Amogh Madanyakanahalli Kumar
amadanay@ucsc.edu
2005781

Assignment - 2

Recovered Secret key:

In decimal - [174, 34, 0, 175, 47, 184, 153, 17, 221, 0]
In hex - AE2200AF2FB89911DD00
In ASCII - ®"\x00¯/¸\x99\x11Ý\x00

Recovered Plaintext:

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.

Approach:

I assumed a key length size ranging from 1 to 19. For each key length size, I divided the cipher text into streams which would be encrypted by the same key character. Then, I calculated the frequency of each character in the streams and summed it up and fetched the key length providing the max value (according to the below equation). The key length obtained was **10**.

$$max_L \frac{1}{L}\sum_{a}^{L}\sum_{i}^{n}\left(\frac{f_i}{n}\right)^2$$

L - key length (1 to 19).
n - number of characters in the stream.
f_i - frequency of i^{th} character in the stream.

Amogh Madanyakanahalli Kumar
amadanay@ucsc.edu
2005781

Once the key length was found, I iterated through the streams again. In each stream, I tried all possible key characters (0 to 255) and tried to use them to decrypt the ciphertext. If the decrypted value was invalid ASCII for regular English plaintext, i.e. >128, I ignored it. If not, I multiplied the frequency of ciphertext characters in the stream with the frequency of regular English alphabets and ASCII characters (obtained from: https://millikeys.sourceforge.net/freqanalysis.html) and summed it up and fetched the key character providing the max value (according to the below equation).

$$max_j \sum_{i}^{n} (q_i * p_{i+j})$$

j - possible key characters (0 to 255).
n - number of characters in the stream.
q_i - frequency of i^{th} ciphertext character in the stream.
p_{i+j} - frequency of ASCII character obtained after XORing i^{th} ciphertext character with j.

Thus, I obtained all 10 key characters and recovered the secret key.
Then, I decrypted the ciphertext using the secret key and recovered the plaintext.

Note:
The source from where I got the ASCII character frequencies only provided English alphabet character frequencies as a whole and not separately for upper and lowercase letters. I utilized these frequencies and set the frequencies of upper and lower case letters by considering the frequency of 0.1f and 0.9f respectively where f is the frequency of the alphabet.