|  |  |  |  |
| --- | --- | --- | --- |
| Character xor table | | | |
| X ^(xor) Y | a-z  0110 0001 - 0111 1010 | A-Z  0100 0001 – 0101 1010 | Space  0010 0000 |
| a-z | 000\_ \_\_\_\_: in range[0,31](dec) | 001\_ \_\_\_\_ : in range[32,63](dec) | 010\_ \_\_\_\_: in range[64,127](dec) |
| A-Z | 001\_ \_\_\_\_ | 000\_ \_\_\_\_: in range[0,31](dec) | 011\_ \_\_\_\_: in range[96,127](dec) |
| space | 010\_ \_\_\_\_ | 011\_ \_\_\_\_: | 0000 0000 |

Xoring char with space resulting a >= 64(dec) number (01xx xxxx binary)

ASCII (source: https://en.wikipedia.org/wiki/ASCII)

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  |  | |  |

One time pad has the encrypt function: ciphertext = plaintext xor key with key is the given bitstream has the same length as plaintext.(page 36, Understanding Cryptography, Christof Paar, Jan Pelrl)

When we know 2 of 3 component, we can find the last with a xor: key = ciphertext xor plaintext

If the key has been used more than once, we can discard the key from the given ciphertexts:

i.e:

Ciphertext\_1 = plaintext\_1 xor key

Ciphertext\_2 = plaintext\_2 xor key

ciphertext\_1 xor ciphertext\_2 = (plaintext\_1 xor key) xor (plaintext\_2 xor key) = (plaintext\_1 xor plaintext\_2) xor (key xor key) = (plaintext\_1 xor plaintext\_2) xor 0 (clearly: key xor key = 0) = plaintext\_1 xor plaintext\_2

So, ciphertext\_1 xor ciphertext\_2 = plaintext\_1 xor plaintext\_2

Each character is represented by 2 hexa number, so the text could be splited into a list of 2 adjacents haxa number, we only need the first part of the ciphertexts with the length of the target cipher text

By xoring arbitrary 2 pair of 3 ciphertexts, we can find the encrypted space character:

Char\_1 xor char\_2 = x >= 64

Char\_1 xor char\_3 = y >= 64

* Char\_1 = space character

Target text included 166 hex numbers(83 characters), so we must find the 166 hex number length stream cipher

We will find the key for each character by traversing through all possible pair combination, when there is 2 pair with similar index xoring has the result > 63: (s1,s2) and (s1,s3), we could find the key for that index k = strxor(s1,space char)

Result:

|  |  |  |
| --- | --- | --- |
| Possible encrypted space, syntax: index: [encoded space] [key] | | |
|  | | |
| A computer screen shot of a black screen  Description automatically generated | | |
| A screen shot of a computer  Description automatically generated | | |
| Possible character decoded(alphabetical char and space only) | | |
|  |  |  |

Since the decrypt module is not perfect (the encoded text may not consist of space character), it still have a few mismatch index: 7,21,25,49, we will guessing for the result.

|  |
| --- |
| Possible printable character |
| 7: r |
| => 21: ‘:’ |
| 25: e |
| => 49: ‘,’ |