Alex McAfee

Kelley Cislo

CSCI 182: Applied Cryptography

Prof. Schaefer

Project 1

So far we have created a few functions that will assist us in our task in deciphering all of the Cipher Text in the individual files. Since each character (a-z, commas and periods) is represented by 5 bits we have created a function that will take all of the PT and separate it into groups of five. Each group of five is stored in an array for future analysis. In accordance with that, there is a function called find\_rep that will take whichever group of five and search as to which character represents it (a-z, commas or periods). All of these characters are hard coded into that function and will undergo a search method to find the correct representation. For instance, if there group of 5 is 00000 then find\_rep will return ‘a’ etc.

The next function, decoder, takes in a large input string (the PT) and separates it into groups of five, using the turn\_into\_5array function and individually looks up each group's representation. It all is processed by a for loop that will run for as many times as we have a group of 5 to search. Of course, this is the decoding method and so is used only after we have derived the correct PT from the CT.

The lorenz\_matcher function XORs substrings of the input (CT) with itself at varying lengths of substrings knowing that the key is repeated therefore, by doing this, we are effectively utilizing a lorenz cipher attack. Additionally, there is a rolling\_XOR function within the lorenz\_cipher function that keeps extending the “key” that will continuously be XOR’d with the CT that we have in hopes of getting the XOR of the PT with itself.

Our plan from here on out is somewhat simple logically. We have already exploited the fact that XORing two ciphertexts will yield the XOR of the two plaintexts, and we will be creating every possible XOR combination. We will then be looking for the highest number of 0’s which we refer to a coincidences. The logic is as follows, by having a repeated key there should be some sort of repeated pattern of the same lengths, the same pattern XORed with itself with yield all 0’s. Following this logic, the XOR we receive from lorenz\_matcher that has the highest number of coincidences should, statistically speaking, be the most likely to be the XOR of the plaintexts.

Once we have a few candidates of possible XORs of the plaintexts it becomes the issue of “teasing” these plaintexts apart. Our plan for this is two fold: first, we will again abuse the property of XOR that (A^X)^(B^X) = (A^B)^(X^X) = A^B, secondly we will use frequency analysis to find common words or letters. Combining these with the assumption that the XOR key is repeated at a close enough interval, we should be able to be left with a string that is the plaintext XORed with itself at an interval such as newValue = encoded[counter] ^ encoded[counter+1]. Therefore if we apply similar logic to known common words or letters and then look for that pattern we should, theoretically, be able to identify some plaintext elements. Once we have a few it becomes a trivial matter of extracting the rest for each ciphertext.

We see a pitfall in the last step, if X is different from piece to piece then this approach will not work at all and we will need to look into a new strategy. Also, if there is salting, padding, or any other modulation other than encryption and encoding we will need to detect that and account for it.