Question 3

The Algorithm for computing the Simplified AES SBox table does exactly that, it computes a table. However, this algorithm shows us how we can compute the SBox directly, without doing a table look up. Write a Sage function to compute the Simplified AES SBox and Inverse SBox directly. Meaning, write functions that take elements of L and return the element of L that the SBox table (or Inverse SBox) lookup would map to. (This is more than a textbook exercise. Some people consider the AES SBox lookups to be insecure because they can leak information through the cache. Such vulnerabilities are called Side Channels. Computing SBoxes without lookups is one way to mitigate this type of attack. Although there is a conditional statement in this SBox computation, which could be exploited by a side channel attack.)

Solution:

The Algorithm for computing the Simplified AES SBox matrix does exactly that, it computes a table. However, this algorithm shows us how we can compute the SBox directly, without doing a table look up. Write a Sage function to compute the Simplified AES SBox and Inverse SBox directly. Meaning, write functions that take elements of L and return the element of L that the SBox table (or Inverse SBox) lookup would map to. (This is more than a textbook exercise. Some people consider the AES SBox lookups to be insecure because they can leak information through the cache. Such vulnerabilities are called Side Channels. Computing SBoxes without lookups is one way to mitigate this type of attack. Although there is a conditional statement in this SBox computation, which could be exploited by a side channel attack.)

def SAES\_DirectlyComputeSBox(nibble):

state0 = nibble;

if (0 != state0):

state1 = state0^-1;

else:

state1 = 0;

state2 = L(A.transpose()\*V(state1) + b);

return state2;

def SAES\_DirectlyComputeInverseSBox(nibble):

state0 = nibble;

state1 = L(A.transpose().inverse()\*(V(state0) + b));

if (0 != state1):

state2 = state1^-1;

else:

state2 = 0;

return state2;