(Team details can be found at the end of the assignment)

Cipher Method: *EAEAE*

Passcode: rqfaimydxscmehuf

Justification:

i. On the first screen, there was a passage leading to some deep underground well. So, we used ***go***.
ii. On the next screen, we had a free fall and nothing to grab so we used ***go***, but we died.
iii. So, we entered the above commands again and the instead of ***go*** we used ***wave*** and we reached screen 3.
iv. Now, swimming in both direction we couldn't find anything, so we used ***dive***.
v. On the next screen, there was a well-lit passage in the wall, so we used ***go***.
vi. The following screen had a glass panel, so we entered ***read*** and got the EAEAE problem.
vii. The problem had input with a block of size 8 bytes as 8 x 1 vector over F_{128}.
viii. After trying enough inputs, we found that 16 letters in the output are from 'f' to 'u'. Also, we considered that input also consists of these letters only as only 16 letters could be represented by four bits and each alphabet was mapped with a number from 0 to 15.
ix. Also, as the input was of 8 bytes but the field was GF(128), so we could have inputs from 'ff' to 'mu' considering MSB for any letter will always be 0.
x. We noticed that on giving any inputs of the form 'gf' or 'hf', it produced same output as of inputs 'g' or 'h' respectively. That means 'f' has been used as padding.
xi. We then decided to keep input having format as:
C^7P, then only the last Byte of output had different values, all others were constants. So, that tempted us to try out different formats such as C^6PC^1 and so on. And we figured out that on

changing i^{th} Byte of input, it changes all Bytes of output after i^{th} Byte.

xii. This gave us a feeling that the transformation matrix used here is a lower triangular matrix.

xiii. Consider the input plaintext represented as I_0, I_1, I_2 … I_7 and the output cipher text as C_1 ... C_$_7$.

xiv. On varying the inputs for k from 0 to 7, making I_0 to $I_{(k-1)}$ as 'ff' we get the output ciphertext values also as 'ff' for C_0 to C_k. In other words whenever the input plaintext has I_0 … I_j = 'ff' we get the ciphertext as C_0,….C_j = 'ff'.

xv. Also, when the input is changed from I_0 … $I_k I_{(k+1)}$ … I_7 to I_0 … $I_k I'_{(k+1)}$ … I_7, the output changes only after k, this implies that the 'ff' present in each row have to be at the end row,

Thus, we get a Lower Triangular matrix

xvi. Thus, we generated inputs of the form $C^{8-i}PC^{i-1}$ using ***Input Generation.ipynb***.

xvii. Now, (Exponential) E box had elements between 1 to 126 and Linear Transformation Matrix A had elements from GF(128) which was observed to be lower triangular.

xviii. Due to the input format we used, only 1 block is non-zero per input, so we can iterate over all possible values of diagonal elements and exponents, since the matrix is lower triangular if x is the value of non-zero input block (say i), then the corresponding block of output has the value $O = (a_{i,i}(a_{i,i} * x^{ei})^{ei})^{ei}$.

xix. We then implemented operations over finite field of 128 with generator $x^7 + x + 1$ which will be used to carry out operations throughout the code in ***BruteCipher.ipynb***.

xx. Now, for each pair of plaintext-ciphertext; we iterated through all possible values of e_i and $a_{i,i}$ (using the above operations) and compared the output. We kept on storing the possible value pairs of (e_i and $a_{i,i}$) and found out that there are 3 pairs per block.

Block Number	Possible Values of $a_{i,i}$ [0,:2]	Possible Values of e_i [0,:2]
Block 0	[33, 26, 25]	[4, 47, 76]
Block 1	[113, 91, 9]	[2, 38, 87]
Block 2	[75, 22, 22]	[1, 19, 107]
Block 3	[72, 109, 105]	[30, 35, 62]
Block 4	[23, 125, 37]	[64, 73, 117]
Block 5	[82, 38, 18]	[15, 31, 81]
Block 6	[94, 20, 106]	[72, 84, 98]
Block 7	[89, 125, 55]	[22, 37, 68]

xxi. Now, we need to eliminate pairs and also find non-diagonal elements. To do this we will use some more plaintext-ciphertext pairs and iterate over the above pairs to find element within 0 to 127 such that the equation O holds. We do this in triangular fashion such that we use $a_{i,i}$ & $a_{j,j}$ to find $a_{i,j}$. Thus, after this iteration we found every element next to each diagonal element.

xxii. This also helped us to reduce possible pairs to 1 element each as only for certain pairs we could produce element next to diagonal entry.

Block Number	Final value of $a_{i,i}$	Final Value of e_i
Block 0	25	76
Block 1	91	38
Block 2	22	19
Block 3	105	62
Block 4	125	73
Block 5	82	15
Block 6	94	72
Block 7	55	68

xxiii. Now, we found each element of this matrix by iterating over all possible values (0 to 127) and using the Final Values of Exponent Matrix and the above found values of Linear Transformation Matrix and checked the validity of function O.

xxiv. Thus, we found the Linear Transformation Matrix as follows:

$$\begin{bmatrix} 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 126 & 57 & 22 & 0 & 0 & 0 & 0 & 0 \\ 103 & 105 & 59 & 105 & 0 & 0 & 0 & 0 \\ 16 & 122 & 74 & 19 & 125 & 0 & 0 & 0 \\ 0 & 126 & 46 & 107 & 125 & 82 & 0 & 0 \\ 112 & 100 & 86 & 105 & 104 & 40 & 94 & 0 \\ 43 & 69 & 72 & 83 & 16 & 40 & 76 & 55 \end{bmatrix}$$

And the Exponent Matrix found was as follows:

$$\begin{bmatrix} 76 & 38 & 19 & 62 & 73 & 15 & 72 & 68 \end{bmatrix}$$

xxv. Now to decrypt the password we used the same way i.e. we iterated over all possible values for a block and check if our EAEAE function's output is same as the current password block we have. We did it in two halves as the password contained 32 letters. The decrypted password found out was:

rqfaimydxscmehuf

Team Details:

- Ashish Pal (18111010)
- Darshit Vakil (18111013)
- Mayank Rawat (18111040)