EECS 4980:805 Inside Cryptography

DES Analysis Project

David Carek

Analysis of DES

The purpose of this analysis is to understand the result of applying the Data Encryption Standard (DES) algorithm. The data collected will help to paint a picture of what the algorithm does at a macroscopic scale. In this analysis we will discuss various aspects of the results that DES yields. These aspects include bit frequencies, byte frequencies, di-gram frequencies, tri-gram frequencies, octet frequencies, and same block encryption. Each of these levels of analysis have important results that show the benefits and disadvantages of what was the standard for encrypting data.

The first set of data collected was on the individual bits of the unencrypted and encrypted files. The number of high bits(1s) in the unencrypted file came in at 17,829,132 and for the low bits(0s) 22,598,668. We will compare these values with those of the encrypted file which had 20,226,928 high bits and 20,200,976 low bits. Initially we are able to tell that running the DES algorithm on the Shakespeare text more evenly distributes the high and low bits. The difference between the high and low bits for the unencrypted text is 4,769,536. The difference for the encrypted file is 25,952. The encrypted text has a much more random distribution of high and low bits. Another important thing to note is that since the file we are encrypting is all ASCII text the MSB is always 0. This helps explain why the plaintext has 4,769,536 more low bits than high. To summarize this data, we have taken text that does not have a very random distribution of bits and made it closer to a random distribution.

The next data collected is on byte frequencies. The graphs listed as figure 1a and 1b illustrate the byte frequencies before and after encryption. Prior to encryption the byte frequencies have widely varying values. The most important take away from these graphs is the variation in the range. In the plaintext the space character occurs 767,104 times. No character in the encrypted file comes anywhere near to having this many occurrences. The most frequent character that occurs in the encrypted file occurs 20724 times and is not even an ASCII character. At a glance it is fairly easy to tell that the points are much more evenly distributed. It may be easier to see these differences by looking at figures 1d and 1e. These are sorted graphs for the frequencies and produce a clearer picture of what is happening. Figure 1e lets you see the average of all the frequencies better and it shows that the byte distributions are not quite ideal. The preferred distribution of the bytes would be flat but we have a gradual slope. However, this is still much better than the unencrypted byte frequencies which has a very steep slope at the beginning and tapers off after 100 bytes or so. This validates the requirements for the confusion that DES is supposed to produce. However, there are still outliers for encrypted bytes. This is one of the spots where DES can be improved at and this is likely to be caused by one of the flaws of electronic code book(ECB) mode. This will be discussed more later in the analysis though since it fits in better with the octet data analysis.

The data in the charts is a helpful tool to visualize what happens to the individual bytes but the statistics are a better way of quantifying the results of running DES. When DES is run on the Shakespeare text using the key ‘Pa$$w0rd’ we drastically bring the min and max closer together. The range drops from 767,104 in the plaintext to 1,800 in the ciphertext. Perhaps the most important statistic in the table on the next page is the standard deviation of the byte frequencies. Again, in an ideal algorithm we would have a standard deviation as close to 0 as possible. DES does a decent job at doing this for the byte frequencies. The standard deviation of the encrypted text is just over one half of one percent of the standard deviation of the unencrypted byte frequencies. The smaller the standard deviation on the encrypted text, the harder it is to discern a pattern.

|  |  |  |
| --- | --- | --- |
| Statistics of sorted byte frequencies | Unencrypted | Encrypted |
| Min | 0.00 | 18924.00 |
| Max | 767104.00 | 20724.00 |
| Range | 767104.00 | 1800.00 |
| Mean | 19740.14 | 19740.19 |
| Median | 0.00 | 19692.00 |
| Standard Deviation | 70869.18 | 372.69 |

Another set of data collected for this analysis was the di-gram frequencies. In figures 2a and 2b we see a similar pattern as with the byte frequencies. The unencrypted file has frequency counts that start very high and drop quickly, then taper off. The encrypted di-gram frequencies are also very similar to the byte frequencies. The differences that are most obvious are the sharper decreases at the beginning and end of the graph. Again, if we look at the standard deviation in the statistics table below it is drastically reduced. Therefore, eliminating most of the pattern in the output di-grams.

|  |  |  |
| --- | --- | --- |
| Statistics of sorted di-gram frequencies | Unencrypted | Encrypted |
| Min | 1.00 | 27.00 |
| Max | 183790.00 | 358.00 |
| Range | 183789.00 | 331.00 |
| Mean | 2840.63 | 77.11 |
| Median | 267.00 | 74.00 |
| Standard Deviation | 9405.39 | 21.22 |

The more interesting trend in the di-grams comes from figures 2c and 2d. If we look at figures 1f and 1g we begin to see that the most common bytes also occur inside the most frequent di-grams. In figures 1f and 2c the hex byte 0x20, the most frequent byte, appears in the hex di-gram 0x6520, which is the second most frequent di-gram. Also, hex byte 0x65, the second most frequent byte appears in the second hex di-gram 0x6520 as well. When we take a look at the encrypted figures 1g and 2d we see that hex byte 0x87, the most common encrypted byte, appears in 0x87A0, the third most frequent encrypted di-gram. The same happens with 0xCB(2nd) and 0x81CB(3rd). This would make sense because of how ECB mode works with DES. Again, more on that later. At this point we appear to have a pattern in an algorithm meant to hide patterns. As we keep going this pattern will make more sense.

The next set of data collected was on the tri-grams. Figures 3a and 3b look very similar to the di-gram charts, just more pronounced. Every time we go a level deeper the charts become more and more defined with the same general shape. The statistics table on the next page also yields similar results. The min and max get closer together, the range decreases, and so does the standard deviation. All things that we expect at this point. We also still see the pattern of frequent di-grams and bytes making up some of the contents of the trigrams. The encrypted di-gram 0xA014 appears in the 23rd encrypted tri-gram 0xD0A014. The 2nd most frequent encrypted byte 0xCB appears in the 4th most frequent tri-gram 0xCBDB13.

|  |  |  |
| --- | --- | --- |
| Statistics of sorted tri-gram frequencies | Unencrypted | Encrypted |
| Min | 2.00 | 2.00 |
| Max | 67523.00 | 242.00 |
| Range | 67521.00 | 240.00 |
| Mean | 368.06 | 3.49 |
| Median | 41.00 | 2.00 |
| Standard Deviation | 1565.19 | 4.65 |

Now we will go into more detail about ECB mode and why it has been giving us a pattern where more frequent bytes appear in more frequent di-grams and tri-grams etc. One of the more interesting parts of this project was comparing a bitmap image with its encrypted pair. One of the problems with the DES in ECB mode is that any given block always encrypts to the same encrypted block. If there is a reoccurring pattern of 8 bytes in the plaintext, then the ciphertext will contain similar patterns. This is most easily seen by looking at the images below. The white background and the yellow text of the Toledo Rocket image always encrypts to the same pattern. This makes the original image more visible in the ciphertext even though it has been encrypted. When patterns are present in the plaintext, we start to lose some of the confusion that DES gives us. If we made an analysis of frequent octets in the English language and had a large enough ciphertext, then we might be able to back into the plaintext more easily. The term electronic code book comes from the idea that DES could be set up like a lookup table. If we enumerated all of the 8 byte blocks of input and ran DES on them with a given key, we would have list of outputs for every possible input. Since each block is independent of the others we can run DES just by looking up its value in the table.

Unencrypted and Encrypted Bitmap File:



Some of the consequences of the way ECB works is that if an 8 byte block of data is repeated in the plaintext we will see the encrypted version of that block in the output more frequently. If we take a look at figures 4a and 4b we get the same patterns as the bytes, di-grams, and tri-grams. The statistics on the next page get interesting at this point because of ECB mode again. When we look at all of the 8 byte blocks in the unencrypted and encrypted files we get the same distribution for both. Since one block in yields one encrypted block out the frequency counts for the blocks are the same. This leads to the statistics being the same for blocks in both files even though the contents of the block are different. We also see the same common bytes, di-grams, and trigrams in the octets. In figure 5 I highlighted some of the bytes that get through all of the stages. Some do overlap and may have a different color associated with them.

|  |  |  |
| --- | --- | --- |
| Statistics of sorted octet frequencies | Unencrypted | Encrypted |
| Min | 1.00 | 1.00 |
| Max | 241.00 | 241.00 |
| Range | 240.00 | 240.00 |
| Mean | 1.62 | 1.62 |
| Median | 1.00 | 1.00 |
| Standard Deviation | 2.80 | 2.80 |

Overall DES in ECB mode has its flaws but when it was the standard it worked well enough. There are other modes that DES runs in to improve on some of these flaws and one of them will be explored in the next analysis.

Figures

Figure 1a.

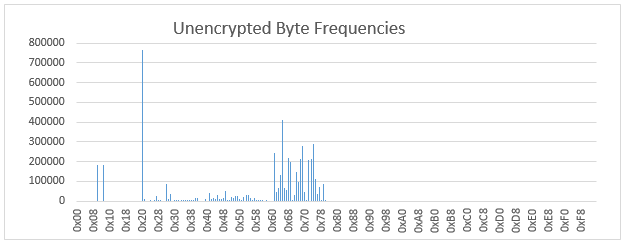


Figure 1b.

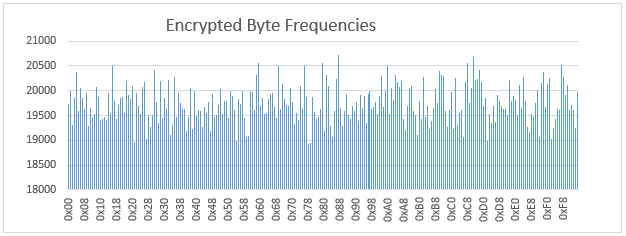


Figure 1c.

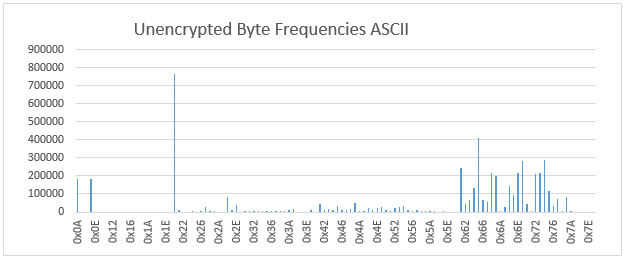


Figure 1d.

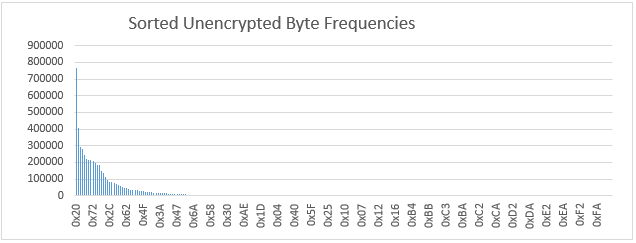


Figure 1e.

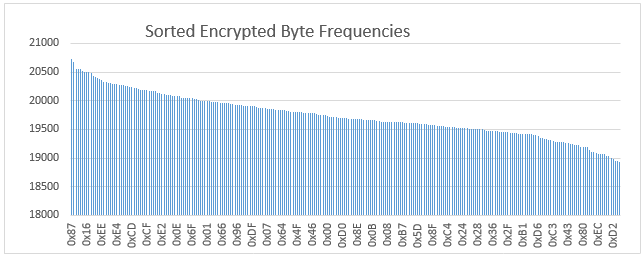


Figure 1f.

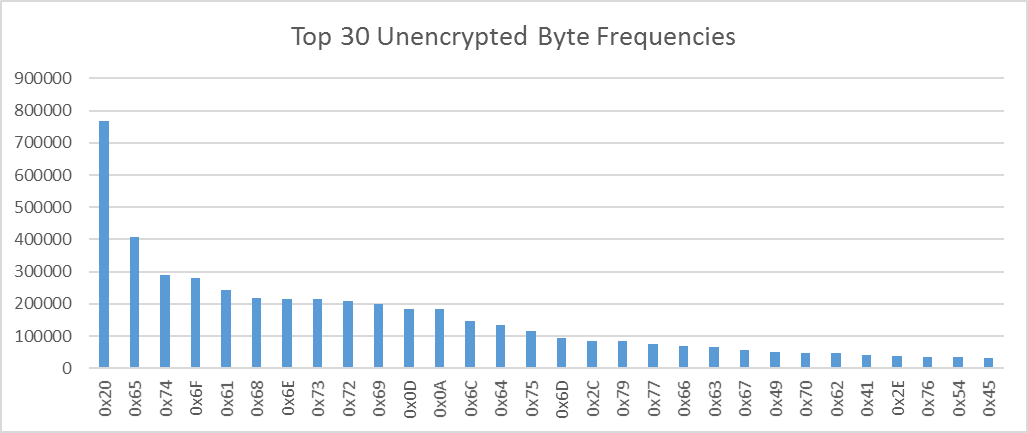


Figure 1g.

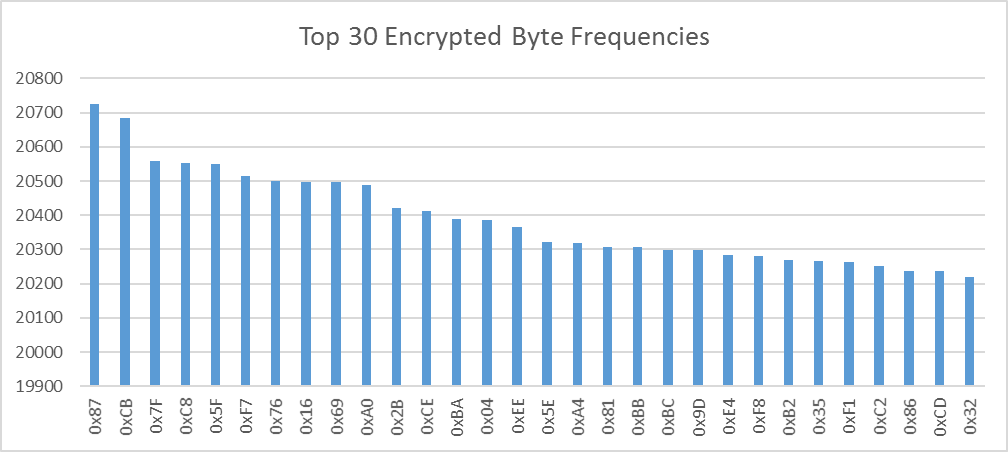


Figure 2a.

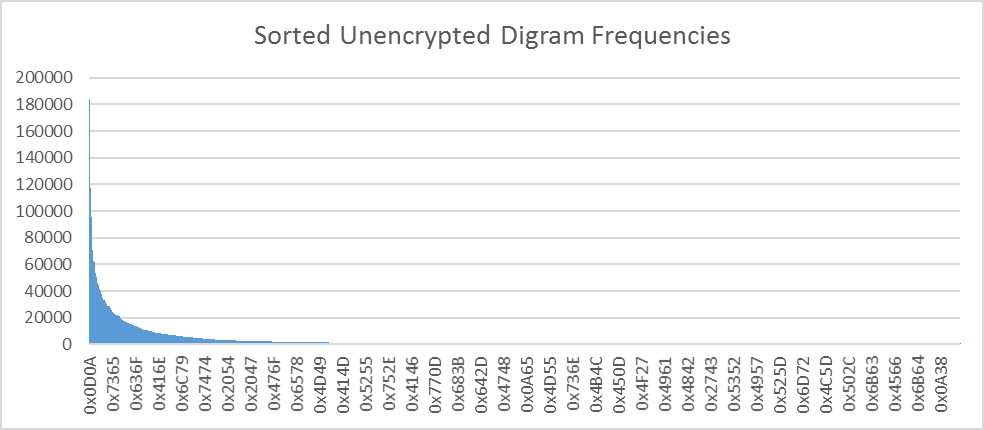


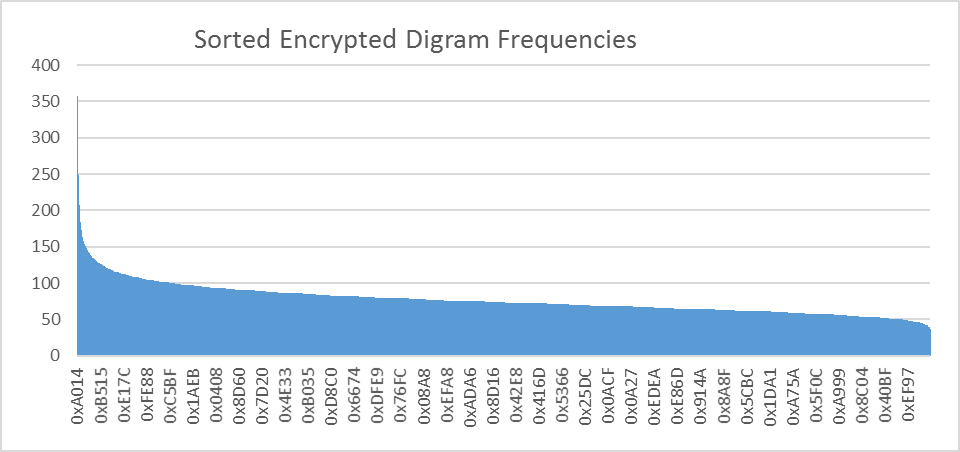
Figure 2b. 

Figure 2c.

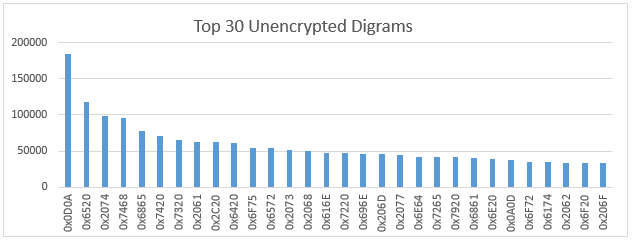


Figure 2d.

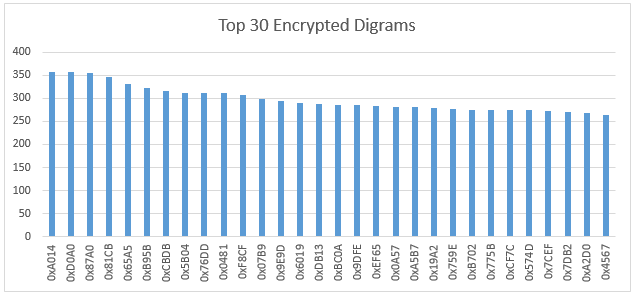


Figure 3a.

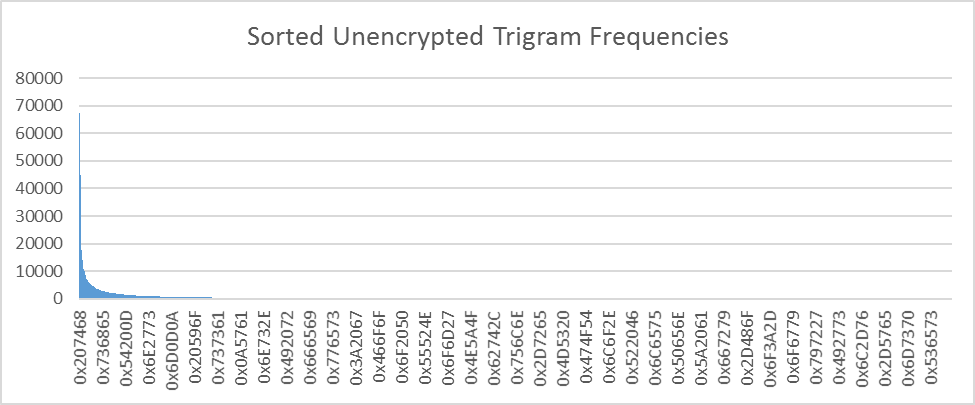


Figure 3b.

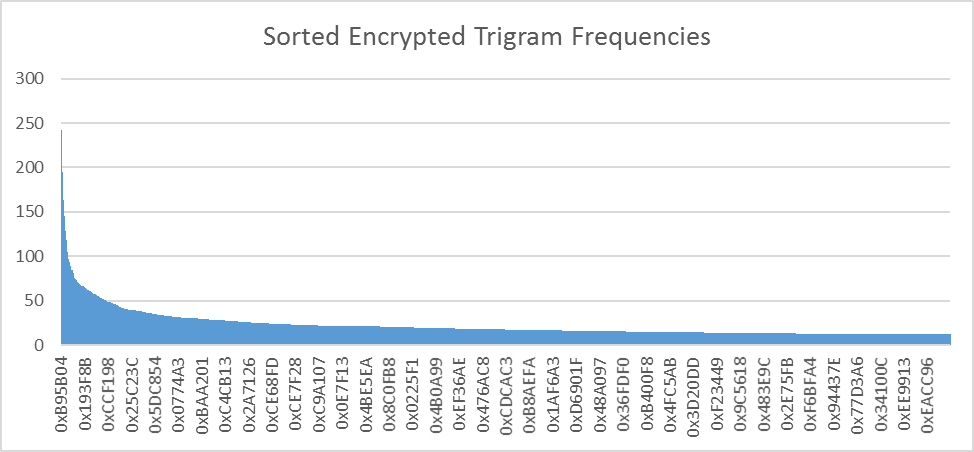


Figure 3c.

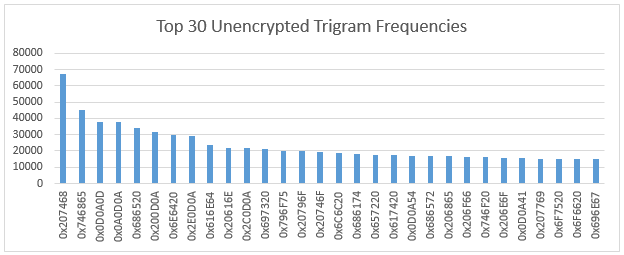


Figure 3d.

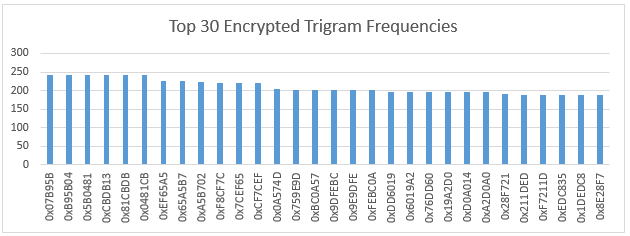


Figure 4a.

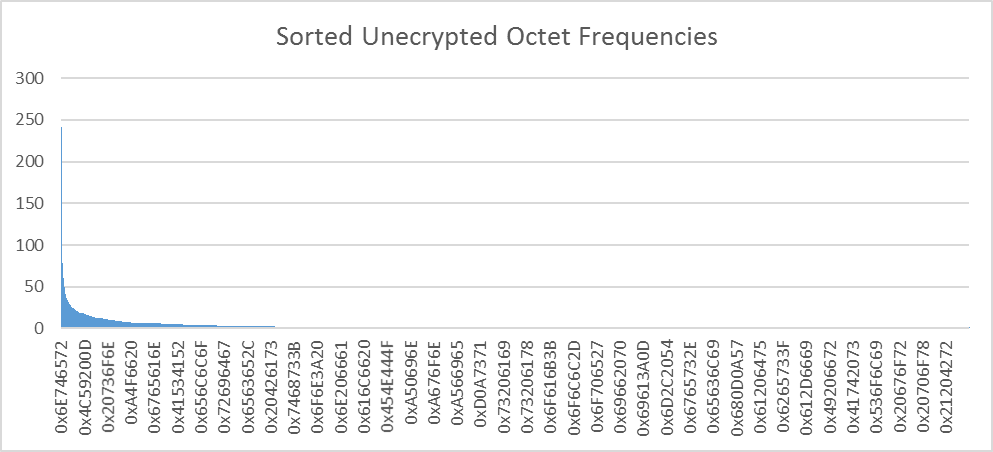


Figure 4b.

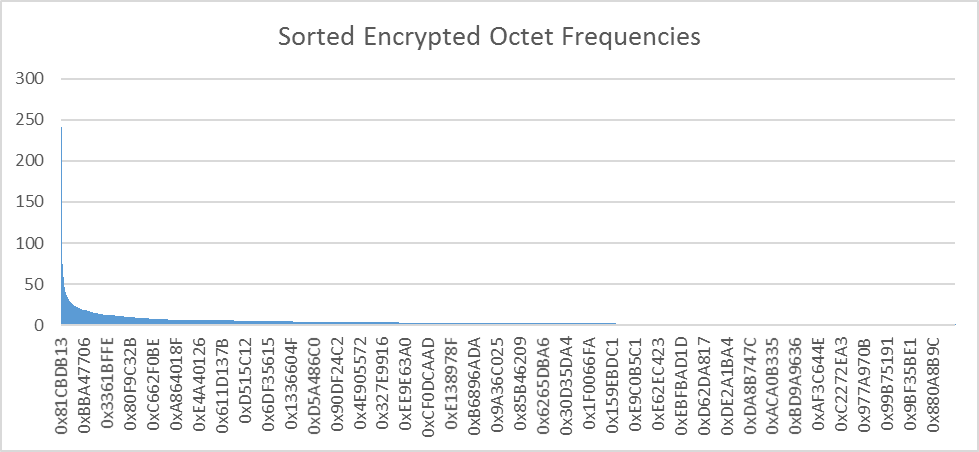


Figure 4c.

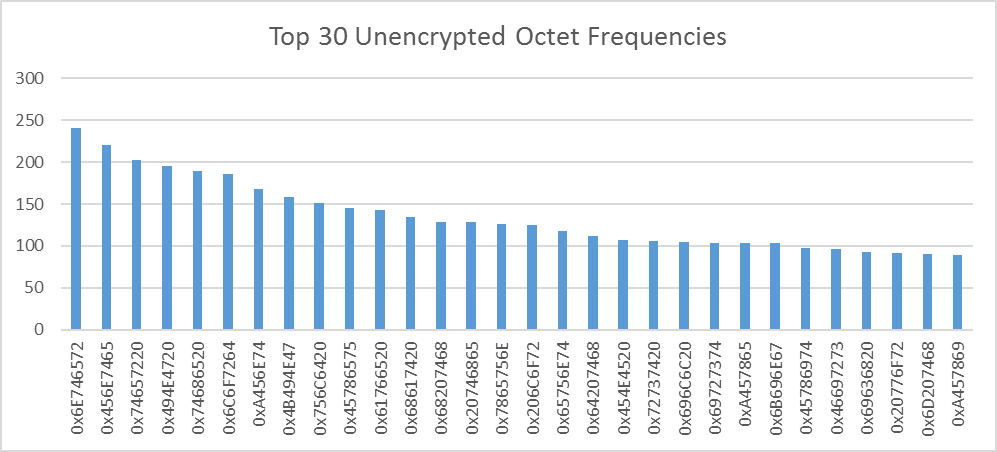


Figure 4d.

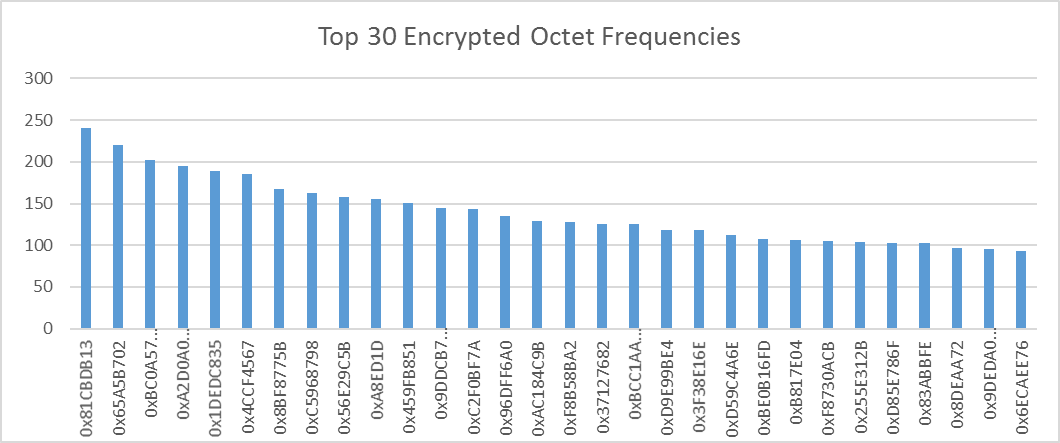


Figure 5.

|  |  |  |  |
| --- | --- | --- | --- |
| **Octet Value** | **Trigram value** | **Digram value** | **Byte value** |
| 0x81CBDB13 | 0x07B95B | 0xA014 | 0x87 |
| 0x65A5B702 | 0x5B0481 | 0xD0A0 | 0xCB |
| 0xBC0A574D | 0xB95B04 | 0x87A0 | 0x7F |
| 0xA2D0A014 | 0x0481CB | 0x81CB | 0xC8 |
| 0x1DEDC835 | 0x81CBDB | 0x65A5 | 0x5F |
| 0x4CCF4567 | 0xCBDB13 | 0xB95B | 0xF7 |
| 0x8BF8775B | 0xEF65A5 | 0xCBDB | 0x76 |
| 0xC5968798 | 0x65A5B7 | 0x0481 | 0x16 |
| 0x56E29C5B | 0xA5B702 | 0x5B04 | 0x69 |
| 0xA8ED1D | 0xF8CF7C | 0x76DD | 0xA0 |
| 0x459FB851 | 0x7CEF65 | 0xF8CF | 0x2B |
| 0x9DDCB7C2 | 0xCF7CEF | 0x07B9 | 0xCE |
| 0xC2F0BF7A | 0x0A574D | 0x9E9D | 0xBA |
| 0x96DFF6A0 | 0x759E9D | 0x6019 | 0x04 |
| 0xAC184C9B | 0x9DFEBC | 0xDB13 | 0xEE |
| 0xF8B58BA2 | 0x9E9DFE | 0x9DFE | 0x5E |
| 0x37127682 | 0xBC0A57 | 0xBC0A | 0xA4 |
| 0xBCC1AADF | 0xFEBC0A | 0xEF65 | 0x81 |
| 0xD9E99BE4 | 0xDD6019 | 0x0A57 | 0xBB |
| 0x3F38E16E | 0x19A2D0 | 0xA5B7 | 0xBC |
| 0xD59C4A6E | 0x6019A2 | 0x19A2 | 0x9D |
| 0xBE0B16FD | 0x76DD60 | 0x759E | 0xE4 |
| 0xB817E04 | 0xA2D0A0 | 0x775B | 0xF8 |
| 0xF8730ACB | 0xD0A014 | 0xB702 | 0xB2 |
| 0x255E312B | 0x28F721 | 0xCF7C | 0x35 |
| 0x83ABBFE | 0x1DEDC8 | 0x574D | 0xF1 |
| 0xD85E786F | 0x211DED | 0x7CEF | 0xC2 |
| 0x8DEAA72 | 0x8E28F7 | 0x7DB2 | 0x86 |
| 0x9DEDA0CD | 0xEDC835 | 0xA2D0 | 0xCD |
| 0x4AAE8773 | 0xF7211D | 0x4567 | 0x32 |