/\* C Programming Language Program to count Number of Existing linear relation in a 4-bit Crypto Substitution Box or S-Box\*/

#include<stdio.h>

int main(){

// prototype of The function read\_s\_box to read 16 elements of a 4-bit Crypto S-Box into a 4-bit linear array.

void read\_s\_box(int,int\*);

// prototype of The function s\_box\_2\_bfs to convert 16 elements of 4-bit S-Box to 4, 4-bit 16 bit long output BFs or OPBFs.

void s\_box\_2\_bfs(int\*,int[][16]);

// prototype of The function bfs\_2\_anfs to calculate 4, 4-bit 16 bit long anfs from 4, 4-bit 16 bit long output BFs or OPBFs.

void bfs\_2\_anfs(int[][16],int[][16]);

// prototype of The function linear\_approximation\_analysis to calculate Existing Linear relations with in the given S-box.

int linear\_approximation\_analysis(int[][16],int[][16]);

// prototype of The function output to display and store output in a file entitled. Linear Approximation Analysis.txt.

void output(int,int\*);

FILE \*fp;

int i,j; // Loop Variable.

int sb[16]; // Integer Linear array ro store 16 elements to store 16 elements of the 4-bit S-box.

int temp[16]; // Temporary linear array to store all elements of S-box.

int bf[4][16]; // integer two dimensional array to store each 4-bit, 16 bit long OPBF in each column of bf[4][16].

int anf[4][16];// integer two dimensional array to store each 4-bit, 16 bit long oanf in each column of anf[4][16].

int no\_elr; // To store the count of number of (no) existing linear relations(elr).

fp = fopen(" Linear Approximation Analysis.txt","w");

fprintf(fp," S-box noelr\n");

fclose(fp);

i=0;

// While loop to calculate number of Existing linear relations with in the given 4-bit S-Box.

while(i<32){

 read\_s\_box(i,sb);

 for(j=0;j<16;j++) temp[j]=sb[j];

 s\_box\_2\_bfs(sb,bf);

 bfs\_2\_anfs(bf,anf);

 no\_elr = linear\_approximation\_analysis(bf,anf);

 output(no\_elr,temp);

 i++;

 }

 return 0;

}

/\* Function to read and store each element of The given S-box in chsb[16] or character Array

first and convert each element of character array to integer and Store it to integer array sb.\*/

void read\_s\_box(int i,int\* sb){

FILE \*fp; // declaration of File Pointer.

char chsb[16]; // declaration of character array to store 16 read character elements of S-box in the linear character array chsb.

long int pos; // declaration of position variable to set the cursor in the read file.

int j; // Loop variable.

/\* block of programming to open the file S-boxes.txt in read mode and read each element of S-box in linear character array chsb.\*/

fp = fopen("S-boxes.txt","r");

pos = i\*18;

fseek(fp,pos,SEEK\_SET);

chsb[ 0] = fgetc(fp);

chsb[ 1] = fgetc(fp);

chsb[ 2] = fgetc(fp);

chsb[ 3] = fgetc(fp);

chsb[ 4] = fgetc(fp);

chsb[ 5] = fgetc(fp);

chsb[ 6] = fgetc(fp);

chsb[ 7] = fgetc(fp);

chsb[ 8] = fgetc(fp);

chsb[ 9] = fgetc(fp);

chsb[10] = fgetc(fp);

chsb[11] = fgetc(fp);

chsb[12] = fgetc(fp);

chsb[13] = fgetc(fp);

chsb[14] = fgetc(fp);

chsb[15] = fgetc(fp);

fclose(fp);

// Conversion of Character to integer.

for(j=0;j<16;j++){

 if(chsb[j]<='9') sb[j] = chsb[j]-48;

 else sb[j] = chsb[j]-55;

}

}

/\* The function is dedicated to convert Each element of the given S-box into 4-bit

Binary Numbers where each bit of 16 4-bit binary numbers constitutes an OPBF, 1st bit

of each binary number from MSB constitutes 4th OPBF, 2nd bit of each binary number

from MSB constitutes 3rd OPBF, 3rd bit of each binary number from MSB constitutes 2nd OPBF

and 4th bit of each binary number from MSB constitutes 1st OPBF\*/

void s\_box\_2\_bfs(int\* sb,int bf[][16]){

int i; int j; // Loop Variable.

int bin[4]; // Linear array binary[bin] to store corresponding 4-bit Bit patterns for each S-box element.

i = 0;

while(i<16){

// Decimal to binary conversion.

for(j=3;j>=0;j--) {

 bin[j]= sb[i]%2;

 sb[i]= sb[i]/2;

 }

// Generation of 4 OPBFs by storing it to corresponding positions of value i, in two dimensional array bf[4][16].

 for(j=0;j<4;j++)

 bf[j][i]=bin[j];

i++;

}

}

/\*This function has been dedicated to calculate 16 anf coefficients from 16 bf values in each position of bf and anf\*/

void bfs\_2\_anfs(int bf[][16],int anf[][16]){

int i = 0; // Loop variable

// ANF Coefficients Calculation block for all 4 OPBFs of an S-box.

while(i<4){

anf[i][ 0] = bf[i][ 0];

anf[i][ 1]=anf[i][ 0]^bf[i][ 1];anf[i][ 2]=anf[i][ 0]^bf[i][ 2];anf[i][ 3]= anf[i][ 0]^bf[i][ 4];anf[i][ 4]=anf[i][ 0]^bf[i][ 8];

anf[i][ 5] = anf[i][ 0]^anf[i][ 1]^anf[i][ 2]^bf[i][ 3];anf[i][ 6] = anf[i][ 0]^anf[i][ 1]^anf[i][ 3]^bf[i][ 5];

anf[i][ 7] = anf[i][ 0]^anf[i][ 1]^anf[i][ 4]^bf[i][ 9];anf[i][ 8] = anf[i][ 0]^anf[i][ 2]^anf[i][ 3]^bf[i][ 6];

anf[i][ 9] = anf[i][ 0]^anf[i][ 2]^anf[i][ 4]^bf[i][10];anf[i][10] = anf[i][ 0]^anf[i][ 3]^anf[i][ 4]^bf[i][12];

anf[i][11] = anf[i][ 0]^anf[i][ 1]^anf[i][ 2]^anf[i][ 3]^anf[i][ 5]^anf[i][ 6]^anf[i][ 8]^bf[i][ 7];

anf[i][12] = anf[i][ 0]^anf[i][ 1]^anf[i][ 2]^anf[i][ 4]^anf[i][ 5]^anf[i][ 7]^anf[i][ 9]^bf[i][11];

anf[i][13] = anf[i][ 0]^anf[i][ 1]^anf[i][ 3]^anf[i][ 4]^anf[i][ 6]^anf[i][ 7]^anf[i][10]^bf[i][13];

anf[i][14] = anf[i][ 0]^anf[i][ 2]^anf[i][ 3]^anf[i][ 4]^anf[i][ 8]^anf[i][ 9]^anf[i][10]^bf[i][14];

anf[i][15] = anf[i][ 0]^anf[i][ 1]^anf[i][ 2]^anf[i][ 3]^anf[i][ 4]^anf[i][ 5]^anf[i][ 6]^anf[i][ 7]

 ^anf[i][ 8]^anf[i][ 9]^anf[i][10]^anf[i][11]^anf[i][12]^anf[i][13]^anf[i][14]^bf[i][15];

i++;

 }

}

// Function Block to Calculate Number of Existing Linear relations.

int linear\_approximation\_analysis(int bf[][16],int anf[][16]){

int i,j;// Loop Variables.

int np;// Nonlinear part of equation.

int lp;//linear part of equation.

// Input vectors in each row of two dimensional array variable x[4][16].

int x[4][16] = {{0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1},

 {0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1},

 {0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1},

 {0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1}};

// Counting Number of existing Linear Relation for four OPBFs of the given S-box.

int count = 0;

for(i=0;i<4;i++){

 for(j=0;j<16;j++){

np = (anf[i][ 5]&x[ 3][ j]&x[ 2][ j])^(anf[i][ 6]&x[ 3][ j]&x[ 1][ j])^(anf[i][ 7]&x[ 3][ j]&x[ 0][ j])^(anf[i][ 8]&x[ 2][ j]&x[ 1][ j])^(anf[i][ 9]&x[ 2][ j]&x[ 0][ j])^(anf[i][10]&x[ 1][ j]&x[ 0][ j])^(anf[i][11]&x[ 3][ j]&x[ 2][ j]&x[ 1][ j])^(anf[i][12]&x[ 3][ j]&x[ 2][ j]&x[0][ j])^(anf[i][13]&x[ 3][ j]&x[ 1][ j]&x[ 0][ j])^(anf[i][14]&x[ 2][ j]&x[ 1][ j]&x[ 0][ j])^(anf[i][15]&x[ 3][ j]&x[ 2][ j]&x[ 1][ j]&x[ 0][ j]);

lp = anf[i][0]^(anf[i][ 1]&x[1][ j])^(anf[i][2]&x[2][ j])^(anf[i][3]&x[3][ j])^(anf[i][4]&x[4][ j]);

if((np==0)&&(lp==bf[i][j])) count++;

 }

 }

 return count;

}

// Output Block.

void output(int no\_elr,int\* sb){

FILE \*fp;

int i;//Loop Variable.

// File printing for create a data file.

fp = fopen(" Linear Approximation Analysis.txt","a");

for(i=0;i<16;i++)fprintf(fp,"%x",sb[i]);

fprintf(fp," ");

fprintf(fp,"%d",no\_elr);

fprintf(fp,"\n");

fclose(fp);

}