Edward Lee (el1926), Shikang (Steven) Zheng (sz1389)

Applied Cryptography

10/24/16

Project 1

We were given a permutation cipher and we were tasked to break it. We had access to the ciphertext, a plaintext dictionary, an english word dictionary, and partial knowledge of the encryption algorithm. There were two main parts, a known plaintext attack and a known ciphertext attack. The first part was fairly simple. We had access to the plaintexts so we were able to analyze the number of characters per word. If the ciphertext matched these character counts, then we would try the matched plaintexts. We made sure to keep track of the number of key values used per letter to make sure the decryption was correct. If none of the matched plaintexts worked, then we would move onto using the dictionary, which was part two of the project.

Part two of the project was a bit more complicated. We both brainstormed ideas of how to approach this and eventually came up with a suitable backtracking algorithm. First, we stored the dictionary in the program and grouped them by word lengths. Through this method, we were able to count how many words of each length there were and use this in our algorithm. The groups with smaller word lengths would be easier to decrypt because there are less possibilities. Next, we take the ciphertext and re-order every word except the last one based on which words have higher value. We did not re-order the last word because it can be truncated and will produce bad results for the algorithm we are using. To get the value of each word, we took the word count for the length and divided it by how many character decryptions the word would produce. Ex: the value of 1,2,3,4 in ciphertext “1,2,3,4 2,2,3,7 6,8,4” would be: $\frac{words\ of\ length\ 4}{8}$ because decrypting 1,2,3,4 would produce 8 character deciphers. This greatly helps us when we are checking to see if a partially decrypted word is in a

dictionary because more decrypted characters mean less possible matches. We performed this check for matches using regular expressions. The algorithm takes in a partially or fully encrypted word, and produces a regular expression that we can use from it. It uses this regular expression and goes through the dictionary words of the same length that are possible matches for it, and produces a list. It iterates through this list of matches and tries them out. Every time a key is substituted, it tests that partially decrypted word to see if it is in the dictionary. If it is, then it is safe to continue, if it is not, then we have tried a wrong word and we backtrack. The algorithm will find all the key mappings relatively fast using this algorithm. Once it finds all the key values, it fixes the order of the cipher text and outputs the decrypted result.

The project tasks were fairly distributed. We both brainstormed ideas together, and we implemented the resulting algorithm through pair programming. In order to test the program, Steven implemented a random message generator using the dictionary, and Edward implemented the encryption algorithm using random key values. We used these two supplementary programs to ensure that our decryption is correct.