

Scilab Manual for  
Data Compression & Encryption  
by Prof S. Chaya  
Electronics Engineering  
Anjuman I Islam's Kalsekar Technical  
Campus<sup>1</sup>

Solutions provided by  
Prof chaya s  
Electronics Engineering  
Anjuman I Islam's Kalsekar Technical Campus

April 25, 2024

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT,  
<http://spoken-tutorial.org/NMEICT-Intro>. This Scilab Manual and Scilab codes  
written in it can be downloaded from the "Migrated Labs" section at the website  
<http://scilab.in>



# Contents

<b>List of Scilab Solutions</b>	<b>3</b>
<b>1 Huffman Encoding of a sequence</b>	<b>5</b>
<b>2 Huffman Decoding of a compressed bit sequence.</b>	<b>6</b>
<b>3 Implementation of RSA Algorithim</b>	<b>7</b>
<b>4 Diffie Hellman Key Exchange Method</b>	<b>10</b>
<b>5 Image Compression Using Block Truncation Coding</b>	<b>12</b>

# List of Experiments

Solution 1.1	Huffman encoding . . . . .	5
Solution 2.1	Huffman Decoding . . . . .	6
Solution 3.1	RSA Algorithim . . . . .	7
Solution 4.1	Diffie Hellman Key Exchange . . . . .	10
Solution 5.5	block truncation in image . . . . .	12
AP 1	LENA IMAGE . . . . .	16

# List of Figures

3.1 RSA Algorithim . . . . .	8
5.1 block truncation in image . . . . .	13

# Experiment: 1

## Huffman Encoding of a sequence

Scilab code Solution 1.1 Huffman encoding

```
1 //OS: Linux
2 //Tool Box–Huffman coding
3 //Scilab Version: Scilab 5.4.1
4 // Generate a Testmatrix
5 A=testmatrix('frk',10)+1;
6 A=A(:)';
7 [QT,QM]=huffcode(A);
8 disp('compressed Bit sequence:');
9 disp(QT);
10 disp('Code Table:');
11 disp(QM);
12 // End of Demo
```

---

# Experiment: 2

## Huffman Decoding of a compressed bit sequence.

Scilab code Solution 2.1 Huffman Decoding

```
1 //program to perform huffman decoding
2 //OS: Linux
3 //Tool Box–Huffman coding
4 //Scilab Version: Scilab 5.4.1
5 // Generate a Testmatrix
6 A=testmatrix('frk',10)+1;
7 A=A(:)';
8 [QT,QM]=huffcode(A);
9 disp('compressed Bit sequence:');
10 disp(QT);
11 disp('Code Table:');
12 disp(QM);
13 // Now, the reverse operation
14 B=huffdeco(QT,QM);
15 disp('Original:');
16 disp(A);
17 disp('Result after Uncompress:');
18 disp(B);
```

---

# Experiment: 3

## Implementation of RSA Algorithm

Scilab code Solution 3.1 RSA Algorithm

```
1 //OS: Windows 7
2 //Scilab Version: Scilab 5.4.1
3 clc;
4 clear all;
5 p=input("Enter the 1st prime no."); //Input: Taking
    the first prime no. for RSA
6 //Input ex. p=11
7 q=input("Enter the 2nd prime no."); //Input: Taking
    the second prime no. for RSA
8 //Input ex. q=5
9 n=p*q;
10 phi=(p-1)*(q-1); //Tuotient Function
11 printf("Enter the value of e >400"); //Input: value
    of e (such that phi and the no. entered by you
    are relatively prime)
12 //Input ex. phi=7
13 e=input("");
```

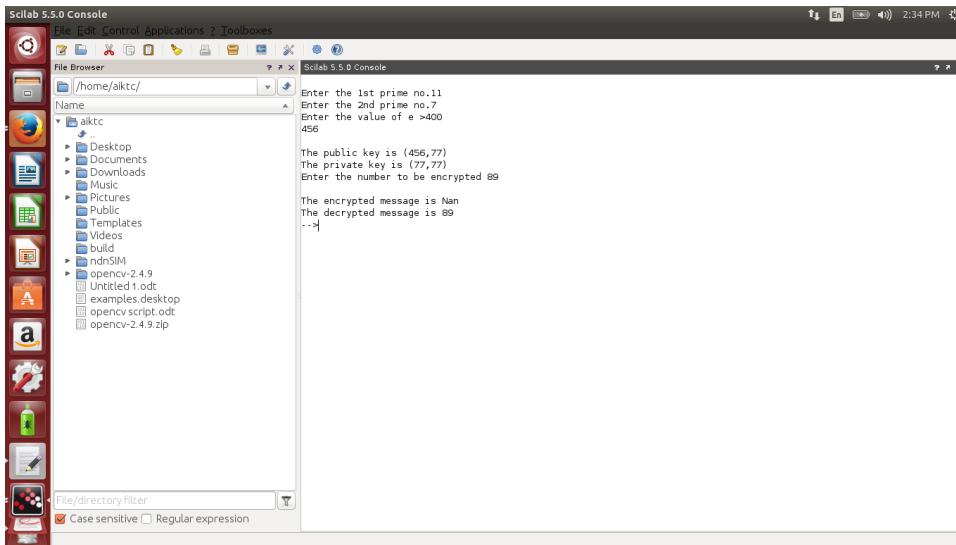


Figure 3.1: RSA Algorithim

```

14 for i = 1:n
15     z=modulo((i*e),phi);
16     if z == 1 then
17         break;
18 end
19 end
20 printf("\nThe public key is (%.d",e); //Output: The
    public key is (e,n)
21 //Output for ex. public key (7,55)
22 printf("%d",n);
23 printf("\nThe private key is (%.d",i); //Output:
    The private key is (i,n)
24     //Output ex. private key (23,55)
25 printf("%d",n);
26 m=input(" Enter the number to be encrypted "); //
    Input: Taking the message to be encrypted
27 //Input ex. 5
28 a=m^e;
29 c=modulo(a,n);
30 printf("\nThe encrypted message is %.d ",c); //Output

```

```
        : Printing the encrypted message
31 //Output for ex. 25
32 b=c^i;
33 t=modulo(b,n);
34 printf("\nThe decrypted message is %.d ",m); //Output
        : Decrypted Message
35 //Output for ex. 5
```

---

# Experiment: 4

## Diffie Hellman Key Exchange Method

**Scilab code Solution 4.1** Diffie Hellman Key Exchange

```
1 //OS: Windows 7
2 //Scilab Version: Scilab 5.4.1
3 clc;
4 clear all;
5 p=[13];//input(" Enter the common prime number(p) ")
    ;//Input: taking common prime number as input
6 //Input ex. p=13
7 g=[6];//input(" Enter the primitive root(g) (any no.)
    "); //Input: taking primitive root as input
8 //Input ex. g=6
9 a=[3];//input(" Enter secret key of first user (any
    no.) ");//Input: Taking secret key for user 1
10 //Input ex. a=3
11 b=[10];//input(" Enter secret key of second user (any
    no.) ");//Input: Taking secret key for user 2
12 //Input ex. b=10
13 A=modulo(g^a,p);//public key of user 1
14 B=modulo(g^b,p);//public key of user 2
15 common_key=modulo(A^b,p);//common key
```

```
16 printf("Common Key is %d", common_key) ; //Output:  
        Produced common key  
17 //Output for ex. Common Key=12
```

---

## Experiment: 5

# Image Compression Using Block Truncation Coding

check Appendix [AP 1](#) for dependency:

lena.png

**Scilab code Solution 5.5** block truncation in image

```
1
2 //Caption: Program performs Block Truncation Coding(
3 // BTC) by choosing different
4 //block sizes of 8
5 //Software version
6 //OS Windows7
7 //Scilab5.4.1
8 //Image Processing Design Toolbox 8.3.1-1
9 //Scilab Image and Video Proccesing toolbox
10 // 0.5.3.1-2
11
12 close;
13 clc;
```

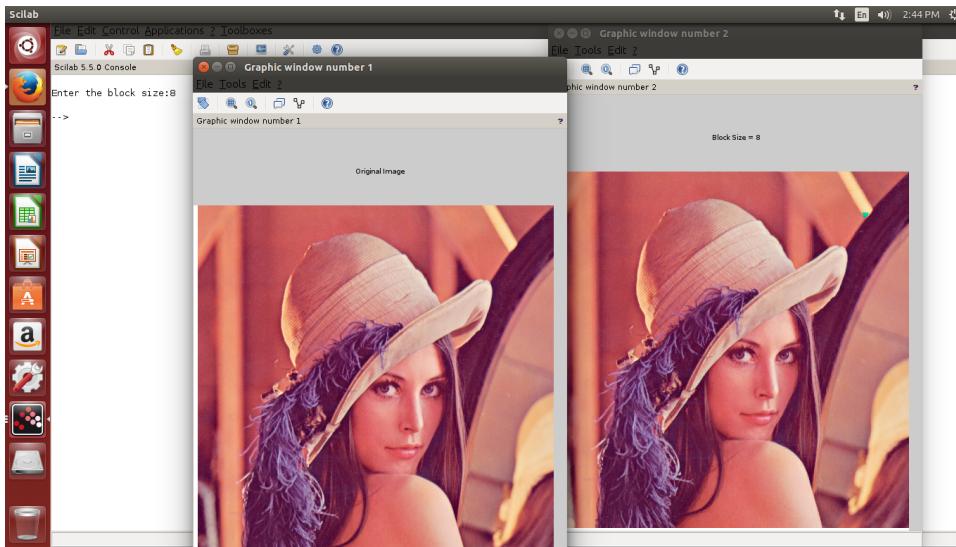


Figure 5.1: block truncation in image

```

12 x =imread( '/home/aiktc/Desktop/lena.png' ); //SIVP
    toolbox
13 //x=imresize(x,[256 256]);
14 x1=x;
15 x=double(x);
16 [m1 n1]=size(x);
17 blk=input('Enter the block size: '); //enter a block
    size of 8 only
18 for i = 1 : blk : m1
19 for j = 1 : blk : n1
20     y = x(i:i+(blk-1),j:j+(blk-1)) ;
21     m = mean(mean(y));
22     sig=std2(y);
23     b = y > m ; //the binary block
24     K = sum(sum(b));
25     if (K ~= blk^2) & ( K ~= 0)
26         ml = m-sig*sqrt(K/((blk^2)-K));
27         mu = m+sig*sqrt(((blk^2)-K)/K);
28         x(i:i+(blk-1), j:j+(blk-1)) = b*mu
            +(1- b)*ml;

```

```
29         end
30 end
31 end
32 //imshow( uint8(x) )
33 //title( 'Reconstructed Image' )
34 x = uint8(x);
35 figure(1)
36 imshow(x1)
37 title('Original Image');    //IPD toolbox
38 figure(2)
39 imshow(x);    //IPD toolbox
40 title('Block Size = 8')
```

---

# Appendix



LENA IMAGE