Report

**Analysis**

Tasks 1 and Task 2 were the building blocks of this complete assignment as they involved understanding the implantation of Feistel Cipher through code. We used the Python language to complete all the tasks. Furthermore, the completion of the aforementioned tasks was fundamental in order to accomplish both Task 5 and Task 7, as only a change of parameters was required. Lastly, the numpy library was imported as numpy arrays were more malleable, user friendly and memory effective thus perfectly suiting our requirements.

**Task 1 and Task 2**

We formed four distinct functions: Subkey Generation, Round Function, Encryption and Decryption. In order to encrypt a cipher function, it is necessary to first know the following information: Plaintext, Master key, number of rounds, and Encrypting Function. All this information was available in the task description. Our Subkey generation function used the master key to generate subkeys equal to the number of rounds which was accomplished through a ‘For loop’. The Round Function was formed by converting the arithmetic function to its python equivalent which was then passed through a loop for the defined ranges. There were two separate functions as there were two distinct ranges and they were distinguished by (If – else) statement.

The encryption function first split the plaintext into two equal parts and converted these parts to their binary equivalent. It was followed by performing substitution, linear transformation and transposition on the plaintext. Substitution was done through calling the Round Function however both the linear transformation i.e. XOR and the transposition were performed in this function by running a for loop. In the end the two blocks of the ciphertext are concatenated.

The decryption function works in the same manner however rather than using the plaintext, we used the cipher text. Similarly, in the end we concatenated the two blocks of plaintext.

**Task 5 and Task 7**

These tasks were completed in the same manner as task 1. The only difference being the round function and the other basic parameters.

**Task 3**

We needed to find the two matrixes A and B in order to check whether the overall behavior of the Feistel Cipher was linear (x = Ak + Bu). Therefore, there were two cases that needed to be fulfilled. The first was that the round function was linear and the second was that the subkey generation was linear. We started by assuming values for plaintext, master key and the number of rounds which we then used to find the ciphertext through the encryption function. This was followed by forming a function “Find Matrices that used “message length”, “ciphertext length”, “key length” and number of rounds as input. We initialized the different parameters in the form of matrixes.

**Task 4**

The intruder knows the system as in the plaintext/ciphertext pair and wishes to find the key. We read the file KPA-data-Zurich and use it to create arrays of plaintext and ciphertext in the format that we require. As it is initially in hexadecimal, we convert it to binary for further computation. Lastly, we find k given (u,x) by computing the formula k=A^(-1)\*(x+B\*u). We calculate the inverse of the matrix by using a build-in python function.

**Task 8**