KUMAR SANATAN
ROLL NO – 2211AI24

CS 579
Cyber Security with Blockchain
ASSIGMENT 1:

INDIAN INSTITUTE OF TECHNOLOGY
PATNA

Date: 23th August 2022 **Deadline:** 30th August 2022

OBJECTIVE

Hash Function Understanding

CODE SUMMARY :

PART ONE

- We are generating/using input as string of 1000 1's so as to implement SHA 256, SHA 512 and RIPEMD. We have imported SHA256, SHA512 and RIPEMD from Crypto.Hash library.

```
from Crypto.Hash import SHA256, SHA512 , RIPEMD
from collections import Counter

#generate 1000 length input as string as all 1's
inputData="1"
for i in range(999):
    inputData = inputData + "1"
```

- For SHA 256 implementation we are running the loop for 1000 times each time incorporating one bit change in the input and observing the difference by XOR operation with respect to the hashed value of original input that is stored as variable hash256 futher storing the change in sumSHA256 variable so as to calculate the average change in the output for one bit changes(1000 times).

```
#SHA 256
hash256=int(SHA256.new(inputData.encode()).hexdigest(),base=16)
sumSHA256 = 0
for i in range(1000):
    #changing one bit
    stringWithOneBitChange = inputData[:i] + '0' + inputData[i+1:]
    #hashing
    ihash256 =  int(SHA256.new(stringWithOneBitChange.encode()).hexdigest(),base=16)
    sumSHA256+=Counter(bin(hash256^int(ihash256)))['1']
avg_change=sumSHA256/(256*1000)
print('Average change in output of SHA256 for one bit change in input=',round(avg_change*100,2),'%')
```

- For SHA 512 and RIPEMD we are performing the same operation and logic as for SHA 256 just here implementing the SHA 512 and RIPEMD hashing and thereby observing the average change in the output for one bit changes in the input for 1000 times.

```
#SHA 512
hash512=int(SHA512.new(inputData.encode()).hexdigest(),base=16)
sumSHA512 = 0
for i in range(1000):
    #changing one bit
    stringWithOneBitChange = inputData[:i] + '0' + inputData[i+1:]
    #hashing
    ihash512 =  int(SHA512.new(stringWithOneBitChange.encode()).hexdigest(),base=16)
    sumSHA512+=Counter(bin(hash512^int(ihash512)))['1']
avg_change=sumSHA512/(512*1000)
print('Average change in output of SHA512 for one bit change in input=',round(avg_change*100,2),'%')
```

```
#RIPEMD
hashRIPEMD=int(RIPEMD.new(inputData.encode()).hexdigest(),base=16)
sumRIPEMD = 0
for i in range(1000):
    #changing one bit
    stringWithOneBitChangeRIPEMD = inputData[:i] + '0' + inputData[i+1:]
    #hashing
    ihashRIPEMD =  int(RIPEMD.new(stringWithOneBitChangeRIPEMD.encode()).hexdigest(),base=16)
    sumRIPEMD+=Counter(bin(hashRIPEMD^int(ihashRIPEMD)))['1']
avg_change=sumRIPEMD/(160*1000)
print('Average change in output of RIPEMD for one bit change in input=',round(avg_change*100,2),'%')
```

PART TWO

- For part two we have implemented SHA 256 to the input and then implementing RIPEMD on top of that on f(g()) operation and compared it to the original input hashed to RIPEMD and thereby compared the average change in the output.

```
hash256=int(SHA256.new(inputData.encode()).hexdigest(),base=16)
hashRIPEMDandSHA256=int(RIPEMD.new(str(hash256).encode()).hexdigest(),base=16)
sumRIPEMDandSHA256=Counter(bin(hashRIPEMD^int(hashRIPEMDandSHA256)))['1']
avg_change=sumRIPEMDandSHA256/(160)
print('Average change in output of SHA 256 followed by RIPEMD change =',round(avg_change*100,2),'%')
```

RESULT

- Following results were obtained with defined change percentage for the above carried operations. The average changes were close to 50% mark and roundabout.

```
Average change in output of SHA256 for one bit change in input= 49.96 %
Average change in output of SHA512 for one bit change in input= 50.12 %
Average change in output of RIPEMD for one bit change in input= 50.1 %
Average change in output of SHA 256 followed by RIPEMD change = 43.12 %
```