International Journal of Novel Research in Computer Science and Software Engineering
Vol. 1, Issue 1, pp: (30-36), Month: September-December 2014, Available at: www.noveltyjournals.com

Kareem Naaz (KN) Encryption and Decryption using Sorting

Shaik Kareem Basha

Asst. Professor CSE

Abstract: **In Computer networks, Messages are transmitted from sender to receiver with high security. Encryption Algorithm is used by sender to convert plain text into cipher text before sending text to receiver. At the receiver, Decryption Algorithm is used to convert cipher text into plain text. Sorting is a process of arranging data in a specific order according to requirements of Application. In the proposed paper I am using sorting algorithm to encrypt plain text into cipher text at the sender and at the receiver also sorting algorithm is used to convert cipher text into plain text. At the sender plain text is sorted in alphabetical order and the indices of characters of plain text are also sorted in that order and placed into index array. To the receiver cipher text and index array are sent. At the receiver index array, which contains indices of sorted plain text, is sorted in increasing order and characters of cipher text are also sorted in the same order to form plain text. I followed the various stages of Software Development Life Cycle to demonstrate the proposed sorting algorithm. Section I will give introduction about proposed sorting algorithm. In section II, proposed Algorithm is Analyzed and Designed. In section III, proposed algorithm is Implemented using C programming Language. In section IV, I tested the implementation of proposed algorithm using different test cases. In section V, I concluded the proposed Algorithm.**

I. INTRODUCTION

In Computer networks, Messages are transmitted from sender to receiver with high security. Encryption is the process of converting plain text of n characters into cipher text of n characters. Plain text is the readable text where as cipher text is unreadable text, which is difficult to understand. Decryption is the process of converting cipher text of n characters into plain text of n characters. Encryption Algorithm is used by sender to convert plain text into cipher text before sending text to receiver. At the receiver, Decryption Algorithm is used to convert cipher text into plain text. Sorting is a process of arranging data in a specific order according to requirements of Application. In the proposed paper I am using sorting algorithm to encrypt plain text into cipher text at the sender and at the receiver also sorting algorithm is used to convert cipher text into plain text. At the sender side I am using plain text and index array, which stores the indices of characters of plain text. In KNEnrypt() algorithm, plain text is sorted in alphabetical order and the indices of characters of plain text with in index array are also sorted in that order to form cipher text. To the receiver cipher text and index array are sent. At the receiver KNDecrypt() algorithm uses index array, which contains indices of sorted plain text characters, is sorted in increasing order and characters of cipher text are also sorted in the same order to form plain text. In the proposed paper, I am using sorting technique at the sender to encrypt plain text into cipher text and at the receiver, sorting technique is used to decrypt cipher text into plain text.

II. ANALYSIS AND DESIGN OF PROPOSED ALGORITHM

The proposed KareemNaaz (KN) Encryption and Decryption using Sorting Algorithms converts plain text of n characters into cipher text of n characters after Encryption and converts cipher text of n characters into plain text of n characters after Decryption. The proposed Encryption Algorithm sorts all the characters in Alphabetical Order and indices of input plain text in that order to make it as cipher text. The proposed Decryption Algorithm sorts all the indices in Increasing Order and characters of cipher text in that order to make it as plain text. The proposed Encryption Algorithm sort’s characters of input plain text in Alphabetical Order, and the indices of characters are also sorted to form cipher text. The proposed Decryption Algorithm sorts indices of cipher text in Increasing Order and characters of cipher text are also sorted to form plain text. Fig 2.1 shows the tasks of Encryption and Decryption Algorithms using Sorting.

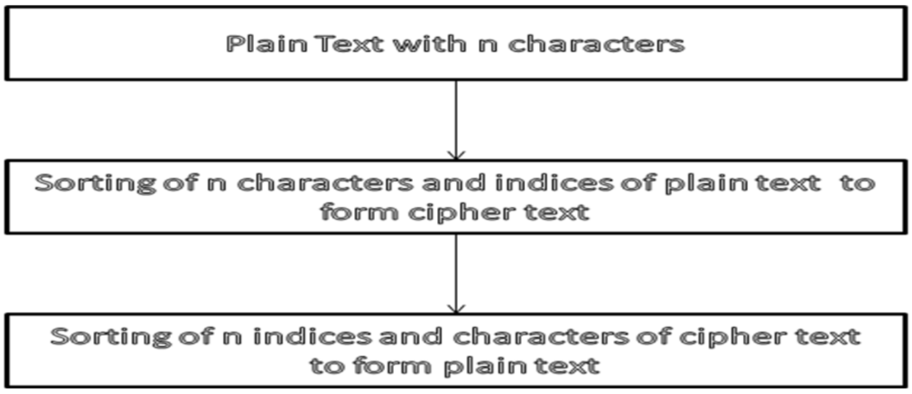

Plain Text with n characters
Sorting of n characters and indices of plain text to form cipher text
Sorting of n indices and characters of cipher text to form plain text

Fig 2.1 Encryption and Decryption using Sorting

Fig 2.2 shows the characters and indices of input Plain Text, which consists of 26 characters starting from index 0 to index 25.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
I	N	D	I	A	I	S	M	Y	C	O	U	N	T	R	Y	A	L	L	I	N	D	I	A	N	S

Fig 2.2 Plain Text with 26 characters

Fig 2.3 shows the encrypted cipher text, which consists of 26 characters of plain text sorted in Alphabetical order and indices of plain text characters also sorted in that order to form cipher text.

4	16	23	9	2	21	0	3	5	19	22	17	18	7	1	12	20	24	10	14	6	25	13	11	8	15
A	A	A	C	D	D	I	I	I	I	I	L	L	M	N	N	N	N	O	R	S	S	T	U	Y	Y

Fig 2.3 Cipher Text with 26 Sorted Characters and Indices (Encryption)

Fig 2.4 shows the decrypted plain text, which consists of indices of characters of cipher text from 0 to 25, which are sorted in increasing order and characters of cipher text are also sorted in that order to form plain text .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
I	N	D	I	A	I	S	M	Y	C	O	U	N	T	R	Y	A	L	L	I	N	D	I	A	N	S

Fig 2.4 Plain Text with 26 Sorted Indices and Characters (Decryption)

Algorithm 2.1 swaps the two given elements a and b. This algorithm is repeatedly called by KNEncrypt() and KNDecrypt() Algorithms to swap characters of plain text, characters of cipher text, indices of plain text and indices of cipher text.

```
Algorithm Swap(a,b)  { // a and b are elements which are to be swapped
 temp=a;
a=b;
b=temp;
}
```

Algorithm 2.1 swaps two indices or characters of text:

The time taken by swap algorithm is 3 units, since it consists of 3 assignment statements.

Algorithm 2.2 is used to convert plain text of n characters into cipher text of n characters. This algorithm takes plain text of n characters, index array which contains indices of plain text characters as input. It uses the concept of Insertion sort to sort all the plain text characters and indices of characters of plain text. Outer for loop repeats from first character to last character of plain text, inner for loop repeats from current character of text to first character of text in reverse order. If the current character is smaller than the previous character then current character is swapped with previous character and index of current character is also swapped with the index of previous character. This process continues until all the characters of plain text are alphabetically sorted and indices of plain text characters are also sorted in that order to form cipher text.

The proposed KNEncrypt Algorithm is as follows:

```
Algorithm KNEncrypt(plain, n,index) { // plain is the input text with n characters
// index is an integer array which contains indices of characters of input text
for i=0  to  n-1  step 1  do {  // loop repeats from first character of text to last character of text
for j=i  to 1 step -1  do { // loop repeats from current character of text to first character of text
if (plain[j] < plain[j-1]) then  { // if current character of text is smaller than previous character
swap(plain[j],plain[j-1]); // swap current and previous characters of input text
swap(index[j],index[j-1]); // swap indices of current and previous characters of input text
}
else break;  // if current character of text is greater or equal to previous character
}  }
cipher=plain; // make the modified input plain text as cipher text
return cipher; // return cipher text
}
```

Algorithm 2.2 Encryption of plain text using Sorting:

In Algorithm 2.2, Outer for loop repeats n times and for I iteration of outer for loop, inner for loop repeats 1 time, for II iteration of outer for loop, inner for loop repeats 2 times. In the same way for n^{th} iteration of outer for loop, inner for loop repeats n times. So the overall time taken by KNEncrypt() algorithm is as follows:

$T(n)=1+2+3+4+$---------------$+n$

$T(n)= n(n+1)/2 = (n^2 + n)/2 = n^2$

$T(n)=O(n^2)$

The time taken by KNEncrypt() Algorithm can also be reduced by using efficient sorting algorithm to sort characters of plain text in Alphabetical order, indices of characters in that order. The proposed algorithm uses Insertion Sort technique to sort characters and indices of plain text.

Algorithm 2.3 is used to convert cipher text of n characters into plain text of n characters. This algorithm takes cipher text of n characters, index array which contains indices of cipher text characters as input. It uses the concept of Insertion sort to sort all the cipher text characters and indices of characters of cipher text. Outer for loop repeats from first character to last character of cipher text, inner for loop repeats from current character of text to first character of text in reverse order. If the index of current character is smaller than the index of previous character then index of current character is swapped with index of previous character and current character is also swapped with previous character. This process continues until all the indices of cipher text are sorted in increasing order and cipher text characters are also sorted in that order to form plain text.

The proposed KNDecrypt Algorithm is as follows:

```
Algorithm KNDecrypt(cipher, n,index) { // cipher is the cipher text with n characters obtained //after Encryption.
// index is an integer array, which contains indices of characters of cipher text.
for i=0  to  n-1  step 1  do { // loop repeats from first character to last character of cipher text.
for j=I  to 1 step -1  do { // loop repeats from current character to first character of cipher text.
if (index[j] < index[j-1]) then  {if index of current character is smaller than index of previous //character of cipher text
swap(cipher[j],cipher[j-1]); // swap current and previous characters of cipher text
swap(index[j],index[j-1]);// swap indices of current and previous characters of cipher text
}
else break; // if index of current character is greater or equal to index of previous character of //cipher text.
}  }
plain=cipher; // make modified cipher text as plain text
return plain; // return plain text
}
```

Algorithm 2.3 Decryption of cipher text using Sorting:

In Algorithm 2.3, Outer for loop repeats n times and for I iteration of outer for loop, inner for loop repeats 1 time, for II iteration of outer for loop, inner for loop repeats 2 times. In the same way for nth iteration of outer for loop, inner for loop repeats n times. So the overall time taken by KNDecrypt() algorithm is as follows:

$T(n)=1+2+3+4+\text{---------------}+n$

$T(n)= n(n+1)/2 = (n^2 + n)/2 = n^2$

$T(n)=O(n^2)$

The time taken by KNDecrypt() Algorithm can also be reduced by using efficient sorting algorithm to sort indices of cipher text in increasing order, cipher characters in that order. The proposed algorithm uses Insertion Sort technique to sort indices and characters of cipher text.

III. IMPLEMENTATION OF PROPOSED ALGORITHM USING C

The Proposed Algorithm is implemented by using C Programming Language, which is a robust, structure oriented programming language, which provides different concepts like functions, pointers, structures, arrays and so on to solve complex problems.

```
/* Program to implement Encryption and Decryption of text using Sorting*/
# include <stdio.h>
# include <conio.h>
char input[50]; // input array to store plain text with n characters
int index[50]; // index array to store indices of characters of plain text
int n; // number of characters of plain text
void CharSwap(char *a,char *b)    { // function to swap two characters of text
 char temp=*a;
```

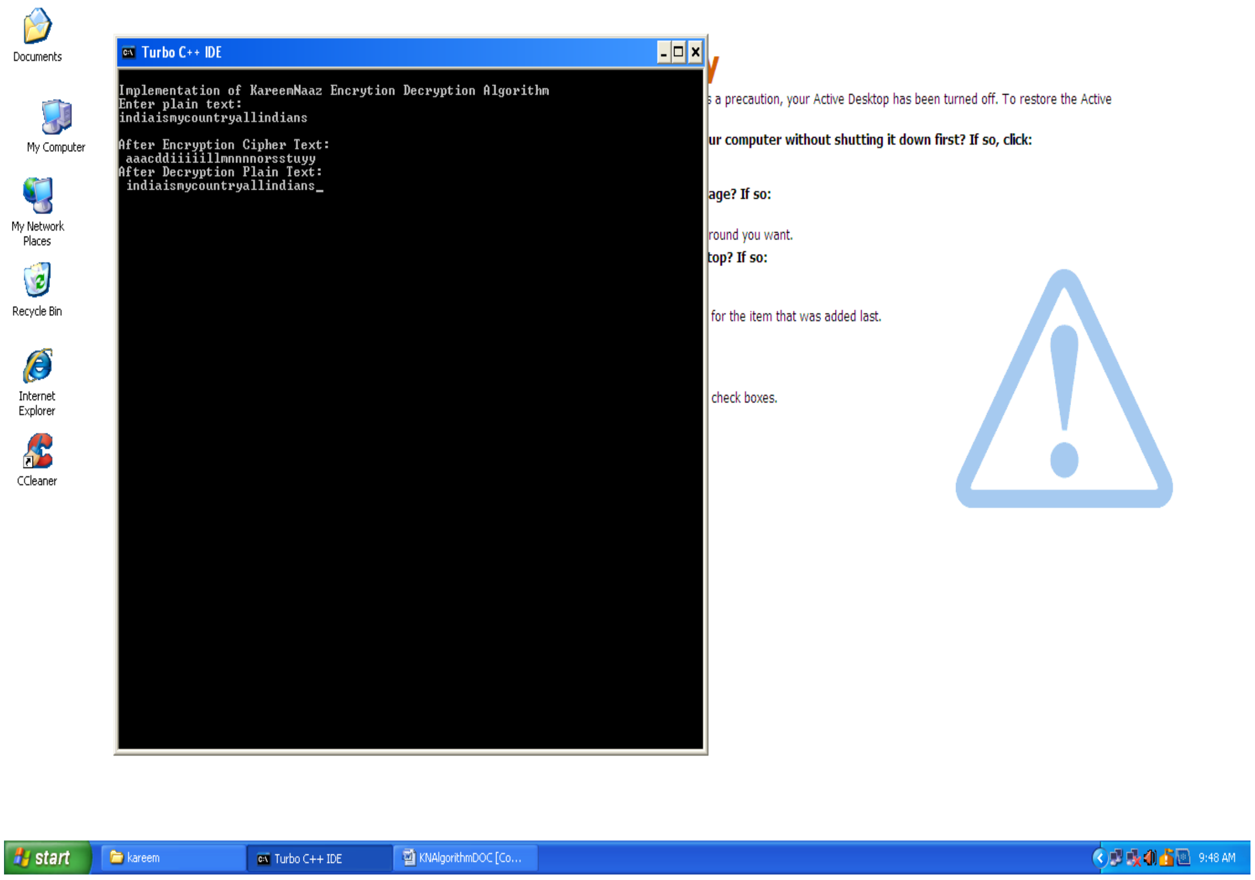
```
*a=*b;
*b=temp;
}
void IntSwap(int *a,int *b)   { // function to swap two indices of characters of text
 int temp=*a;
 *a=*b;
 *b=temp;
}
void KNEncrypt()  { // Encryption of plain text by sorting characters and indices of plain text
 int i,j;
 for(i=0;i<n;i++) // outer loop repeats from first character to last character of plain text
 for(j=i;j>0;j--) // inner loop repeats from current character to first character of plain text
 if(input[j]<input[j-1])  { // if current character is smaller than previous character of plain text
  CharSwap(&input[j],&input[j-1]); // swap current and previous characters of plain text
  IntSwap(&index[j],&index[j-1]); // swap indices of current and previous characters of text
 }
 else // if current character is greater or equal to previous character of text
 break; // stop comparing characters of plain text
}
void KNDecrypt(){ //Decryption of cipher text by sorting indices and characters of cipher text
 int i,j;
 for(i=0;i<n;i++)// loop repeats from first character to last character of cipher text
 for(j=i;j>0;j--) // loop repeats from current character to previous character of cipher text
 if(index[j]<index[j-1])  { // if index of current character is smaller than index of previous //character of cipher text
  IntSwap(&index[j],&index[j-1]); // swap indices of current and previous characters of cipher
  CharSwap(&input[j],&input[j-1]);//swap current and previous characters of cipher text
 }
 else // if index of current character is greater or equal to index of previous character of cipher
 break; // stop comparing characters of cipher text
}
void main() {
  int i;
  clrscr();
  printf("\nImplementation of KareemNaaz Encrytion Decryption Algorithm");
  printf("\nEnter plain text:\n");
```

```
scanf("%s",input); // read plain text from key board
n=strlen(input); // n is the length of plain text
for(i=0;i<n;i++) // placing indices of plain text into index array
index[i]=i;
KNEncrypt(); // Encrypt plain text of n characters to cipher text
printf("\nAfter Encryption Cipher Text:\n %s",input);
KNDecrypt(); //Decrypt cipher text of n characters to plain text
printf("\nAfter Decryption Plain Text:\n %s",input);
getch();
}
```

Program 3.1 To Encrypt and Decrypt text by using Sorting:

IV. TESTING OF PROPOSED ALGORITHM

Different Test Cases are considered to test the result of Program 3.1. The following screen shot, shows the result of proposed algorithm. Consider the following Test Case in which plain text consists of 26 characters "indiaismycountryallindians" starting from index 0 to index 25. After Encryption, cipher text is "aaacddiiiiillmnnnnorsstuyy". After Decryption, plain text is "indiaismycountryallindians". Fig 4.1 shows the encrypted cipher text and decrypted plain text.



```
Implementation of KareemNaaz Encrytion Decryption Algorithm
Enter plain text:
indiaismycountryallindians

After Encryption Cipher Text:
 aaacddiiiiillmnnnnorsstuyy
After Decryption Plain Text:
 indiaismycountryallindians_
```

Fig 4.1 output screen shot of program 3.1

V. CONCLUSION

I conclude that In the proposed KareemNaaz (KN) Encryption and Decryption using Sorting paper, KNEncrypt() algorithm takes plain text of n characters as input, sort all the characters of plain text in alphabetical order and also sorts the indices of plain text characters in that order to form cipher text of n characters. Proposed KNDecrypt() algorithm will take cipher text of n characters as input, sort all the indices of cipher text in increasing order and also sorts characters of cipher text in that order to form plain text of n characters. In both KNEncrypt() and KNDecrypt() algorithm, I am using Insertion Sort technique to sort characters of plain text in alphabetical order, indices of cipher text in increasing order, characters of cipher text and indices of plain text. To reduce complexity some other sorting algorithms can also be preferred.

REFERENCES

[1] Aho, Alfred V. and Jeffrey D. Ullman [1983]. Data Structures and Algorithms. AddisonWesley,Reading, Massachusetts.

[2] Cormen, Thomas H., Charles E. Leiserson and Ronald L. Rivest [1990]. Introduction to Algorithms. McGraw-Hill, New York.

[3] Knuth, Donald. E. [1998]. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, Reading, Massachusetts.