CS641: Level 5
Team: TrojanHorse

Members:
Ayush Kumar, 180174
Rishav Kumar, 180612
Nilay Majorwar, 180483

March 9, 2020

Reaching the instructions:

Reaching the instructions was tricky in this level:

1. Speak `go` to proceed on the passage. Soon, we start falling into the lake.
2. `wave` the magic wand to slow down the fall and avoid death.
3. `dive` into the lake to reach the internal pool.
4. `go` into the cave, to reach the main chamber.
5. `read` the glass panel to get instructions for the cryptosystem.

The spirit's instructions:

"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 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. You can see the coded password by simply whispering 'password' near the screen..."

Encrypted password: gkftmrfrfolimtgrlrihgniummflhkio

Breaking the encoding:

The encoding is almost the same as in Level 4. Since we are using the F_{128} field here, each pair of characters must encode to a number from 0 to 127. Thus, the set of character-pairs goes from `ff` to only `mu` (and not `uu`) (`mu` $\equiv$ `(7, 15)` $\equiv$ `7*16 + 15` $\equiv$ `127`)

Observation:

Consider the following plaintext-ciphertext pairs:

```
ff ff ff ff ff ff ff fg → ff ff ff ff ff ff ff lt
ff ff ff ff ff fg ff ff → ff ff ff ff ff li jf ih
ff ff ff fg ff ff ff ff → ff ff ff kk lq ig gm gh
ff fg ff ff ff ff ff ff → ff kn gh lp im ls hj mr
```

The most important observation to make here, is that the i^{th} block of the plaintext only affects the corresponding i^{th} block of ciphertext and the blocks after that. Or in other terms, **the i^{th} block of the plaintext affects the j^{th} block of ciphertext iff** $i \leq j$.So the 1^{st} block of the ciphertext depends only on the 1^{st} block of the plaintext, the 2^{nd} block of the ciphertext depends only on the first two blocks of the plaintext, and so on.

Getting the first block of decrypted password:

The above observation makes it tremendously straightforward to find the decrypted password. Note that **the first block of encrypted password depends only on the first block of decrypted password**. Also, the correspondence must be one-one, otherwise one ciphertext will correspond to multiple plaintext which cannot happen.

So, we can create a group of 128 plaintext blocks that have all blocks set to zero, except the first block which takes the values `0 to 127`. We know that the first block of the encrypted password is `gk`, and depends only on the first block of the plaintext. Now, **one of the plaintexts of the group must result in the ciphertext with the first block as** `gk`. The first block of this plaintext is then the first block of the decrypted ciphertext.

Getting subsequent blocks of decrypted password:

Consider the second block. The second block of the encrypted password depends on the first two blocks of the plaintext. We already know the first block, and just need to loop over the 128 possible values of the second block.

So, we create a group of plaintexts which have the first block equal to the value found in previous section, the second block takes the 128 possible values in F_{128}, and rest of the blocks are set to zero.

Again, the second block also holds a one-to-one correspondence between plaintext and ciphertext, thus we wll get a unique value for the second block.

Similarly, fixing the values of blocks and looping over the next block, we can get the value of each block easily. Repeating this process for both the blocks of the encrypted password, we get the full decrypted password as `mkmlmgmhlololulnmllulkmolpllmpmh`.

Each block requires 127 encryptions to get the corresponding block of decrypted password. So, the total number of chosen-plaintexts required is 127×8**.**

The final decoding:

Entering the final decrypted password `mkmlmgmhlololulnmllulkmolpllmpmh` into the console, the next level does not open up - only the encrypted password is shown (which is the same as the output on entering `password`). This implies that there is another decoding to be applied on this 32-character password.

Now **notice that each pair of characters of the decrypted password starts with `l` or `m`**. This hints that each pair of characters can be decoded to another character. The most probable guess here would be to use ASCII. Let us consider the first few pairs of the decrypted password:

`mk` ≡ `(7, 5)` ≡ `0111 0101` ≡ `01110101` ≡ `117` ≡ `u`
`ml` ≡ `(7, 6)` ≡ `0111 0110` ≡ `01110110` ≡ `118` ≡ `v`
`mg` ≡ `(7, 1)` ≡ `0111 0001` ≡ `01110001` ≡ `113` ≡ `q`
...

Proceeding in a similar manner, we get the final, decoded password to be **uvqriiohvoeyjfzr**. Entering this into the console, we proceed to the next level.

– – – – – – – – – – – – – – – – – – – –