Security

1. Introduction

Data Security is a process of protecting files, databases, and accounts on a network by adopting a set of controls, applications, and techniques that identify the relative importance of different datasets, their sensitivity, regulatory compliance requirements and then applying appropriate protections to secure those resources.

Similar to other approaches like perimeter security, file security or user behavioral security, data security is not the be all, end all for a security practice. It's one method of evaluating and reducing the risk that comes with storing any kind of data.

2. Algorithm

Data security is a method which is used to cover the important information. Data security methods control the privacy and integrity of the important information. The access to the database of the companies has improved. Now companies store their business data more on computer than before. Most of the company data is for internal use and not for the general public because business data is highly confidential. At present, cryptographic block cipher is being used with some logical operation and the main drawback in this method is the generation of the secret key which is totally based on the alphabets. So with the help of loop concept, there is a chance for the hackers to find out the secret key. But I propose advanced algorithm for cryptography which is totally dependent on hashing function technique to generate a secret key which is further used to encrypt and decrypt the important information. The secret key will be generated by using different key generation algorithms which will be of higher sets of alphanumeric characters.

3. Encryption

Data Encryption is the process of converting the plaintext into Encoded form (non-readable) and only authorized person/parties can access it. Data security is an essential part of an Individual/organization; it can be achieved by the using various methods. The encrypted data is safe for some time but never think it is permanently safe. After the time goes on there is chance of hacking the data by the hacker. Fake files are transmitted in the same manner as one can sends the encrypted data. There are many algorithms available in the market for encrypting the data. Encryption Key has the major role in the overall process of data.

4. Decryption

The conversion of encrypted data into its original form is called Decryption. It is generally a reverse process of encryption. It decodes the encrypted information so that an authorized user can only decrypt the data because decryption requires a secret key or password.

One of the reasons for implementing an encryption-decryption system is privacy. As information travels over the Internet, it is necessary to scrutinise the access from unauthorized organizations or individuals. Due to this, the data is encrypted to reduce data loss and theft. Few common items that are encrypted include text files, images, e-mail messages, user data and directories. The recipient of decryption receives a prompt or window in which a password can be entered to access the encrypted data. For decryption, the system extracts and converts the garbled data and transforms it into words and images that are easily understandable not only by a reader but also by

a system. Decryption can be done manually or automatically. It may also be performed with a set of keys or passwords.

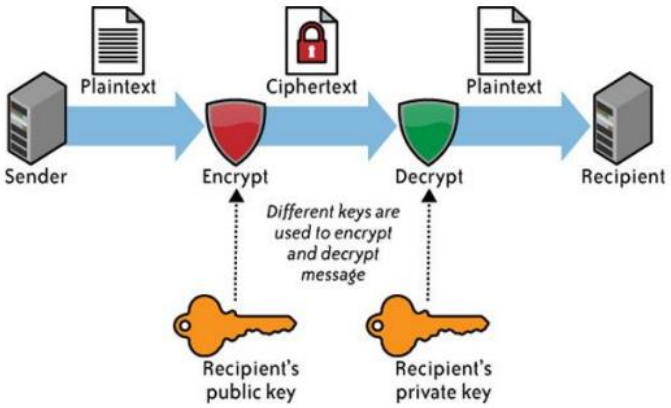


Fig1. Encryption and Decryption process

5. Types of Algorithm

1. **Number Theory.**
2. **GCD & GCD using Euclid's.**
3. **Hill cipher (Inverse of the matrix).**
4. **Transposition cipher.**
    a. **rail fence.**
    b. **more complex scheme.**
5. **substitution.**
    a. **caser cipher.**
    b. **multiplicative.**
    c. **shift + multiplicative.**
6. **vigenere cipher.**
7. **vernam cipher.**
8. **playfair cipher.**

9. DES.

5.1 Number Theory.

The main goal of number theory is to discover interesting and unexpected relationships between different sorts of numbers and to prove that these relationships are true. In this section we will describe a few typical number theoretic problems, some of which we will eventually solve, some of which have known solutions too difficult for us to include, and some of which remain unsolved to this day.

Sums of Squares. Can the sum of two squares be a square? The answer is clearly "YES";

for example $3^2 + 4^2 = 52$ and $5^2 + 12^2 = 132$. These are examples of Pythagorean triples

$$
\begin{array}{llll}
3 = \text{NO}, & 5 = 1^2 + 2^2, & 7 = \text{NO}, & 11 = \text{NO}, \\
13 = 2^2 + 3^2, & 17 = 1^2 + 4^2, & 19 = \text{NO}, & 23 = \text{NO}, \\
29 = 2^2 + 5^2, & 31 = \text{NO}, & 37 = 1^2 + 6^2, & \ldots
\end{array}
$$

Sums of Higher Powers. Can the sum of two cubes be a cube? Can the sum of two fourth powers be a fourth power? In general, can the sum of two n^{th} powers be an n^{th} power? The answer is "NO." This famous problem, called Fermat's Last Theorem, was first posed by Pierre de Fermat in the seventeenth century, but was not completely solved until 1994 by Andrew Wiles. Wiles's proof uses sophisticated mathematical techniques that we will not be able to describe in detail, but in Chapter

30 we will prove that no fourth power is a sum of two fourth powers, and in Chapter 46 we will sketch some of the ideas that go into Wiles's proof.

Number Shapes. The square numbers are the numbers 1, 4, 9, 16, . . . that can be arranged in the shape of a square. The triangular numbers are the numbers 1, 3, 6, 10, . . . that can be arranged in the shape of a triangle.

$$1+2+3+\cdots+(n-1)+n=\frac{n(n+1)}{2}.$$

5.2 GCD & GCD using Euclid's.

Euclid's algorithm, also known as the Euclidean algorithm, can be used to efficiently calculate the greatest common divisor (GCD) for two integer values. This article describes the algorithm and provides several C# methods that calculate the GCD

which is often called the Euclidean algorithm, is an algorithm described by the Greek mathematician, The algorithm seeks to calculate the greatest common divisor (GCD) of two arbitrarily large integers; the GCD being the largest whole number that can the two values can be divided by without remainder. The algorithm uses the fact that when two numbers share a common divisor, subtracting the smaller number from the larger yields a result that also shares the common divisor.

As an example, let's consider the values 210 and 126. The GCD of these two numbers is 42. Note that 210 / 42 = 5 and 126 / 4 = 3. If we were to subtract the smaller value from the larger we get 210 - 126 = 84. This too divides equally by 42, giving the result, If the process of subtracting the smaller number from the larger is repeated, eventually one of the values will become zero. At this point, the other value will

contain the GCD for the original pair. This iterative process for our example values yields the following results:

```
210 - 126 = 84
126 - 84 = 42
84 - 42 = 42
42 - 42 = 0
```

The final calculation gives a zero result, which would leave the two values 42 and zero. Therefore, the GCD is the value 42.

For the first implementation of the algorithm we will follow the steps described above directly. We will create a that has two integers and returns a third integer containing the GCD. The is static so that it can be called easily from the Main method of a console application without instantiating objects. You may decide to implement it as an instance method for your own project. The method has several steps. Initially a is created that will continue processing until one of the values has been reduced to zero. Within the loop the smaller of the two values is subtracted from the larger. Once the loop has exited, one value will be zero and the other will contain the GCD. The GCD is returned by finding the larger of the two values. NB: The code in this article assumes that both integers are positive.

To create the method, add the following code:

```
public static int GetGCDBySubtraction(int value1, int value2)
{
   while (value1 != 0 && value2 != 0)
   {
```

```
        if (value1 > value2)
            value1 -= value2;
        else
            value2 -= value1;
    }
    return Math.Max(value1, value2);
}
```

To test the method, try executing the following command. This finds the GCD of 116,150 and 232,704. The result should be 202.

```
int gcd = GetGCDBySubtraction(116150, 232704); // 202
```

5.3 Hill cipher (Inverse of the matrix)

Hill cipher is a polygraphic substitution cipher based on linear algebra.Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

Examples:

Input : Plaintext: ACT Key: GYBNQKURP Output : Ciphertext: POH

Input : Plaintext: GFG Key: HILLMAGIC Output : Ciphertext: SWK

5.3.1 Encryption

We have to encrypt the message 'ACT' (n=3).The key is 'GYBNQKURP' which can be written as the nxn matrix:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

The message 'ACT' is written as vector:

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

The enciphered vector is given as:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} (\text{mod } 26)$$

which corresponds to ciphertext of 'POH'

5.3.1 Decryption

To decrypt the message, we turn the ciphertext back into a vector, then simply multiply by the inverse matrix of the key matrix (IFKVIVVMI in letters).The inverse of the matrix used in the previous example is:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} - 1 \equiv \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} (\text{mod } 26)$$

For the previous Ciphertext 'POH':

$$\begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \equiv \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} (\text{mod } 26)$$

which gives us back 'ACT'.

Assume that all the alphabets are in upper case.

Below is the implementation of the above idea for n=3.

```
using System;


class GFG
{


// Following function generates the
// key matrix for the key string
static void getKeyMatrix(String key,
                         int [,]keyMatrix)
```

```
{
    int k = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            keyMatrix[i, j] = (key[k]) % 65;
            k++;
        }
    }
}

// Following function encrypts the message
static void encrypt(int [,]cipherMatrix,
                    int [,]keyMatrix,
                    int [,]messageVector)
{
    int x, i, j;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 1; j++)
        {
            cipherMatrix[i, j] = 0;

            for (x = 0; x < 3; x++)
```

```
            {
                cipherMatrix[i, j] += keyMatrix[i, x] *
                                    messageVector[x, j];
            }

            cipherMatrix[i, j] = cipherMatrix[i, j] % 26;
        }
    }
}

// Function to implement Hill Cipher
static void HillCipher(String message, String key)
{

    // Get key matrix from the key string
    int [,]keyMatrix = new int[3, 3];
    getKeyMatrix(key, keyMatrix);

    int [,]messageVector = new int[3, 1];

    // Generate vector for the message
    for (int i = 0; i < 3; i++)
        messageVector[i, 0] = (message[i]) % 65;

    int [,]cipherMatrix = new int[3, 1];
```

```
    // Following function generates
    // the encrypted vector
    encrypt(cipherMatrix, keyMatrix, messageVector);

    String CipherText = "";

    // Generate the encrypted text from
    // the encrypted vector
    for (int i = 0; i < 3; i++)
        CipherText += (char)(cipherMatrix[i, 0] + 65);

    // Finally print the ciphertext
    Console.Write("Ciphertext: " + CipherText);
}

// Driver code
public static void Main(String[] args)
{
    // Get the message to be encrypted
    String message = "ACT";

    // Get the key
    String key = "GYBNQKURP";
```

```
    HillCipher(message, key);
}
}
```

Output:

Ciphertext: POH

5.4 Transposition cipher

Given a plain-text message and a numeric key, cipher/de-cipher the given text using Columnar Transposition Cipher.

The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one.

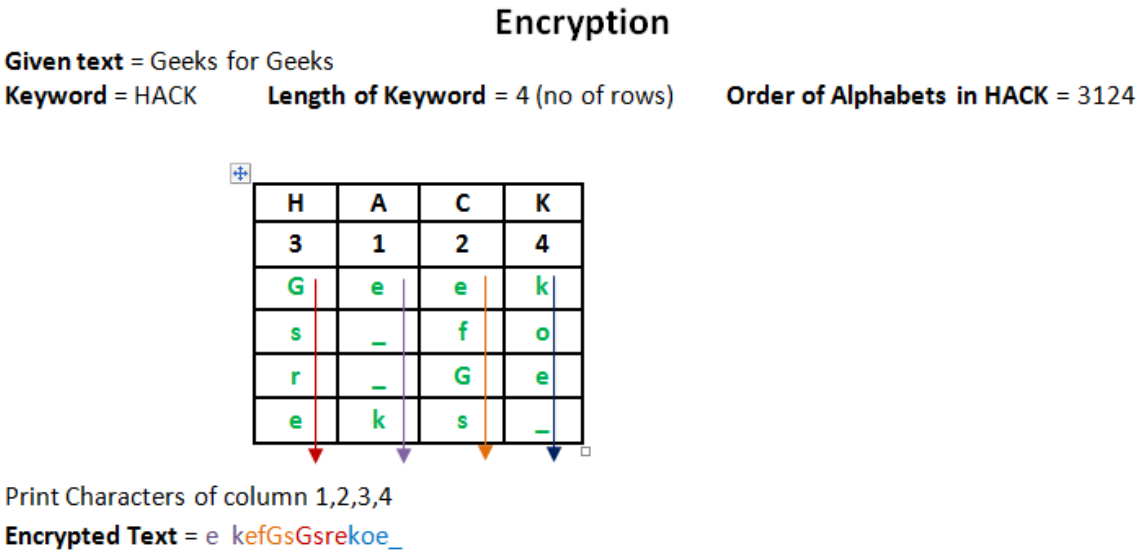
Examples:

```
Encryption
Input : Geeks for Geeks
Key = HACK
Output : e  kefGsGsrekoe_
Decryption
Input : e  kefGsGsrekoe_
Key = HACK
Output : Geeks for Geeks
```

5.4.1 Encryption

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be “3 1 2 4”.
4. Any spare spaces are filled with nulls or left blank or placed by a character .
5. Finally, the message is read off in columns, in the order specified by the keyword.



5.4.2 Decryption

1. To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length.
2. Then, write the message out in columns again, then re-order the columns by reforming the key word.

   Example:

```
Private Shared Function GetShiftIndexes(key As String) As
Integer()
    Dim keyLength As Integer = key.Length
    Dim indexes As Integer() = New Integer(keyLength - 1) {}
    Dim sortedKey As New List(Of KeyValuePair(Of Integer,
Char))()
    Dim i As Integer

    For i = 0 To keyLength - 1
            sortedKey.Add(New KeyValuePair(Of Integer,
Char)(i, key(i)))
    Next

    sortedKey.Sort(Function(pair1 As KeyValuePair(Of
Integer, Char), pair2 As KeyValuePair(Of Integer, Char))
pair1.Value.CompareTo(pair2.Value))

    For i = 0 To keyLength - 1
            indexes(sortedKey(i).Key) = i
```

```
    Next

    Return indexes
End Function

Public Shared Function Encipher(input As String, key As
String, padChar As Char) As String
    input = If((input.Length Mod key.Length = 0), input,
input.PadRight(input.Length - (input.Length Mod
key.Length) + key.Length, padChar))
    Dim output As New StringBuilder()
    Dim totalChars As Integer = input.Length
    Dim totalColumns As Integer = key.Length
    Dim totalRows As Integer =
CInt(Math.Truncate(Math.Ceiling(CDbl(totalChars) /
totalColumns)))
    Dim rowChars As Char(,) = New Char(totalRows - 1,
totalColumns - 1) {}
    Dim colChars As Char(,) = New Char(totalColumns - 1,
totalRows - 1) {}
    Dim sortedColChars As Char(,) = New Char(totalColumns
- 1, totalRows - 1) {}
    Dim currentRow As Integer, currentColumn As Integer, i
As Integer, j As Integer
    Dim shiftIndexes As Integer() = GetShiftIndexes(key)
```

```
    For i = 0 To totalChars - 1
        currentRow = i \ totalColumns
        currentColumn = i Mod totalColumns
        rowChars(currentRow, currentColumn) = input(i)
    Next

    For i = 0 To totalRows - 1
        For j = 0 To totalColumns - 1
            colChars(j, i) = rowChars(i, j)
        Next
    Next

    For i = 0 To totalColumns - 1
        For j = 0 To totalRows - 1
            sortedColChars(shiftIndexes(i), j) = colChars(i,
j)
        Next
    Next

    For i = 0 To totalChars - 1
        currentRow = i \ totalRows
        currentColumn = i Mod totalRows
        output.Append(sortedColChars(currentRow,
currentColumn))
```

```
    Next

    Return output.ToString()
End Function

Public Shared Function Decipher(input As String, key As
String) As String
    Dim output As New StringBuilder()
    Dim totalChars As Integer = input.Length
    Dim totalColumns As Integer =
CInt(Math.Truncate(Math.Ceiling(CDbl(totalChars) /
key.Length)))
    Dim totalRows As Integer = key.Length
    Dim rowChars As Char(,) = New Char(totalRows - 1,
totalColumns - 1) {}
    Dim colChars As Char(,) = New Char(totalColumns - 1,
totalRows - 1) {}
    Dim unsortedColChars As Char(,) = New
Char(totalColumns - 1, totalRows - 1) {}
    Dim currentRow As Integer, currentColumn As Integer, i
As Integer, j As Integer
    Dim shiftIndexes As Integer() = GetShiftIndexes(key)

    For i = 0 To totalChars - 1
            currentRow = i \ totalColumns
```

```
            currentColumn = i Mod totalColumns
            rowChars(currentRow, currentColumn) = input(i)
    Next

    For i = 0 To totalRows - 1
            For j = 0 To totalColumns - 1
                    colChars(j, i) = rowChars(i, j)
            Next
    Next

    For i = 0 To totalColumns - 1
            For j = 0 To totalRows - 1
                    unsortedColChars(i, j) = colChars(i,
shiftIndexes(j))
            Next
    Next

    For i = 0 To totalChars - 1
            currentRow = i \ totalRows
            currentColumn = i Mod totalRows
            output.Append(unsortedColChars(currentRow,
currentColumn))
    Next

    Return output.ToString()
```

```
    End Function



    Dim text As String = "The quick brown fox jumps over the
    lazy dog"
    Dim key As String = "pangram"
    Dim cipherText As String = Encipher(text, key, "-"c)
    Dim plainText As String = Decipher(cipherText, key)



    Output:
        cipherText: "hk mra-uo oed- bosty-iwjv o-e fp z-
Tcnuelgqrx h -"
        plainText: "The quick brown fox jumps over the lazy dog-
-----"
```

5.5 substitution

In cryptography, a simple substitution cipher is a cipher that has been in use for many hundreds of years. It is an improvement to the Caesar Cipher. Instead of shifting the alphabets by some number, this scheme substitutes every plaintext character for a different ciphertext character.

With 26 letters in alphabet, the possible permutations are 26! (Factorial of 26) which is equal to 4x1026. The sender and the receiver may choose any one of these possible permutation as a ciphertext alphabet. This permutation is the secret key of the scheme.

Example:

Here is a quick example for the encryption and decryption of simple substitution cipher. The text we will encrypt is 'cryptography'.

Keys for the simple substitution cipher usually consist of 26 letters. An example key is:

Plain Alphabet:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher Alphabet:	y	h	k	q	g	v	x	f	o	l	u	a	p	w	m	t	z	e	c	j	d	b	s	n	r	i

An example encryption using the above key:

Plain Alphabet:	c	r	y	p	t	o	g	r	a	p	h	y
Cipher Alphabet:	k	e	r	t	j	m	x	e	y	t	f	r

It is easy to see how each character in the plaintext is replaced with the corresponding letter in the cipher alphabet. Decryption is just as easy, by going from the cipher alphabet back to the plain alphabet.

Simple substitution cipher is a considerable improvement over the Caesar Cipher. The possible number of keys is large (26!) and even the modern computing systems are not yet powerful enough to comfortably launch a brute force attack to break the

system. However, the simple substitution cipher has a simple design and it is prone to design flaws, say choosing obvious permutation, this cryptosystem can be easily broken.

Example:

```
Private Shared Function Cipher(input As String, oldAlphabet As
String, newAlphabet As String, ByRef output As String) As Boolean
        output = String.Empty

        If oldAlphabet.Length <> newAlphabet.Length Then
                Return False
        End If

        For i As Integer = 0 To input.Length - 1
                Dim oldCharIndex As Integer =
oldAlphabet.IndexOf(Char.ToLower(input(i)))

                If oldCharIndex >= 0 Then
                        output += If(Char.IsUpper(input(i)),
Char.ToUpper(newAlphabet(oldCharIndex)),
newAlphabet(oldCharIndex))
                Else
                        output += input(i)
                End If
        Next
```

```
        Return True
End Function

Public Shared Function Encipher(input As String, cipherAlphabet As
String, ByRef output As String) As Boolean
        Dim plainAlphabet As String = "abcdefghijklmnopqrstuvwxyz"
        Return Cipher(input, plainAlphabet, cipherAlphabet, output)
End Function

Public Shared Function Decipher(input As String, cipherAlphabet As
String, ByRef output As String) As Boolean
        Dim plainAlphabet As String = "abcdefghijklmnopqrstuvwxyz"
        Return Cipher(input, cipherAlphabet, plainAlphabet, output)
End Function

Dim text As String = "The quick brown fox jumps over the lazy dog"
Dim cipherAlphabet As String = "yhkqgvxfoluapwmtzecjdbsnri"
Dim cipherText As String
Dim plainText As String

Dim encipherResult As Boolean = Encipher(text, cipherAlphabet,
cipherText)
Dim decipherResult As Boolean = Decipher(cipherText,
cipherAlphabet, plainText)
```

Output

cipherText: "Jfg zdoku hemsw vmn ldptc mbge jfg ayir qmx"

plainText: "The quick brown fox jumps over the lazy dog"

5.6 Vigenere Cipher

Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenere *square or* Vigenere *table*.

- The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows.
- The alphabet used at each point depends on a repeating keyword.

Example:

Input : Plaintext : GEEKSFORGEEKS

Keyword : AYUSH

Output : Ciphertext : GCYCZFMLYLEIM

For generating key, the given keyword is repeated

in a circular manner until it matches the length of

the plain text.

The keyword "AYUSH" generates the key

"AYUSHAYUSHAYU"

The plain text is then encrypted using the process

explained below.

5.6.1 Encryption

The first letter of the plaintext, G is paired with A, the first letter of the key. So use row G and column A of the Vigenere square, namely G. Similarly, for the second letter of the plaintext, the second letter of the key is used, the letter at row E and column Y is C. The rest of the plaintext is enciphered in a similar fashion.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

5.6.2 Decryption

Decryption is performed by going to the row in the table corresponding to the key, finding the position of the ciphertext letter in this row, and then using the column's

label as the plaintext. For example, in row A (from AYUSH), the ciphertext G appears in column G, which is the first plaintext letter. Next we go to row Y (from AYUSH), locate the ciphertext C which is found in column E, thus E is the second plaintext letter.

A more easy implementation could be to visualize Vigenère algebraically by converting [A-Z] into numbers [0–25].

Encryption

The plaintext(P) and key(K) are added modulo 26.

$E_i = (P_i + K_i) \bmod 26$

Decryption

$D_i = (E_i - K_i + 26) \bmod 26$

Note: D_i denotes the offset of the i-th character of the plaintext. Like offset of A is 0 and of B is 1 and so on.

Below is the implementation of the idea.

```
Private Shared Function [Mod](a As Integer, b As Integer) As
Integer
        Return (a Mod b + b) Mod b
End Function

Private Shared Function Cipher(input As String, key As String,
encipher As Boolean) As String
        For i As Integer = 0 To key.Length - 1
```

```
            If Not Char.IsLetter(key(i)) Then
                Return Nothing ' Error
            End If
        Next

        Dim output As String = String.Empty
        Dim nonAlphaCharCount As Integer = 0

        For i As Integer = 0 To input.Length - 1
            If Char.IsLetter(input(i)) Then
                Dim cIsUpper As Boolean =
Char.IsUpper(input(i))
                Dim offset As Integer =
Convert.ToInt32(If(cIsUpper, "A"c, "a"c))
                Dim keyIndex As Integer = (i -
nonAlphaCharCount) Mod key.Length
                Dim k As Integer =
Convert.ToInt32(If(cIsUpper, Char.ToUpper(key(keyIndex)),
Char.ToLower(key(keyIndex)))) - offset
                k = If(encipher, k, -k)
                Dim ch As Char =
ChrW(([Mod](((Convert.ToInt32(input(i)) + k) - offset), 26)) +
offset)
                output += ch
            Else
```

```
                output += input(i)
                nonAlphaCharCount += 1
            End If
        Next

        Return output
End Function

Public Shared Function Encipher(input As String, key As String)
As String
        Return Cipher(input, key, True)
End Function

Public Shared Function Decipher(input As String, key As String)
As String
        Return Cipher(input, key, False)
End Function

Dim text As String = "Hello, World!"
Dim cipherText As String = Encipher(text, "cipher")
Dim plainText As String = Decipher(cipherText, "cipher")
```

Output

```
  cipherText:      "Jmass, Nqzak!"
  plainText:       "Hello, World!"
```

5.7 Vernam cipher

Vernam Cipher is a method of encrypting alphabetic text. It is simply a type of substitution cipher. In this mechanism we assign a number to each character of the Plain-Text, like (a = 0, b = 1, c = 2, … z = 25).

Method to take key:

In Vernam cipher algorithm, we take a key to encrypt the plain text which length should be equal to the length of the plain text.

Encryption Algorithm:

1. Assign a number to each character of the plain-text and the key according to alphabetical order.
2. Add both the number (Corresponding plain-text character number and Key character number).
3. Subtract the number from 26 if the added number is greater than 26, if it isn't then leave it.

Example:

```
Plain-Text: RAMSWARUPK
Key: RANCHOBABA
```

Now according to our encryption algorithm we assign a number to each character of our plain-text and key.

```
PT:  R  A  M  S  W  A  R  U  P  K
NO:  17 0  12 18 22 0  17 20 15 10

KEY: R  A  N  C  H  O  B  A  B  A
```

```
NO:  17  0  13  2  7  14  1  0  1  0
```

Now add the number of Plain-Text and Key and after doing the addition and subtraction operation (if required), we will get the corresponding Cipher-Text character number.

```
CT-NO: 34  0  25  20  29  14  18  20  16  10
```

In this case, there are two numbers which are greater than the 26 so we have to subtract 26 from them and after applying the subtraction operation the new Cipher text character numbers are as follow:

```
CT-NO:  8  0  25  20  3  14  18  20  16  10
```

New Cipher-Text is after getting the corresponding character from the number.

```
CIPHER-TEXT: I  A  Z  U  D  O  S  U  Q  K
```

5.8 Playfair cipher

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet.

It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.

Encryption Technique

For the encryption process let us consider the following example:

```
Key:monarchy
Plaintext:instruments
```

The Playfair Cipher Encryption Algorithm:

The Algorithm consistes of 2 steps:

1. Generate the key Square(5×5):
   - The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
   - The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

   For example:

```
The key is "monarchy"
Thus the initial entires are
'm', 'o', 'n', 'a', 'r', 'c', 'h', 'y'
followed by remaining characters of
a-z(except 'j') in that order.
```

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

For example:

```
PlainText: "instruments"
After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'
```

Rules for Encryption:

- If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).

  For example:

```
Diagraph: "me"
Encrypted Text: cl
Encryption:
 m -> c
 e -> l
```

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).

For example:

```
Diagraph: "st"
Encrypted Text: tl
Encryption:
 s -> t
 t -> l
```

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

For example:

```
Diagraph: "nt"
```

Encrypted Text: rq

Encryption:

n -> r

t -> q

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

For example:

Plain Text: "instrumentsz"

Encrypted Text: gatlmzclrqtx

Encryption:

i -> g

n -> a

s -> t

t -> l

r -> m

u -> z

m -> c

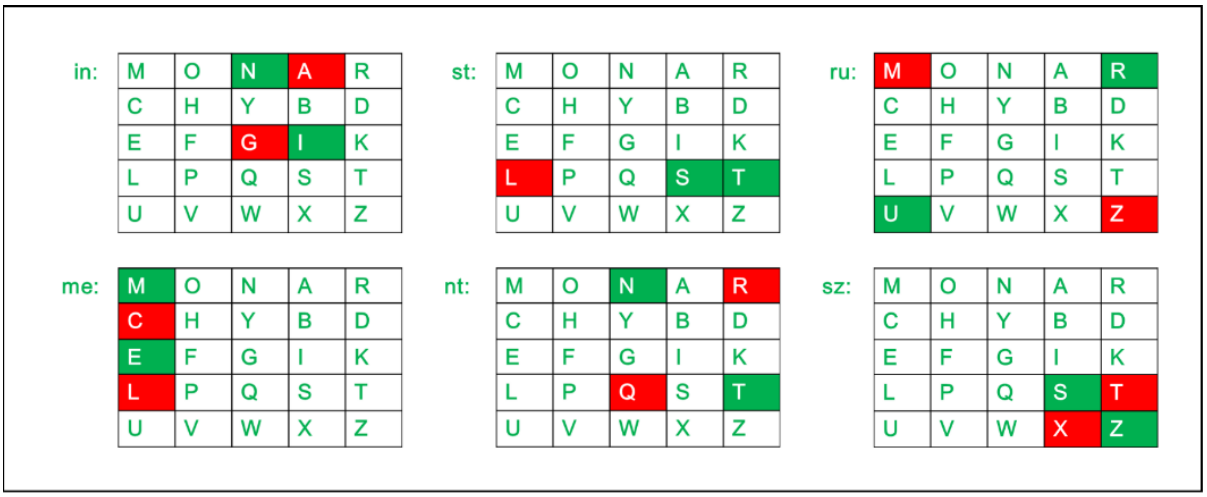
e -> l

n -> r

t -> q

s -> t

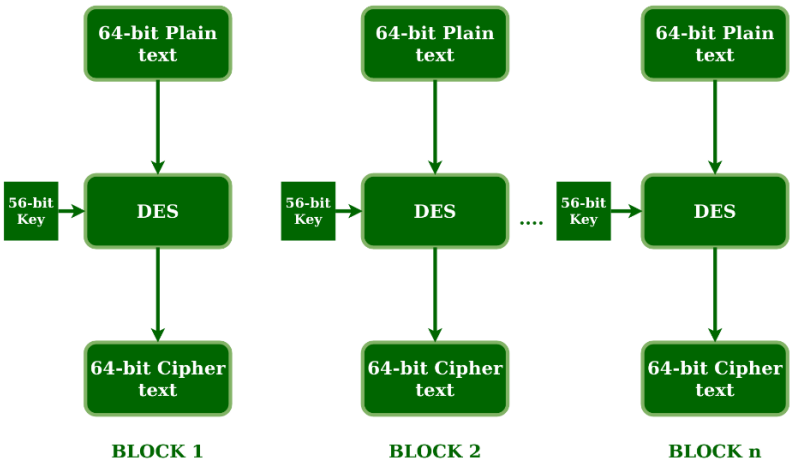
z -> x



5.9 DES (Data Encryption standard)

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline.

DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is show in figure.



We have mention that DES uses a 56 bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.

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	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Initial Permutation (IP)

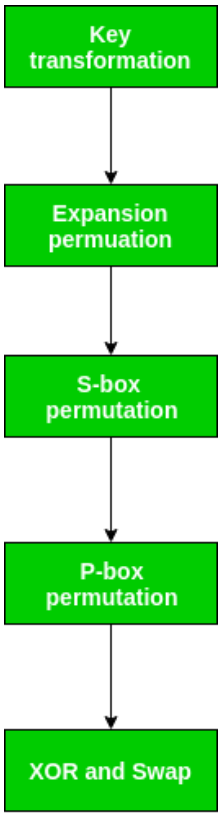
As we have noted, the Initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as show in figure.

For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block and so on.

This is nothing but jugglery of bit positions of the original plain text block. the same rule applies for all the other bit positions which shows in the figure.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

As we have noted after IP done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad level steps outlined in figure.



Step-1: Key transformation

We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called as

key transformation. For this the 56 bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

For example, if the round number 1, 2, 9 or 16 the shift is done by only position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is show in figure.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

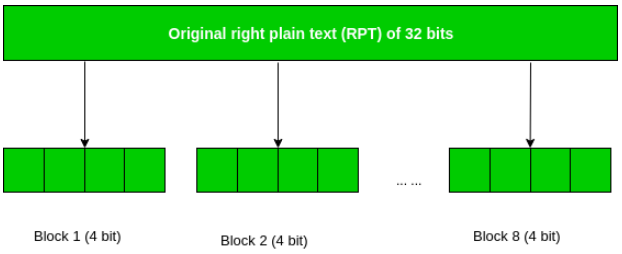
After an appropriate shift, 48 of the 56 bit are selected. for selecting 48 of the 56 bits the table show in figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub set of the original 56-bit key it is called Compression Permutation.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Because of this compression permutation technique, a different subset of key bits is used in each round. That's make DES not easy to crack.

Step-2: Expansion Permutation

Recall that after initial permutation, we had two 32-bit plain text areas called as Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called as expansion permutation. This happens as the 32 bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4 bit block of the previous step is then expanded to a corresponding 6 bit block, i.e., per 4 bit block, 2 more bits are added.



This process results into expansion as well as permutation of the input bit while creating output. Key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits. Now the 48-bit key is XOR with 48-bit RPT and resulting output is given to the next step.

Example:

```
Imports System.Security
Imports System.Security.Cryptography
Imports System.Text
Imports System.IO

Public Class Tester
   Public Shared Sub Main()
```

```
        Try

        Dim encryptor As DESCryptoServiceProvider = New
DESCryptoServiceProvider()

        encryptor.Key =
ASCIIEncoding.ASCII.GetBytes("12345678")

        encryptor.IV =
ASCIIEncoding.ASCII.GetBytes("12345678")

        Dim encryptation As ICryptoTransform =
encryptor.CreateEncryptor(encryptor.Key, encryptor.IV)

        Dim Sourcefile As FileStream = New
FileStream("TextFile.txt", FileMode.Open, FileAccess.Read)
        Dim Outputfile As FileStream = New
FileStream("OutputTextFile.txt", FileMode.Create,
FileAccess.Write)
        Dim encrprocess As CryptoStream = New
CryptoStream(Outputfile, encryptation,
CryptoStreamMode.Write)

        Dim inputarray(Sourcefile.Length - 1) As Byte
```

```
            Sourcefile.Read(inputarray, 0, inputarray.Length)
            encrprocess.Write(inputarray, 0, inputarray.Length)

            encrprocess.Close()
            Sourcefile.Close()
            Outputfile.Close()

        Catch a As Exception
            Console.WriteLine(a.Message)

        End Try
        Console.ReadLine()
    End Sub
End Class
```

6- Resources

1. **Advanced RSA Cryptographic Algorithm for Improving Data Security.**
2. **Novel Data Encryption Algorithm**, School of Computer and System Sciences, Jaipur national University Jaipur, Rajasthan, India-302025
3. **Visual Basic 2008 Programming Black Book**, Platinum Edition.
4. **Security Engineering (Engineering and Management of Security),** Eric Conrad, ... Joshua Feldman, in CISSP Study Guide (Third Edition), 2016.

5. **Modern Cryptography Primer**, Kościelny, M Kurkowski, M Srebrny - 2013 - Springer