Q1 Team Name

0 Points

Group Name

team_9

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 => wave => dive => go =>read =>password(to get the encrypted password)
after decryption type actual password without the padding zeroes.

Q3 Cryptosystem

5 Points

What cryptosystem was used at this level?

EAEAE form of cryptography where E is Exponentiation and A is Linear transformation with 8X8 matrix and both E and A is part of key.

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.

Cryptanalysis of the Cryptosystem

1. This particular form of Cryptography uses a pattern based encryption involving linear and Affine transformations of input texts. As earlier the magical wand and the spirit helped by giving the hints in order to decrypt the cipher. By observing multiple inputs and outputs we got that the cipher text contains letters only from f to u. As the input block is of 8 bytes and the output is also 8 bytes. We map 16 letters from f to u in chronological order with 4 bits each from 0000 to 1111 forming two letters equal to 1 byte. Also from the hint, it was given that each byte is an element of the field F_{128} in range 0 to 127, so the MSB is always 0. Therefore possible letter pairs are from $'ff'$ to $'mu'$. Also from the each element of the input vector over F_{128} constructed using the degree 7 irreducible polynomial $x^7 + x + 1$ over F_2. This modulo operation we take care while doing the element wise multiplication of the matrix.

2. **Observations**

(a) If input plaintext is $''ffffffffffffffff''$ then output is also $''ffffffffffffffff''$.

(b) If first i bytes of plaintext is$f$ then first i bytes of cipher text is also f.

(c) If we change the i^{th} byte of the plaintext $''ffffffffffffffff''$then the output will change from the i^{th} byte onwards. That means for any two inputs if the dissimilarity is after i^{th} byte then the same pattern of dissimilarity can be observed after i^{th} byte only.

From the above observations it is clear that matrix A $(8X8)$ is a lower triangular matrix.

3. **Calculations**

The matrix A is of dimension $(8X8)$ and E is of dimension $(8X1)$.

Let $a_{i,j}$ ϵ A where i is row index and j is column index and let e_i ϵ E.

We proceeded with the cryptanalysis with choosen plaintext attack to see how encryption took place block by block. In order to do this we started generating the plaintexts by **plaingen.py**, which generated 8 sets of 128 plaintexts each of size 8 bytes long. These plaintexts are stored in **plaintexts.txt**. The 16 letters were changed in a format where we have changed the pair of letters in each iterations. For the first set, First iterations we have proceded to change by (fff*,fgf*,fhf*.....fuf*) and second iteration (gff*,ggf*,ghf*.....guf*) and so on till (mff*,mgf*,mhf*....muf*).

<u>*Note:*</u> *f* denotes all letters are f after this till 16th letter*

In the second set we have shifted the similar changes for 3th and 4th letters and then for the third set these changes were done for 5th and 6th letters and this way we have prepared 8 sets of plaintexts and each set contains 128 plaintexts. So total no of plaintexts are $(8X128 = 1024)$. This is min requirement of plain-ciphertext pair for choosen plaintext attack in order to break the cipher because each letter is represented by 4 bits so 2^4 combinations are possible for each letter and each block of plaintext or ciphertext is of 8 bytes or 2^6 combinations. Then we proceeded for getting the corresponding ciphertexts from the server by running **server.py**, and the output is stored in **ciphertexts.txt** file.

As we know that matrix A is a lower triangular matrix cipher text C was found by the following operation on plaintext P

$C = (A * (A * (P)^E)^E)^E$

So, we started finding the possible diagonal elements using brute force method. Referring the hint given in the question we define the **exponentiate, linearTransform** functions over field F_{128}. We have used these libraries *from pyfinite import ffield* and *import galois* to define the field F_{128} and then we declare $'0x83'$ as generator to ensure the modulo operation is perform by the irreducible polynomial $X^7 + X + 1$. To find out the possible diagonal elements of matrix A we exponentiate the input strings by iteration over values $[1, 126]$ for each of the input strings from $[0, 127]$ for 8 sets of input. Then we match with the corresponding ciphertexts, if the value matches then we store the value of the diagonal element and corresponding exponent value. Then we got the list of possible values of diagonal elements of matrix A i.e $a_{i,i}$ and the corresponding exponent values i.e e_i. The list of the possible values are given below:-

Set No.	Possible diagonals	Possible Exponents
1.	$[84, 87, 7]$.	$[23, 48, 56]$
2.	$[77, 70]$.	$[20, 108]$
3.	$[108, 43, 86]$.	$[29, 43, 55]$
4.	$[124, 12, 88]$.	$[51, 80, 123]$
5.	$[28, 112, 62]$.	$[58, 86, 110]$
6.	$[17, 110, 11]$.	$[29, 43, 55]$
7.	$[123, 27, 63]$.	$[18, 21, 88]$
8.	$[38, 11, 58]$.	$[29, 43, 55]$

Then we started checking for exact diagonal elements and the corresponding exponents by taking each individual plain-ciphertext pairs. In the code, firstly we compute each pair of hexadecimal digits in the cipher text, converts them into an integer, and then converted those integers to compute the corresponding ASCII character. For each iteration of the outermost loop, we tried all possible combinations of exponents and diagonal values for the two adjacent diagonal elements of the matrix, and all possible values for the off-diagonal element between them. Then, we checked if resulting matrix encrypts the input string into the output string, using the formulas for the encryption algorithm over the finite field F_{128}. If it does, the possible values of the diagonal and off-diagonal elements are stored along-with the corresponding exponents.
The values of each byte in the plaintext and their corresponding $a_{i,i}s$ and e_is are given in the list below:-

$ByteNo.$	$a_{i,i}Values$	$e_iValues$
1.	84	23
2.	70	108
3.	43.	43
4.	12.	80
5.	112.	86
6.	11.	55
7.	27.	21
8.	38.	29

In the EAEAE function we have taken the input and output pairs and this function returns the Linear Transformation Matrix A and the Exponent vector E. As we have the Exponent values and the diagonal element values, we compute the EAEAE operation on each input strings(on corresponding ASCII) and then the off-diagonal values are calculated. The matrix A and Vector E is given below:-

```
      [[ 84   0    0   0     0     0   0   0]
       [119  70    0   0     0     0   0   0]
       [ 20  27   43   0     0     0   0   0]
       [125  26   31  12     0     0   0   0]
A  =   [ 96  57   71  23   112     0   0   0]
       [ 31  42   25  51    98    11   0   0]
       [ 23 124   14  98     2    82  27   0]
       [ 95  12   77  29    25    68   6  38]]
```

E = $[23, 108, 43, 80, 86, 55, 21, 29]$

From the game Server when we entered the the word $"password"$ we got the cipher text as $"mshsmqmomsjsfokmmnjtjsgukqikhriq"$. As we know two letter forms a byte of the input block and the size of input block is 8 bytes the actual password must have been padded with extra $0's$. So, we divided this encrypted password into two equal blocks as $"mshsmqmomsjsfokm"$ and $"mnjtjsgukqikhriq"$. Then we have performed the inverse operation to decrypt the actual password i.e $P = E^{-1}(A^{-1}(E^{-1}(A^{-1}(E^{-1}(C)))))$
and we get the password plaintext in two blocks given below:
$1.''wmtvxbnz''$
$2.''hz000000''$
We remove the extra padded $0s$ from the second block and combine them to get the actual password as $'wmtvxbnzhz'$ and entered this to clear the level.

Q5 Password
10 Points

What was the password used to clear this level?

"wmtvxbnzhz"

Q6 Code
0 Points

Please add your code here. It is MANDATORY.

Assignment_5_Final.zip Download

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

Assignment 5

● Graded