APPROACH TO DECIPHER THE MULTI PAD CIPHER TEXTS

Note : ‘+’ means xor everywhere

1. Using the fact that xor is commutative,
   m1 + k = c1
   m2 + k = c2
   Taking Xor of both equations, we get
   m1 + m2 = c1 + c2, (k + k = 0 and x + 0 = x)

2. Thus, we can apply “Only ciphertext attack” in case of multi-pad stream ciphers.

3. We now make use of the property of xor of any ASCII value with space (“ ”), hexadecimal code : 20h.
   m + 20h = x
   So, x is randomly generated if m is any character other than alphabets or space.
   If m = 20h, x = 00h
   If ‘m’ is a lower-case alphabet (‘a’ to ‘z’) we get ‘x’ as corresponding upper-case alphabet (‘A’ to ‘Z’).
   If ‘m’ is a upper-case alphabet (‘A’ to ‘Z’) we get ‘x’ as corresponding lower-case alphabet (‘a’ to ‘z’).
   Thus, taking xor with space gives us a special property that we can recover the other corresponding text as well. But this is not found with any other character in the ASCII set.

4. Thus, we initially xor every cipher with all the other ciphers. If while taking xor of cipher[i] with cipher[j], we get a character which lies with the ascii range of alphabets, then we may conclude that one of the ciphers contain a space and the other conatins the corresponding alphabets. But we do not know which one contains a space and which one contains the alphabet. Here we do our statistical test. We assume if a position in cipher[i] generates a ascii character while xorring with cipher[j] atleast 6 / 7 / 8 out of 9 times, we consider that position corresponding to space in cipher[i].

5. Thus, we also preprocess the xor all the ciphers texts with a string of spaces.

6. From the assumed position of spaces with particular cipher, we try to construct the key using the pre-processed strings, obtained by xorring cipher iwth string of spaces. We simultaneously also mark the corresponding positions in key as been found.

7. After the above process is done with all the strings with varying statistics, we try to generate the plaintext from the assumed key and cipher text using the following manner :

m + k = c
Taking xor with ‘k’ on both sides
m = c + k (k + k = 0 and x + 0 = x)

8. By doing this, we get most of the plaintext recovered. The position which were not discovered in key were marked as astericks (‘*’) in the plaintext. While doing so, we assumed that cipher text only consisted of spaces and alphabets, so other charcters, like punctuation marks, numbers etc. are not discovered by this technique. These has to be guessed or found out by the below mentioned processes.

9. Now, we try to read the almost complete plaintext we generated and try to guess the missing characters and correct the few faulty ones.

10. Finally, we generate the correct key from the guessed plaintext as follows:
m + k = c
Taking xor with ‘m’ on both sides
k = m + c (m + m = 0 and x + 0 = x)

11. Now we generate all the other messages from the ciphertexts and key we generated. We read all the messages thus generated and check it is gramatically correct and spelled correctly. If there are some errors, we repeat steps 9 and 10 and continue the process till we are satisfied with the plaintexts we get.

12. Now the above process will (with almost 100% probability) generate the plaintext of all strings, if the size of all strings is same. But here, the size of every plaintext is different. So, we start by guessing the plaintext in increasing order of their length. This is because the more information (the pairwise xor of different ciphers) we have, the better our chances to decode the cipher into plaintext and correctly guess the key.

13. Towards the end, where the information we have is quite less, we can use other techniques like “Crib dragging” to recover the plaintext. More information on this can be found at http://travisdazell.blogspot.in/2012/11/many-time-pad-attack-crib-drag.html and the dynamic code for it can be found at https://github.com/SpiderLabs/cribdrag .

14. Thus, doing all the above steps, we can successfully recover the plaintexts from the cipher texts only.

Note : Since, in the assignment we need to guess the cipher-text 10, which has length reasonable to get enough information, step 13 was not applied to recover the plaintext 10.