CS641 - Modern CryptoGraphy
Assignment 5

Shashvat Singham
(200922)

On entering chapter 5, we found a passage that curved downwards steeply.We tried to go ahead with courage, but suddenly found ourselves falling forward with nothing other than thin air below our foot. Desperately we tried to grab something but there was nothing to grab there. Just then we remembered that we had a wand and this could be the chance we can make use of it.We waved it and magically, our fall got arrested. Almost in slow motion, we hit water. We dived in and found a hole, passing which we found ourselves in a pool. We came out of it and found a passage again. Hesitantly we entered in it and there was a screen. On reading, we found that the only way out was to decrypt an encrypted password "fgffgnkqjfjlgrkolglnkklqirlgjlkh". The encoding scheme was also described there which was as follows:

The Cipher used in this round had five stages: EAEAE. The input to the cipher is an 8x1 vector with elements coming from F_{128}. In the E-stage, the input to the stage is element-wise exponentiated. The vector containing exponents is unknown and is part of the key. In the A-round, the input vector is multiplied by an unknown matrix of size 8x8. All the operations are carried out in F_{128}. Matrix A is also part of the key. We now detail the steps to break the algorithm:

Encoding String

The input/output encoding for this level was same as the one used in Chapter 4 with the only difference that since each element is in F_{128}, two characters mapped to 7-bits. So, assumed that in each two character pair, the first character is from f-m and the second from f-u. This was verified by looking at the outputs for different inputs.

Initially we thought of using SASAS attack to break the cipher because in the cipher used in the level, E function acts as the substitution function and A functions as the affine transformation. Thus, the setting is exactly the same as that required by the SASAS attack. However, we observed some property of the cipher which eliminated the use of SASAS and pointed to a much simpler attack:

An Observation about Cipher

We observed that on changing the last byte of input only the last byte of cipher changes. For example, the cipher for the input "fffffffffffffg" is "fffffffffffffgn". Similarly, when the second-last byte is changed only the last two bytes of the output change. Speaking generally, changing the i^{th} byte of input resulted in change in all the bytes including and after i^{th} byte of the cipher output. This implies that the i^{th} byte of input is not used in computing the output values at byte number 0 $...to...$ $(i-1)$. This points out that the matrix used in the cipher is a lower triangular matrix.

The Attack

Let the matrix A and the vector E be:

$$A = \begin{bmatrix} a_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{10} & a_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{20} & a_{21} & a_{22} & 0 & 0 & 0 & 0 & 0 \\ a_{30} & a_{31} & a_{32} & a_{33} & 0 & 0 & 0 & 0 \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & 0 & 0 & 0 \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & 0 & 0 \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & 0 \\ a_{70} & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \end{bmatrix} \qquad E = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \end{bmatrix}$$

We find the diagonal elements of the matrix A and the elements of vector E as follows: To find the i^{th} element, consider the two inputs: First with 1 at i^{th} position and the second with 2 at the i^{th} position (all other bytes were 0). The corresponding output values at the i^{th} position will be $a_{ii}^{e_i(e_i+1)}$ and $2^{e_i^3} a_{ii}^{e_i(e_i+1)}$ respectively. Dividing, second by the first, we get $2^{e_i^3}$. Since this is cubic, we get three potential values for e_i and for each e_i, we get a value of a_{ii}. Thus for each i, we get three pairs $(a_{ii},\ e_i)$. Incorrect pairs will be eliminated while finding the first line of A (i^{th} line, all throughout this writeup, means set $\{a_{i0}, a_{(i+1),1}....a_{7,(7-i)}$, thus diagonal is the 0^{th} line) (placing 3 or any other value at i^{th} location would not have helped in eliminating the possibilities).

If the value at i^{th} position of input is t, then for all $j > i$, the j^{th} byte of output is given by:

$$\left(\sum_{k=i}^{j} t^{e_i e_k} a_{jk} a_{ki}^{e_k} \right)^{e_j}$$

We find the second line of matrix A as follows:
To find $a_{i,(i-1)}$ we use two plaintexts, one with 1 and second with 2 at the $(i-1)^{th}$ position and 0 at all other locations. For each of these inputs we look at the outputs at the i^{th} place. Using the expression above, we calculate the output in terms of $a_{i,(i-1)}$(all other values are known, precisely three possibilities, we try each) and solve it to get $a_{i,(i-1)}$. Out of the three possibilities for 0^{th} line of A and the vector E, only for one pair we get a value of $a_{i,(i-1)}$, except for a_{77} and e_7, for which two pairs still remains, for each of these two values we get a potential value of a_{76}. While Calculating a_{75}, one of these was eliminated.

We now proceed to find all the other unknown lines of A in order $2^{nd}, 3^{rd}, ..., 7^{th}$. Proceeding in this order ensures that while finding a_{ij}, all the values required other than a_{ij} are known. Also, to calculate a_{ij}, we use two inputs, one with 1 and second with 2 at j^{th} byte and all other bytes 0 and look at the outputs at byte i. Proceeding in this way we were able to find matrix A completely. We needed just 16 chosen plaintexts to break the cipher. The matrix A and the vector E were found to be:

$$A = \begin{bmatrix} 26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 71 & 46 & 0 & 0 & 0 & 0 & 0 & 0 \\ 50 & 75 & 57 & 0 & 0 & 0 & 0 & 0 \\ 112 & 0 & 79 & 101 & 0 & 0 & 0 & 0 \\ 41 & 119 & 91 & 116 & 40 & 0 & 0 & 0 \\ 120 & 7 & 12 & 60 & 58 & 100 & 0 & 0 \\ 11 & 112 & 37 & 15 & 13 & 118 & 4 & 0 \\ 25 & 32 & 106 & 4 & 75 & 13 & 11 & 118 \end{bmatrix} \qquad E = \begin{bmatrix} 102 \\ 107 \\ 72 \\ 42 \\ 65 \\ 106 \\ 6 \\ 53 \end{bmatrix}$$

Decrypting Password

We calculated the inverse of matrix A (A') and the inverse of exponentiation function E'. Decryption then is E'A'E'A'E'. We first used the same encoding that is used for inputs to encode the password. However, that didn't work. We then encoded the password using ASCII and it worked. The decrypted password was: "clqzxlooayqyzzly".

Code

The code decrypt.py has the decryption code. It can be run as:

```
python3 decrypt.py
```

It takes about 5 minutes to break the cipher. We used python's BitVector module to do operations in the field. To invert the matrix we used pyFinite module. Python's requests module was used to query server for encrypted inputs. To run the code please install these modules. They can be installed using pip3.