International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 7, July 2013 ISSN 2319 - 4847

# Improved Cryptographic Technique for Data Security

**Ankit Agarwal[1], Sahil Nyati[2] and Dr.Kapil Kumar Bansal[3]**

[1]Ankit Agarwal B-tech student, Department of Computer Science and Engineering, SRM University

[2]Sahil Nyati B-tech student, Department of Computer Science and Engineering, SRM University

[3] Dr.Kapil Kumar Bansal Head Research and Publication, SRM University

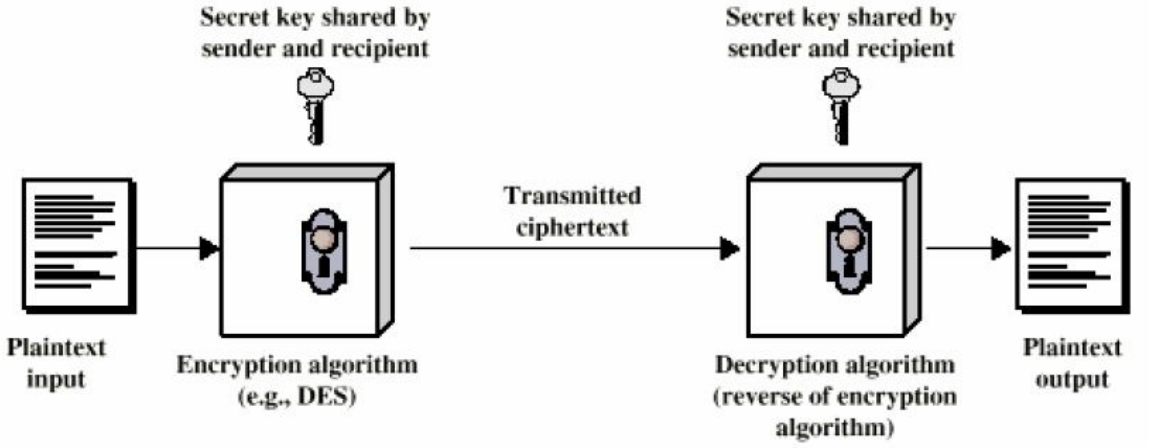
## ABSTRACT

***Cryptography is the process of securing the information. It protects availability, privacy and integrity of the message. Access to stored information on computer databases has increased greatly. More companies communicate with computer than ever before. Most of the information is highly confidential and not for public viewing. In this paper I have developed a new cryptography algorithm which is based on cipher concept. In this algorithm I have used logical operation like substitution and shifting operation. Experimental results show that proposed algorithm is much secured. To write this paper I have Study about various cryptography techniques. After the detailed study of cryptography, I am presenting my proposed work. A typical example of message includes your social-security number, or your account number, or private information, or confidential information coming from some trusted infrastructure. In many situations, however, the third party (intruder) always tries to spy on your private messages gaining an upper hand over a corporation or an industry. In this paper, we develop a new algorithm based on cryptographic techniques- substitution and position shift to enhance the capability of securing the messages.***

**Keywords:** Cryptography**,** Database, Algorithm.

## 1. INTRODUCTION

Protecting information by encrypting it into an unreadable format, called ciphertext. Only those who possess a secret key can decipher the message into plain text. Encrypted messages can sometimes be broken by cryptanalysis, also called code breaking, although modern cryptography techniques are virtually unbreakable. As the Internet and other forms of electronic communication become more prevalent, electronic security is becoming increasingly important. Cryptography is used to protect e-mail messages, credit card information, and corporate data. One of the most popular cryptography systems used on the Internet is Pretty Good Privacy because it's effective and free.




The main feature of the encryption/decryption program implementation is the generation of the encryption key. If we are protecting confidential information then cryptography is provide high level of privacy of individuals and groups. However,

the main purpose of the cryptography is used not only to provide confidentiality, but also to provide solutions for other problems like: data integrity, authentication, non-repudiation. Cryptography is the methods that allow information to be sent in a secure from in such a way that the only receiver able to retrieve the information.

**Substitution Cipher:**

A substitution cipher is a method of encoding by which units of plaintext are replaced with ciphertext, according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution. Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered. There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraph. A mono-alphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.

**Position Cipher:**

In position-based cryptography the quantum setting, the shifting of the position in the message is performed. The aim is to use the geographical position of a party as its only credential. On the negative side, we show that if adversaries are allowed to share an arbitrarily large entangled quantum state, no secure position-verification is possible at all. We show a distributed protocol for computing any unitary operation on a state shared between the different users, using local operations and one round of classical communication. Using this surprising result, we break any position-verification scheme of a very general form. On the positive side, we show that if adversaries do not share any entangled quantum state but can compute arbitrary quantum operations, secure position-verification is achievable. Jointly, these results suggest the interesting question whether secure position-verification is possible in case of a bounded amount of entanglement. Our positive result can be interpreted as resolving this question in the simplest case, where the bound is set to zero. In models where secure positioning is achievable, it has a number of interesting applications. For example, it enables secure communication over an insecure channel without having any pre-shared key, with the guarantee that only a party at a specific location can learn the content of the conversation. More generally, we show that in settings where secure position-verification is achievable, other position-based cryptographic schemes are possible as well, such as secure position-based authentication and position-based key agreement.

## 2. IDEA OF NAASS ALGORITHM:

The blank column at e6 signifies SPACE.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| **a** | a | 1 | y | @ | b | . | ( | 2 | c |
| **b** | d | ! | w | 9 | e | # | x | * | f |
| **c** | g | 7 | = | 8 | h | ) | z | 6 | i |
| **d** | j | ; | t | 0 | k | $ | 5 | , | l |
| **e** | m | % | v | / | n | | u | 3 | o |
| **f** | p | 4 | ? | - | q | + | ^ | s | r |

Idea is based on the box drawn above; it is to replace the message into an alphanumeric code and then applying position shift so that the numeric and the letters can be placed to one side. Here while replacing a definite letter with another one can also use a set of words with same length (like tap, mat, cat, dog) instead of a,b,c, d, e, f; in order to baffle the intruder to read the message.

**For example:**

Message: - attack the enemy

In the message, a is converted to a1, t to d3, c to a9, k to d5, (space) is defined by e6 and similarly as shown in the box above. Then using quantum position cipher the letters and digits are stored together.

Encoded Message:addaadedcbebebea1331956355655513

## 3. JAVA PROGRAM FOR ENCODING AND DECODING THE MESSAGE:

**Program to ENCODE the message:**

```
import java.io.*;
public class encode_message
{
public static void main(String S[])throws IOException
     {
char x;
BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the text");
          String inp=br.readLine();
          int inp_len=inp.length();
          int ct=0;
           char arr[]=new char[(inp_len*2)];
          for(int i=0;i<inp_len;i++)
{
              x=inp.charAt(i);
              if(x=='a'||x=='A')
              {
arr[ct]='a';
arr[(inp_len+ct)]='1';
ct++;
              }
              if(x=='b'||x=='B')
              {
arr[ct]='a';
arr[(inp_len+ct)]='5';
ct++;
              }
              if(x=='c'||x=='C')
              {
arr[ct]='a';
arr[(inp_len+ct)]='9';
ct++;
              }
              if(x=='d'||x=='D')
              {
arr[ct]='b';
arr[(inp_len+ct)]='1';
ct++;
              }
              if(x=='e'||x=='E')
              {
arr[ct]='b';
arr[(inp_len+ct)]='5';
ct++;
              }
              if(x=='f'||x=='F')
              {
arr[ct]='b';
arr[(inp_len+ct)]='9';
ct++;
              }
               if(x=='g'||x=='G')
              {
arr[ct]='c';
arr[(inp_len+ct)]='1';
ct++;
              }
              if(x=='h'||x=='H')
              {
```

```
arr[ct]='c';
arr[(inp_len+ct)]='5';
ct++;
            }
            if(x=='i'||x=='I')
            {
arr[ct]='c';
arr[(inp_len+ct)]='9';
ct++;
            }
            if(x=='j'||x=='J')
            {
arr[ct]='d';
arr[(inp_len+ct)]='1';
ct++;
            }
            if(x=='k'||x=='K')
            {
arr[ct]='d';
arr[(inp_len+ct)]='5';
ct++;
            }
            if(x=='l'||x=='L')
            {
arr[ct]='d';
arr[(inp_len+ct)]='9';
ct++;
            }
            if(x=='m'||x=='M')
            {
arr[ct]='e';
arr[(inp_len+ct)]='1';
ct++;
            }
            if(x=='n'||x=='N')
            {
arr[ct]='e';
arr[(inp_len+ct)]='5';
ct++;
            }
            if(x=='o'||x=='O')
            {
arr[ct]='e';
arr[(inp_len+ct)]='9';
ct++;
            }
            if(x=='p'||x=='P')
            {
arr[ct]='f';
arr[(inp_len+ct)]='1';
ct++;
            }
            if(x=='q'||x=='Q')
            {
arr[ct]='f';
arr[(inp_len+ct)]='5';
ct++;
            }
            if(x=='r'||x=='R')
```

```
            {
arr[ct]='f';
arr[(inp_len+ct)]='9';
ct++;
            }
             if(x=='s'||x=='S')
            {
arr[ct]='f';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='t'||x=='T')
            {
arr[ct]='d';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='u'||x=='U')
            {
arr[ct]='e';
arr[(inp_len+ct)]='7';
ct++;
            }
               if(x=='v'||x=='V')
            {
arr[ct]='e';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='w'||x=='W')
            {
arr[ct]='b';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='x'||x=='X')
            {
arr[ct]='b';
arr[(inp_len+ct)]='7';
ct++;
            }
            if(x=='y'||x=='Y')
            {
arr[ct]='a';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='z'||x=='Z')
            {
arr[ct]='c';
arr[(inp_len+ct)]='7';
ct++;
            }
            if(x=='1')
            {
arr[ct]='a';
arr[(inp_len+ct)]='2';
ct++;
            }
```

```
            if(x=='2')
            {
arr[ct]='a';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='3')
            {
arr[ct]='e';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='4')
            {
arr[ct]='f';
arr[(inp_len+ct)]='2';
ct++;
            }
            if(x=='5')
            {
arr[ct]='d';
arr[(inp_len+ct)]='7';
ct++;
            }
            if(x=='6')
            {
arr[ct]='c';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='7')
            {
arr[ct]='c';
arr[(inp_len+ct)]='2';
ct++;
            }
            if(x=='8')
            {
arr[ct]='c';
arr[(inp_len+ct)]='4';
ct++;
            }
            if(x=='9')
            {
arr[ct]='b';
arr[(inp_len+ct)]='4';
ct++;
            }
            if(x=='0')
            {
arr[ct]='d';
arr[(inp_len+ct)]='4';
ct++;
            }
            if(x=='@')
            {
arr[ct]='a';
arr[(inp_len+ct)]='4';
ct++;
```

```
            }
            if(x=='.')
            {
arr[ct]='a';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x==' ')
            {
arr[ct]='e';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x=='*')
            {
arr[ct]='b';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='^')
            {
arr[ct]='f';
arr[(inp_len+ct)]='7';
ct++;
            }
            if(x=='(')
            {
arr[ct]='a';
arr[(inp_len+ct)]='7';
ct++;
            }
            if(x==')')
            {
arr[ct]='c';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x=='=')
            {
arr[ct]='c';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='#')
            {
arr[ct]='b';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x=='!')
            {
arr[ct]='b';
arr[(inp_len+ct)]='2';
ct++;
            }
            if(x==';')
            {
arr[ct]='d';
arr[(inp_len+ct)]='2';
```

```
ct++;
            }
            if(x==',')
            {
arr[ct]='d';
arr[(inp_len+ct)]='8';
ct++;
            }
            if(x=='$')
            {
arr[ct]='d';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x=='+')
            {
arr[ct]='f';
arr[(inp_len+ct)]='6';
ct++;
            }
            if(x=='-')
            {
arr[ct]='f';
arr[(inp_len+ct)]='4';
ct++;
            }
            if(x=='?')
            {
arr[ct]='f';
arr[(inp_len+ct)]='3';
ct++;
            }
            if(x=='/')
            {
arr[ct]='e';
arr[(inp_len+ct)]='4';
ct++;
            }
            if(x=='%')
           {
arr[ct]='e';
arr[(inp_len+ct)]='2';
ct++;
           }
      }
System.out.println("Encoded Message");
for(int i=0;i<(inp_len*2);i++)
System.out.print(arr[i]);
System.out.println();
}
}
```

Program to DECODE the message:

```
import java.io.*;
public class decode_crypt
{
      public static void main(String S[])throws IOException
      {
```

```
        char x;
        char y;
BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the text");
        String inp=br.readLine();
        int inp_len=inp.length();
        int ct=0;
        int n_ct=inp_len/2;
        char arr[]=new char[(n_ct)];
        for(int i=0;i<n_ct;i++)
        {
           x=inp.charAt(i);
           y=inp.charAt((n_ct+i));
           if(x=='a' && y=='1')
           {
arr[ct]='a';
ct++;
           }
           if(x=='a' && y=='2')
           {
arr[ct]='1';
ct++;
           }
           if(x=='a' && y=='3')
           {
arr[ct]='y';
ct++;
           }
           if(x=='a' && y=='4')
           {
arr[ct]='@';
ct++;
           }
           if(x=='a' && y=='5')
           {
arr[ct]='b';
ct++;
           }
           if(x=='a' && y=='6')
           {
arr[ct]='.';
ct++;
           }
           if(x=='a' && y=='7')
           {
arr[ct]='(';
ct++;
           }
           if(x=='a' && y=='8')
           {
arr[ct]='z';
ct++;
           }
           if(x=='a' && y=='9')
           {
arr[ct]='c';
ct++;
           }
           if(x=='b' && y=='1')
```

```
            {
arr[ct]='d';
ct++;
            }
            if(x=='b' && y=='2')
            {
arr[ct]='!';
ct++;
            }
            if(x=='b' && y=='3')
            {
arr[ct]='w';
ct++;
            }
            if(x=='b' && y=='4')
            {
arr[ct]='9';
ct++;
            }
            if(x=='b' && y=='5')
            {
arr[ct]='e';
ct++;
            }
            if(x=='b' && y=='6')
            {
arr[ct]='#';
ct++;
            }
            if(x=='b' && y=='7')
            {
arr[ct]='x';
ct++;
            }
            if(x=='b' && y=='8')
            {
arr[ct]='*';
ct++;
            }
            if(x=='b' && y=='9')
            {
arr[ct]='f';
ct++;
            }
            if(x=='c' && y=='1')
            {
arr[ct]='g';
ct++;
            }
            if(x=='c' && y=='2')
            {
arr[ct]='7';
ct++;
            }
            if(x=='c' && y=='3')
            {
arr[ct]='=';
ct++;
            }
```

```
            if(x=='c' && y=='4')
            {
arr[ct]='8';
ct++;
            }
            if(x=='c' && y=='5')
            {
arr[ct]='h';
ct++;
            }
            if(x=='c' && y=='6')
            {
arr[ct]=')';
ct++;
            }
            if(x=='c' && y=='7')
            {
arr[ct]='z';
ct++;
            }
            if(x=='c' && y=='8')
            {
arr[ct]='6';
ct++;
            }
            if(x=='c' && y=='9')
            {
arr[ct]='i';
ct++;
            }
            if(x=='d' && y=='1')
            {
arr[ct]='j';
ct++;
            }
            if(x=='d' && y=='2')
            {
arr[ct]=';';
ct++;
            }
            if(x=='d' && y=='3')
            {
arr[ct]='t';
ct++;
            }
            if(x=='d' && y=='4')
            {
arr[ct]='0';
ct++;
            }
            if(x=='d' && y=='5')
            {
arr[ct]='k';
ct++;
            }
            if(x=='d' && y=='6')
            {
arr[ct]='$';
ct++;
```

```
            }
            if(x=='d' && y=='7')
            {
arr[ct]='5';
ct++;
            }
            if(x=='d' && y=='8')
            {
arr[ct]=',';
ct++;
            }
            if(x=='d' && y=='9')
            {
arr[ct]='l';
ct++;
            }
            if(x=='e' && y=='1')
            {
arr[ct]='m';
ct++;
            }
            if(x=='e' && y=='2')
            {
arr[ct]='%';
ct++;
            }
            if(x=='e' && y=='3')
            {
arr[ct]='v';
ct++;
            }
            if(x=='e' && y=='4')
            {
arr[ct]='/';
ct++;
            }
            if(x=='e' && y=='5')
            {
arr[ct]='n';
ct++;
            }
            if(x=='e' && y=='6')
            {
arr[ct]=' ';
ct++;
            }
            if(x=='e' && y=='7')
            {
arr[ct]='u';
ct++;
            }
            if(x=='e' && y=='8')
            {
arr[ct]='3';
ct++;
            }
            if(x=='e' && y=='9')
            {
arr[ct]='o';
```

```
ct++;
            }
            if(x=='f' && y=='1')
            {
arr[ct]='p';
ct++;
            }
            if(x=='f' && y=='2')
            {
arr[ct]='4';
ct++;
            }
            if(x=='f' && y=='3')
            {
arr[ct]='?';
ct++;
            }
            if(x=='f' && y=='4')
            {
arr[ct]='-';
ct++;
            }
if(x=='f' && y=='5')
            {
arr[ct]='q';
ct++;
            }
            if(x=='f' && y=='6')
            {
arr[ct]='+';
ct++;
            }
if(x=='f' && y=='7')
              {
arr[ct]='^';
ct++;
               }
            if(x=='f' && y=='8')
              {
                arr[ct]='s';
      ct++;
            }
            if(x=='f' && y=='9')
           {
     arr[ct]='r';
     ct++;
            }
     }
     System.out.println("Decoded Message");
     for(int i=0;i<(inp_len/2);i++)
          System.out.print(arr[i]);
  System.out.println();
   }
}
```

**4. CONCLUSION**

This type of technique is never implemented in cryptography to encode a message; the technique is simple, efficient as well as ensures privacy of the messages. It is also interesting in its own right, intersecting and connecting the two methods substitution cipher and position cipher. From a purely cryptographic point of view, the challenge is to find a set

of parameters that leads not only to hard problems but also to reasonably efficient implementations. We hope that this paper will revive the interest for the technique used above, and will be useful to those willing to study the hardness of the subjacent cryptographic problems.

**References**

[1] https://en.wikipedia.org/w/index.php?title=Substitution_cipher&action=edit&section=6

[2] http://en.wikipedia.org/wiki/Quantum_cryptography

[3] Cryptography and network security, principles and practice, 2nd edition - William Stallings

**AUTHOR**

**Ankit Agarwal** is currently a 4rd year undergraduate, in department of Computer Science & Engineering, from SRM University, Batch 2010-2014.