COMP3491 Codes and Cryptography 2020-21 Summative Assessment

1 Cryptography

There are many ways to attempt to crack the DES cipher, with some being more practical than others. Techniques such as linear cryptanalysis and differential cryptanalysis, whilst theoretically more efficient than a simple brute force attack, are not practical due to the requirement of a large number of plain-text/ cipher-text pairs, 2^{43} in the case of linear cryptanalysis.

As a first attempt to brute force the code, I attempted to recreate the encoder and use the known plain-text/ cipher-text pair of 0123456789abcdef -> a80f2c74f235484e to test random keys until the right one was found that encoded the correct cipher text from the plain-text. Immediately it was clear that this attempt, whilst theoretically possible, was also not practical on my machine [1]. My program could approximately test around 2500 per second, which would mean that to test all 2^{56} combinations, it would take:

$$2^{56}/2500 = 1.44x10^{13}\ seconds$$
$$= 912636.422\ years$$

As this was unfeasible given the hardware available, instead of cracking the key, I decided to try and use the encoder provided and test all the possible plain-text combinations to try and produce the output given. This sort of solution would work to crack the particular code given, but then wouldn't be reproducible on a different code-word without trying all the combinations again.

Since the cipher-text was 16 bytes, the plain-text must also be the same length. with 2 dots between the 3 words, this leaves 14 letters. According to the what3words website, the smallest word length they use is words of 4 letters. This means all 3 words can only consist of four, five and six letters. I downloaded a dictionary or four, five and six-letter words. From the files I downloaded, there were 3130 four-letter words, 5757 five -letter words, and 15788 six-letter words.

At first I attempted to try all combinations of these words. Knowing that there must be 14 characters, there could only be a combination of 2 four-letter words and a six-letter word, or a combination of 2 five-letter words and a four-letter word. For each combination of words, there are 6 permutations.

Therefore let C be the total number of combinations, and W_n be the number of words length n:

$$C = (W_4 \cdot W_4 \cdot W_6 \cdot 6) + (W_5 \cdot W_5 \cdot W_4 \cdot 6)$$
$$\therefore C = (3130 \cdot 3130 \cdot 15788 \cdot 6) + (5757 \cdot 5757 \cdot 3130 \cdot 6)$$
$$= 1.55x10^{12}$$

In an initial attempt to brute force these combinations, my program crated combinations of these words, called the encoder and compared the output to the cipher-text. The initial rate was about 6 attempts per second, which would require around 8194 years to check all attempts.

[1] All tests performed on a Windows PC with AMD Ryzen 5 5600x 3.7GHz processor

However after some thought, I realised that since the encoder was using DES in ECB mode, I could pass through it multiple strings at once, have them all encoded and then split the strings, since every 8 bytes would be encoded the same. This meant I could check around 2000 strings per second since calling the encoder was the main bottleneck of the system. However even at this rate it would still take around 24 years to check all the combinations.

This lead me to my final attack plan. Since the encoder was encoding in blocks of 8 bytes, I could attempt to check all the combinations of the first 8 letters. This could either be a six-letter word followed by a '.' and another letter, a five-letter word followed by a '. and two letters, or a four-letter word followed by a '.' and 3 letters. This dramatically decreased the number of combinations to check through, since now there only needs to be:

$$
\begin{aligned}
C &= (W_6 \cdot 26) + (W_5 \cdot 26^2) + (W_4 \cdot 26^3) \\
\therefore C &= (15788 \cdot 26) + (5757 \cdot 26^2) + (3130 \cdot 26^3) \\
&= 410,488 + 3,891,732 + 55,012,880 \\
&= 59,315,100
\end{aligned}
$$

At a rate of around 2500 per second, this would take around 6.5 hours to check all possibilities on my machine. After the combination is found, the process simply needs to be repeated checking all the words that could fit as the last word, with all the words that could fit in the middle word that begin with the letters found at the end of the first half. The time to check the second half will be much less, since instead of checking combinations with all letters, combinations of a 3rd word and a part of a 2nd word that fits with the first half will be checked.

After running on my machine, for 6 hours. I found that the first half of the word was tile.bil, and after running a further 51,372 tests for 18 seconds, the second half of the word was ls.print, thus making the full word tile.bills.print, which corresponds to a square near Four Seasons Total Landscaping in Philadelphia, Pennsylvania, USA.