Q1 Team Name

0 Points

Group Name

ela

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

Q3 Cryptosystem

5 Points

What cryptosystem was used at this level?

EAEAE cipher was used at this level
E is exponentiation function
A is Linear Transformation in finite fields

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.

The encryption used is a block cipher 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. The two transformations are as follows:
i) A linear transformation given by invertible 8 x 8 key matrix A with elements from F_{128}
ii) An exponentiation vector given by 8 x 1 vector E whose elements are numbers between 1 and 126, and E is applied 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.

The input “password” gives us output “jghrlgmnhsgfffjkgkikhrmjkrgigjmu”.

Observations from Input-Output pairs:

The analysis of a few ciphertexts revealed that the encoding involves 16 characters from 'f' to 'u'. Hence it can be concluded that the letters can be mapped to hexadecimal numbers from 0-15 as follows: f = 0000; g = 0001; h = 0010 i = 0011; j = 0100; k = 0101; l = 1011; m = 0111; n = 1000; o = 1001 p = 1010; q = 1011; r = 1100; s = 1101; t = 1110; u = 1111

Hence, every byte would consist of a pair of characters from the above list. On further analysis, we observed that ciphertext can consist of 128 pairs of characters starting from “ff”, “fg”, “fh” and so on upto “mu”. Here, odd positions will include characters from f - m, whereas even positions will contain characters from f - u. We end at “mu” since F_{128} contains only 128 elements. So, “ff” to “mu” can be mapped to 0-127 in the decimal system.

After further analysis of the plaintexts and ciphertexts pairs, following observations were made on passing as input to the cryptosystem, an 8x1 vector over F_{128}:
i) When the input plaintext was all 0's i.e all bytes have ‘ff’, the ciphertext was all 0's as well
ii) If first k bytes of input plaintext were “ff” i.e 0, the first k bytes of ciphertext were also 0
iii) If we just changed the k_{th} byte of the input, the output begins to change from k_{th} byte onwards.

Hence, it can be assumed that the transformation matrix A might be a lower triangular matrix.
Explaination: Consider
i) $p_0, p_1, \ldots.., p_7$ as the 8 byte input plaintext
i) $c_0, c_1, \ldots.., c_7$ as the 8 byte output ciphertext

Now “ff” maps to 0.

When input is “ff ff ff ff ff ff ff ff ff” the output is also “ff ff ff ff ff ff ff ff ff” i.e.,

$$\begin{bmatrix} ff \\ ff \\ ff \\ ff \\ ff \\ ff \\ ff \\ ff \end{bmatrix} \Rightarrow \begin{bmatrix} ff \\ ff \\ ff \\ ff \\ ff \\ ff \\ ff \\ ff \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

When input is “ff ff ff fg ff ff ff ff ” the output is “ff ff ff lh jk lu mj gh ” i.e.,

$$\begin{bmatrix} ff \\ ff \\ ff \\ fg \\ ff \\ ff \\ ff \\ ff \end{bmatrix} \Rightarrow \begin{bmatrix} ff \\ ff \\ ff \\ lh \\ jk \\ lu \\ mj \\ gh \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ * \\ * \\ * \\ * \\ * \end{bmatrix}$$

Here ‘*’ corresponds to some non-zero value.

Hence, when the input text has a non-zero byte p_k, and rest bytes 0, then it would result in $c_0, \ldots, c_{k-1}$ bytes as 0 and $c_k, \ldots, c_7$ bytes to be corresponding non-zero bytes, i.e., the ciphertext changes from k-th byte onwards.

In other words, for any byte p_k, changing any byte p_j such that $j > k$ has no effect on the corresponding ciphertext byte c_k. However, changing byte p_j such that $j < k$ changes the corresponding ciphertext byte c_k.

Next we need to decipher the exponentiation transformation E as well as linear transformation A. A is the key matrix of 8x8 elements, and its elements are referred to as $a_{i,j}$, where i and j represent the element's row and column, respectively. Also, e_i is the i-th element of the 8x1 vector E.

Plaintext and Ciphertext pairs generation:

The inputs of the form $C^{-1}PC^{8-k}$ were generated using the file "plaintext_generation.py" and stored in "plaintext.txt". In simple words, the input had only one non zero block at a time. Every block pick has 128 possible plaintext values, i.e., from ff to mu. We generated 8 sets of 128 input plaintexts, each having only one non-zero byte, i.e., 128 x 8 plaintexts were employed in the attack.

The file "generating_script.py" generates "script.sh" from the text file "plaintext.txt." We make a temporary file called "output.txt" with "command.sh. Next using "cipertext_generation.py" we generated and stored ciphertexts in "ciphertext.txt".

Finding the Matrices A and E:

Now we know that A is a lower-triangular matrix, hence, if x be the i-th non-zero input block value, then the corresponding output block will be having the value $(a_{i,i} * (a_{i,i} * x^{e_i})^{e_i})^{e_i}$. The diagonal element of vector A, as well as all of the elements of vector E, can then be computed. In addition, if the linear transform is lower-triangular, then the i-th output block is dependent on the j-th input block if and only if $i >= j$.
Now we iterate over all possible values of diagonal elements $a_{i,i}$ (from 0-127) of Matrix A and elements e_i (from 1-126) of exponential vector E, and note down values for which input maps to output. For each block, we get three tuples of values.

Block Number	Possible pairs of $a_{i,i}$ and e_i
0	[(84, 24), (46,28), (96,75)]
1	[(127,59), (9,90), (70,105)]
2	[(108,29), (43,43), (86,55)]
3	[(68,15), (95,31), (12,81)]
4	[(73,45), (112,93), (57,116)]
5	[(106,10), (11,54), (70,63)]
6	[(123,18), (27,21), (63,88)]
7	[(38,24), (33,28), (53,75)]

Next we need to find the non-diagonal elements of A as well as eliminate some pairs from the above result. Using more plaintext ciphertext pairs we iterate over the above $a_{i,i}$ and e_i pairs to find these elements in the range 0-127 in a way that satisfies the transformation equation.

Element $a_{i,j}$ is obtained by looking at the i-th output block where the corresponding j-th input block is non-zero. To find $a_{i,j}$, all of the components of set $S_{i,j}$ must be known. The set $S_{i,j}$ is provided by:

$$S_{i,j} = \{a_{y,x} | y > x, j <= y, x <= i\} \cap \{a_{y,y} | j <= y < i\}$$

These elements can be graphically visualized as forming a right- angled triangle with corners at $\{a_{i,i}, a_{i,j}, a_{j,j}\}$. The elements adjacent to each diagonal element are then known, in addition to one element at each diagonal location of matrix A and vector E. As a result, we get the following final pairs:

Block Number	Final pairs of $a_{i,i}$ and e_i
0	[(84, 24)]
1	[(70,105)]
2	[(43,43)]
3	[(12,81)]
4	[(112,93)]
5	[(11,54)]
6	[(27,21)]
7	[(38,24)]

Alternate Multiply and Exponentiate functions were utilised to implement this. The addition in F_{128} is analogous to the XOR operation in integers. The values were stored in 128*128 matrices after modifying the above functions accordingly. We then defined our Encrypt function, which was utilised during the brute force attack on each of the $a_{i,j}, e_k$ pairs to see if our encrypted output matched the actual encrypted output.

Using the final values of the diagonal components of A and the exponent vector E, we iterate through all possible values (0-127) for the remaining elements of A and remove those values that do not match the transformation equation, starting with $a_{i+1,1}$ and continuing in this fashion, and eliminating the extra possibilities from previous list. "eaeas.ipynb" contains the code needed to do these tasks.

The final key matrix A is:

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 118 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 27 & 27 & 43 & 0 & 0 & 0 & 0 & 0 \\ 122 & 17 & 30 & 12 & 0 & 0 & 0 & 0 \\ 96 & 60 & 12 & 113 & 112 & 0 & 0 & 0 \\ 28 & 49 & 18 & 46 & 99 & 11 & 0 & 0 \\ 22 & 125 & 1 & 98 & 5 & 89 & 27 & 0 \\ 94 & 12 & 72 & 22 & 19 & 71 & 29 & 38 \end{bmatrix}$$

The Exponentiation vector E is:

E = [24, 105, 43, 81, 93, 54, 21, 24]

Decrypting the Password:

Using these E and A we decrypted the actual password. We assumed the input blocks were unknown. The encrypted password was separated into two blocks. Then, each block of encrypted password was subjected to the transformation: E^-1(A^-1(E^-1(A^-1(E^-1(p))))); where p is the encrypted password.
The transformations gave us the password blocks as follows:
[120, 106, 116, 119, 101, 97, 110, 117] and [97, 102, 48, 48, 48, 48, 48, 48]

The code for this can be found in the "eaeas.ipynb". We tried mapping the numbers back to their corresponding character pairs using our initially defined map, however it wasn't the correct password. Hence we tried using the standard ASCII table to map these values back to their corresponding characters. Hence, we obtained the string xjtweanuaf000000. Assuming 0's to be padding, we omitted them and finally obtained the plaintext password: xjtweanuaf

Q5 Password
10 Points