AtharvK9 / Simplified-Data-Encryption-Standard-S-DES-Algorithm Public

forked from AkshayChavan7/Simplified-Data-Encryption-Standard-S-DES-Algorithm

Code Pull requests Actions Projects Wiki Security Insights Se

Simplified-Data-Encryption-Standard-S-DES-Algorithm / SDES.cpp

AkshayChavan7 4 years ago

474 lines (385 loc) · 8.97 KB

Code Blame

```
/*
TITLE: Program to implement S-DES Algorithm

Akshay S. Chavan                BE-B(25)

NOTE: Refer the PDF that I've kept in the same folder. It contains the information about the fu
*/

#include<iostream>

using namespace std;


static int round1_key[8],round2_key[8];
void p10(int key[])
{
        //Input:  1 2 3 4 5 6   7 8 9 10
        //Output: 3 5 2 7 4 10  1 9 8 6
        int out[10]={3,5,2,7,4,10,1,9,8,6};
        int temp[10];

        for(int i=0;i<10;i++)          //backup key
                temp[i]=key[i];
        for(int i=0;i<10;i++)
        {
                key[i]=temp[out[i]-1];
                //cout<<key[i]<<"\t";
        }

}

void p8(int key[])
{
        //Input:  1 2 3 4 5 6 7  8 9 10
        //Output: 6 3 7 4 8 5 10 9
        int out[8]={6,3,7,4,8,5,10,9};
        int temp[10];

        for(int i=0;i<10;i++)           //backup key
                temp[i]=key[i];

        //cout<<"New key"<<endl;
        for(int i=0;i<8;i++)
        {
                key[i]=temp[out[i]-1];
                //cout<<key[i]<<"\t";
        }

}

void p4(int s0s1[])
{
        //Input:  1 2 3 4
        //Output: 2 4 3 1
        int out[4]={2,4,3,1};
        int temp[4];

        for(int i=0;i<4;i++)            //backup array
                temp[i]=s0s1[i];

        for(int i=0;i<4;i++)
        {
                s0s1[i]=temp[out[i]-1];
        }

}

void left_shift(int left_half[], int right_half[], int shift_count)            //left shift by
{
        int temp1=left_half[0];
        int temp2=right_half[0];

        for(int i=0;i<4;i++)
        {
                left_half[i]=left_half[i+1];
                right_half[i]=right_half[i+1];
        }
        left_half[4]=temp1;
        right_half[4]=temp2;

        if(shift_count==2)
                left_shift(left_half,right_half,1);

}

int* generate_key(int key[],int round)
{
        int left_half[5],right_half[5];
        static int key1[10],key2[8];
        p10(key);
        for(int i=0;i<10;i++)
        for(int i=0;i<10;i++)
        {
                if(i<5)
                {
                        left_half[i]=key[i];
                        //cout<<left_half[i]<<"\t";
                }
                else
                {
                        //cout<<"right"<<endl;
                        right_half[i-5]=key[i];
                        //cout<<right_half[i-5]<<"\t";
                }
        }

        left_shift(left_half,right_half,1);

        for(int i=0;i<5;i++)            //combine left_half and right_half to form key1
        {
                key1[i]=left_half[i];
                key1[i+5]=right_half[i];
        }
        if(round==1)
        {
                p8(key1);
                return key1;
        }
        else
        {
                left_shift(left_half,right_half,2);
                for(int i=0;i<5;i++)            //combine left_half and right_half to form key1
                {
                        key2[i]=left_half[i];
                        key2[i+5]=right_half[i];
                }
                p8(key2);
                return key2;
        }
}


void initial_permutation(int pt[])
{
        //Input:  1 2 3 4 5 6 7 8
        //Output: 2 6 3 1 4 8 5 7
        int out[8]={2,6,3,1,4,8,5,7};
        int temp[8];

        for(int i=0;i<8;i++)            //backup Plain Text Array
                temp[i]=pt[i];

        for(int i=0;i<8;i++)
        {
                pt[i]=temp[out[i]-1];
```

```
            //cout<<pt[i]<<"\t";
        }
}


void inverse_initial_permutation(int pt[])
{
        //Input:  2 6 3 1 4 8 5 7
        //Output: 1 2 3 4 5 6 7 8
        int out[8]={2,6,3,1,4,8,5,7};
        int temp[8];

        for(int i=0;i<8;i++)            //backup Plain Text Array
                temp[i]=pt[i];

        for(int i=0;i<8;i++)
        {
                pt[out[i]-1]=temp[i];
        }
}

int* expand_and_permute(int right_half[])
{
        //Input:  1 2 3 4
        //Output: 4 1 2 3 2 3 4 1
        int out[8]={4,1,2,3,2,3,4,1};
        int temp[4];
        static int expanded_right[8];

        for(int i=0;i<4;i++)            //backup Plain Text Array
                temp[i]=right_half[i];

        for(int i=0;i<8;i++)
        {
                expanded_right[i]=temp[out[i]-1];
                //cout<<expanded_right[i]<<"\t";
        }
        return expanded_right;
}

int get_S0(int row,int column)
{
        int s0[4][4]={
                        {01,00,11,10},
                        {11,10,01,00},
                        {00,10,01,11},
                        {11,01,11,10}
                        };
        return s0[row][column];
}

int get_S1(int row,int column)
{
        int s1[4][4]={
```

```
                        {00,01,10,11},
                        {10,00,01,11},
                        {11,00,01,00},
                        {10,01,00,11}
                        };
                return s1[row][column];
        }

        int* rounds(int pt[],int key[],int round_no,int flag)
        {
                int left[4],right[4],*expanded_right,s0[4],s1[4],temp_key[10];

                /*cout<<"\n\n Text to be decoded:\n";
                for(int i=0;i<8;i++)
                {
                        cout<<pt[i];
                }*/

                //cout<<"\n\nKey:";

                cout<<"\nROUND-"<<round_no;
                for(int i=0;i<10;i++)
                {
                        //cout<<key[i]<<"\t";
                        temp_key[i]=key[i];             //backup initial key as key gets changed furthe
                }

                if(round_no==1)
                initial_permutation(pt);        //step1 initial permutation of plain text


                //cout<<"\n\nleft half:\n";
                //divide into two halves
                for(int i=0;i<4;i++)
                {
                        left[i]=pt[i];
                        right[i]=pt[i+4];
                        //cout<<left[i];
                }
                expanded_right= expand_and_permute(right);

                /*cout<<"\n\nexpanded_right:\n";
                for(int i=0;i<8;i++)
                        cout<<expanded_right[i];

                */
                static int* key1;
                if(flag==0)             //flag=0 is for encoding
                {       key1=generate_key(key,round_no);                //key1 for round1 and key2 for
                        if(round_no==1)
                        {
                                for(int i=0;i<8;i++)
                                        round1_key[i]=key1[i];          //backup key for decoding
                        }
```

```
            else
            {
                for(int i=0;i<8;i++)
                    round2_key[i]=key1[i];
            }
            cout<<"\n\nEncode Key of Round "<<round_no<<endl;
            for(int i=0;i<8;i++)
            {
                cout<<key1[i];
            }
        }
        else                        //else flag=1 ie. for decoding
        {
            //cout<<"\n\n\nInside decode";
            //for decoding we use the keys in reverse order

            if(round_no==1)                 //if round1 use key2
            {
                //cout<<"\n Inside round1";
                for(int i=0;i<8;i++)
                {
                    key1[i]=round2_key[i];
                    //cout<<round2_key[i];
                    //cout<<"test";
                }
            }
            else                            //if round2 use key1
            {
                //cout<<"\n Inside round2";
                for(int i=0;i<8;i++)
                {
                    //cout<<round1_key[i];
                    key1[i]=round1_key[i];
                }
            }

            cout<<"\n\nDecode Key of Round "<<round_no<<endl;
            for(int i=0;i<8;i++)
            {
                cout<<key1[i];
            }
        }
        /*cout<<"\n\nExpanded right\n";
        for(int i=0;i<8;i++)
        {
            cout<<expanded_right[i]<<"\t";
        }
        cout<<"\n\n";*/

        for(int i=0;i<8;i++)
        {
            expanded_right[i]=expanded_right[i] ^  key1[i];
            if(i<4)
                s0[i]=expanded_right[i];
                else
                        s1[i-4]=expanded_right[i];
        }

        int row=s0[3]+(s0[0]*2);                        //step 4
        int column=s0[2]+(s0[1]*2);
        static int s0s1[4];
        int ss0=get_S0(row,column);
        //cout<<"\nRow: "<<row<<"Column: "<<column;
        row=s1[3]+(s1[0]*2);
        column=s1[2]+(s1[1]*2);
        //cout<<"\nRow: "<<row<<"Column: "<<column;
        int ss1=get_S1(row,column);

        s0s1[1]=ss0%10;
        s0s1[0]=ss0/10;
        s0s1[3]=ss1%10;
        s0s1[2]=ss1/10;

        /*cout<<"\n\nBefore P4:\n";
        for(int i=0;i<4;i++)
                cout<<s0s1[i];
        */
        p4(s0s1);

        static int new_plain_text[8];
        //s0s1 EXOR Left_Half from step 1
        for(int i=0;i<4;i++)
        {
                s0s1[i]=s0s1[i] ^ left[i];
                //swap the s0s1 and right half from step 1 to generate plain text for next roun
                if(round_no!=2)         //if round is not 2nd one and it's not for decoding
                {
                        new_plain_text[i]=right[i];
                        new_plain_text[i+4]=s0s1[i];
                }
                else                                            //else don't swap
                {
                        new_plain_text[i+4]=right[i];
                        new_plain_text[i]=s0s1[i];
                }
        }

        /*cout<<"\n\ns0s1:\n";
        for(int i=0;i<4;i++)
                cout<<s0s1[i];
        */
        cout<<"\n\nRound "<<round_no<<" Output:\n";
        for(int i=0;i<8;i++)
                cout<<new_plain_text[i]<<"\t";
        cout<<endl;



```

```
		if(round_no==1)
		{
			//cout<<"\n\ngoing for next round\n";
			if(flag==0)		//if encoding
				rounds(new_plain_text,temp_key,2,0);
			else			//else decoding
				rounds(new_plain_text,temp_key,2,1);
		}
		else
		{
			return new_plain_text;
		}

	}

	int* encode(int pt[],int* round_text,int key[])
	{
		round_text=rounds(pt,key,1,0);
		inverse_initial_permutation(round_text);

		cout<<"\n\n-------------FINAL CIPHER TEXT-------------\n";
		for(int i=0;i<8;i++)
			cout<<round_text[i];

		return round_text;
	}

	void decode(int pt[], int* cipher_text,int key[])
	{
		int *new_ct=rounds(cipher_text,key,1,1);		//flag=1 for decoding
		inverse_initial_permutation(new_ct);

		cout<<"\n\n-------------DECODED TEXT-------------\n";
		for(int i=0;i<8;i++)
			cout<<new_ct[i];
	}
	int main()
	{
		int *round_text, *cipher_text, pt[8],key[10];
		cout<<"\nEnter the plain text (8-bits) :";
		for(int i=0;i<8;i++)
			cin>>pt[i];
		cout<<"\nEnter the key (10-bits) :";
		for(int i=0;i<10;i++)
			cin>>key[i];

		//int pt[8]={0,1,1,1,0,0,1,0};
		//int key[10]={1,0,1,0,0,0,0,0,1,0};

		cout<<"\n-------------ENCRYPTION-------------\n";
		cipher_text=encode(pt,round_text,key);		//Encryption
		cout<<"\n\n\n-------------DECRYPTION-------------\n";
		decode(pt,cipher_text,key);					//Decryption
		return 0;
```

```
}

/*
OUTPUT:

C:\Users\Akshay Chavan\Desktop>g++ SDES.cpp

C:\Users\Akshay Chavan\Desktop>a

Enter the plain text (8-bits) :0 1 1 1  0 0 1 0

Enter the key (10-bits) :1 0 1 0 0 0 0 0 1 0

-------------ENCRYPTION-------------

ROUND-1

Encode Key of Round 1
10100100

Round 1 Output:
1       0       0       1       1       1       0       1

ROUND-2

Encode Key of Round 2
01000011

Round 2 Output:
1       1       1       0       1       1       0       1


-------------FINAL CIPHER TEXT-------------
01110111


-------------DECRYPTION-------------

ROUND-1

Decode Key of Round 1
01000011

Round 1 Output:
1       1       0       1       1       0       0       1

ROUND-2

Decode Key of Round 2
10100100

Round 2 Output:
1       0       1       0       1       0       0       1

```

```

-------------DECODED TEXT-------------
01110010
C:\Users\Akshay Chavan\Desktop>

*/
```