Q1 Team Name
0 Points

flash

Q2 Commands
5 Points

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

1)go
2)wave
3)dive
4)go
5)read
6)password

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.)

#After using the go, wave, dive, go and read commands respectively we were presented with the magical screen that contains the following hints:

" 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 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 the 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..."

#This screen gives us the hint that "EAEAE" transformations are applied on the plaintext to encrypt it.

#Also it is written that whisper the "password". On using the command 'password' we got 'mshjhijrlufshtgqitfnmlhkkuhimgj' as an encoded password. We need to decrypt it in order to get the final command to clear the level.

#For the conversion of string to the binary format (i.e. representing it using 0 and 1), we did the analysis of the encrypted password and observed that it contains alphabets between 'f' to 'u'. There is a total of 16 letters between 'f' and 'u'. So we need a 4bit binary to represent each letter as a total of 16 (i.e., $2^4 = 16$) combinations are possible with the 4 bits. So we used the following representation for alphabets in 4bit:

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

#The finite field F_{128} has a cardinality of 128 so the range of pairs in plaintext should be between 'ff' to 'mu', (not uu) if we keep the first f fixed we can get 16 different options (f-u) for second place. Also for the first place, we have 8 options(f-m). So, in the total
we have $8 * 16 = 128$ elements between 'ff' to 'mu' .

#We manually tried some plaintexts for discovering some patterns. Some of the plaintext that we tried to encrypt are:

ff ff ff ff ff ff ff ff -> ff ff ff ff ff ff ff ff
fg ff ff ff ff ff ff ff -> fo ir gu hh gp ht ln mj
gg gg gg gg gg gg gg gg -> lg ji hr fs jt gp fi fh
hh hh hh hh hh hh hh hh -> iq ft mf fi hm if hr gu

ff gg gg gg gg gg gg gg -> ff ig kf kg gt ir kh hl

#With the help of the above activity, we made an observation that i^{th} block of the plaintext changes the i^{th} block of the ciphertext, and blocks after that, there is no change till $i-1^{th}$ block. It shows that i^{th} block affects some block j if and only if $i \geq j$. This shows that A could be a lower triangular matrix. We can draw the inference that each byte in plaintext affects the corresponding byte and the subsequent bytes in plaintext.

#We also observed that there is a one-to-one correspondence between plaintext and ciphertext.

#If we look at the binary representation of 'ff' will convert to 0 in decimal and binary representation of 'mu' will convert to 127 in decimal. So, we mapped them to their decimal representation.

#From the resources that were provided to us we can conclude that we need to perform structural cryptanalysis of EAEAE.

#In order to compute the matrix A and vector E we performed the below mentioned approach:

#We generated the input of the form C^iPC^{8-i-1} such that each input has at least one non-zero byte here possible values of P is from 'ff' to 'mu' (i.e., 0 to 127), value of C is 'ff' and value of i is from 0 to 7. We took this condition of atleast one non zero byte since if we have all zero bytes then the output will be all 0 bytes. We generated 128 x 8 plaintexts and write them in the plaintexts.txt

#Since, after getting the password if we write the plaintext in the server we will get the ciphertext corresponding to it. So, we obtained the encrypted text (ciphertext) for the corresponding plaintexts that we generated by automating the process of reading each input block from plaintexts.txt, writing it in the command line of the server, fetching the corresponding ciphertext, and writing it in ciphertexts.txt.

#As we figured out that A is lower-triangular matrix, so if the i^{th} non-zero input block has value y_i, then the corresponding output block can be calculated by $z_i = (a_{i,i} * (a_{i,i} * y_i^{e_i})^{e_i})^{e_i} - equation1,$ as mentioned in the hint to use "EAEAE" transformation. This is used in calculating the diagonal elements of A and elements of vector E.

For encrypting the plaintext, exponential and multiplication functions have been used one after the other. The addition is implemented using the bitwise XOR since the field is F_{128}. Values were calculated as mentioned above and were stored into 128*128 matrices. The above operations were used while carrying out the brute force attack on each of the $a_{i,j}, e_k$ to check whether encrypted output matched with the actual encrypted output.

We used the plaintext-ciphertext pairs that we generated and iterated over all possible elements e_i of exponentiation vector E and diagonal elements $a_{i,i}$ of A to see if inputs produce the same output as produced by the server. We add those values to a list of possible candidates for A and E.

#In order to calculate the non-diagonal elements of A and prune some pairs from the above result, we made use of more plaintext- ciphertext pairs and iterate over the above $(a_{i,i}, e_i)$ pairs for finding these elements between 0 to 127 so that equation1 is satisfied.
We can find element $a_{i\,j}$ with the help of i^{th} output block such that j^{th} input block is non-zero. To calculate $a_{i\,j}$ all elements of the following set $S_{i\,j}$ need to be known, where the set $S_{i\,j}$ can be written as:
$S_{i\,j} = \{a_{n,m} | n > m, j \leq n, m \leq i\} \cap \{a_{n,n} | j \leq n \leq i\}$. If we plot the elements of $S_{i\,j}$ they will form the right-angled triangle with vertices as $\{a_{i\,i}, a_{i\,j}, a_{j\,j}\}$.

#After the above step we discovered elements next to the diagonal and one element at each diagonal position for the vectors A and E.

#In order to obtain the remaining elements of A, we considered each possible value i.e., 0-127, and selected the values such that they satisfy the equation1 where we use the final values of A's diagonal elements and the exponent vector E, starting from

$a_{i+1,i}$ and so on. We can remove other possibilities from the previous list if we find $a_{i+1,i}$

Linear transformation matrix A comes out to be,

A=[[84, 0, 0, 0, 0, 0, 0, 0],
[119, 70, 0, 0, 0, 0, 0, 0],
[19, 31, 43, 0, 0, 0, 0, 0],
[96, 20, 0, 12, 0, 0, 0, 0],
[99, 56, 0, 114, 112, 0, 0, 0],
[18, 47, 18, 55, 97, 11, 0, 0],
[12, 124, 9, 102, 31, 92, 27, 0],
[64, 15, 76, 26, 16, 73, 3, 38]]

Exponentiation vector E comes out to:

E = [23, 113, 39, 75, 88, 44, 24, 28]

#We used the final key matrix A and final vector E for decrypting the password. As for getting the plain text, we applied the following transformation:

$$EncryptedPassword = E(A(E(A(E(Originalpassword)))))$$

Thus to get the original password we need to apply these transformation in reverse order so,

$$Originalpassword = E^{-1}(A^{-1}(E^{-1}(A^{-1}(E^{-1}(Encryptedpassword)))))$$

We partitioned the password into two blocks and applied the transformation mentioned above in each block.

As a result for each block of encrypted password we have the following decrypted values for the original password:

Block1: [119, 114, 112, 113, 122, 113, 113, 121]
Block2: [101, 105, 48, 48, 48, 48, 48, 48]

Since earlier we have mentioned that we represented pairs from 'ff' to 'mu' with values from 0 to 127. Thus doing reverse mapping of the numbers to these pairs we got the original

password in the string format.
119 ->mm, 114 -> mh, 112 -> mf, 113 -> mg, 122 -> mp, 113 -> mg, 113 -> mg, 121 ->mo, 101-> lk, 105 -> lo, 48 -> if, 48 -> if, 48 -> if, 48 -> if, 48 -> if, 48 -> if

So the password comes to 'mmmhmfmgmpmgmgmolkloififififififif'.But on trying that password we got an unknown command message. That means it is also decrypted. Then I tried encrypting them considering ASCII values as they are between 0 to 127.

119 -> w, 114 -> r, 112 -> p, 113 -> q, 122 -> z, 113 -> q, 113 -> q, 121 -> y, 101-> e, 105 -> i, 48 -> 0, 48 -> 0, 48 -> 0, 48 -> 0, 48 -> 0, 48 -> 0

Using this we obtained "wrpqzqqyei000000" as the password. We tried this but it didn't work so we thought that maybe some extra zeroes are padded as it was in the previous assignment. So we again tried after removing 0's that worked for us to clear the level. So our final password is "wrpqzqqyei"

We did all the tasks related to this assignment in the Flash.py file. Also, we provided a make file. You can run it with the help of readme.

References -
https://www.researchgate.net/publication/2386882_Structural_cryptanalysis_of_SASAS

No files uploaded

Q4 Password

5 Points

What was the final commands used to clear this level?

wrpqzqqyei

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.

Flash.zip Download

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

Assignment 5

● GRADED

2 DAYS, 11 HOURS LATE

GROUP
Ravi Shankar Das
Nimit Jain
Prashant Mishra
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 **50** / 50 pts

QUESTION 4
Password **5** / 5 pts

QUESTION 5
Codes **0** / 0 pts