

## Q1 Teamname

0 Points

CryptoCreeks

## Q2 Commands

5 Points

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

go -> wave -> dive -> go -> read ->  
password

## Q3 Analysis

50 Points

Give a detailed description of the cryptanalysis used to figure out the password. (Explain in less than 100 lines and 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.)

On the first screen, we heard the sound of a rock hitting the water, and then we also heard the splashing sound. So we thought that the passage seems to be leading to some underground well and hence we used the command "*go*". After then we accidentally hold on to a rock, but there was nothing to grab, so we used the command "*wave*". We couldn't find anything by swimming in both the directions, so we used the command "*dive*". Then after that we saw a well lit passage carved into the wall which appeared to be leading somewhere deep inside and hence we used the command "*go*". Then we saw a glass panel and used the command "*read*". There were got our problem statement.

From there we got to know that we are required to apply the given

transformations in the sequence  $EAEAE$  on the input block to obtain block, where  $E$  is a  $8 \times 1$  vector whose elements are numbers between 1 and 126 and it is used to apply exponentiation on a block by taking the  $i$ th element of the block and raising it to the power given by  $i$ th element in  $E$ , and  $A$  is an invertible  $8 \times 8$  key matrix with elements from  $F_{128}$ . So Let

$$E = [e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7], \text{ and}$$

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{70} & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

On whispering "*password*", we got the coded password as:  
**ijfomumrlqiljikgmngfiuiriffnjtkp**

After observing ciphertexts of many inputs, we found that the ciphertexts consists of letters from  $f$  to  $u$  only. That means, only 16 characters are being used, which can be represented by 4bits in binary. So we used the mapping such that  $\{f : 0000 \dots u : 1111\}$ . But, it was given that the Field is of size 128, so each byte can be from  $ff$  to  $mu$ . So, we thought to use the plaintexts of the similar form. We also noticed that each byte from  $ff \dots mu$  mapped into some unique ciphertext byte.

Since the password contains 32 characters, we need to divide it into 2 blocks of 16 character each (since blocksize = 64 bits and each character is represented using 4 bits).

$encrypted\_password1 = \text{ijfomumrlqiljikg}$   
 $encrypted\_password2 = \text{mngfiuiriffnjtkp}$

We analyzed several ciphertexts and observed that on changing the  $i^{th}$  byte of the input, all the bytes of the output from  $i^{th}$  byte got changed. From this, we concluded, that the matrix  $A$  is a lower triangular matrix. Therefore, our  $A$  matrix will be like :

$$A = \begin{bmatrix} a_{00}, 0, 0, 0, 0, 0, 0, 0 \\ a_{10}, a_{11}, 0, 0, 0, 0, 0, 0 \\ a_{20}, a_{21}, a_{22}, 0, 0, 0, 0, 0 \\ a_{30}, a_{31}, a_{32}, a_{33}, 0, 0, 0, 0 \\ a_{40}, a_{41}, a_{42}, a_{43}, a_{44}, 0, 0, 0 \\ a_{50}, a_{51}, a_{52}, a_{53}, a_{54}, a_{55}, 0, 0 \\ a_{60}, a_{61}, a_{62}, a_{63}, a_{64}, a_{65}, a_{66}, 0 \\ a_{70}, a_{71}, a_{72}, a_{73}, a_{74}, a_{75}, a_{76}, a_{77} \end{bmatrix}$$

So, by using the input format  $C^{i-1}PC^{8-i}$ , we generated the plaintexts of this form.

And for each plaintext, we generated its corresponding ciphertext. Here,  $C$  represents string  $ff$  and  $P$  represents any byte string in  $\{ff\dots mu\}$ . So, our input format is of type  $\{ff\}^{i-1}\{p_1p_2\}\{ff\}^{8-i}$ , where  $p_1p_2$  is any one element of the set  $\{ff\dots mu\}$ .

We automated the process of extracting the required ciphertexts using a function. This function, runs the ssh command by taking an input text file which contains the commands as well as plaintexts. The output gets stored into a list, from which the function extracts the ciphertexts.

We used two approaches to find the required decrypted password ::

## APPROACH 1 ::

From the input format we chose, we will have only 1 non-zero block per input. We can use that to find the diagonal elements of the key matrix  $A$ . For any input, if  $i^{th}$  block is the non-zero block, then the corresponding output block can be obtained via the formula,  $(a_{ii}(a_{ii} * x^{e_i})^{e_i})^{e_i}$ . Now, we can find the values of the diagonal elements i.e.  $a_{ii}$  by iterating over the values 0-127 and the elements of the  $E$  matrix  $e_i$  by iterating over the values 1-126.

After this we obtained possible values for each diagonal element of matrix  $A$ , which are as shown below::

$$a_{00} : \{84, 28, 113\}$$

$$a_{11} : \{47, 70, 70\}$$

$$a_{22} : \{43, 78, 87\}$$

$$a_{33} : \{68, 95, 12\}$$

$$a_{44} : \{47, 112, 96\}$$

$$a_{55} : \{5, 11, 121\}$$

$$a_{66} : \{52, 27, 27\}$$

$$a_{77} : \{38, 35, 39\}$$

For matrix E, we got the following possible values:

$$e_0 : \{18, 21, 88\}$$

$$e_1 : \{1, 19, 107\}$$

$$e_2 : \{36, 42, 49\}$$

$$e_3 : \{17, 41, 69\}$$

$$e_4 : \{65, 92, 97\}$$

$$e_5 : \{36, 42, 49\}$$

$$e_6 : \{1, 19, 107\}$$

$$e_7 : \{18, 21, 88\}$$

We can find the non-diagonal elements of matrix A, using the values of the diagonal elements which we have already obtained above. We can find the value of  $a_{ij}$  (for  $i > j$ ) using the value of the elements  $a_{ii}$  and  $a_{jj}$  by iterating over them in a triangular fashion. During this process, we also get the exact value of each diagonal element and the elements just below it in matrix A from the possible values we had found earlier.

For the rest of the elements of A, which we have not got yet, we iterated over all the values from 0-127 and used the EAEAE technique over the available matrix A and exponentiation matrix E, and used the formula  $(a_{ii}(a_{ii} * x^{e_i})^{e_i})^{e_i}$  over various plaintext and ciphertext pairs to check if the equation holds true. This gave us all the remaining values of the lower triangular key matrix A.

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 124 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 16 & 43 & 0 & 0 & 0 & 0 & 0 \\ 102 & 16 & 2 & 12 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
[110 , 38, 2,122,112, 0, 0, 0],
[25 , 51, 29, 45, 111, 11, 0, 0],
[ 2 , 119, 15, 105, 25, 88, 27, 0],
[115 , 2 , 82, 24, 24, 66, 31, 38]]
```

$E =$  [18, 107, 36, 69, 92, 42, 19, 18]

Hence , now we have both the E and A matrix available with us and now using these matrices we can decrypt both the blocks of password.

We got the following results:

**decrypted\_password1** = mhlrlslqljlulrjt  
**decrypted\_password2** = lilgifififififkq

We tried this but it did not work, so we tried to convert them using ASCII character codes taking 2 characters at a time. We first converted the *decrypted\_password* into 0s and 1s using the mapping {*f* : 0000.....*u* : 1111}, and then we converted it into ASCII characters taking 8 bits at a time. We then got the password as: **rlmkdoNoca00000** [. This we tried submitting as our password but it didn't get accepted. Also we saw the letter "N" and "[" appeared to be different from the rest of the values. We observed that we got both these values from the last 2 characters of *decrypted\_password1* and *decrypted\_password2*, respectively. So we tried to bruteforce on those last 2 characters of both the halves. For the last 2 characters of **decrypted\_password1**, we iterated over the values from *ff....mu* to find the plaintext which will generate the ciphertext as *encrypted\_password1* = **ijfomumrlqiljikg**. We got the resultant plaintext as **mhlrlslqljlulrlu**. We repeated this for last 2 characters of *decrypted\_password2* as well and we got the resultant plaintext as **lilgifififififif**. So we got

**decrypted password : mhlrlslqljlulrlulilgififif**

We converted it using ASCII character codes, and got **rlmkdoloca000000**. This also didn't work, so we removed the trailing 0s, and that got accepted.

**final password : rlmkdoloca**

## APPROACH 2 ::

As discussed above , we have a lower triangular matrix  $A$  , which means on changing the  $i_{th}$  block of the input , all blocks from  $i$  and onwards get changed , which implies that every block depends only on its previous blocks. Since we know that for each byte we can have values from  $ff....mu$ , so we can iterate over these values for each byte to find the value which generates the required ciphertext for the corresponding plaintext.

Like, for first half of our encrypted password, i.e.,

*encrypted\_password1*, the first two bytes are **ij**, so we can iterate over  $ff....mu$  to get such a plaintext byte  $p_1p_2$  which generates the  $1^{st}$  byte of ciphertext as **ij**. We got the  $1^{st}$  plaintext byte as **mh**. Now we can fix this byte, and again iterate over  $ff....mu$  to find the characters of  $2^{nd}$  byte of plaintext as  $p_1p_2$  so that the plaintext **mhp<sub>1</sub>p<sub>2</sub>** generates the ciphertext with  $1^{st}$  two bytes as **ijfo**. We got first 4 bytes of plaintext (*decrypted\_plaintext1*) as **mhlr** .Now we can repeat this for the rest of the bytes fixing the previous bytes as described to get the required decrypted password. We will repeat this procedure for second block of encrypted password ,i.e., *encrypted\_password2*.

We get the following results:

**decrypted\_password1 = mhlrlslqljlulrlu**

**decrypted\_password2 = lilgifififififif**

**decrypted password : mhlrlslqljlulrlulilgififif**

We tried this but it did not work, so we tried to convert them using ASCII character codes taking 2 characters at a time. We first converted the *decrypted\_password* into 0s and 1s using the mapping  $\{f : 0000.....u : 1111\}$ , and then we converted it into

ASCII characters taking 8 bits at a time. We then got the password as: **rlmkdoloca000000**. This again did not work, so we removed the trailing 0s and got the final password.

**final password : rlmkdoloca**

**NOTE : All the operations used here are according to tl**

 No files uploaded

## Q4 Password


5 Points

What was the final command used to clear this level?

rlmkdoloca

## Q5 Codes

0 Points

 No files uploaded

# Assignment 5

● GRADED

### GROUP

PRADEEP KUMAR TIWARI

TANISHA RASTOGI

SHRUTI WASNIK

 [View or edit group](#)

TOTAL POINTS

**60 / 60 pts**

QUESTION 1

Teamname **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**