Mohan Krishna Hasti

hastim@oregonstate.edu

Assignment Homework 1

1. **AES**

   Plaintext = 00 00 00 00 00 00 C1 A5 51 F1 ED C0 FF EE B4 BE
   Key = 00 00 01 02 03 04 DE CA F0 C0 FF EE 00 00 00 00

   After Add round key operation Plaintext (bitwise XOR) Key
   Cipher text = 00 00 01 02 03 04 1F 6F A1 31 12 2E FF EE B4 BE

   After first sub bytes operation
   Cipher text = 63 63 7C 77 7B F2 C0 A8 32 C7 C9 31 16 28 8D AE

2. **Review the code.**
   a. **What is the type of problem here?**
      Since there is an if condition in MixColumns_Mult_by2 there is a possibility of a timing side channel attack.
   b. **Identify the function(s) with undesired behavior.**
      unsigned char MixColumns_Mult_by2(unsigned char Input)

   c. **Suggest a code fragment that solves the problem under idealized assumptions.**
      We can use a LookUpTable to solve this issue.

```
unsigned char MixColumns_Mult_by2(unsigned char Input) {
        unsigned char LUT[256] = {precomputed LUT of input carry check and XOR values};

        return (Output[input]);

}
```

   d. **Is your solution processor independent? Please provide appropriate reasoning.**

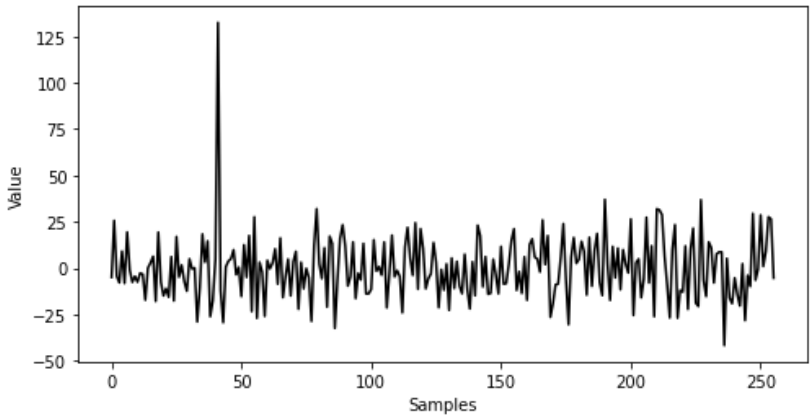
      Yes, it is processor independent as the code involves only lookup for the input and does not have any operations or a if condition for timing side channel attack to take place.

3. **Perform a timing side-channel analysis.**
   a. **Recover the key.**
      Key = 41 142 79 30 183 104 193 19 15 246 189 223 236 119 47 176

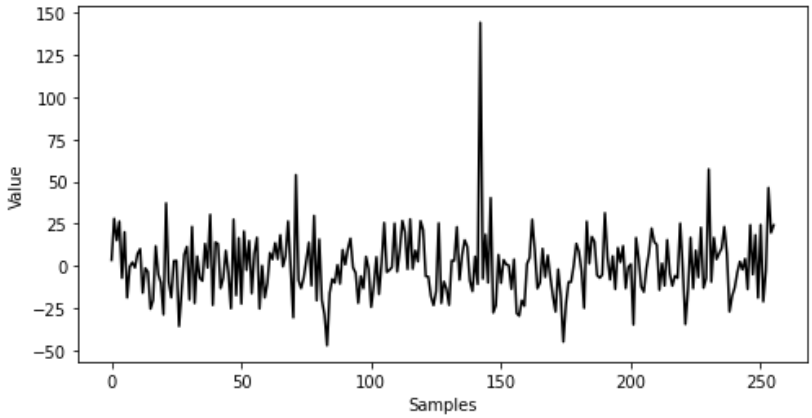
Mohan Krishna Hasti

hastim@oregonstate.edu

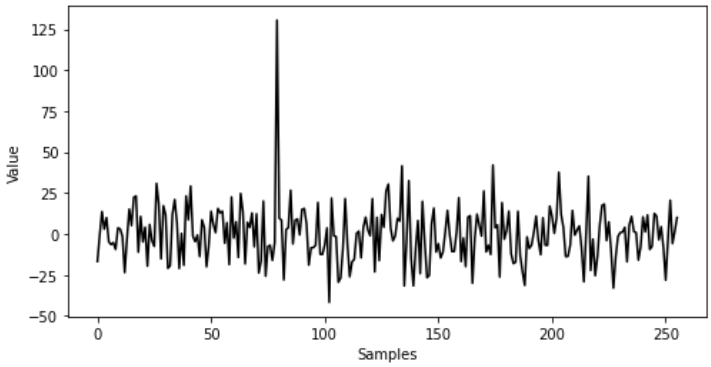
Index = 0 key = 41 Time = 13.663481712341309



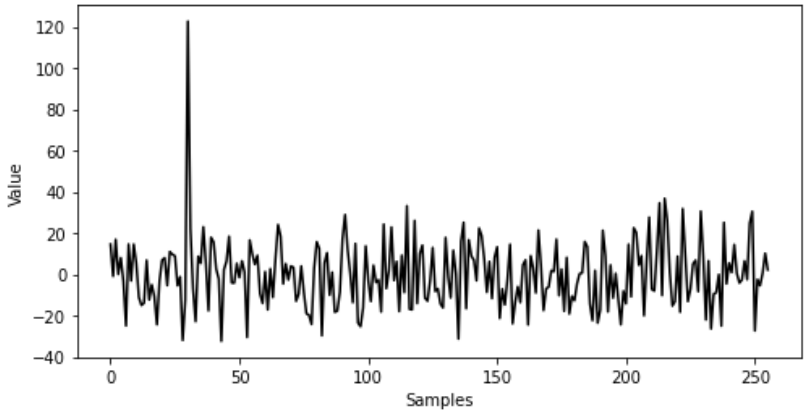
Index = 1 key = 142 Time = 12.162993669509888



Index = 2 key = 79 Time = 12.24288558959961



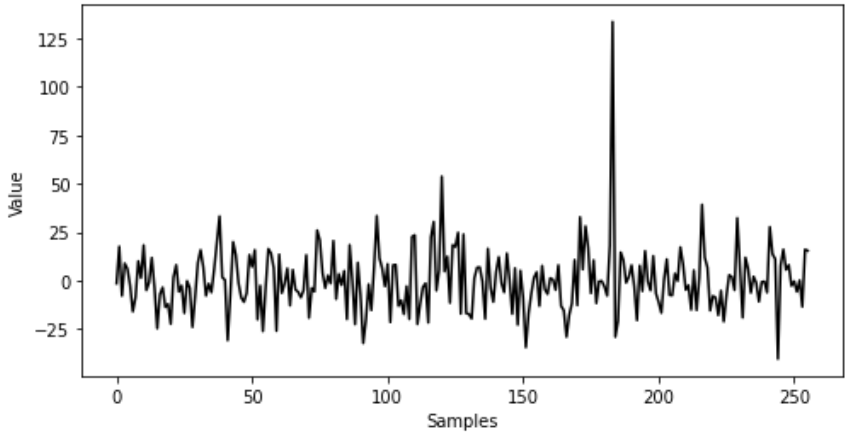
Index = 3 key = 30 Time = 12.142224073410034



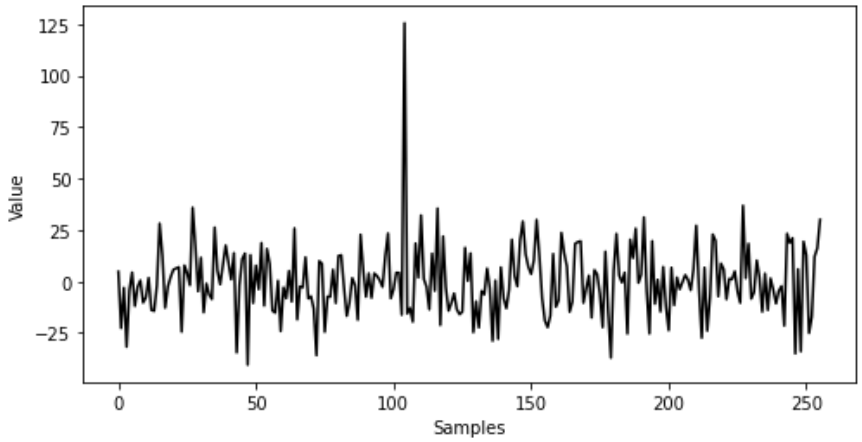
Mohan Krishna Hasti

hastim@oregonstate.edu

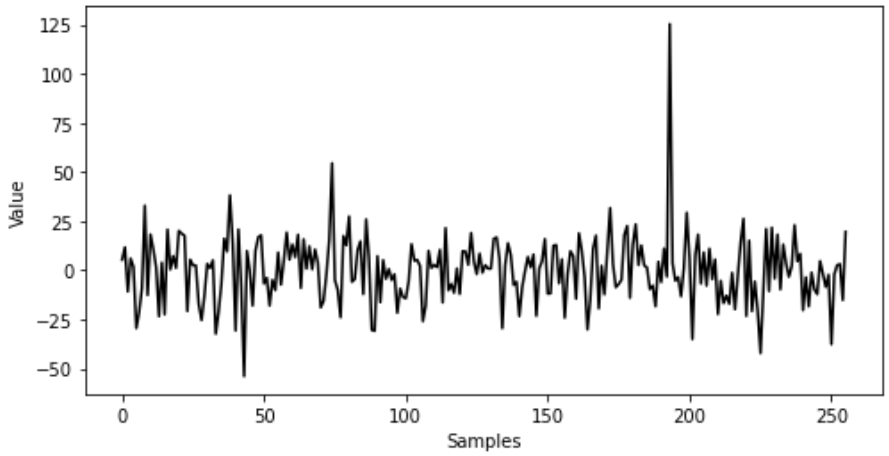
Index = 4 key = 183 Time = 16.299654722213745



Index = 5 key = 104 Time = 16.783077716827393



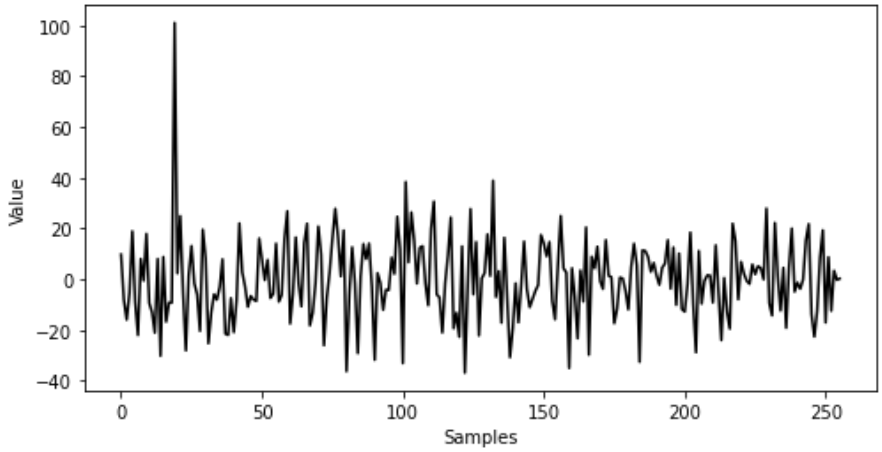
Index = 6 key = 193 Time = 12.327324151992798



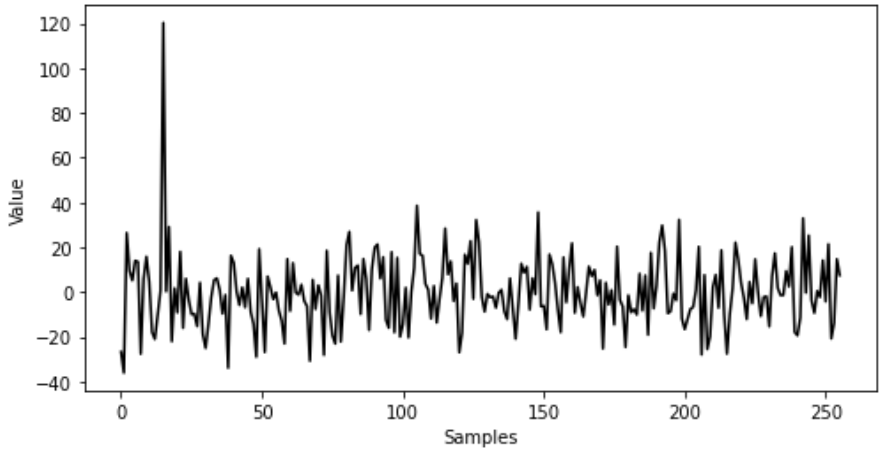
Mohan Krishna Hasti

hastim@oregonstate.edu

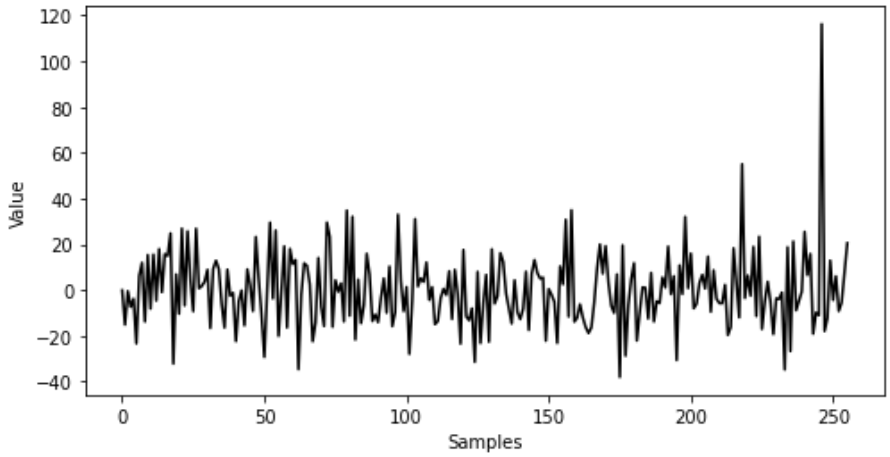
Index = 7 key = 19 Time = 13.930487632751465



Index = 8 key = 15 Time = 12.363195419311523



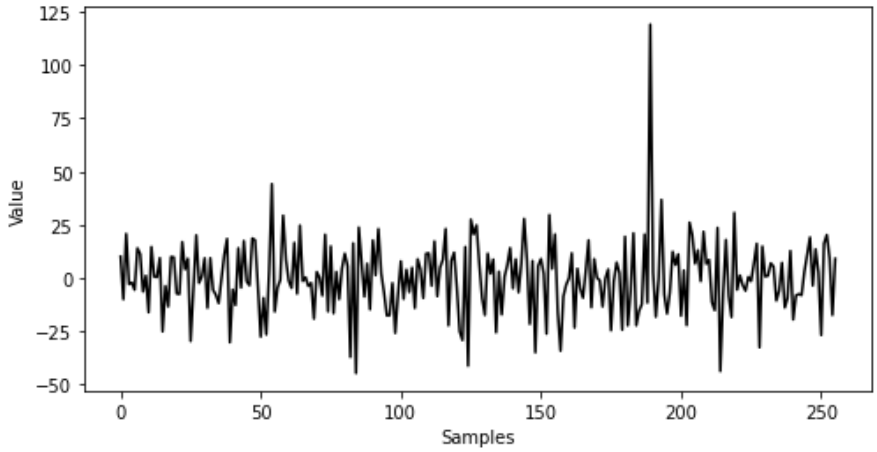
Index = 9 key = 246 Time = 12.946616172790527



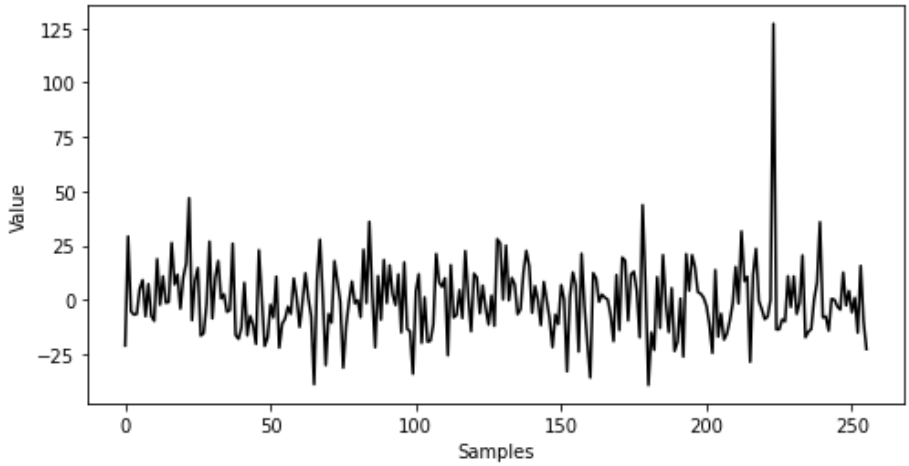
Mohan Krishna Hasti

hastim@oregonstate.edu

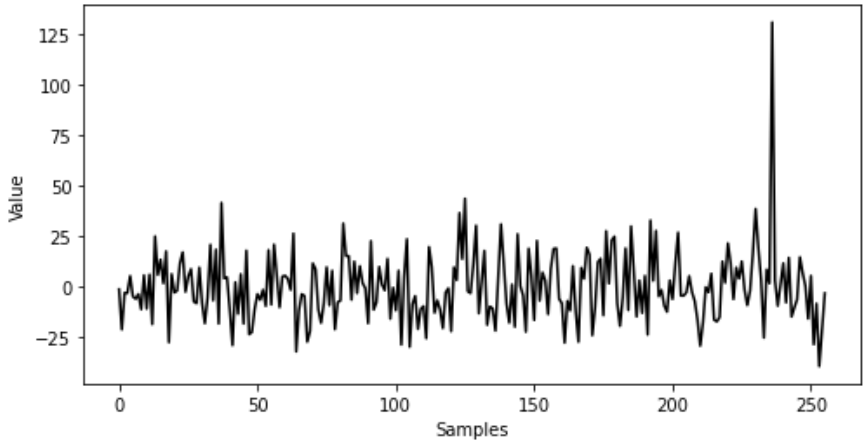
Index = 10 key = 189 Time = 15.955958843231201



Index = 11 key = 223 Time = 14.478906869888306



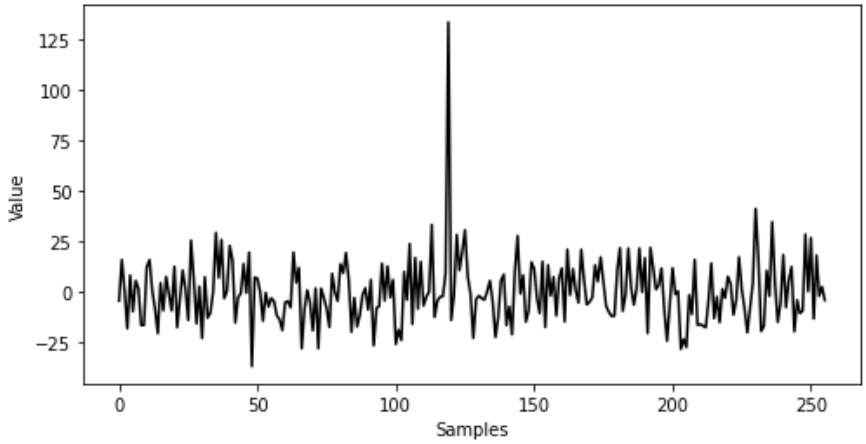
Index = 12 key = 236 Time = 15.146339654922485



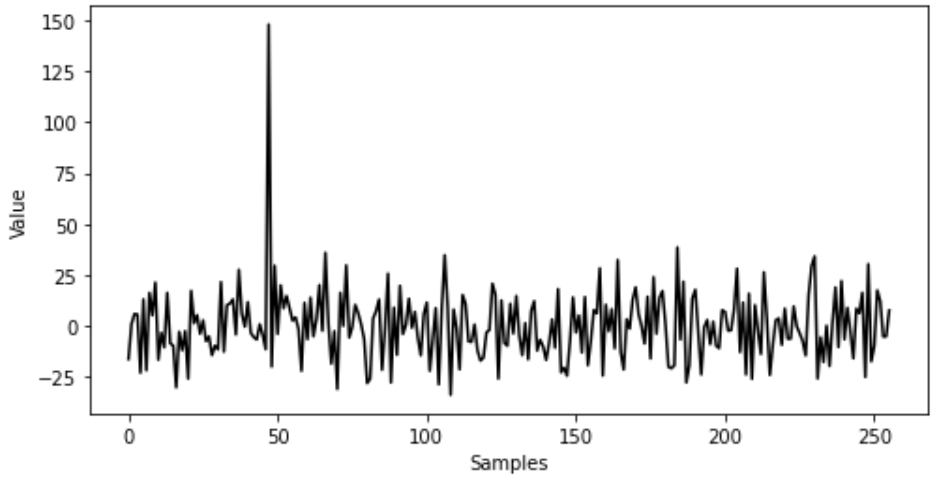
Mohan Krishna Hasti

hastim@oregonstate.edu

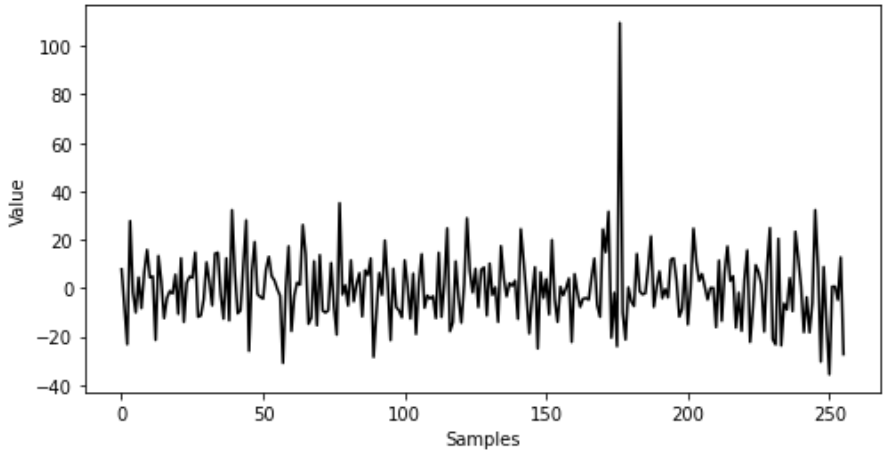
Index = 13 key = 119 Time = 12.232941627502441



Index = 14 key = 47 Time = 16.01475691795349



Index = 15 176 Time = 12.450374364852905



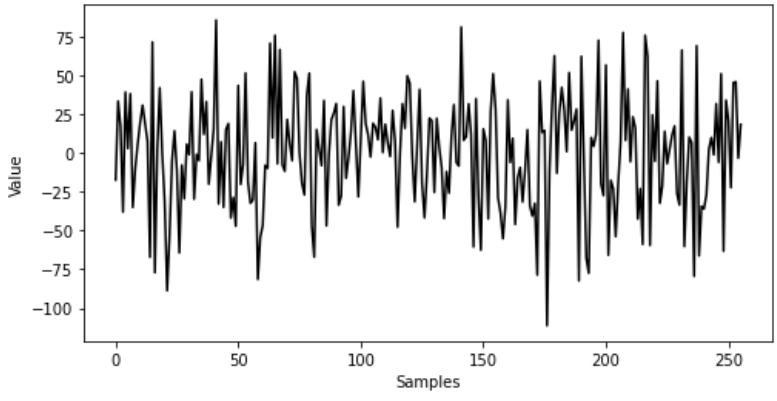
Total Time 224.07885074615479 -> single thread process

Mohan Krishna Hasti

hastim@oregonstate.edu

b. Check how many samples you need for each key byte.
For all Key bytes, N = 150,000

41



N = 100,000

c. Optimize your code

I used numpy functions for all my operations
First I converted **sbox into and np.array**
Created a **k_array=np.array(range(256))** which will create an np.array of 0-255 values
Then I created an

and_128 = np.empty(256, dtype = int)
and_128.fill(128) with all 128 values
used this to compute the np.bitwise_xor

Then I created a **p_i = np.array** of 256 which is a **256x256** 3D array each p_i[0-255] will have the corresponding timings which can be later used to compute the XOR and find the MSB.

created 2 groups group1 and group0 to put the timings for MSB 1 and MSB 0 then used that to compute the average and find the delta value.

I used numpy functions wherever possible to minimise the timings. Creating numpy arrays and eliminating the use of multiple nested pyhton for loops increased the performance of computation drastically.

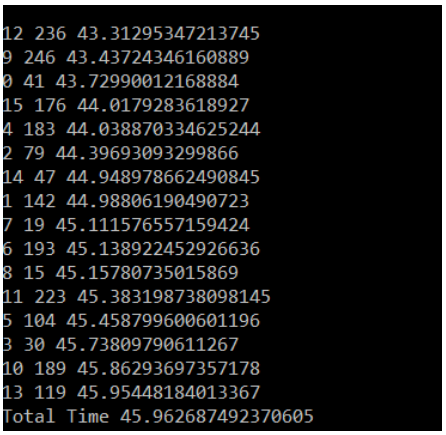
Used python multiprocessing library to process the computations parallelly

with Pool(2) as p:

p.map(fun, range(16)) - I used to parallelly process the 16 key bytes which creates 16 different threads and each computes for 1million rows. Which increased the performance.
Using multiprocessor the Total time to compute = 45.962687492370605

Mohan Krishna Hasti

hastim@oregonstate.edu



Processor : Intel(R) Xeon(R) CPU X5650 @ 2.67GHz – 6 core, 12 threads X 2 processors Total: 12 cores, 24 threads

Ram : 94GB

OS: Linux

I used Flip server to run this.