CS641 - Assignment 5

Nishtha - 180489

Text on the wall :

“You come up to the closed door and look at the screen ...
... there is nothing written on it as earlier.
As you wonder if the spirit is around to help you out, you hear
it whispering in your ears ...
"This is another magical screen. And this one I remember perfectly...
Consider a block of size 8 bytes as 8 x 1 vector over F_{128} --
constructed using the degree 7 irreducible polynomial x^7 + x + 1
over F_2. Define two transformations: first a linear transformation
given by invertible 8 x 8 key matrix A with elements from F_{128} and
second an exponentiation given by 8 x 1 vector E whose elements are
numbers between 1 and 126. E is applied on a block by taking the ith
element of the block and raising it to the power given by ith element in E.
Apply these transformations in the sequence EAEAE on the input block
to obtain the output block. Both E and A are part of the key.
You can see the coded password by simply whispering 'password' near
the screen..."”

Cryptanalysis

- **Commands to get to cipher**

  Go -> wave -> dive -> go -> read

- **Analysis (EAEAE Cipher)**

  - It has been given in the magic screen that it is an EAEAE cipher. So firstly, we tried to find out the different letters involved and on trying various random inputs we figured that the output had 16 letters from 'f' to 'u' so we got an intuition that maybe it has something to do with hex base. Also, the input vector is 8 x 1 and in a finite field over $$F_{128}$$ thus we could have inputs from 'ff' to 'mu'. Next on trying all pairs as inputs from ff to mu, we realised that they all mapped to different strings which gave us a intuition that the may have something to do with ASCII values as they have 128

characters matching the 128 different strings of ff to mu. Also, for a 8 character long plain text, we get a 16 character long cipher text. The binary output of encryption is therefore 64 bit long. Which makes sense as we have 4 bits for every character which brings us to a conclusion that there is a mapping from 'f' - '0000' to u - '1111' just like the previous assignment.

- Further, we observed that inputs like 'gf', 'gff', gave same output which means 'f' has been used for padding. Next, we tried inputs of the form $$C^{(8-1)}PC^{(i-1)}$$ where C corresponded to the constant bits (in this case the padding) and P corresponded to the variable bits and we observed that only the bytes after the $$i^{th}$$ byte changed on changing the $$i^{th}$$ byte. This gave us a slight intuition of a lower triangular matrix. We further have, say input text $$t_0t_1t_2.....t_7$$and corresponding output cipher as $$o_0o_1o_2....o_7$$, then on taking $$t_o$$ to $$t_i$$ as say, 'ff' we have $$o_0$$ to $$ o_i$$ also as 'ff'. If we use other string than 'ff' in $$t_o$$ to $$t_i$$, then the output will have their corresponding cipher of that string and the bytes after the $$i^{th}$$ bit will change. as For linear transformation matrix $$A = a_{ij}$$, for input vector to last layer is T, then output vector can be drawn with O = [$$o_0o_1o_2....o_7$$] where $$o_i = \sum_{j=0}^{7} a_{i,j}*t_{i}$$. Thus the observation that only the bytes after the $$i^{th}$$ byte changed on changing the $$i^{th}$$ byte will hold in case of a LOWER TRIANGULAR MATRIX only. This is a very crucial observation which has been used throughout to draw the password.

- We have Exponent matrix with elements from 1 to 126 and the linear transformation matrix with elements from field $$F_{128}$$. Rather than trying to crack matrices, we have a go at the password directly using our above observation.

- We get our ciphertext to be "jgjtmofogoifghmsgjgffompkmmglhjn" and we break it into two halves of 16 characters as h1 = "jgjtmofogoifghms" and h2 = "gjgffompkmmglhjn" and we break them one by one.

- breaking h1 - We take as inputs the range varying first byte ir ‘fffffffffffffff’ to ‘mufffffffffffff’ and we generate the output cipher using the output1.sh script and clean using the myclean.py script. In the output, we find the only string starting from ‘jg’ as ‘jggmlmlpmjmmipfh’ and find the corresponding input string to be ‘mnfffffffffffff’. Next we give as input, keeping only second byte variable as ‘mnfffffffffffff’ to ‘mnmufffffffffff’ and we find the only string starting with ‘jgjt’ to be ‘jgjtilfnkgkkfmgf’ and the corresponding input string was ‘mnmnfffffffffff’. Seeing the

pattern, we keep on following and varying the inputs and finding the corresponding input for our encrypted password byte by byte. We have the following table:

Block	Input (varying block i)	Required Cipher	Corresponding Input
1	‘ffffffffffffffff’ - ‘muffffffffffffff’	‘jggmlmlpmjmmipfh’	‘mnfffffffffffffff’
2	‘mnfffffffffffffff’ - ‘mnmufffffffffffff’‘	jgjtilfnkgkkfmgf’	‘mnmnfffffffffffff’
3.	‘mnmnfffffffffffff’ - 'mnmnmufffffffffff'	'jgjtmolihlfmhsmi'	'mnmnlufffffffffff'
4	'mnmnlufffffffffff' - 'mnmnlumufffffffff'	'jgjtmofomsjllogo'	'mnmnlumiffffffff'
5	'mnmnlumiffffffff' - 'mnmnlumimufffffff'	'jgjtmofogogjgmjf'	'mnmnlumilgffffff'
6	'mnmnlumilgffffff' - 'mnmnlumilgmuffff'	'jgjtmofogoifloff'	'mnmnlumilgltffff'
7	'mnmnlumilgltffff' - 'mnmnlumilgltmuff'	'jgjtmofogoifghji'	'mnmnlumilgltmgff'
8	'mnmnlumilgltmgff'-'mnmnlumilgltmgmu'	'jgjtmofogoifghms'	'mnmnlumilgltmgmm'

Thus, we find the decrypted first half to be 'mnmnlumilgltmgmm'. When we type this we get the encrypted cipher 'jgjtmofogoifghms' and thus we know our decryption is correct. Then we change this 16 character string to 8 cahracter by combining two cahracters to get blocks as :
Say we have 'mn', we have corresponding hex to be '0111' & '1000', combining '01111000' -> 120 thus ascii character 'x'. Following this pattern, we get the first half of password as 'xxosanqw'.
- Breaking h2 - For the second half, we follow the exact same pattern and the table is as follows:

Block	Input (varying block i)	Required Cipher	Corresponding Input
1	‘ffffffffffffffff’ - ‘muffffffffffffff’	‘gjhrgrlumukuhkgi’	‘ljfffffffffffff’
2	‘ljfffffffffffff’ - ‘ljmufffffffffff’	‘gjgfgqgrhsismsij’	‘ljloffffffffff’
3.	‘ljloffffffffff’ - 'ljlomufffffffff'	'gjgffoiniskhipjm'	'ljloiffffffffff'
4	'ljloiffffffffff' - 'ljloifmufffffff'	'gjgffompggkohpjo'	'ljloififfffffff'
5	'ljloififfffffff' - 'ljloififmuffffff'	'gjgffompkmlggflo'	'ljloifififffffff'
6	'ljloififfffffff' - 'ljloifififmuffff'	'gjgffompkmmgllfl'	'ljloififififffff'
7	'ljloifififfffff' - 'ljloififififmuff'	'gjgffompkmmglhlt'	'ljloififififffff'
8	'ljloifififfffff' - 'ljloififififmu'	'gjgffompkmmglhjn'	'ljloifififififf'

Thus, the text obtained corresponding to h2 was 'ljloifififfffif' and following the conversion method above, we obtained 'di000000'.

- Password - Combining both, we got 'xxosanqwdi000000'. We tried this but still got connection closed and after removing the padding (0's), we finally cleared this level.

- Attachments - We have attached the code to get cipher and clean the obtained output. Since, we gave inputs 128 at a time, and each block of inputs was different, so we changed it accordingly. We have attached a txt file containing all inputs given 'input.txt' and a output file 'clean.txt' containing all the corresponding ciphers.

- **Password**

  xxosanqwdi