CS641: THE GREAT CAVES

DEDSEC

Guntas Singh Brar 180274

Divyanshu Bhardwaj 180253

Devanshu Gupta 180233

March 8, 2020

Chapter 5 :The Spirit

Reaching the Cipher Text

- We enter a passage with very little light. We try to walk and are stuck in a steeping passage holding both sides of the wall.
- We attempt to go back but eventually fall in what seems to be a vertical tunnel. We "go" but we die.
- So next time, we "wave" the wand which seemed to be the only sensible option. We slow down magically and we fall in water where there's only rock walls ashore.
- We "dive" into the water and find light coming out of a hole. We enter it and find a passage with torches. We "go" further the passage ends in a small door into a hall with marbles and chandelier. There is an exit with a glass panel. We "read" the glass panel but its blank. We then hear the spirit whispering in our ears:

  "This is another magical screen. 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 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 i^{th} element of the block and raising it to the power given by i^{th} 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..."

Cracking The Cipher Text

- So as we know it is EAEAE Encryption. We tried out random inputs on the encryption and it was evident that all the characters in the output were from "f" to "u".
- Also for 8 character long plain text(64bit), we get cipher text 16 character long. The binary output of the encryption surely will be 64bit long(and the cipher text has to be the ASCII translation of binary into characters, it should've been 8 character long)
- One character for every 4 bits(16bit long cipher-text) means that it must be hexadecimal. But we don't see any numbers in the cipher-text for any plain-text.So we conclude that there is a mapping of "0000" to "1111" to "f" to "u" like it was in the last chapter.
- Also "a", "af", "aff" and so on give the same output meaning that the input is padded with "f".
- Now the input block size is 8bits but in a finite field GF_{127} so the inputs can be from "ff" to "mu"(Taking the most significant bit as 0).

*

- Now we tried various input formats like C^7P, C^6PC^1, where C means corresponding bit takes constant value and P means it can take different values, and we saw that on changing the n^{th} byte, only the bytes after the n^{th} byte changed and the rest were the same.
- The above behaviour gave us an intuition that the transformation matrix used here must be a lower triangular matrix.
- Say for input text $T_0T_1T_2T_3T_4T_5T_6T_7$ we have the corresponding cipher-text $C_0C_1C_2C_3C_4C_5C_6C_7$
- Now we take plain-text with T_0 to T_i as "ff" and we get corresponding cipher text with C_0 to C_i as "ff".
- Also cipher-texts corresponding to plain-texts $T_0T_1...T_iT_{i+1}...T_7$ and $T_0T_1...T_iT^{'}_{i+1}...T_7$ differ only after the i^{th} bit meaning that the 'ff' in each row should be at the end. Hence, a Lower Triangular Matrix
- We generate inputs of the form $C^{8-i}PC^{i-1}$.
- Exponent Matrix has elements between 1 to 126 and Linear Transformation Matrix has elements from GF_{128} which is Lower Triangular.
- Due to the input format we used, only 1 block is non-zero per input, so we can iterate over all possible values of diagonal elements and exponents, since the matrix is lower triangular if x is the value of non-zero input block (say i), then the corresponding block of output has the value $O = (a_{i,i}(a_{i,i} * x^{ei})^{ei})^{ei}$.
- Now, we need to eliminate pairs and also find non-diagonal elements. To do this we will use some more plaintext-ciphertext pairs and iterate over the above pairs to find element within 0 to 127 such that the equation O holds.
- We do this in triangular fashion such that we use $a_{i,i}\ \ a_{j,j}$ to find $a_{i,j}$.we found every element next to each diagonal element. This also helped us to reduce possible pairs to 1 element each as only for certain pairs we could produce element next to diagonal entry.
- Now, we found each element of this matrix by iterating over all possible values (0 to 127) and using the Final Values of Exponent Matrix and the above found values of Linear Transformation Matrix and checked the validity of function O.
- Thus, we found the Linear Transformation Matrix as follows:

$$\begin{bmatrix} 100 & 122 & 6 & 10 & 58 & 9 & 104 & 13 \\ 0 & 56 & 121 & 97 & 14 & 76 & 30 & 91 \\ 0 & 0 & 40 & 77 & 78 & 114 & 98 & 54 \\ 0 & 0 & 0 & 50 & 10 & 116 & 92 & 58 \\ 0 & 0 & 0 & 0 & 16 & 92 & 104 & 113 \\ 0 & 0 & 0 & 0 & 0 & 87 & 44 & 17 \\ 0 & 0 & 0 & 0 & 0 & 0 & 14 & 37 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 103 \end{bmatrix}$$

  And the Exponent Matrix found was as follows:

$$[85 \quad 52 \quad 38 \quad 72 \quad 116 \quad 38 \quad 66 \quad 50]$$

- We iterated over all possible values for a block and check if our EAEAE function's output is same as the current password block we have. We did it in two halves as the password contained 32 letters. The decrypted password found out was:

"batuqkizynqckgar"

Attachments

The following files are attached:

- **generate_input.py** :Generates inputs and writes them in **inputs.txt**
- **fetch_outputs.py** :Fetches outputs for corresponding inputs and stores them in **outputs.txt**
- **brute_force_search.py** : Searches for possible values of transformation and exponent matrix.
- **decrypt_password.py** : Uses the found matrix values to decryot the password.