 & 111 & 11 & 0 & 0 \\ 10 & 120 & 21 & 101 & 31 & 94 & 27 & 0 \\ 90 & 12 & 94 & 25 & 12 & 71 & 7 & 38 \end{bmatrix}$$

Also, we found out the vector E to be : $E = \begin{bmatrix} 19 & 117 & 40 & 75 & 90 & 50 & 21 & 14 \end{bmatrix}$.

Our encrypted password was **gsiiipillogpftijmomkfqmrfqmmmhfk**. Using the encoding we mentioned earlier, we converted it into bits : length of the password was 16 bytes. Therefore, we split it into 2 parts of length 8 byte each. To get the corresponding input for each part, we operated each byte in sequence from 1 to 8 and tried all possible values from 0 to 127 for it. Then we selected the values for which the output values matched with the given value till that byte (after the EAEAE encryption). As stated in our previous analysis, the i^{th} byte of the output would depend only on the bytes $< i$ in the input. So we found the correct values as we operated over the bytes from $i = 1$ to $i = 8$.

We converted it from bits to alphabetic representation using the ASCII representation. The password which we got was : **svqqbwnkdm000000**. Then we removed the zeroes at the end, because they would have been added to make text length a multiple of the size of the block.
This gave us our final password : **svqqbwnkdm**.

No files uploaded

Q4 Password
5 Points

What was the final commands used to clear this level?

svqqbwnkdm

Q5 Codes
0 Points

It is mandatory that you upload the codes used in the cryptanalysis. If you fails to do so, you will be given 0 for the entire assignment.

Cryptophilic.zip Download

```
1  Binary file hidden. You can download it using the button above.
```