Q1 Team Name
0 Points

Group Name

the_cryptonics

Q2 Commands
5 Points

List all the commands in sequence used from the start screen of this level to the end of the level

go->enter----- dead!!!

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

Q3 Cryptosystem
5 Points

What cryptosystem was used at this level?

Cryptography Technique used here is EAEAE (a variant of AES)

Q4 Analysis
80 Points

Knowing which cryptosystem has been used at this level, give a detailed description of the cryptanalysis used to figure out the password.

Initial Observations

> *whisper of spirit*

1. From the whisper of spirit we got to know about the cryptosystem used. The spirit gives us a hint on properties of cryptosystem that was used.

~ 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.

~two transformations
i)first a linear transformation given by invertible 8 x 8 key matrix A with elements from F_{128}.
ii)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.

> *Obseravation of text and mapping*

2. We started with trying random inputs many a times our connection to the server got terminated so we figured out that we need to craft our input ina specific manner.

We observed that the output contains 16 letters, ranging from "**f**" to "**u**" Additionally, we took into account that the input only consists of these 16 letters. 16 letters could be represented by four bits sp each alphabet was mapped with a number from **0** to **15**

correspondingly.

f->0000, g-> 0001, h->0010, i->0011, j-> 0100, k->0101, l->0110, m->0111, n-> 1000,
o->1001, p->1010, q-> 1011, r-> 1100, s->1101,t->1110, u->1111.

As GF(128) has 128 elements and characters to be considered are from 'f' to 'u' we have concluded that the 128 element values from 0-127 are to be from **ff** to **mu**.
odd positions corresponds to [**f-m**]
even positions corresponds to [**f-u**]

The coded password we got is
gsjmluhphmfjgshriujmjumiklghlkmj

Decryption

We have used a single file **MAIN.ipynb for the whole analysis from generation of inputs-outputs to getting the password.**

3. After identifying the types of inputs to send to the server (to receive ciphertext values), we noticed someÂ key elements.

a)We discovered that any inputs of the pattern "gf" or "hf" resulted in the same output as inputs of type "g" or "h," respectively. That indicates that "f" was used as padding.
b) On input "fffffffffffffff" output is "fffffffffffffff".
c) Changing the last byte of the plain text does change only the last byte value of the ciphertext. For instance: Cipher Text Obtained: "ffffffffffffflq"; Plain Text: "fffffffffffffmn".
d) Changing the i_{th} Byte of the plain text effects â€˜ i_{th} to 8_{th}â€™ Bytes of the corresponding ciphertext.
eg. ffffffffffffff > ffffffffffffff

eg. ffffffffffff -> ffffffffffff
ffffffffffffkf -> ffffffffffffki
ffffffffffffkff -> ffffffffffflhho

This indicates that the transformation matrix used here can be a **lower triangular matrix**.

4. So, the â€˜logicsâ€™ derived from above are as follows:
a) The first byte of the ciphertext is entirely dependent on the First byte of the plain text.

b) The second byte of the ciphertext is dependent only on the 1st and 2nd byte of the plain text and so on.

5. Based on the above-mentioned logic; we can apply brute force i.e given an encrypted text we can find the corresponding plain text. The process can be explained in two steps:

a) We generated 128 plain texts where the first-byte value is different across all the 128 plain texts but all the other 7 byte values are the same (all 7 bytes are zeroes i.e ff). Then, we have passed the 128 plain text values to the game server to obtain the ciphertext. From the logic (4a) i.e â€œThe first byte of the ciphertext is entirely dependent on the First byte of the plain textâ€ we figured out which of our plain texts first byte gives the encrypted text first byte. That value will be considered as the first byte of plain text.

b) As we have the first byte of our plain text, the next step is to use this first byte of plain text and generate 128 number of plain texts where the second byte takes all the 128 values (ff to mu) and from 3 to 8-byte values are kept as zeroes (i.e ff). From the logic (4b) i.e â€œSecond byte of the ciphertext is dependent only on the 1st and 2nd byte of the plain text and so onâ€ we figured out which of our plain texts second byte gives

the encrypted text second byte. That value will be considered as the second byte of plain text. This process continues till we get all the 8 bytes of our plain text.

inputs.txt containts plaintexts
outputs.txt containts corresponding ciphertexts

Eg: For obtaining the 3rd byte of plain text; we craft 128 plain text values wherein all 128 plain texts; we already knew 1st and 2nd-byte values and 3rd byte takes all the 128 values (ff to mu) and from 4 to 8-byte values are kept as zeroes (i.e ff).

Our input having format is as follows:
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. Thus, we generated inputs of the form $C^{8-i}PC^{i-1}$ using the '**Generating Required Inputs**' section of the **MAIN.ipyn file** and the corresponding ciphertexts are produced by using **script.sh**.

> *Finding Preimage*
''

6. 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.

>. 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 = $(a_{i,i}\,(a_{i,i} * x^{ei})^{ei})^{ei}$

> 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 section **Brute force EAEAE Cipher** in **MAIN.ipynb**.

> 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.

$Block\ No.$	$Possible\ Values\ of\ a_i, i[0, :2]$	$Possi$
$b0$	$[27, 84, 84]$	
$b1$	$[122, 62, 70]$	
$b2$	$[105, 43, 107]$	
$b3$	$[9, 12, 6]$	
$b4$	$[73, 112, 57]$	
$b5$	$[51, 11, 53]$	
$b6$	$[27, 66, 70]$	
$b7$	$[38, 56]$	

>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 ai,i & aj,j to find ai,j. Thus, after this iteration we found every element next to each diagonal element.

>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.

$Block\ No.$	$Possible\ Values\ of\ a_i, i[0,:2]$	$Poss$
$b0$	84	
$b1$	70	
$b2$	43	
$b3$	12	
$b4$	112	
$b5$	11	
$b6$	27	
$b7$	38	

> 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.

> Thus, we found the Linear Transformation Matrix as follows:

$$\begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 115 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 29 & 43 & 0 & 0 & 0 & 0 & 0 \\ 124 & 17 & 13 & 12 & 0 & 0 & 0 & 0 \\ 97 & 32 & 6 & 111 & 112 & 0 & 0 & 0 \\ 25 & 52 & 28 & 44 & 110 & 11 & 0 & 0 \\ 20 & 120 & 23 & 100 & 3 & 89 & 27 & 0 \\ 77 & 13 & 84 & 28 & 13 & 66 & 24 & 38 \end{bmatrix}$$

And the Exponent Matrix found was as follows:

$$\begin{bmatrix}19 & 115 & 41 & 79 & 93 & 41 & 22 & 20\end{bmatrix}$$

Getting password from preimage

7. Since our password is a 32-bit string i.e. â€œ *gsjmluhphmfjgshriujmjumiklghlkmj*â€□, we have split into two halves and applied the above-mentioned brute force logic to obtain plaintext values for the corresponding ciphertexts.
Cipher Text: gsjmluhphmfjgshr & Obtained Plain Text: mimjmhmklkltlumg
Cipher Text: iujmjumiklghlkmj & Obtained Plain Text : lmlrifififififif

8. We converted "*mimjmhmklkltlumglmlrifififififif*" through ASCII conversion method for the above string and obtained â€œstruenoqgl000000â€□ after removing padding "**struenoqgl**" which got accepted to clear level-5.

Q5 Password
10 Points

What was the password used to clear this level?

struenoqgl

Q6 Code
0 Points

Please add your code here. It is MANDATORY.

main.zip	Download
1	Binary file hidden. You can download it using the button above.

Assignment 5

Graded

Group

TANEYA SONI
SONAM
ABHISHEK KUMAR PATHAK
View or edit group

Total Points

85 / 100 pts

Question 1

Team Name — **0** / 0 pts

Question 2

Commands — **5** / 5 pts

Question 3

Cryptosystem — **5** / 5 pts

Question 4

Analysis — **65** / 80 pts

Question 5

Password — **10** / 10 pts

Question 6

Code — **0** / 0 pts