Q1 Teamname

0 Points

Cipherberg

Q2 Commands

5 Points

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

go, wave, dive, go, read

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.)

We have the following information available from the screen text:

1. The encryption used is a block cipher.
2. Block size = 8 bytes
3. The irreducible polynomial for the linear transformation is $x^7 + x + 1$.
4. Elements in the matrix A are from F_{128} and that of vector E are between 1 and 126.
5. The encrypted password is "gsjgjljmlkfmigggmojufnipisjklimq" which we obtain by typing "password" as told by the spirit.

***Encoding* :**

To find out the encoding that converted our string input to binary

block, we analyzed the ciphertexts. After analyzing the ciphertexts we found the characters to be in the range 'f' to 'u' which are 16 in number. This made us think about the hexadecimal base system in which each character is represented using 4 bits. Therefore, we proceeded by mapping letters f-u to 0-15 respectively:

{f : 0000, g : 0001, h: 0010, i : 0011, j : 0100, k : 0101, l : 0110, m : 0111, n: 1000, o : 1001, p : 1010, q : 1011, r : 1100, s : 1101, t : 1110, u : 1111}

Since each character is represented using 4 bits, therefore a byte consists of 2 characters. The field F_{128} consists of 128 elements, therefore we analyzed all the 128 pairs from ff-mu, and observed that all these pairs are encrypted to unique strings, this leads us to the hint that encryption is somehow related to ASCII values. Therefore from ff to mu, we assumed the mapping 0 to 127 in decimal.

Observation:

The problem at hand is a structural cryptanalysis of an EAEAE cryptosystem. The input to the cryptosystem is a block of size 8 bytes as a 8*1 vector over the field F_{128}.
-When the input plaintext was all 0s, the ciphertext came out to be all 0s.
-When the first i bytes of input plaintext were 0, then also the first i bytes of ciphertext came to be all 0.
-If we changed the input plaintext at the i-th bit, the output from i-th bit onwards started changing.
These observations give us the hint that the transformation matrix A could be a lower triangular matrix.

More formally, Let $p_0, ..., p_7$ be the 8 byte input plaintext. Let $c_0, ..., c_7$ be 8 byte output ciphertext.
- If input text is $ff\ ff\ ff\ ff\ ff\ ff\ ff\ ff$ then ciphertext obtained is $ff\ ff\ ff\ ff\ ff\ ff\ ff\ ff$.
-If the input text has p_j as non-zero byte and the byte $p_i = 0$, $\forall i \neq j$, we observe ciphertext of form, $c_i = 0$ for $i < j$.
-On changing input from $p_0, ...p_k, p_{k+1}, ..., p_7$ to $p_0, ...p_k, p'_{k+1}, ..., p_7$, ciphertext also changes after k^{th} byte.

This tells us that all the 0s present in each row are present at the end of row.
Hence, A could be a lower triangular matrix.

Cracking the transformations E and A:

Now, we need to crack the exponentiation transformation E and linear transformation A.
The elements of $8 \text{ x } 8$ key matrix A are referred as $a_{i,j}$, where i = row, j = column of element. e_i is i-th element of $8 \text{ x } 1$ vector E.

Step 1: Generation of input plaintexts

The inputs were generated using the file $Generate\text{-}inputs.ipynb$ of the form $C^{i-1}PC^{8-i}$ i.e. each input had atmost one non-zero block at a time. For every choice of such a block, there are 128 possible plaintext values (from ff to mu). We generated 8 sets of 128 input plaintexts where in each set only one byte is non-zero. Overall, 128 x 8 plaintexts were used for attack. All these input texts are saved in $plaintexts.txt$.

Step 2: Obtaining the Ciphertexts corresponding t

generated plaintexts

To automate collection of ciphertexts corresponding to the plaintexts, we used Python's pexpect to establish connection to the server using valid credentials. We used $server.py$ to generate the ciphertexts for the plaintexts stored in $plaintexts.txt$ and stored them in $ciphertexts.txt$.

Step 3: Finding the Matrices A and E

Since A is lower-triangular, so if the i-th non-zero input block has value m $(m_i \neq 0)$, then the corresponding output block has value
$n = (a_{i,i} * (a_{i,i} * m^{e_i})^{e_i})^{e_i}$ ——— eqn (1).

This helps us in finding the diagonal elements of A and all elements of vector E.
Also, if linear transform is lower-triangular, then i-th output block is dependent on j-th input block iff $i \geq j$.

The finite field F_{128} with irreducible generator polynomial $x^7 + x + 1$ over F_2 is used to perform the operations. Alternate Multiply and Exponent functions are called for encryption during brute force attack to check if encrypted output matches the ciphertext. Addition is implemented using XOR of integers since the field is F_{128}.

Using these plaintext-ciphertext pairs, we iterate over all possible elements e_i of exponentiation vector E [1-126] and diagonal elements $a_{i,i}$ of A [0-127] to see if inputs maps to the output. If it does, we append those values to a corresponding possible list. E's elements are numbers between 1 and 126 and key matrix A has elements from F_{128}. We get 3 tuples for each block.

Block	Possible $a_{i,i}$	Possible e_i
Block 0	[27, 84, 84]	[1, 19, 107]
Block 1	[91, 70, 17]	[13, 120, 121]
Block 2	[43, 14, 72]	[40, 89, 125]
Block 3	[124, 12, 88]	[51, 80, 123]
Block 4	[64, 100, 112]	[18, 21, 88]
Block 5	[11, 73, 103]	[52, 99, 103]
Block 6	[27, 66, 70]	[22, 37, 68]
Block 7	[38, 61, 125]	[15, 31, 81]

Now, we need to find the non-diagonal elements of A and also eliminate some pairs from above result. We use more plaintext-ciphertext pairs and iterate over the above $(a_{i,i}, e_i)$ pairs to find these elements in range [0-127] such that eqn (1) holds. An element $a_{i,j}$ can be found by looking at i-th output block where j-th input block is non-zero. To find $a_{i,j}$, all elements of the following set $S_{i,j}$ need to be known :
$S_{i,j} = \{a_{n,m} | n > m, j <= n, m <= i\} \cap \{a_{n,n} | j <= n <= i\}$.
Graphically, these elements form a right-angled triangle with corners at $\{a_{i,i}, a_{i,j}, a_{j,j}\}$
After this, we get to know elements next to each diagonal element and one element at each diagonal position of A and vector E.

Block	Final $a_{i,i}$	Final e_i

Block 0	84	19
Block 1	70	120
Block 2	43	40
Block 3	12	80
Block 4	112	88
Block 5	11	52
Block 6	27	22
Block 7	38	15

For the remaining elements of A, we iterate over all possible values (0-127) and eliminate ones that don't satisfy eqn (1) using final values of diagonal elements of A and exponent vector E, starting from $a_{i+1,i}$ and so on. We can discard other possibilities from previous list if we find $a_{i+1,i}$.

Final key matrix A is,

A = [[84, 0, 0, 0, 0, 0, 0, 0],
[115, 70, 0, 0, 0, 0, 0, 0],
[8, 28, 43, 0, 0, 0, 0, 0],
[125, 20, 24, 12, 0, 0, 0, 0],
[97, 43, 3, 110, 112, 0, 0, 0],
[30, 47, 19, 43, 111, 11, 0, 0],
[17, 120, 18, 101, 2, 92, 27, 0],
[88, 13, 95, 25, 12, 69, 3, 38]]

Final exponentiation vector E is
E = [19, 120, 40, 80, 88, 52, 22, 15]

The code to perform these tasks is included in
$Cryptanalysis.ipynb$

Step 4: Finding the Password

Using final A and E, we can decrypt the encrypted password "gsjgjljmlkfmigggmojufnipisjklimq" using the code included in $Decrypt.ipynb$. The password was divided into two blocks and then we applied the following transformations on each block of encrypted password:

$$E^{-1}(A^{-1}(E^{-1}(A^{-1}(E^{-1}(encrypted\_password)))))$$

Using these transformations, we obtained the following decrypted values for both the blocks of password:
$[115, 121, 113, 118, 122, 121, 111, 108]$ and
$[100, 109, 48, 48, 48, 48, 48, 48]$

We mapped these numerical values using the mapping {ff: 0,..., mu:127} and obtained the decrypted text. However, this text was not our password and thus we guessed that these numerical values could instead be ASCII codes. After mapping these numerical values to ASCII values, we obtain the decrypted password as - syqvzyoldm000000. We ignored 0's from this password as they must have been used for padding and cleared the level using "syqvzyoldm" as password.
Note: The matrix A was guaranteed to be invertible as mentioned on the panel in the game.

Ref - Structural Cryptanalysis of SASAS , Alex Biryukov

No files uploaded

Q4 Password
5 Points

What was the final command used to clear this level?

syqvzyoldm

Q5 Codes
0 Points

Assignment-5-Cipherberg.zip Download

```
1  Binary file hidden. You can download it using the button
   above.
```

Assignment 5

GRADED

GROUP

SAMBHRANT MAURYA
DEEKSHA ARORA
SHRUTI SHARMA

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