ASSIGNMENT 4
CS666: HARDWARE SECURITY OF INTERNET OF THINGS

Group G2 Nov, 2023

1.Group Members

1. VIKRANT CHAUHAN : MS CYBERSERCURITY : 231110407
2. SOUVIK MUKHERJEE : MS CYBERSECURITY : 231110405
3. VISHAL KUMAR : MTECH CYBERSECURITY : 231110058
4. VALETI LOKESH : MS CYBERSECURITY : 231110406
5. M DHILIPKUMAR : MTECH CYBERSECURITY : 231110026

2.Helping instruction for running the code files

- Kindly do *cd* to the specific folder
- Run the command python3 Group2_Assign4_Final.py
- See Output

IMPORTANT NOTE:-

Since we had more than 10k entries in PUF CRP's in Board3_10K_2_S1,S2 & S3 in the board no 3, we trimmed it to 10k CRP's.

3. Question

In this assignment, each group will be provided with 10K CRPs for Arbiter-PUF implemented on three different FPGA devices(Boards). Using these CRPs you have to calculate the Uniqueness, Reliability, and Uniformity of the 64-bit Arbiter-PUF. To calculate Uniqueness and Uniformity, 10k CRPs of board1, board2, and board3 will be used. Uniqueness will be calculated for the following cases:
• Uniqueness for: (Board1, Board2)
• Uniqueness for: (Board1, Board3)
• Uniqueness for: (Board2, Board3)

And Uniformity will be calculated for individual FPGA boards (Board1, Board2, and Board3).
The Reliability of the PUF has to be calculated for a particular FPGA board e.g., Board3. To calculate reliability each group is provided with 15 sets of 10k CRPs for Board3.

Solution

Introduction :

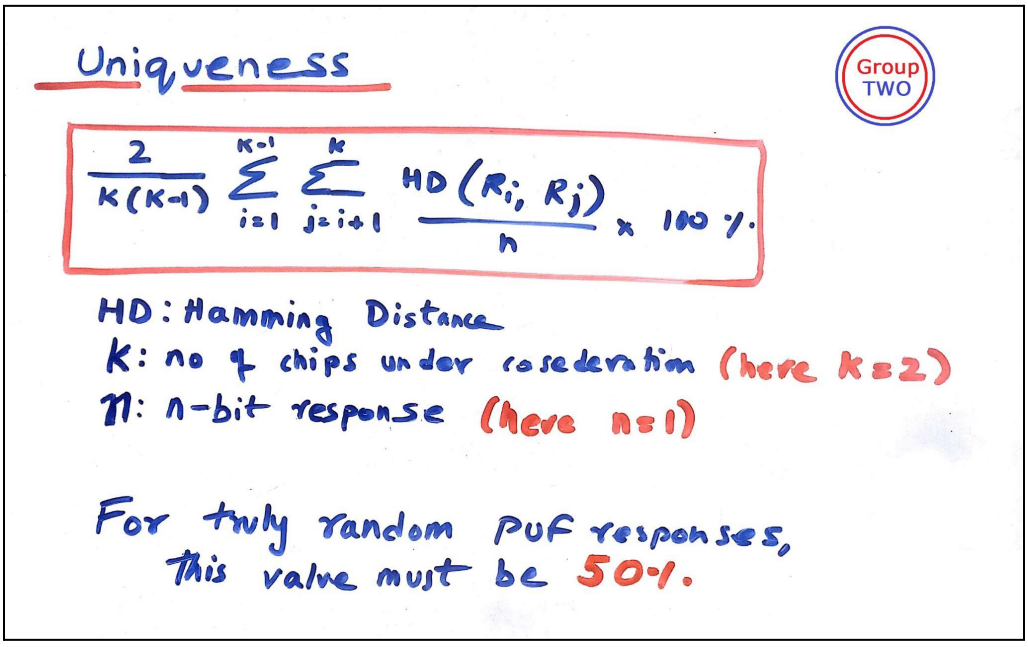
In this assignment we learn about various performance metrics for quality evaluation of PUFs(Physically Unclonable Functions).With the help of these matrices we can evaluate the quality of our PUFs.

The various performance metrics for quality evaluation of PUFs(Physically Unclonable Functions) which we learned in this assignment are:-

1.Uniqueness:-

Uniqueness is the ability of a PUF to uniquely distinguish itself among a set of PUFs instances of the same PUF type on different chips. Average Hamming Distance (HD) of PUF is used to evaluate uniqueness and the uniqueness value should be close to 50%, which means half the bits are different on average.

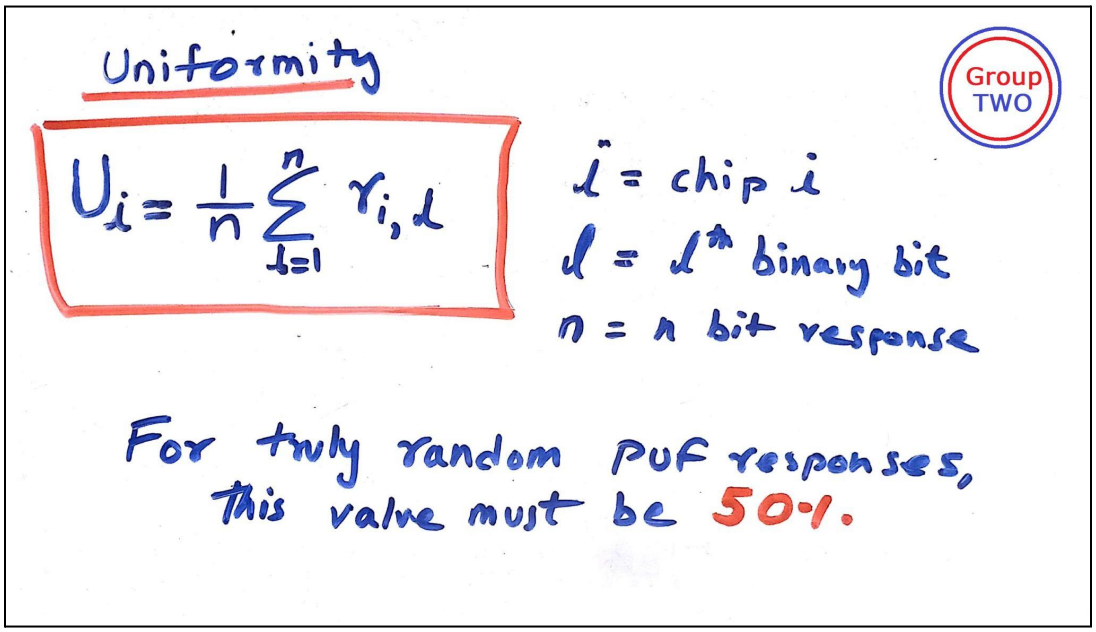
The uniqueness is given by:-



2.Uniformity:-

Uniformity estimates how uniform the proportion of 0s and 1s in the response bits of the PUF There should be a uniform distribution of 0's and 1's in a given PUF. For truly random PUF responses, this proportion must be 50%.

The uniformity is given by:-

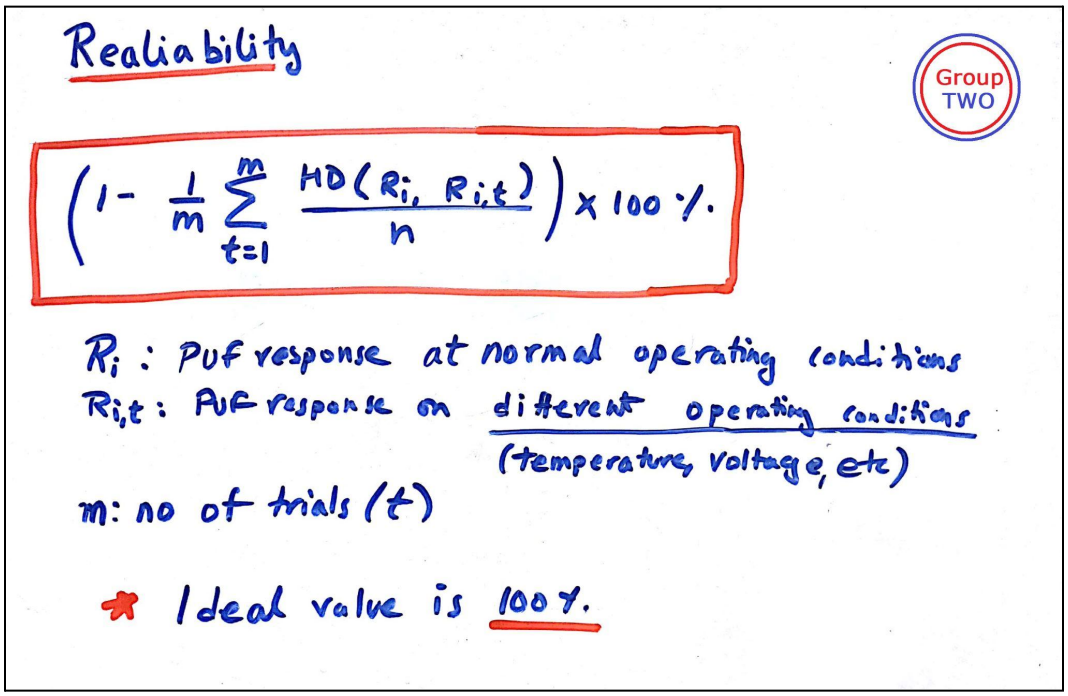


3. Reliability:-

Reliability tells us how efficient our PUF is in reproducing the response bits.The ideal PUF should have perfectly consistent response for a given challenge.But the dynamic noise, due to variation in operating voltage and temperature, affects the different circuit components in a non-uniform manner.

To evaluate reliability we evaluate the same n-bit response is extracted at a different operating condition (different ambient temperature or different supply voltage).

The reliability is given by:-



Algorithms Used :

Algorithm 1 Function For Calculating Uniqueness By Group 2

Input: Challenge Response Pairs (CRPS) where C is the Challenge Set and R is the corresponding response for each challenge present in the challenge set.Board_x contains Response set of Board X and .Board_y contains Response set of Board y.All the responses are ‘n’ bit responses.
Output: Uniqueness of Board X

```
1: Hamming_Distance=0
2: for i in range(len(Board_x)) do
3:         hd=Board_x[i]^Board_y[i]
4:          Hamming_Distance=Hamming_Distance+hd
5: end for
6: tmp=Hamming_Distance/len(Board_x)
7: uniqueness=tmp*100.00
8: return(uniqueness)
```

Algorithm 2 Function For Calculating Uniformity By Group 2

Input: Challenge Response Pairs (CRPS) where C is the Challenge Set and R is the corresponding response for each challenge present in the challenge set.Board contains Response set of Board for which we want to calculate uniformity
Output: Uniformity of Board

```
1: sum=0
2: count_responses=0
3:for row in Board do
4:        for res in row do
5:                sum=sum+res
6:                count_responses=count_responses+1
7:        end for
8: end for
9: uniformity_tmp=(sum/count_responses)*100
10: uniformity_main=round(uniformity_tmp,2)
11: return(uniformity_main)
```

NOTE:- For calculation of uniformity and uniqueness for board 3 we were given 15 files of challenge response pairs but we needed only one file so what we did is we with the help of multi-voting among all the 15 files of CRPs for the board 3 we made a single CRP filethese CRPs file responses are stored in a Board3_responses list which would be the most accurate for the calculation of the uniformity and uniqueness for the board 3 calculations and then we have used that file for calculation of uniqueness and uniformity of Board 3.

Algorithm 3: Function For Calculating Reliability By Group 2

Input: A string with “Relative address of the folder containing the 15 different (10K CRP’s) for PUF no.3 drawn out from different operating/environment conditions” embedded in it (enter the filename, exclude the “.txt” extension) [Let it be ‘g’]

Output: Reliability of the PUF no. 3 in percentage

```
1.  Initialize two arrays, a[ ] & b[ ]
2.  for (i=1;i<16;i++) do
3.      s= g+str(i)+".txt"
4.      call the “Make_Challenge_Response_list(b,s)”
5.      a.append(b)
6.      b=[]
7.  end for
8.  reliability = 0
9.  for (i=0;i<10000;i++) do
10.     f=[ ]
11.     for (j=0; j<15; j++) do
12.         f.append(int(a[j][i]))
13.     if (f.count(1)>f.count(0)) do
14.         Board3_Responses.append([1])
15.         rel=(1-(f.count(0)/15))*100
16.     end if
17.     else do
18.         Board3_Responses.append([0])
19.         rel=(1-(f.count(1)/15))*100
20.     end else
21.     end for
22.     reliability = reliability+rel
23. end for
24. return (reliability/10000)
```

The ***“Make_Challenge_Response_list(array,absolute_location)”*** takes in a “.txt” file containing 10K CRP’s, and uses file handling to open and read it. It copies every alternate line into an array (handled by count%2==0).

We used to identify our response by extracting the number between “:” and “/n” and appending it in our “array”. The pseudocode for same is:

Algorithm 4: Make_Challenge_Response_list(array,absolute_location)

Input: File's absolute location of the ".txt" file containing the CRP's
Output: Populates the array with responses extracted from the ".txt" file

```
1.  def Make_Challenge_Response_list(array,absolute_location):
2.      with open(absolute_location,"r") as file:
3.          File_data=file.readlines()
4.          count=1
5.          for line in File_data do
6.              colon_index=line.index(":")
7.              newline_index=line.index("\n")
8.              if(count%2==0) do
9.                  sliced_string_responses=line[colon_index+1:newline_index]
10.                 array.append(sliced_string_responses) #appends response into array
11.             end if
12.             count+=1
13.         end for
14.         print(count) #prints the no of lines present in the file
```

Code Snapshots :

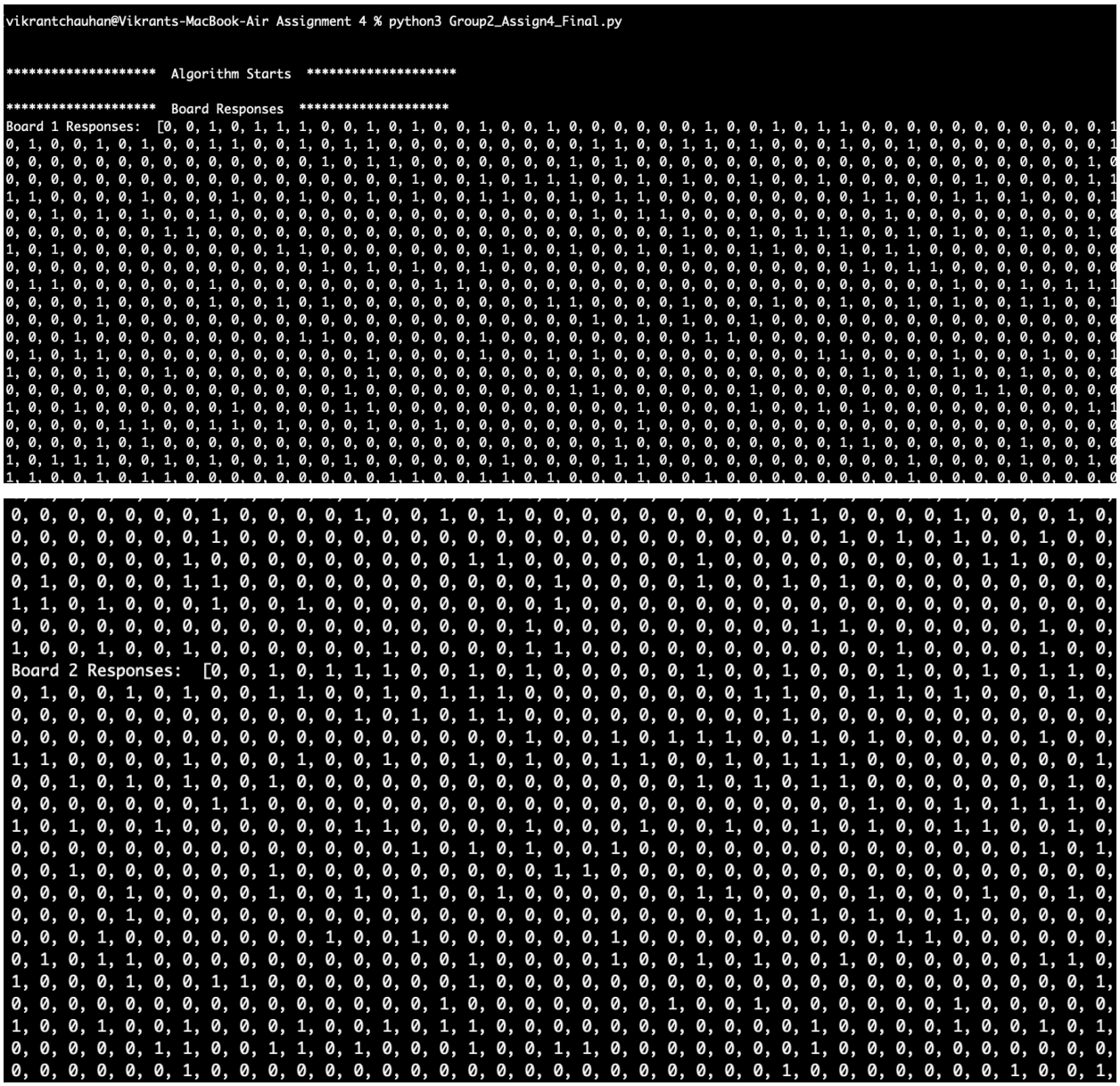
```
6   def Make_Challenge_Response_list(Board_Responses,file_name):
7       with open(file_name,"r") as file:
8           File_data=file.readlines()
9           count=1
10          for line in File_data:
11              colon_index=line.index(":")
12              newline_index=line.index("\n")
13              if(count%2==0):
14                  sliced_string_responses=line[colon_index+1:newline_index]
15                  Board_Responses.append(sliced_string_responses)
16              count+=1
17
```

```
19  def calculate_uniformity(Board):
20      sum=0
21      count_responses=0
22      for row in Board:
23          for res in row:
24              sum+=int(res)
25              count_responses+=1
26      uniformity_tmp=(sum/count_responses)*100
27      uniformity_main=round(uniformity_tmp,2)
28      return uniformity_main
```

```
35  def calculate_uniqueness(Board_x,Board_y):
36      Hamming_Distance=0
37      for i in range(len(Board_x)):
38          hd=Board_x[i]^Board_y[i]
39          Hamming_Distance=Hamming_Distance+hd
40      tmp=Hamming_Distance/(len(Board_x)/100)
41      uniqueness=tmp
42      return uniqueness
```

```
46  def reliabilityOfPUF(g):
47      a=[]
48      b=[]
49      for i in range(1,16):
50          s=g+str(i)+".txt"
51          Make_Challenge_Response_list(b,s)
52          a.append(b)
53          b=[]
54
55      realibility=0
56      for i in range(10000): #10000
57          f=[]
58          for j in range(15):
59              f.append(int(a[j][i]))
60          if(f.count(1)>f.count(0)):
61              Board3_Responses.append([1])
62              rel=(1-(f.count(0)/15))*100
63          else:
64              Board3_Responses.append([0])
65              rel=(1-(f.count(1)/15))*100
66          realibility=realibility+rel
67      return(realibility/10000)
```

Output Snapshots:-



```
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
Board 3 Responses:  [0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
```

```
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,


********************Output********************
The Realibility For Board 3 is:  99.17866666666669
The Uniformity For Board 1 is:  19.95
The Uniformity For Board 2 is:  22.0
The Uniformity For Board 3 is:  17.46
The Uniqueness For Board 1 and Board 2 is:  3.51
The Uniqueness For Board 2 and Board 3 is  5.9
The Uniqueness For Board 1 and Board 3 is:  3.93


********************  Algorithm End  ********************


vikrantchauhan@Vikrants-MacBook-Air Assignment 4 %
```

References:-

[1] CS666:Hardware Security For Internet Of Things Slide (Lecture No. 14, 15)

[2] Mukhopadhyay, D., & Chakraborty, R.S. (2014). Hardware Security: Design, Threats, and Safeguards (1st ed.). Chapman and Hall/CRC. (Indian Institute of Technology Kharagpur,West Bengal, Indian textbook)

[3] A. Maiti, V. Gunreddy, and P. Schaumont, A System-atic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. New York, NY:Springer New York, 2013, pp. 245–267