CS641 Assignment 5

Instructor

Dr. Manindra Agrawal

Authors

Roll No	Name	Email
200771	Rashmi Waghmare	rashmibw20@iitk.ac.in
200291	Vidish Chandekar	cvvijay20@iitk.ac.in
200419	Harsh Saroha	harshsar20@iitk.ac.in

Feb 18, 2023

1 Question

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 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.

2 Solution

Use the following command to reach the ciphertext password: ***go -> wave -> dive -> go -> read -> password***. Cryptosystem used in this level is ***EAEAE Cipher*** which is a weaker variation of ***AES (Advanced Encryption Standard)***. The following strategy was used to break this level:-

1. The 8 x 1 vector looks as follows

$$v = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_8 \end{bmatrix}$$

   where b_i represents 1 byte (8-bits).

2. From the first line of question we can conclude that last 4 bits of b_i range from 0000 to 1111 and first 4 bits range from 0000 to 0111. Hence b_i ranges from 00000000 (0 in decimal) to 01111111 (127 inn decimal). We know from frequency analysis that f : 0000 and u : 1111 so b_i in terms of plaintext ranges from *ff* to *mu*.

3. We observed that the first i bytes of the ciphertext came out to be zero whenever a chosen plaintext of a 8 byte vector was passed with the first i bytes equal to zero. Also, changing ith byte of input caused a change only in the bytes after the ith byte in the output. We therefore concluded that the key matrix A used is a *lower triangular matrix*. Here it is worth noting that when a byte is 0, it implies that its plaintext value is *ff*.

4. Hence we used ***generate_inputs.py*** to generate all vectors where only one byte ranges from 0 to 127 (and all other bytes are 0) and stored it in ***input.txt***. Then we used ***input_to_alphabet.py*** which converts numbers from ***input.txt*** to binary (stored in

bin_input.txt and then use ***bin_input.txt*** to convert it into plaintext format using the same mapping (f to u) ans store it in ***alphabet_input.txt***.

5. We wrote a script file ***script.sh*** which feeds input to server (***alphabet_input.txt***) and fetches corresponding output stored in ***output.txt***.

6. Files ***alphabet_input.txt*** and ***output.txt*** are then fed to ***refined_input.py*** and ***refined_output.py*** in order to refine the format for further analysis. They are respectively stored in ***refined_input.txt*** and ***refined_output.txt***.

7. Say b_i is the non-zero byte of the vector b. If we trace the mathematical manipulation of b_i, we can see that the corresponding byte of the ciphertext will always be

$$c_i = (a_{i,i} \cdot (a_{i,i} \cdot b_i^{e_i})^{e_i})^{e_i}$$

and the subsequent bytes from i will be changed. $a_{i,i}$ is the ith diagonal element of A and e_i is the ith element of E.

8. So for each value of $a_{i,i}$ (from 1 to 127 and e_i (from 1 to 126), we iterate over all plaintext-ciphertext pairs, compare them and store the possible values of $a_{i,i}$ and e_i. This is done using ***ae.py***. This will give us the following output:-

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 119 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 30 & 43 & 0 & 0 & 0 & 0 & 0 \\ - & - & 7 & 12 & 0 & 0 & 0 & 0 \\ - & - & - & 118 & 112 & 0 & 0 & 0 \\ - & - & - & - & 96 & 11 & 0 & 0 \\ - & - & - & - & - & 89 & 27 & 0 \\ - & - & - & - & - & - & 28 & 38 \end{bmatrix}$$

$$E = \begin{bmatrix} 23 & 118 & 38 & 76 & 93 & 47 & 24 & 23 \end{bmatrix}$$

9. We then used ***finding_matrix.py*** to find the remaining elements of the matrix A. We do so by brute forcing all the possible values of $a_{i,j}$ and checking which value satisfies for the same plaintext-ciphertext pairs (from ***refined_input.txt*** and ***refined_output.txt***).

We hence get the following output:-

$$A = \begin{bmatrix} 84 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 119 & 70 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14 & 30 & 43 & 0 & 0 & 0 & 0 & 0 \\ 97 & 17 & 7 & 12 & 0 & 0 & 0 & 0 \\ 99 & 35 & 13 & 118 & 112 & 0 & 0 & 0 \\ 24 & 48 & 29 & 32 & 96 & 11 & 0 & 0 \\ 9 & 124 & 20 & 103 & 30 & 89 & 27 & 0 \\ 3 & 15 & 87 & 23 & 20 & 67 & 28 & 38 \end{bmatrix}$$

$$E = \begin{bmatrix} 23 & 118 & 38 & 76 & 93 & 47 & 24 & 23 \end{bmatrix}$$

10. We now plug these values of A and E in ***decrypting.py*** in order to find the password. We do so by simple brute force. For the following ciphertext:

$$msjjkrhffnfmhqfrgthtmghjhkfsghkq$$

the password/plaintext was found to be:

wworetqtbe

It should be noted that we will actually get a 16 decimal password which has to be converted to ASCII. Doing so we will realize that a lot of them are 0s which are to be ignored.