Part 1:

Key (in hex): 0x502f087c5f3000

Plaintext:

Tonight I can write the saddest lines.

Write, for example,'The night is shattered

and the blue stars shiver in the distance.'

The night wind revolves in the sky and sings.

Tonight I can write the saddest lines.

I loved her, and sometimes she loved me too.

Through nights like this one I held her in my arms

I kissed her again and again under the endless sky.

She loved me sometimes, and I loved her too.

How could one not have loved her great still eyes.

Tonight I can write the saddest lines.

To think that I do not have her. To feel that I have lost her.

To hear the immense night, still more immense without her.

And the verse falls to the soul like dew to the pasture.

What does it matter that my love could not keep her.

The night is shattered and she is not with me.

This is all. In the distance someone is singing. In the distance.

My soul is not satisfied that it has lost her.

My sight searches for her as though to go to her.

My heart looks for her, and she is not with me.

The same night whitening the same trees.

We, of that time, are no longer the same.

I no longer love her, that's certain, but how I loved her.

My voice tried to find the wind to touch her hearing.

Another's. She will be another's. Like my kisses before.

Her voide. Her bright body. Her inifinite eyes.

I no longer love her, that's certain, but maybe I love her.

Love is so short, forgetting is so long.

Because through nights like this one I held her in my arms

my sould is not satisfied that it has lost her.

Though this be the last pain that she makes me suffer

and these the last verses that I write for her.

In Order to determine the key, we dumped the hex of the encrypted file into an online frequency analyzer that showed that there is a sequence repeated 9 times. To get the length of the key, it had to be a common factor of all the distances between these 9 repetitions. That was 14. Then the hex file was columned into 7 columns, each containing a byte per line, then the frequency analyzer was run again to show for frequent bytes in the file. These most frequent bytes have to be mapped to the space character, as it is the most used in plain text. This gave us the full key of 0x502f087c5f3000.

The process of obtaining the key was not automated, however the process of decryption was.

If the file was compressed before it was encrypted, then the method of finding the most common bytes to represent the space would not work, as the format of the original plaintext would have fundamentally changed, and the space would no longer be the most common character.

Part 2:

Key = 53.503563N,-113.528894W

Plaintext: See attached file plaintext2.txt

JPEG: See attached file picture.jpg

For Part 2, We quickly realized that the key to solving this part of the assignment was to take the given file signatures (or Known-Plaintext) and attempt to find a valid key that would allow those headers.

This was done by formatting a list of file formats and iterating over them attempting to find a possible match (given possible key characters).

This led to 4 possible matches, and knowing that the key length was longer than Part 1, indicated that the longest possible match would be our best bet, which was a key that produced a plaintext Exif header.

The format of the resulting key led to the realization that the key was actually a pair of coordinates, which helped narrow the search of possible keys to key characters that would follow the format of geographic data

From there, the strategy was to iterate over possible keys conforming to those rules that would produce a valid JPEG file containing Exif data.

Comparing the file to a non-encrypted file with Exif data gave insight as to the format and metadata contained within, specifically date-time signature, which was used to verify the key without fully decrypting the file each time.

Part 3:

1. DES encryption in OFB and CFB modes both resulted in a file the same size as the plaintext. CBC and ECB resulted in a file 8 bytes longer. The file encrypted in CBC-mode is 8 bytes longer due to the inclusion of its Initilization Vector (initial random number used to XOR plaintext before encryption), whereas ECB is 8 bytes longer due to padding its last block to 64-bits. The files encrypted in OFB and CFB modes are likely the same size because they have had their initilization vectors omitted.

Since ECB splits files into 8-byte blocks and our input was the same 8 bytes repeated, the resulting ciphertext is also a repetition of 8-blocks.

The first 8 bytes of the OFB and CFB are the same because in both cases they take the same secret key and XOR it with the plaintext. However, the following bytes are different because where in CFB mode the resulting ciphertext is used in the next steps block cipher encryption, in OFB mode only the block cipher encrypted key (with initialization vector if it exists) is passed to the next step.

Assuming from it's length that CBC mode included an initilization vector, the reason its first line differs from CFB and OFB is due to the inclusion of the vector (whether it be the vector itself, or that the plaintext was XOR'd with the vector)

2. In OFB mode it resulted in a replacement of one plaintext character at position 19. This is because the decryption simply does an XOR on the ciphertext using the corresponding one-time pad, so the change of one byte of input doesn't affect the output of another, only the corresponding byte of output is changed.

In CBC mode it resulted in a mangled block 3 and a replacement of one plaintext character at position 27. This is because in decryption the ciphertext is passed with the key to the block cipher, and the one character change causes a completely different 8 byte value to be returned from the block cipher, which is then XOR'd with the plaintext causing an equally mangled output. The same ciphertext is then used in the next step and XOR'd with the next block, resulting in a change at the index of our modified character, in block 4.

In CFB mode it resulted in a mangled block 4 and a replacement of one plaintext character at position 19. Because in decryption the ciphertext of a given block X is used as input for the block cipher of block X+1 (resulting in a different output from the block cipher), a one character change in the ciphertext results in a change of the corresponding index in the same block, as well as a mangled next block.

In ECB mode it resulted in only a mangled block 3. This is because all blocks are encrypted and decrypted independently of one another, so a change in one block does not affect any others, but because the ciphertext is passed a block cipher a one character change affects the entire output of that block.

Workload Division:

Part 1:	key find – Omar
	Decrypt – Theo
Part 2:	Get Signatures – Omar
	Find pattern in key – Omar
	Find possible fileTypes – Theo
	Decrypt – Theo
Part 3:	All - Theo