

PROGRAMMING ASSIGNMENT 1 DOCUMENTATION

Submitted By:- Amal Majeed Mucheth Abdulmajeed
200415928

This program was written in **Python** programming language to be executed with python interpreter version **2.7**. The aim of the program is to implement two functions PRGA and inverse PRGA(IPRGA) of RC4 stream cipher encryption to move from one 256 bit RC4 state (S_n, i_n, j_n) to the next 256 bit state ($S_{n+1}, i_{n+1}, j_{n+1}$) with PRGA algorithm and vice versa for IPRGA algorithm.

Implementation

***Note** : - In python , an array is called a list and can add any data type of values to it even other lists .

The function **PRGA** takes in 5 arguments :

- 1) **S** (int list) : - the 256 bit random stream array used for encryption, here we implement it as an array of size 256 with values from 0 - 255 populated randomly to emulate the Key Scheduling Algorithm result.
- 2) **i** (int) - 8 bit index pointer for swapping procedure to further randomise the array S
- 3) **j** (int) - 8 bit index pointer for swapping procedure to further randomise the array S
- 4) **n** (int) - number of times the PRGA algorithm has to be applied on initial state (S, i, j)
- 5) **seq1** (list of lists) - a list to append/keep track of the intermediate states of S while moving forward with PRGA algorithm. This list will be initially empty and will be populated with intermediate values of 'S' as PRGA execution progresses

The function **IPRGA** takes in 5 arguments :

- 6) **S** (int list): - the 256 bit random resultant array from PRGA implementation on the initial S after n steps.
- 7) **i** (int) - 8 bit index pointer value at the end of n steps of PRGA on S
- 8) **j** (int) - 8 bit index pointer value at the end of n steps of PRGA on S
- 9) **n** (int) - number of times the IPRGA algorithm has to be applied on Final state (S_n, i_n, j_n) to obtain the initial state (S, i, j)
- 10) **seq2** (list of lists) - a list to append/keep track of the intermediate states of S while moving backward with IPRGA algorithm. This list will be initially empty and will be populated with intermediate values of 'S' as IPRGA execution progresses

The function **diffr** takes in 2 arguments :

- 11) **l1** (list of lists) - the list of states that PRGA takes the random array S to reach S'
- 12) **l2** (list of lists) - the list of states that IPRGA takes the random array S' to reach S

Working

```
Initial Random state 'S' after KSA
```

```
[116, 143, 117, 205, 76, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166,
 31, 201, 203, 110, 10, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190,
 75, 105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97,
 235, 22, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 123, 234, 30, 1
93, 237, 227, 74, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6
, 152, 65, 177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221,
 197, 114, 172, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109,
 137, 199, 233, 69, 146, 228, 107, 128, 23, 214, 232, 130, 217, 48, 245, 224,
 136, 200, 249, 12, 20, 148, 19, 161, 87, 185, 238, 126, 253, 151, 189, 27, 84
, 118, 243, 15, 198, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66,
 215, 104, 95, 218, 165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 2
36, 156, 53, 191, 125, 140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179,
 170, 45, 108, 89, 26, 159, 3, 149, 70, 162, 246, 24, 36, 134, 182, 178, 195,
 122, 252, 18, 133, 64, 11, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196
, 240, 25, 226, 239, 225, 21, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 15
4, 90, 28, 210]
```

Initially an array S will be initialized using python's **random library** to emulate implementation of KSA algorithm before PRGA and give a random 256 length array with values from 0 to 255. Next the 'i' and 'j' pointer values to be swapped will be prompted to be entered by the user to simulate any random point of execution in the algorithm , along with 'n' the number of steps to move forward :

```
Enter the number of steps to proceed along PRGA : 5
Enter the current state of pointer 'i' : 0
Enter the current state of pointer 'j' : 0
```

Next , the swapping procedure begins 'n' number of times and if the programmer wants to see the intermediate states after swapping , keep pressing 'y' at the prompt when asked , if pressed 'n' then the algorithm directly takes to the end state (S_n, i_n, j_n) without further prompting :

PRGA Logic

Do you want to see next state ?(PRESS y - Yes , n - No) y
Indices swapped : 1 & 143

```
[116, 185, 117, 205, 76, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 31, 201, 203, 110, 1, 0, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75, 105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235, 22, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 1, 23, 234, 30, 193, 237, 227, 74, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65, 177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 172, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233, 69, 146, 228, 107, 128, 23, 214, 2, 32, 130, 217, 48, 245, 224, 136, 200, 249, 12, 20, 148, 19, 161, 87, 143, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 198, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218, 165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125, 140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 159, 3, 149, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 226, 239, 225, 21, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]
```

Do you want to see next state ?(PRESS y - Yes , n - No) y
Indices swapped : 2 & 4

```
[116, 185, 76, 205, 117, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 31, 201, 203, 110, 1, 0, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75, 105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235, 22, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 1, 23, 234, 30, 193, 237, 227, 74, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65, 177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 172, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233, 69, 146, 228, 107, 128, 23, 214, 2, 32, 130, 217, 48, 245, 224, 136, 200, 249, 12, 20, 148, 19, 161, 87, 143, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 198, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218, 165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125, 140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 159, 3, 149, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 226, 239, 225, 21, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]
```

Do you want to see next state ?(PRESS y - Yes , n - No) n
Indices swapped : 3 & 209

Indices swapped : 4 & 70

Indices swapped : 5 & 238

Towards the end of the algorithm , the final state (S_n, i_n, j_n) will be printed out to the programmer and the program enters IPRGA logic to reverse the effects of PRGA on the array S.

Random state 'S' after PRGA

```
[116, 185, 76, 149, 74, 226, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 31, 201, 203, 110, 10, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75, 105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235, 2, 2, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 123, 234, 30, 193, 237, 227, 117, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65, 177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 1, 72, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233, 69, 146, 228, 107, 128, 23, 214, 232, 130, 217, 48, 245, 224, 136, 200, 249, 1, 2, 20, 148, 19, 161, 87, 143, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 19, 8, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218, 165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125, 140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 1, 59, 3, 205, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 168, 239, 225, 21, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]
```


Inside the IPRGA logic , again the programmer is prompted whether he/she wants to see the intermediate array while swapping and works exactly like PRGA but all the operations are opposite in nature are order in order to reverse or inverse the effects of PRGA on S :-

```

IPRGA Logic

Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  5  &  238

[116, 185, 76, 149, 74, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 31
, 201, 203, 110, 10, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75,
105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235, 2
2, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 123, 234, 30, 193, 237,
227, 117, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65,
177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 1
72, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233,
69, 146, 228, 107, 128, 23, 214, 232, 130, 217, 48, 245, 224, 136, 200, 249, 1
2, 20, 148, 19, 161, 87, 143, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 19
8, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218,
165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125,
140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 1
59, 3, 205, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11
, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 226, 239, 225, 21
, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]

Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  4  &  70

[116, 185, 76, 149, 117, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 3
1, 201, 203, 110, 10, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75,
105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235,
22, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 123, 234, 30, 193, 237,
227, 74, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65,
177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 1
72, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233,
69, 146, 228, 107, 128, 23, 214, 232, 130, 217, 48, 245, 224, 136, 200, 249, 1
2, 20, 148, 19, 161, 87, 143, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 19
8, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218,
165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125,
140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 1
59, 3, 205, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11
, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 226, 239, 225, 21
, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]

```

And as long as the programmer keeps pressing 'y' at the prompt , he/she will be able to see the intermediate snapshots of the array S while the IPRGA algorithm works for 'n' steps.

```

Do you want to see next state ?(PRESS y - Yes , n - No)  n
Indices swapped :  3  &  209

Indices swapped :  2  &  4

Indices swapped :  1  &  143

Random state 'S' after IPRGA

[116, 143, 117, 205, 76, 168, 244, 80, 1, 102, 184, 85, 14, 37, 39, 222, 166, 3
1, 201, 203, 110, 10, 138, 2, 219, 150, 72, 208, 115, 55, 94, 60, 119, 190, 75,
105, 132, 141, 13, 194, 38, 47, 250, 120, 163, 247, 86, 83, 88, 121, 97, 235,
22, 153, 223, 78, 49, 157, 77, 204, 144, 160, 175, 187, 123, 234, 30, 193, 237,
227, 74, 98, 231, 0, 131, 248, 29, 135, 242, 92, 61, 112, 57, 230, 6, 152, 65,
177, 158, 58, 192, 164, 33, 127, 167, 50, 9, 43, 46, 34, 216, 221, 197, 114, 1
72, 251, 207, 40, 93, 186, 111, 52, 51, 254, 173, 206, 103, 109, 137, 199, 233,
69, 146, 228, 107, 128, 23, 214, 232, 130, 217, 48, 245, 224, 136, 200, 249, 1
2, 20, 148, 19, 161, 87, 185, 238, 126, 253, 151, 189, 27, 84, 118, 243, 15, 19
8, 202, 229, 211, 41, 68, 8, 73, 42, 220, 169, 174, 99, 66, 215, 104, 95, 218,
165, 96, 212, 100, 91, 142, 113, 181, 255, 5, 106, 188, 236, 156, 53, 191, 125,
140, 59, 82, 7, 17, 32, 4, 241, 71, 183, 81, 209, 179, 170, 45, 108, 89, 26, 1
59, 3, 149, 70, 162, 246, 24, 36, 134, 182, 178, 195, 122, 252, 18, 133, 64, 11
, 213, 54, 44, 139, 16, 63, 155, 124, 180, 145, 196, 240, 25, 226, 239, 225, 21
, 147, 56, 171, 79, 129, 101, 67, 35, 62, 176, 154, 90, 28, 210]

```

While all the above intermediate states were generated their snapshots kept on being appended to 2 python lists **seq1** and **seq2**, one for each PRGA and IPRGA algorithm and the function **diff** in the end reverses the second list , as IPRGA starts with end state of PRGA the states will be in the opposite order. This reversed list from IPRGA is compared with the list from PRGA and if they are equal , the program prints out the following :-

```

The path trace list for PRGA and IPRGA were compared and are equal and inverted

```

Hence the function confirms that the **states from PRGA can be reversed by IPRGA**

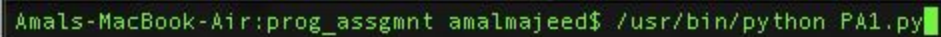
Dependencies / Software Requirements

- **Python 2.7** interpreter (On how to install if not already installed , visit : <https://www.python.org/downloads/>)

Execution

The command environment varies from system to system , the below specifications are for linux based operating systems (Ubuntu , MacOS etc) and might be different for windows systems.

1. In the command prompt type in the path to the interpreter followed by the name of the program file as follows : -

A screenshot of a terminal window with a dark background. The prompt is 'Amals-MacBook-Air:prog_assgmt amalmajeed\$'. The command entered is '/usr/bin/python PA1.py' followed by a green cursor. The rest of the terminal area is dark and mostly obscured.

```
Amals-MacBook-Air:prog_assgmt amalmajeed$ /usr/bin/python PA1.py
```

Where **PA1.py** is the name of the program and **/usr/bin/python** is the full system path for the python2.7 interpreter (this varies from system to system).