International Journal of Theoretical and Applied Sciences **4**(2): 25-32(2012) **ISSN No. (Print) : 0975-1718**
ISSN No. (Online) : 2249-3247

Implementation of MULET (Multilanguage Encryption Technique) Algorithm

Kuldeep Bhardwaj

Department of Mathematics, Dr. B.R. Ambedkar University, IBS, Khandari, Agra, (U.P.)

(Received 11 May, 2012, Accepted 12 July, 2012)

ABSTRACT : The multilingual approach in cryptography is not commonly used and hence is not so prevalent now. The MULET algorithm which is focuses on encryption of plain teat over a range of languages supported by a Unicode. It is helpful in the localization of cryptographic software took. This paper involves the implementation of MULET algorithm in.net.

Keywords : Unicode, Encryption, Decryption, Software Localization and Cryptanalysis.

I. INTRODUCTION

The Internet has recorded a rapid growth in the recent times and extremely broad spectrums of available networks have resulted into powerful, creative and useful applications. Practically all the software applications have become online, what to talk of Google Docs and Microsoft Office Live. As a result the networks have become more open and accessible consequently an adversary is not confined only to eavesdropping but has become capable to perform a more important role of acting like a man in the Middle Attack. A large number of such types of attacks were witnessed during the last decade [1][7]. Therefore, the security of the large amount of data transferred during the last decade is at stake.

The cryptology as a science dates back to Caesar's time. Since then a large variety of heuristics have been proposed for safe and secured communication. However, cryptanalysis has successfully and almost simultaneously cracked these encryption techniques from time to time [2]. Therefore the basic and fundamental task of cryptography is not only restricted to protect the secrecy of messages transmitted over public communication lives but also to subvert such cryptanalytic attacks which tend to become rampant with the passage of time.

The broad classification of data encryption techniques can be undertaken as symmetric and asymmetric key cryptography. In case of symmetric key cryptography, the same key is used by the sender and the receiver for encryption and decryption respectively. The representative algorithms of this approach are AES, TDES, RCS [3-4] and the likes. But, Asymmetric or public key cryptography makes use of two keys namely, the private key which is kept by the receiver and the public key which is announced to the public cryptosystems such as RSA, PGP and ECC [6] come under this category some of the newer and recent data encryption techniques include Quantum cryptography and GnuPG [7].

In spite of the fact that a wide variety of techniques have been employed for encryption and decryption, making use of the multilingual approach is still not very common and prevalent. This has motivated us to propose a new novel algorithm that focuses on encryption of plain text over a range of languages supported by Unicode [8]. Making use of the mapping techniques enable algorithm to become very fast efficient and easier to implement. Besides, the replacement strategy adopted here ensures better safety and security.

II. MULET ALGORITHM

A. Notation

M : Mapping Constant

ch_map : A set of *M*-characters from the universal character set is considered as a Mapping Array.

chno : A set of character for universal character set is considered as a substitution array.

Quo : Quotients required for decryption (key).

Enc : Ciphered text.

Dec : Deciphered text.

B. Encryption

The text selected for encryption is read character by character and the Unicode value of each character is obtained. This value is thereafter divided by the mapping constant *M*. The remainder *R* so determined is used as the index of mapping array *ch_map* and *ch_map[R]* and it is the corresponding encrypted character from the cipher text Enc. The questioned obtained after division is stored in and there array *Quo*. The quotient is put to use in decryption. In brief it can be said that the remainder attains the encrypted character and the quotient holds the key for decryption of the corresponding character.

The cipher text *Enc* is likely to have repetition of characters. This is because of the fact that the encryption technique maps the characters of the original message to the mapping array *ch_map*. In view of this a replacement strategy is incorporated and it helps in maintaining checks

over successive repetitions of characters. In case, such repetitive patterns are observed, they are replaced with a character in substitution array chno and corresponding to the number of such repetitions. It has been observed that this replacement strategy inducts some non-regular characters into the cipher text and consequently crypto analysis become evident from the cipher text. The chances of substitution of multiples to great extent depend upon the mapping consonant *M* in addition to the plain text.

C. Decryption

Scanning of the cipher text is carried out for characters in the substation array. If *chno[i]* is found to be a number 'm' the character preceding *chno[i]* in the cipher text is in number of times to obtain the temporary repeated '*m*' number of times to obtain the temporary decrypted massage. A comparison of the characters of the temporary decrypted message is undertaken with the mapping array ch_map. In case these characters match, the corresponding index of the mapping array is the remainder *R*. The Unicode values of the characters of the original massage are there after determined by adding *R* to the product of *M* and *Quo*. There values provide us the corresponding characters of the plain text and these accounts for the decryption procedure.

D. The Algorithm:

The MULLET Algorithm [7] basically comprises of two functions *viz.* Encryption ( ) and decryption ( ) as described above, Encryption of the plain text is used to obtain the cipher text which is normally obtained by substitution of multiples. The transmitted encrypted message Enc is received by the recipient as *Dec*. Undoing substitutions followed by decryption of the cipher text gives back the original message.

A plain text takes as input in the function *Encryption* ( ) and obtains the cipher text *enc* as output.

```
Begin
 while ( ! End of plain text)
  Begin
   Read a character from the original file and store the Unicode value in a variable n ;
    R: = n%k;
    Quo [i] : = n / k ;
    Enc [i] : = ch_map [R]
    Increment i ;
  End
 while ( ! end of Enc )
  Begin
   while (Enc [i] = = Enc [i+1] )
    Begin
     Increment count ;
     Increment  i ;
    End
  if (count >=2)
    Replace the repetitions with
    chno [ count ] in enc
   Reset count to zero
   End
 End
```

With the cipher text enc as input, the function *Decryption ( )* obtains the original message *dec.*

```
Begin
 While (! end of enc)
  Beg in
   If (character is chno [ i ])
    Remove the character from enc and the character preceding chno [ i ] in the  cipher text is repeated ' i ' number of times and store in dec
  End
 While (! end of Dec)
  Begin
   Compare the character with the mapping array ch_map;
   Position of the character in ch_map is the required remainder R ;
    U: = Quo [ i ] * M + R ;
    Convert U to the corresponding character;
  End
 End
```

III. IMPLEMENTATION OF MULET ALGORITHM

Existing methodology about various encryption techniques and algorithms most of them are implemented In English in those techniques the cipher text produced is not in a readable for and thus reveal that the message is encrypted. We are going to implement in Hindi and going to use Unicode for it.

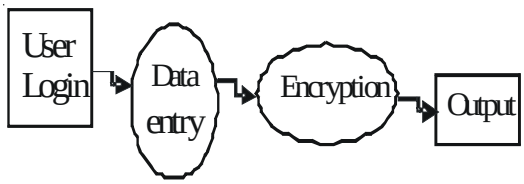
A. Modules Description

There are basically three modules :

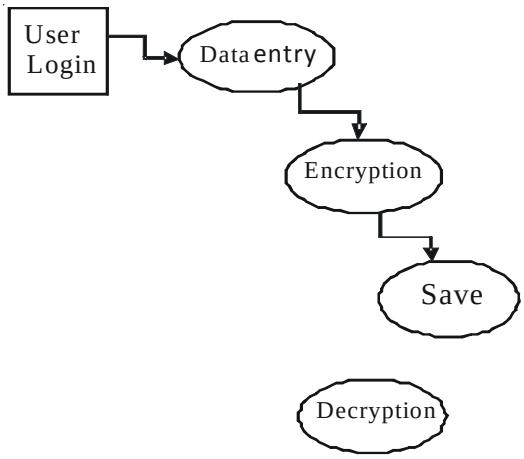
- **(*a*) Encryption Module :** In this module we are going to convert our plain text into cipher text by using any of the following algorithms RSA, RC4 and Elgamal.
- **(*b*) Decryption Module :** In this module we are going to convert cipher text into plain text by using the same algorithm that we have used for encryption.
- **(*c*) Mapping Module :** In this module we required to map the cipher text thus produced with the Hindi dictionary so that the cipher text is in readable form.

Data Flow Diagrams :

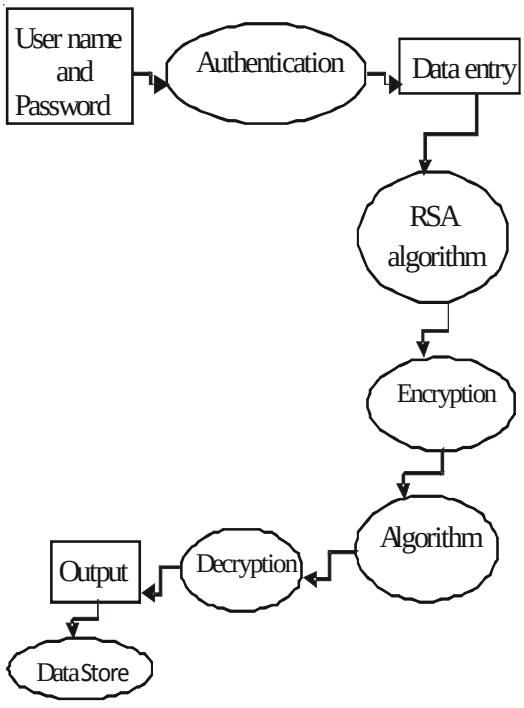
Level 0



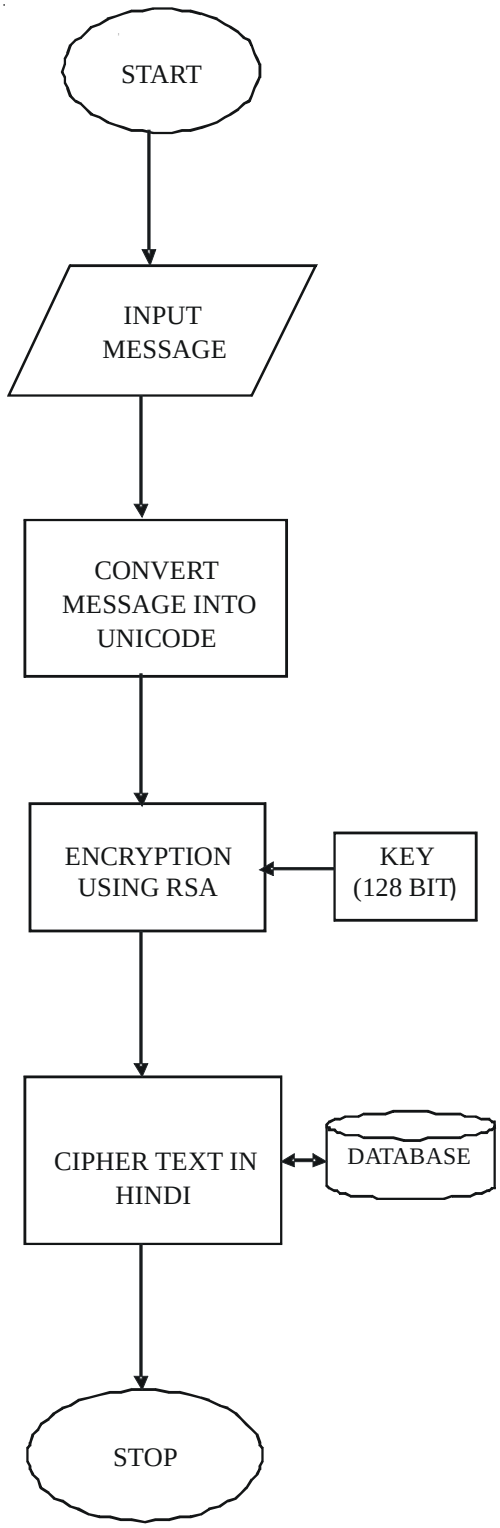
Level 1



Level 2



Flow Chart :



The code for implementation of MULET algorithm is listed below as:

//Encoder Code

Imports System.Security.Cryptography

Imports System.Text

```
Imports System.Net.Sockets
Public Class frm_Unicode
    Dim textbytes, encryptedtextbytes As Byte()
    Dim rsa As New RSACryptoServiceProvider
    Dim encoder As New UTF8Encoding
    Dim clientSocket As New
System.Net.Sockets.TcpClient()
    Dim server Stream As Network Stream
    Dim readData As String
    Dim infinite Counter As Integer
    Private Sub Form1_Load (ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Me.OutputTableAdapter.Fill(Me.Ds.Output)
        TextBox1.Focus()
        TextBox2.Visible = False
        btn_Decode.Visible = False
        TextBox3.Visible = False
        btn_Reset.Visible = False
        lbl_decoded.Visible = False
        lbl_Encoded.Visible = False
        btn_Close.Visible = False
        btn_Save.Visible = False
        lbl_Message.Visible = False
        clientSocket.Connect("127.0.0.1", 8888)
    End Sub
    Private Sub btn_Encode_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btn_Encode.Click
        Dim TexttoEncrypt As String = TextBox1.Text
        Dim outStream As Byte()
        lbl_Encoded.Visible = True
        TextBox2.Visible = True
        btn_Decode.Visible = True
        btn_Save.Visible = True
        TextBox1.Enabled = False
        textbytes = encoder.GetBytes(TexttoEncrypt)
        encryptedtextbytes = rsa.Encrypt(textbytes, True)
        TextBox2.Text =
Convert.ToBase64String(encryptedtextbytes)
        readData = "Conected to Encoder / Decoder
Server ..."
        msg()
        serverStream = clientSocket.GetStream()
        outStream =
System.Text.Encoding.ASCII.GetBytes("Admin" + "$")
        serverStream.Write(outStream, 0,
outStream.Length)
        serverStream.Flush()
        Dim ctThread As Threading.Thread = New
Threading.Thread(AddressOf getMessage)
        ctThread.Start()
        outStream =
System.Text.Encoding.ASCII.GetBytes(TextBox1.Text +
"$")
        serverStream.Write(outStream, 0,
outStream.Length)
        serverStream.Flush()
    End Sub
    Private Sub btn_Decode_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btn_Decode.Click
        lbl_decoded.Visible = True
        TextBox3.Visible = True
        btn_Reset.Visible = True
        btn_Close.Visible = True
        textbytes = rsa.Decrypt(encryptedtextbytes, True)
        TextBox3.Text = encoder.GetString(textbytes)
        btn_Reset.Focus()
    End Sub

    Private Sub btn_Reset_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btn_Reset.Click
        TextBox2.Visible = False
        btn_Decode.Visible = False
        TextBox3.Visible = False
        btn_Reset.Visible = False
        lbl_decoded.Visible = False
        lbl_Encoded.Visible = False
        TextBox1.Enabled = True
        TextBox1.Text = ""
        TextBox1.Focus()
        btn_Close.Visible = False
        lbl_Message.Visible = False
        btn_Save.Visible = False
```

```
        End Sub
        Private Sub btn_Close_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_Close.Click
            Me.Close()
        End Sub
        Private Sub btn_Save_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_Save.Click
            Me.OutputTableAdapter.Insert(TextBox1.Text, TextBox2.Text)
            lbl_Message.Text = "Data saved to Database"
            lbl_Message.Visible = True
            btn_Close.Visible = True
            btn_Reset.Visible = True
        End Sub
        Private Sub msg()
            If Me.InvokeRequired Then
                Me.Invoke(New MethodInvoker(AddressOf msg))
            End If
        End Sub
        Private Sub getMessage()
            For Me.infiniteCounter = 1 To 2
                infiniteCounter = 1
                serverStream = clientSocket.GetStream()
                Dim buffSize As Integer
                Dim inStream(10024) As Byte
                buffSize = clientSocket.ReceiveBufferSize
                serverStream.Read(inStream, 0, buffSize)
                Dim returndata As String = _
                System.Text.Encoding.ASCII.GetString(inStream)
                readData = "" + returndata
                msg()
            Next
        End Sub
    End Class
    //Database Connectivity
    Imports System.Data.OleDb
    Public Class frm_Login
        Dim cn As OleDbConnection
        Dim cmd As OleDbCommand
        Dim dr As OleDbDataReader
        Private Sub OK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OK.Click
            Try
                cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=e:\projects\juhi\unicode.accdb;")
                cn.Open()
                cmd = New OleDbCommand("select * from users", cn)
                dr = cmd.ExecuteReader
                Dim user As String
                Dim pass As String
                Dim flag1 As Boolean
                Dim flag2 As Boolean
                flag1 = False
                flag2 = False
                While dr.Read()
                    user = dr(0)
                    pass = dr(1)
                    If txt_Username.Text.ToLower = user Then
                        flag1 = True
                        If txt_Password.Text = pass Then
                            flag2 = True
                        End If
                    End If
                End While
                If flag1 = True Then
                    If flag2 = True Then
                        MsgBox("Valid User & Password")
                        If txt_Username.Text.ToLower = "admin" Then
                            MsgBox("Welcome Admin, You are allowed to run Encoder & Decoder")
                            frm_Unicode.Show()
                        End If
                    Else
                        MsgBox("Incorrect Password")
                    End If
                Else
                    MsgBox("Unknown User")
                End If
```

```
            Catch
            End Try
            dr.Close()
            cn.Close()
        End Sub
        Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Cancel.Click
            Me.Close()
        End Sub
        Private Sub frm_Login_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        End Sub
    End Class
    //Server
    Imports System.Net.Sockets
    Imports System.Text
    Module Module1
        Dim clientsList As New Hashtable
        Sub Main()
            Dim serverSocket As New TcpListener(8888)
            Dim clientSocket As TcpClient
            Dim infiniteCounter As Integer
            Dim counter As Integer
            serverSocket.Start()
            msg("Encoder / Decoder Server Started ....")
            counter = 0
            infiniteCounter = 0
            For infiniteCounter = 1 To 2
                infiniteCounter = 1
                counter += 1
                clientSocket = serverSocket.AcceptTcpClient()
                Dim bytesFrom(10024) As Byte
                Dim dataFromClient As String
                Dim networkStream As NetworkStream = _
                    clientSocket.GetStream()
                networkStream.Read(bytesFrom, 0,
    CInt(clientSocket.ReceiveBufferSize))
                dataFromClient = System.Text.Encoding.ASCII.GetString(bytesFrom)
                dataFromClient = _
                    dataFromClient.Substring(0, dataFromClient.IndexOf("$"))
                clientsList(dataFromClient) = clientSocket
                broadcast(dataFromClient + " connected", dataFromClient, False)
                'msg(dataFromClient + " connected to Encoder / Decoder Server ")
                Dim client As New handleClinet
                client.startClient(clientSocket, dataFromClient, clientsList)
            Next
            clientSocket.Close()
            serverSocket.Stop()
            msg("exit")
            Console.ReadLine()
        End Sub
        Sub msg(ByVal mesg As String)
            mesg.Trim()
            Console.WriteLine(" >> " + mesg)
        End Sub
        Private Sub broadcast(ByVal msg As String, _
        ByVal uName As String, ByVal flag As Boolean)
            Dim Item As DictionaryEntry
            For Each Item In clientsList
                Dim broadcastSocket As TcpClient
                broadcastSocket = CType(Item.Value, TcpClient)
                Dim broadcastStream As NetworkStream = _
                    broadcastSocket.GetStream()
                Dim broadcastBytes As [Byte]()
                If flag = True Then
                    broadcastBytes = Encoding.ASCII.GetBytes(uName + " says : " + msg)
                Else
                    broadcastBytes = Encoding.ASCII.GetBytes(msg)
                End If
                broadcastStream.Write(broadcastBytes, 0, broadcastBytes.Length)
                broadcastStream.Flush()
            Next
        End Sub
        Public Class handleClinet
            Dim clientSocket As TcpClient
```

```
Dim clNo As String
Dim clientsList As Hashtable
Public Sub startClient(ByVal inClientSocket As TcpClient, _
ByVal clineNo As String, ByVal cList As Hashtable)
Me.clientSocket = inClientSocket
Me.clNo = clineNo
Me.clientsList = cList
Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf doChat)
ctThread.Start()
End Sub
Private Sub doChat()
Dim infiniteCounter As Integer
Dim requestCount As Integer
Dim bytesFrom(10024) As Byte
Dim dataFromClient As String
Dim sendBytes As [Byte]()
Dim serverResponse As String
Dim rCount As String
requestCount = 0
For infiniteCounter = 1 To 2
infiniteCounter = 1
Try
requestCount = requestCount + 1
Dim networkStream As NetworkStream = _
clientSocket.GetStream()
networkStream.Read(bytesFrom, 0, CInt(clientSocket.ReceiveBufferSize))
dataFromClient = System.Text.Encoding.ASCII.GetString(bytesFrom)
dataFromClient = _
dataFromClient.Substring(0, dataFromClient.IndexOf("$"))
msg("From Encoder client - " + clNo + " : " + dataFromClient)
rCount = Convert.ToString(requestCount)
broadcast(dataFromClient, clNo, True)
Catch ex As Exception
MsgBox(ex.ToString)
End Try
Next
End Sub
End Class
End Module
```

B. Technology and System Feasibility:

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not this means that feasibility is the study of the based in outline.

Operating System : Windows 9x/Windows Xp, Windows ME or more

Processor : Pentium 3.0 GHz or higher

RAM : 256 Mb or more

Hard Drive : 10 GB or more

Database : Ms Access

IV. RESULT AND DISCUSSION

First we start Encoder/Decoder Server through server.exe in Fig. 1. Fig. 2 shows main window. After the main window appears login window in Fig. 3. Welcome window shows in Fig. 4. In Fig. 5 there is three options that is Encode, Decode and Save to Database which is encrypt and decrypt the message.

Fig. 1. Server Window.

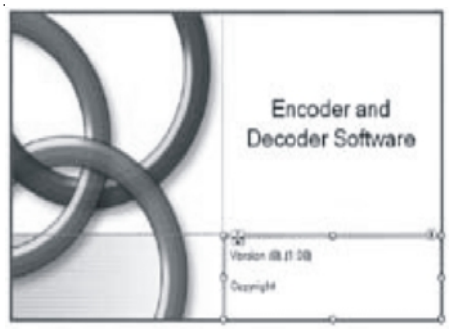


Fig. 2. Main Window.

Fig. 3. Login Window.

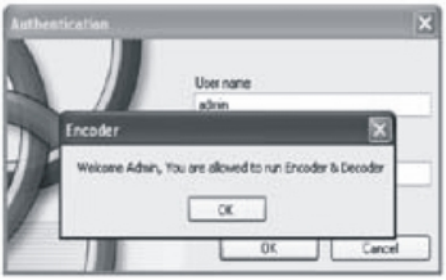


Fig. 4. Welcome Window.

Fig. 5. Encode and Decode Window.

Encryption of various messages from different languages can be easily carried out with the help of MULLET algorithm. This is definitely the characteristic feature of the algorithm as it paves the way for the localization of software in crypto graphic domain [5]. It is also an interesting fact that replacement strategy can be easily applied when we have successive repetition of characters. This mechanism is also highly effective in hiding the number of characters in the cipher text and thus makes it extremely difficult for the intruders to predict two message size.

V. CONCLUSION

The MULLET algorithm described above, direct mapping technique has been used which apart from being simple in implementation also reduces run time complexity. Thus, the ability of the proposed algorithm to work over different language domains would simplify and facilitate the localization of crypto graphic software tools. It has also been observed that the algorithm is immune to intruders and the robustness of this encryption method is attributable to multiple facets of the algorithm.

The algorithm used in this implementation is feasible to use. The algorithm is efficient to use and produces the definite output. There are some manipulations which are done to improve the efficiency for the implementation. Over all we would say that as till now this implementation has been running successfully.

VI. ACKNOWLEDGEMENT

The author wish to thank Dr. Sanjay Chaudhary for his constant encouragement for completion of this work. The author wish to acknowledgement Mis Richa Kakkar for her co-operation for completing this paper, also to Dr. Santosh Kumar Yadav for his encouragement during this work.

REFERENCES

[1] Ross J. Anderson, "Why Cryptosystems Fail", Communications of the ACM, New York, USA, 1994, pp. 32-40, (1994).

[2] Francois-Xavier Standaert, Gilles Piret, Jean-Jacques Quisquater, "Cryptanalysis of Block Ciphers: A Survey", UCL Crypto Group, (2003).

[3] R. L. Rivest, "The RC5 encryption algorithm", Proceedings of the 1994 Leuven Workshop on Fast Software Encryption, Springer-Verlag, pp. 86-96, (1995).

[4] William C. Barker, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", National Institute of Standards and Technology, NIST Special Publication 800-67, (2008).

[5] Collins, R.W., "Software localization for Internet software, issues and methods", Software, *IEEE*, Florida, USA, pp. 74-80, (2002).

[6] Elliptic Curve Cryptography, Certicom Research, (2000).

[7] G. Praveen Kumar, Arjun Kumar Murmu, Biswas Parajuli, Prasenjit Choudhury, "MULET: A Multilanguage Encryption Technique," itng, *Seventh International Conference on Information Technology*, pp.779-782, (2010).

[8] Unicode Character form http://www.unicode.org.