Q1 Team Name
0 Points

DECODERS

Q2 Commands
5 Points

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

go,wave,dive,go,read

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.)

$Information\ obtained\ from\ the\ screen:$

After entering the command "read" on the glass panel, we got the EAEAE problem. The encryption used is a block cipher having block size as 8 bytes. The irreducible polynomial for the linear transformation is $x^7 + x + 1$. In the matrix A, elements are from F_{128} and the elements of vector E are between 1 and 126. By typing "password", as told by the spirit, we obtain the encrypted password as: "ijhjgmklinkpipggjlkqiumigjgmllkj"

$Figuring\ out\ the\ Encoding:$

We analyzed our ciphertexts in order to figure out what encoding

was used to convert our string input to binary block. Upon analyzing the ciphertexts, we discovered that the characters ranged from 'f' to 'u', which are 16 in number. This reminded us of the hexadecimal base system, which uses four bits for each character. We therefore mapped letters f-u to 0-15. The mapping is as follows:
f: 0000
g: 0001
h: 0010
i: 0011
j: 0100
k: 0101
l: 10110
m: 0111
n: 1000
o: 1001
p: 1010
q: 1011
r: 1100
s: 1101
t: 1110
u: 1111

A byte is consists of 2 characters as each character is represented by 4 bits. As the field F_{128} contains 128 elements, we analyzed all these 128 pairs from ff-mu, and determined that all these pairs were encrypted to unique strings, suggesting that encryption is related to ASCII values. Thus, we mapped ff-mu to 0-127 in decimal. Here odd positions will contain characters from f to m and even positions will contain characters from f to u.

We have made some observations after analyzing the encoding of plaintexts to ciphertexts. We are dealing with structural cryptanalysis of an EAEAE cryptosystem. A block of size 8 bytes is passed as an input to the cryptosystem as a 8x1 vector over the field F_{128}. The ciphertext also came out as all 0's when the input plaintext was all 0's. In the case where the first i bytes of input plaintext were 0, also the first i bytes of ciphertext were all 0. The output from i-th bit onwards start changing if we change the input plaintext at i-th bit. Observing these facts suggests that the transformation matrix A could be lower triangular matrix. In a more formal manner consider $p_0, p_1 ... p_7$ as the 8 byte input plaintext and $c_0, c_1 ... c_7$ as the output ciphertext.

The ciphertext obtained is $ff\ ff\ ff\ ff\ ff\ ff\ ff\ ff$ when input text is $ff\ ff\ ff\ ff\ ff\ ff\ ff\ ff$.

When the input text has a non zero byte as p_j and the byte $p_i = 0, \forall i \notin j$, we observe ciphertext of form, $c_i = 0$,for $i < j$. If the input is changed from $p_0, ..., p_k, p_{k+1}, ... p_7$ to $p_0, ..., p_k, p'_{k+1}, ... p_7$ then the ciphertext also changes after k^{th} byte.

It shows that every 0 in a row ends up at the end of that row. The matrix A could therefore be a lower triangular matrix.

$Cracking\ the\ transformations\ E\ and A:$

The next step is to crack the exponentiation transformation E and linear transformation A.
A is the key matrix of 8x8 elements, and the elements are referred to as $a_{i,j}$ where i and j are row and column of the element respectively. e_i is the i-th element of the 8x1 vector E.

$Step\ 1:$ $Generation\ of\ input\ plaintexts\ and\ the\ corresponding\ c$

Using the file "Plaintext_generation.py", the inputs were generated of the form $C^{-1}PC^{8-i}$ which means each input had atmost one non zero block at a time. The plaintexts were stored in "plaintext.txt". There are 128 possible plaintext values (from ff to mu) for every choice of such a block. 8 sets of 128 input plaintexts were generated, with only one byte as non-zero in each set. Thus, a total of 128 x 8 plaintexts were used for attack. "plaintext.txt" contains all these input texts.

"Generating_script.py" generates "script.sh" using "plaintext.txt" file. Using "command.sh", we generate "output.txt" file, which is a temporary file.We used "Cipertext_generation.py" to generate the ciphertexts and store them in "ciphertext.txt".

$Step\ 2 : Finding\ the\ Matrices\ A\ and\ E$

As A is lower-triangular, if the i-th non-zero input block has value m ($m_i \neq 0$), then the corresponding output block will be having the value $n = (a_{i,i} * (a_{i,i} * m^{e_i})^{e_i})^{e_i}$ and let this be eqn(1). The diagonal element of vector A and all the elements of vector E can then be determined. Furthermore, if the linear transform is lower-triangular, then the i-th output block depends on the j-th input block iff i >= j.

The operations are carried out on the finite field F_{128} with irreducible generator polynomial $x^7 + x + 1$ over F_2. During a brute force attack, the Alternate Multiply and Exponent functions are used to check if the encrypted output matches the ciphertext. Because the field isF_{128}, addition is done by XORing the integers.

We iterate over all possible elements e_i of exponentiation vector E [1-126] and diagonal elements $a_{i,i}$ of A [0-127] using these plaintext-ciphertext pairs to see if inputs maps to the output. And if the input maps to the output, we add those values to a corresponding possible list. The elements of E are numbers between 1 and 126, while the elements of key matrix A are from F_{128}. Each block yields three tuples.

Block	Possible $a_{i,i}$	Possible e_i
Block 0	[84, 28, 113]	[18, 21, 88]
Block 1	[8, 115, 70]	[58, 86, 110]
Block 2	[33, 43, 98]	[22, 37, 68]
Block 3	[31, 11, 12]	[9, 44, 74]
Block 4	[5, 31, 112]	[78, 85, 91]
Block 5	[11, 73, 103]	[52, 99, 103]
Block 6	[27, 14]	[20, 108]
Block 7	[38, 61, 125]	[17, 41, 69]

We must now find the non-diagonal elements of A as well as eliminate some pairs from the above result. More plaintext

ciphertext pairs are used, and we iterate over the above $a_{i,i}, e_i$ pairs to find these elements in the range [0-127] in a way that satisfies eqn (1).
Looking at the i-th output block where the j-th input block is non-zero will yield an element $a_{i,j}$. In order to find $a_{i,j}$, all the elements of set $S_{i,j}$ needs to be known. The set $S_{i,j}$ is given by:
$S_{i,j} = \{a_{n,m} | n > m, j <= n, m <= i\} \cap \{a_{n,n} | j <= n < i\}$
These elements, when represented graphically, form a right-angled triangle with corners at $\{a_{i,i}, a_{i,j}, a_{j,j}\}$. The elements next to each diagonal element, as well as one element at each diagonal position of A and vector E, are then known.

Block	Final $a_{i,i}$	Final e_i
Block 0	84	18
Block 1	70	110
Block 2	43	37
Block 3	12	74
Block 4	112	91
Block 5	11	52
Block 6	27	20
Block 7	38	17

We iterate over all possible values (0-127) for the remaining elements of A and eliminate those that do not satisfy eqn (1) using final values of diagonal elements of A and exponent vector E, starting from $a_{i+1,i}$, and so on. If we find$a_{i+1,i}$, we can eliminate the rest of the possibilities from the previous list.

Finally, the key matrix A is:

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 124 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & 17 & 43 & 0 & 0 & 0 & 0 & 0 \\ 99 & 23 & 24 & 12 & 0 & 0 & 0 & 0 \\ 111 & 37 & 1 & 122 & 112 & 0 & 0 & 0 \\ 29 & 45 & 29 & 44 & 110 & 11 & 0 & 0 \\ 13 & 119 & 12 & 104 & 28 & 95 & 27 & 0 \\ 88 & 3 & 81 & 27 & 24 & 70 & 1 & 38 \end{bmatrix}$$

and the exponentiation vector E is:

$$E = \begin{bmatrix} 18 & 110 & 37 & 74 & 91 & 52 & 20 & 17 \end{bmatrix}$$

"EAES_ALGO.py" contains the code needed to do these tasks.

$Step\ 3 : Decrypting\ the\ Password$

The same python file "EAES_ALGO.py" contains the code thorugh which we can decrypt the encrypted password "ijhjgmklinkpipggjlkqiumigjgmllkj" using these final A and E. The password was separated into two blocks, and each block of encrypted password was subjected to the following transformations:
$E^{-1}(A^{-1}(E^{-1}(A^{-1}(E^{-1}(encrypted\_password)))))$

Using these transformations, we were able to decrypt both password blocks as:
$[114, 111, 110, 112, 99, 121, 109, 110]$ and
$[102, 97, 48, 48, 48, 48, 48, 48]$

With the mapping {ff:0,...,mu:127} we mapped these numerical values in order to decrypt the text. But we found out that this text wasn't our password, so we guessed they might be ASCII codes instead. And thus, we mapped these numerical values to ASCII values. By doing thus we get the decrypted password as $\mathbf{ronpcymnfa000000}$. This password contained 0's, which are probably padding. Hence we used the password $\mathbf{ronpcymnfa}$ to clear this level.

No files uploaded

Q4 Password

5 Points

What was the final commands used to clear this level?

ronpcymnfa

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.

DECODERS.zip Download

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

Assignment 5

● GRADED

2 DAYS, 22 HOURS LATE

GROUP

Manthan Kojage

Abhishek Dnyaneshwar Revskar

Akash Gajanan Panzade

View or edit group

TOTAL POINTS

40 / 60 pts

QUESTION 1

Team Name	**0** / 0 pts

QUESTION 2

Commands	**5** / 5 pts

QUESTION 3

Analysis	**50** / 50 pts

QUESTION 4

Password **5** / 5 pts

QUESTION 5

Codes R **-20** / 0 pts