*I. J. Computer Network and Information Security*, **2017, 10, 41-49**
Published Online October 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2017.10.05

# Multi-layer Masking of Character Data with a Visual Image Key

**Asif Karim**
Charles Darwin University, Northern Territory, Australia
E-mail: asif.karim.2012@live.com

Received: 25 May 2017; Accepted: 01 August 2017; Published: 08 October 2017

***Abstract*—**Information is one of the single most important factor for understanding a situation as well as deciding upon a solution by effectively devising a working method. Thus the magnitude of securely delivering information to the correct individual or organization has always been a prime concern. The field of Cryptography deals with such necessities as to encrypt the information in such a way so that only the intended receiver, equipped with the right armoury, can decipher the message. Here in this paper a method for encrypting character data has been presented whereby the ASCII values of individual character is converted into their Hex values before they are turned into their binary form, and randomly assigned a single digit Odd integer value for a 1 and single digit Even integer value for a 0. Going forward we do some more processing of the data to make it even more secure, these will be discussed in due course. The receiver must possess a valid Image key in order to decrypt the text. This image is generated during encryption from two RGB (Red, Green and Blue) values-having a difference of a random number produced within the range of total length of the plaintext.

***Index Terms*—**Cryptanalysis, cryptology, bit masking, visual key encryption, symmetric encryption.

## I. Introduction

The project includes a working application developed in Visual Basic 6.0, mainly for the purpose of demonstration and harnessing VB's inherent rapid development features.

Cryptology is the field where mathematics and security engineering enmesh together to give us tools to protect systems and information. In a very basic sense, Cryptography is the art and science of designing algorithms for performing encryption, while Cryptanalysis refers to the science and arts behind breaking them. Cryptology encompasses the study of both Cryptography and Cryptanalysis.

In today's modern world, *Information Security* has become the most talked about field in computer science, which clearly underlines the gravity of cryptology and its related studies. Cryptology has been in use since the time of Roman emperors [1]. Julius Caesar enciphered A for D, B for E and so on. In modern parlance, it is said that the key had been altered from D to A.

However, Arabs are the most prominent for developing a systematic approach of cryptanalytic methods. Al-Kindi, an Arab mathematician, sometime around AD 800 invented the technique of frequency-analysis for breaking monoalphabetic substitution ciphers, which was a significant valuable resource on cryptanalytic field till World War II [2]

Leon Battista Alberti, the Italian Cryptographer, conveniently known as the "father of Western Cryptology", is the first individual to bring forth the Polyalphabetic Cipher, which is significantly stronger methodology from that of the earlier monoalphabetic crypto algorithms [3].

Gilbert Vernam in 1917 proposed a teleprinter cipher. The method included using a previously prepared key, kept on paper tape, to combine with each of the characters of the plaintext message in a view to produce the cyphertext. This led to the development of electromechanical devices as cipher machines, and to the only unbreakable cipher known as, the *one time pad* [5].

Claude Shannon's "A mathematical theory of cryptology", written in 1945, is considered by most as the inception of the development of current cryptologic advancements [6]. Modern encryptions are designed by employing algorithms having a key to encrypt and decrypt information. The role of these keys are to change the original message and data into seemingly nonsensical data or gibberish thorough encryption and then retrieve the bona-fide through decryption. The critical factor here is the key length, which is directly proportional to the degree of difficulty of breaking the encryption.

## II. Classes of Cryptosystems

The data/information that needs to be encrypted are known as Plaintext, while the output to encryption function is called Ciphertext. Depending upon the algorithm, there are number of cryptographic primitives such as Stream Ciphers, Block Ciphers and Hash Function.

### Block Ciphers

A block cipher is an encryption method which encrypts the inputted information as a **block** of data (for example, 96 contiguous bits) at once as a group instead of one bit at a time. Blocks are of fixed size bytes which can be up to 128 bits long. The block cipher can encrypt, for example, a 128 bit plaintext and generate a 128 bit cipher text as the output result. In order for a block cipher to encrypt data, the function would require a secret key which comes as a string of bits normally 128 to 256 bits long. A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits. As all the blocks need to be of same size, the last block, if necessary, is padded up to the length of desired bits [7].

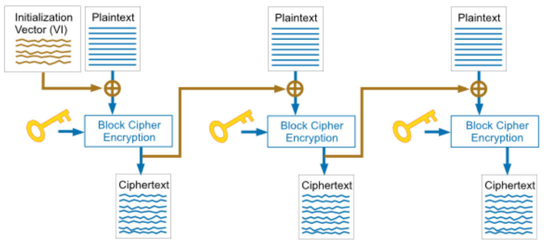


Fig.1. A Graphical Representation of Block Cipher Algorithm

Some of the more popular Block Cipher Encryption systems are Digital Encryption Standard (DES) – gained lots of ground in 90s, Triple DES – A variant of DES, Advanced Encryption Standard (AES) – a newer block cipher algorithm and IDEA – another modern block cipher application with wide acceptability and has a sufficiently robust encryption capability.

## Stream Ciphers

This types of encryption algorithm encrypts a single bit or bytes of plain text at a time. It applies a pseudorandom cipher digit stream (keystream) on each bit of the message to be encrypted. For a stream cipher implementation to remain absolutely secure, its pseudorandom generator has to be unpredictable plus the key should never be reemployed.

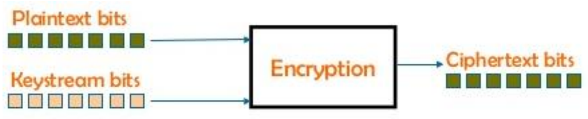
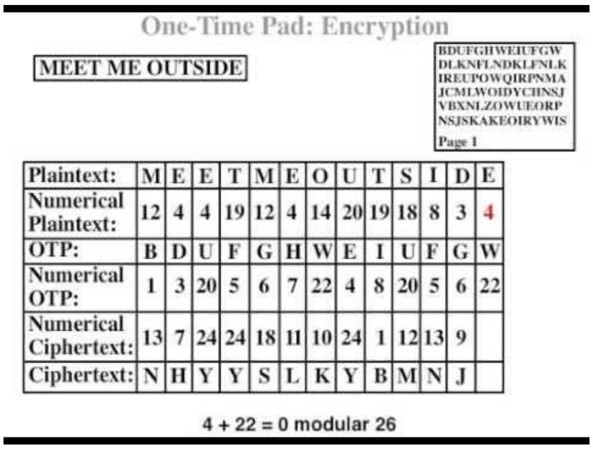


Fig.2. A graphical representation of Stream Cipher Algorithm

Stream ciphers can often be designed to attain the allusive characteristic of "Perfect Secrecy", one such example of an encryption system is the "One Time Pad". It's a technique that cannot be broken, but needs the use of a one-time key which is pre-shared, the key is of the same size, or longer, as that of the message being sent. To begin with, a plaintext is paired with a random secret key (also denoted as *a one-time pad*). Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the. Now if the key used, kept completely secret, is truly random, and at least as long as the plaintext, also if never reused in whole or partially then the resulting ciphertext will be impossible to crack [8].



| Plaintext: | M | E | E | T | M | E | O | U | T | S | I | D | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Numerical Plaintext: | 12 | 4 | 4 | 19 | 12 | 4 | 14 | 20 | 19 | 18 | 8 | 3 | 4 |
| OTP: | B | D | U | F | G | H | W | E | I | U | F | G | W |
| Numerical OTP: | 1 | 3 | 20 | 5 | 6 | 7 | 22 | 4 | 8 | 20 | 5 | 6 | 22 |
| Numerical Ciphertext: | 13 | 7 | 24 | 24 | 18 | 11 | 10 | 24 | 1 | 12 | 13 | 9 | |
| Ciphertext: | N | H | Y | Y | S | L | K | Y | B | M | N | J | |

Fig.3. One Time Pad Encryption Algorithm

However, one time pad did not get wide implementation due its key-length issues, which can sometimes be extremely large. However, one algorithm that did get wide popularity is RC4 (Rivest Cipher 4). RC4 has been used in wireless network security protocols such as WEP and WPA. Besides RC4, which has recently lost some grounds, other stream cipher systems such PANAMA and SALSA may come into foreground.

## Hash Functions

Hash functions, first used around 1960 in computer systems, these days have become extremely important and useful in many sorts of applications. Hash functions are mathematical functions that convert a numerical input value into another numerical value, often compressed. The input to the can be of arbitrary length whilst the output is always of fixed length. The encrypted values are often known as Hash values or message digest as the content is considerably smaller than the inputted data. The process of converting the raw data to a message digest is known as Hashing. Hash functions are computationally faster than symmetric algorithms such as block ciphers. Hash functions have few fundamental properties such the easiness of calculating the hash for any given data, the extreme difficulty of reversing the process and having the same hash for different data, as a slight modification in the original data entails a great change in the hash itself [9].

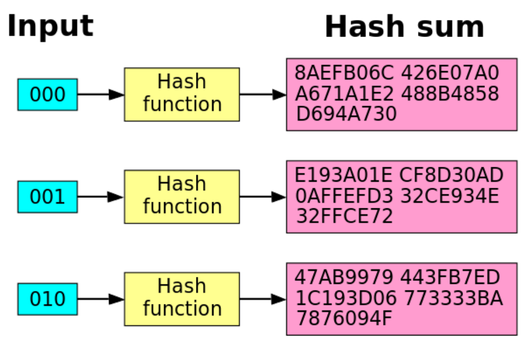

Input
Hash sum
000
Hash function
8AEFB06C 426E07A0 A671A1E2 488B4858 D694A730
001
Hash function
E193A01E CF8D30AD 0AFFEFD3 32CE934E 32FFCE72
010
Hash function
47AB9979 443FB7ED 1C193D06 773333BA 7876094F

Fig.4. Changing of Just a Single Bit Can Bring a Considerably Change in the Corresponding Hash Value

Some popular Hash encryption systems are Message Digest 5 (MD5) and Secure Hash Function (SHA). Storage of passwords and verifying data integrity are two most widely used applications of Hash functions.

## Public-Private Encryption

In a revolutionary 1976 paper, W. Diffie and M. Hellman brought forward the notion of Public-Key cryptography in which two different but mathematically oriented keys are used — a public key and a private key. A public key system is designed in such a way that calculation of one key (the 'private key') is computationally infeasible from the other (the 'public key'), even though they are necessarily interrelated. Instead, both keys are produced secretly, as an interrelated pair. In public-key cryptosystems, the public key may be distributed freely, while its paired private key must remain secret. The public key is typically used for encryption, whilst the private key is employed for decryption. RSA is an example of such cryptography. [4]

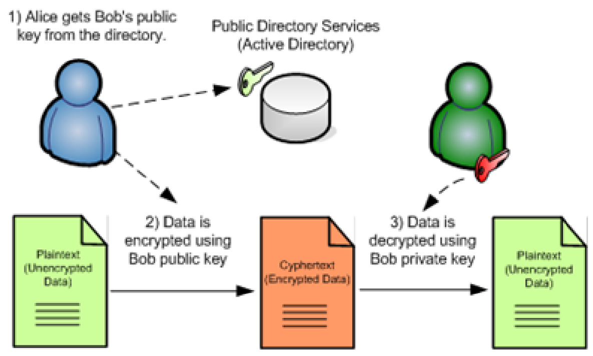

1) Alice gets Bob's public key from the directory.
Public Directory Services (Active Directory)
2) Data is encrypted using Bob public key
3) Data is decrypted using Bob private key
Plaintext (Unencrypted Data)
Cyphertext (Encrypted Data)
Plaintext (Unencrypted Data)

Fig.5. Public-Private Key Cryptosystem

The durability of systems like RSA (*Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman)* lies in the difficulty of the modern computer systems to address the computational intractability of the Integer Factorization problem. RSA is used in most Web browsers and in the SSL protocol. However, to make a system fully secured using RSA, the two prime number needed for the calculation has to be sufficiently large to make it computationally infeasible as far as possible, but still, extremely powerful machines or the advent of quantum computing may challenge this crypto system. Besides repetitive occurrence of a character in a plaintext may sometimes causes some security issues. To tackle such problems, few works have already been done, and one notable schema is the use of different ASCII value from a matrix each time for encryption. [11] RSA have also been used to encrypt audio data, and till now not many algorithms are available for developing cryptosystems on audio signal. [15]

## Visual Cryptography

Visual cryptography is a cryptographic technique which allows visual information such as texts, images etc. to be encrypted in manners by which the decryption can be achieved without using any computers at all, just by using the human visual system if the correct *Key* image is used. It is an encryption system strongly based upon graphical data and the first introduction of such a cryptosystem appeared in 1994 through the work of Naor and Adi Shamir. Visual cryptography uses two transparent images, often known as *Shares*; one image contains random pixels while the other contains the clandestine information. It is not at all possible to retrieve the secret information from one of the images. Both transparent images and layers (shares) are required to get the encrypted information. The shares are superimposed on each other to reveal the desired information [10]. But Visual Cryptography often suffers with the problem of *Pixel Alignment* [14].

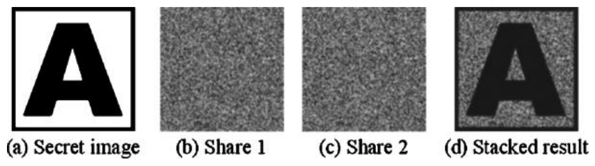


Fig.6. Example of Visual Cryptography.

## Modern Concerns

Developments such as cloud computing have also shown its potential in mobile world, and as mobile devices have storage limitations, the concept of cloud computing has also been attached to such devices. But privacy of client's data, stored on the cloud, often comes under scrutiny. Research has been going on in this field for developing new techniques and cryptosystems, as well as enhancing and modifying the existing one, such as encryption using a hybrid cryptosystem, comprising of Advanced Encryption Standard (AES), Schnorr signature and Blake2b before it can be stored in the cloud. [12]

Further, advancements such as quantum cryptography, where the counter intuitive behaviour of particles such as photons, known as "Uncertainty Principle", is being deployed to develop even more robust cryptosystems. [13]

## III. The Proposed System

The system discussed here will have multiple features to make things secure as much as possible. All the steps have been outlined below in short, and from there on we will go in some details:

### *1. Steps of Encryption*

**A)** Get the required plaintext from a source.
**B)** Convert character by character to the respective ASCII values, and then to corresponding HEX values.
**C)** Convert each of these values to 8 bit binary digits.
**D)** Now we replace each of the **1s** with a randomly chosen one digit Odd integer and **0s** with a single digit randomly chosen *Even* integer.
**E)** We now calculate the final length of the morphed plaintext, chose a random number less than the length, add it up by 100 and make sure it ends up as an odd value. Let us say the value has been put in a variable named ***z***.

**F)** After which we will continue the processing of data- by reversing the string starting from character number ***z*** to the end; let's put the reversed string in a variable named ***strRev***.

**G)** We now concatenate ***strRev*** to the rest of the original string, but bring the reversed portion (***strRev***) before the remaining portion of the original text. That remaining portion, starting from character number 1 to ***z-1***, is not reversed. Let us assume we have put the processed string in a variable named ***proStr***.

**H)** We will now step into the final processing of the ciphertext ***proStr***, by inserting random single-digit values within ***proStr*** in positions determined by each of the digits from ***z***. It will be repeated till the end of ***proStr*** is reached. This completes series of steps which will produce the final ciphertext.

### *2. Preparing the Image Key*

The image key works like a symmetric key (shared between the sender and receiver), except that there is no avenue for manual entry of any code from keyboard, rather, the receiver selects the obtained image, clicks *once* on each of the colours, and initiates the decryption process. The procedure will be illustrated further after we unravel the algorithm behind the key and its advantage. The algorithm works in the following way:

**A)** Once the encryption completes, the user will supply two (2) sets of Red (R), Green (G), Blue (B) values (RGB) and (R1G1B1), the values has to be within the range of 1-255.

**B)** The corresponding *CalculateRGB()* function now prepares *Two* separate integers from the chosen colour values (by putting the values side by side and choosing random values NOT in excess of the given value for each of R, G and B) and keeps on calculating the difference of these two values till the difference equals ***z*** (from section ***III.1.E***).

**C)** Once we have got our two desired values, the user who encrypts the data, has the job of preparing a Bitmap image, (can be easily done from Microsoft Paint). The image will have two separate colours in it, corresponding to the two values calculated in the *CalculateRGB()* function, and it will supplied to the receiver as the key to the data from decryption. The image should NOT be converted to JPEG or any other format as decolouring can occur which will change the underlying RGB values.

At this stage the encryption has been completed and the required Image key will be generated by following the procedure. One of the major reasons of using such a form symmetric key is the fact that almost all the key's composed of numbers, letters or special characters, are susceptible to Brute Force and Dictionary attack. These two schemes, given enough time and computational power, will eventually end up finding the key and will be able to automatically check against the encrypted code. In the proposed system, finding two appropriate numbers having difference equal to ***z***, manually creating an image with those values for decryption and finally manual clicking on each of the two different colours before the decryption process can begin, is a series of tasks that is quite a complicated job to come through in an automatic manner for those attacks based on Dictionary and Brute Force algorithms. We will see the whole process of both encryption and decryption with images in a later section. Now we will have an overview of the decryption process step by step.

### *3. Steps of Decryption*

**A)** Manually select the image key into the image box.

**B)** Click once on each of the two different colours. With each of the clicks the value required to calculate the difference equal to ***z*** (from section ***III.1.E***) is automatically gleaned from the underlying RGB values; these are stored in two different variables, let us say **L1** and **L2**.

**C)** Now select the encrypted data source and start the decryption process. Let us consider we put the data in a variable called ***cryptoData***.

**D)** Next difference of **L1** and **L2** is calculated, for the accurate decryption, this must equal ***z***. Let us say we put this difference in a variable called ***x***.

**E)** Now previously inserted random single-digit values within this encrypted data in positions determined by each of the digits from ***x*** will be removed. It will be repeated till the end of ***cryptoData*** is reached. Note that if the value of ***x*** is not that of ***z***, wrong data bits will be removed the decrypted text will just be a set of random gibberish.

**F)** Following these steps, we will have to reverse back the portion of data (to bring it into original orientation) that have been reversed during encryption.

**G)** The portion of data that have been reversed back in previous step is now concatenated with the rest of the non-reversed portion and position of both these segment are swapped to address the processing in section ***III.1.G***).

**H)** Going forward, all the even digits will be replaced by a '1' and off digits by a '0'.

**I)** Finally, decimal values are determined from this stream of 1's and 0's, and ASCII values are mapped from the decimal outputs, revealing the bona-fide information that had been encrypted. This completes the decryption process.

As we can see, an accurate decryption can only be possible if an image with the correct RGB values are supplied as the key.

## IV. Analysis of the Prototype Developed

As mentioned previously, a prototype has been developed in Visual Basic 6.0 for the demonstration of the algorithm and the general flow of the complete system. Here we will analyse and benchmark the performance and algorithm of each of the core segments mentioned in previous section.

***Sample VB code for Encryption and Performance Evaluation***

**A)** The algorithm mentioned in section ***III.1.B*** to ***III.1.E*** can be achieved using the following function:

```
Private Function LongToBinary$(ByVal long_value&)

Dim hex_string$, _
   digit_num%, _
   digit_value%, _
   nibble_string$, _
   result_string$, _
   factor%, _
   bit%

   hex_string = Hex$(long_value)

   hex_string = Right$(String$(8, "0") & hex_string, 8)

   For digit_num = 8 To 1 Step -1

      digit_value = CLng("&H" & Mid$(hex_string,
digit_num, 1))
      factor = 1

      For bit = 3 To 0 Step -1
         If digit_value And factor Then
            nibble_string = "1" & nibble_string
         Else
            nibble_string = "0" & nibble_string
         End If
         factor = factor * 2
      Next bit
      result_string = nibble_string & result_string

    Next digit_num

    LongToBinary = result_string

 End Function
```

Each of the Hex values are first extended to 8 bits from their 4-bit dimension and then usual processing is carried through, the colouring of the line indicates the following summary:

**B)** The algorithm mentioned in section ***III.1.B*** to ***III.1.E*** can be achieved using the following function:

```
Private Function BintoDec$(bin$)

  Dim s() As String, _
     I&, _
     numstr$, _
     numstr1$, _
     numstr2$, _
     sString$

  ReDim s(1 To Len(bin))

  For I = 1 To UBound(s)
   s(I) = Mid$(bin, I, 1)
   If s(I) = "1" Then
     s(I) = CStr(RandomInteger(True))
   Else
     s(I) = CStr(RandomInteger(False))
   End If
  Next

  For I = 1 To UBound(s)
     numstr = numstr + s(I)
  Next
  Randomize

  z = Int(((Len(numstr)) * Rnd) + 100)
  z = IIf(z Mod 2 <> 0, z, z + 1)

  numstr1 = StrReverse(Mid(numstr, z))
  numstr1 = numstr1 & Mid(numstr, 1, Len(numstr) -
Len(numstr1))

  Dim a() As String
  sString = CStr(z)

  Dim v%
  v = 1
  For I = 1 To Len(numstr1)
     If (I = CInt(Mid(sString, v, 1))) Then
        Randomize
        numstr2 = numstr2 & Mid(numstr1, I, 1) &
CStr(Int((9) * Rnd))
     Else
        numstr2 = numstr2 & Mid(numstr1, I, 1)
     End If
     If v = Len(sString) - 1 Then
        v = 1
     Else
        v = v + 1
     End If
  Next I


  BintoDec = numstr2

 End Function
```

The function *BintoDec()* runs through a set of data having 7,953 characters. The variable ***z*** has a global scope and the function also uses another function called *RandomInteger()*, depicted as follows:

```
Private Function RandomInteger%(sign As Boolean)

    Randomize
    Dim z%

    If sign = True Then
      Do
        z = Int((10) * Rnd)
      Loop While (z Mod 2) <> 0
      RandomInteger = z
    Else
      Do
        z = Int((10) * Rnd)
      Loop While (z Mod 2) = 0
      RandomInteger = z
     End If
End Function
```

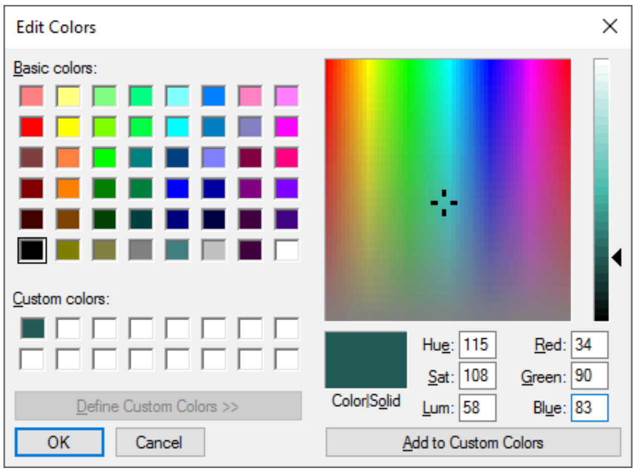
**C)** The algorithm mentioned to devise the two numbers for the preparation of the Image Key in section ***III.2.A*** and ***III.2.B*** can be attained using the following procedure:

```
Private Sub CalculateRGB()

   Dim s$, r$, g$, b$, r1$, g1$, b1$
   Dim t As Long

  s = InputBox("Enter RGB Values, each of the values
must be between 0-255", _
   "", "R=13 G=130 B=130 R1=130 G1=30 B1=130")

  Do Until Abs(t) = z
     Randomize
     r = CStr(Int(Trim(Mid(s, InStr(1, s, "R=") + 2, 3)) _
          * Rnd))
     g = CStr(Int(Trim(Mid(s, InStr(1, s, "G=") + 2, 3)) _
          * Rnd))
     b = CStr(Int(Trim(Mid(s, InStr(1, s, "B=") + 2, 3)) _
          * Rnd))

     r1 = CStr(Int(Trim(Mid(s, InStr(1, s, "R1=") + 3,
3)) _
          * Rnd))
     g1 = CStr(Int(Trim(Mid(s, InStr(1, s, "G1=") + 3,
3)) _
         * Rnd))
     b1 = CStr(Int(Trim(Mid(s, InStr(1, s, "B1=") + 3,
3)) _
         * Rnd))

     t = CLng((r & g & b)) - CLng((r1 & g1 & b1))

  Loop

MsgBox r & " " & g & " " & b & "  - AND -  " & r1 & " " & g1
& " " & b1, , "RGB Sets"
End Sub
```

The procedure shown above works by accepting two sets of RGB values from the user, and then looping through the possible combinations.

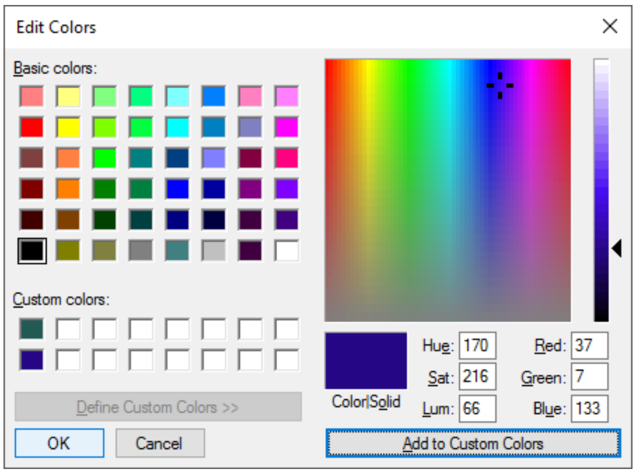
Suppose ***z*** equals 28050, therefore possible RGB sets could be {(2 54 32), (53 4 82)} because $(25432 - 53482)_{\text{Absolute}}$ = 28050; or we may have {(34 90 83), (37 7 133)} because $(349083 - 377133)_{\text{Absolute}}$ = 28050 as well. The algorithm will keep on evaluating possible combinations of RGB that satisfies the above condition and will show the values soon it finds one.

Now we can just use a simple software such as MS paint to create the Image Key with these sets of values as shown below:



Hue: 115 Red: 34
Sat: 108 Green: 90
Lum: 58 Blue: 83

Fig.7. First Pair of Colours (34, 90 and 83)



Hue: 170 Red: 37
Sat: 216 Green: 7
Lum: 66 Blue: 133

Fig.8. Second Pair of Colours (37, 7 and 133)

And the final Image Key is an image that has both of these colours and is sent with the encrypted code as an uncompressed Bitmap file.

Fig.9. The Final Multi-Colour Image Key

***Sample VB code for Decryption and Performance Evaluation***

**A)** Now before anything, the user has to load the Image Key in an appropriate Picture Box, and must click once on each of the colours. With each *Click*, the following routine calculates the underlying RGB values and saves the combination in global variables named **L1** and **L2**; **c** is also global.

```
Private Sub Picturebox_MouseDown(Button As
Integer, _
Shift As Integer, X As Single, Y As Single)

  Dim PixelColor As Long
  Dim ColorRed As Byte, ColorGreen As Byte,
ColorBlue As Byte

  PixelColor = Picturebox.Point(X, Y)
  ColorRed = PixelColor And 255
  ColorGreen = (PixelColor \ 256) And 255
  ColorBlue = (PixelColor \ 65536) And 255

  c = c + 1
  If c = 1 Then
    L1 = CLng(ColorRed & ColorGreen &
ColorBlue)
  Else
    L2 = CLng(ColorRed & ColorGreen &
ColorBlue)
  End If

  If c = 2 Then
    c = 0
  End If

End Sub
```

**B)** The following function takes care of the actual decryption process and it is assumed that **L1** and **L2** already has the numeric values to begin decryption. Note that wrong values in either of **L1** or **L2** will render incorrect decryption of the encrypted data. The algorithm has been stated in section ***III.3.B*** to ***III.3.H.***

```
Public Function Decrypt$()
Dim txt3$
Dim txt$
Dim txt1$
Dim sString$

Open lbl.Caption For Input As #1
 txt3 = Input(LOF(1), 1)
 Close #1
 txt3 = Replace(txt3, vbCrLf, "")

  Dim s() As String, _
    I&, _
    numstr$

  Dim txt2$

  sString = CStr(Abs(L1 - L2))

  Dim v%
  v = 1
  For I = 1 To Len(txt3)
    If (I = CInt(Mid(sString, v, 1)) + 1) Then
    Else
      txt = txt & Mid(txt3, I, 1)
    End If
    If v = Len(sString) - 1 Then
      v = 1
    Else
      v = v + 1
    End If
  Next I

  txt = Trim$(txt)

  txt1 = StrReverse(Mid(txt, 1, (Len(txt) -
CLng(sString))))

  txt2 = Mid(txt, (Len(txt1)) + 2) & "0" & txt1


ReDim s(1 To Len(txt2))
  For I = 1 To UBound(s)
   If Val((Mid$(txt2, I, 1))) Mod 2 = 0 Then
    s(I) = "1"
   Else
    s(I) = "0"
   End If
  Next

  numstr = vbNullString
  For I = 1 To UBound(s)

    numstr = numstr + s(I)
  Next
  Decrypt = numstr

End Function
```

To finalize the process, as mentioned in section ***III.3.I***, decimal values are derived from the binary stream returned by the above function, and subsequently ASCII values are determined from the decimal outputs, unveiling the desired information that had been encrypted.

## V. Conclusion and Future Aspirations

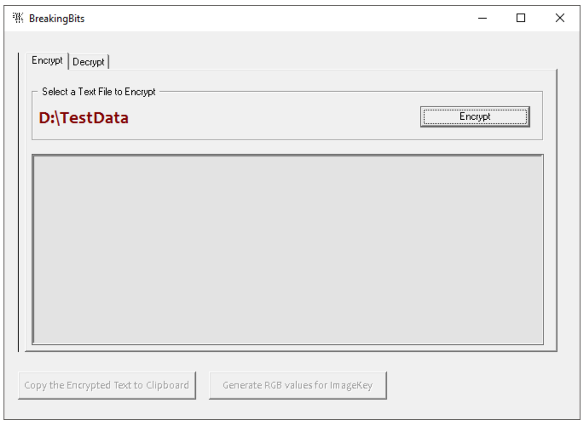
The paper proposed a system where the encryption has been achieved after multiple treatment of the data and a different but effective kind of Key is used encrypt the data. This Image Key and the overall process have been designed in such a way so that the traditional code-breaking systems are either render useless or at least face considerable obstacle in breaking the proposed system in any automatic manner. The algorithms used have all been tested and measured; but needless to say that a lot of performance improvements can be achieved if we can take the code closer to the machine. The system takes some elements out of different cipher systems discussed in Section II, but also adorns and enhances it with completely novel way of taking things forward. One of the cardinal issue that we hope will be address in coming days is the length of the numbers used to generate the Image Key. The lengths of each of the keys should be embedded into the ciphertext itself while encrypting so that the attacker cannot just keep one of the variables (**L1** or **L2**) constant and manipulate the others in some automatic way. These lengths will be checked while decryption and will in fact be used within the decryption logic itself. We can also put more additional logic in the selection of these numbers to make things even more rigorous. It is aspired that more complete and effective systems will be devised in days forward taking the core logic and idea from this paper.

## VI. Testing and Result

The Testing section will include encryption and decryption of reasonable large block of characters and the results have been shown using images for an easy and intuitive demonstration. Note the RGB values shown will keep on altering with every run.

***Screenshots of Encryption Module of the Developed Prototype***

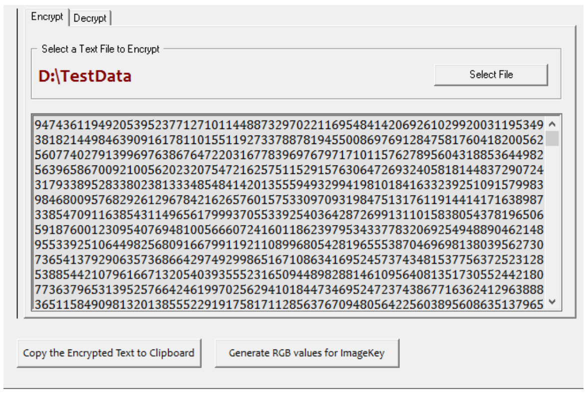
**A)** The initial screen for encryption is shown below, with required data source, having 7,953 characters, has been selected. Pressing *Encrypt* button initiates the encryption.


BreakingBits
Encrypt
Decrypt
Select a Text File to Encrypt
Encrypt
Copy the Encrypted Text to Clipboard
Generate RGB values for ImageKey

D:\TestData

Fig.10. Encryption Screen with Loaded Data Source

**B)** Encryption has been completed as shown in the following screen, with encrypted data available in the *Rich Text Box.*

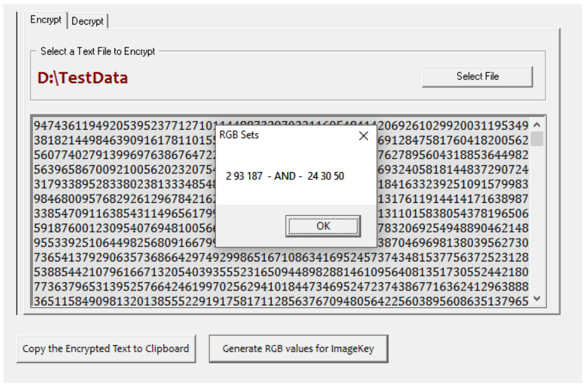

Encrypt
Decrypt
Select a Text File to Encrypt
Select File
Copy the Encrypted Text to Clipboard
Generate RGB values for ImageKey

D:\TestData

```
9474361194920539523771271011448873297022116954841420692610299200311953 49
3818214498463909161781101551192733788781945500869769128475817604182005 62
5607740279139969763867647220316778396976797171011576278956043188536449 82
5639658670092100562023207547216257511529157630647269324058181448372907 24
3179338952833802381333485484142013555949329941981018416332392510915799 83
9846800957682926129678421626576015753309709319847513176119144141716389 87
3385470911638543114965617999370553392540364287269913110158380543781965 06
5918760012309540769481005666072416011862397953433778320692549488904621 48
9553392510644982568091667991192110899680542819655538704696981380395627 30
7365413792906357368664297492998651671086341695245737434815377563725231 28
5388544210796166713205403935552316509448982881461095640813517305524421 80
7736379653139525766424619970256294101844734695247237438677163624129638 88
3651158490981320138555229191758171128563767094805642256038956086351379 65
```

Fig.11. Encrypted Data

**C)** Now Two (2) sets of RGB values are derived as shown below to prepare. … the Image Key.


Encrypt
Decrypt
Select a Text File to Encrypt
Select File
RGB Sets
OK
Copy the Encrypted Text to Clipboard
Generate RGB values for ImageKey

D:\TestData

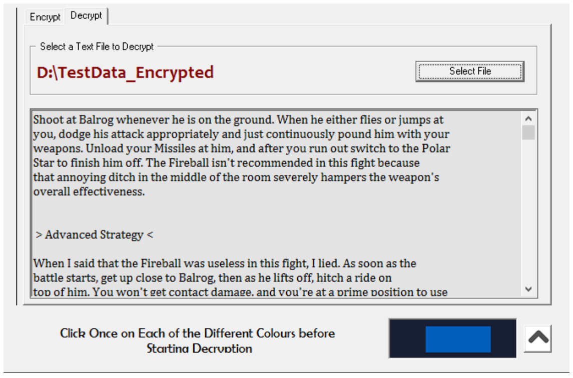
2 93 187 - AND - 24 30 50

```
94743611949205395237712710                        2069261029920031195349
38182144984639091617811015                        6912847581760418200562
56077402791399697638676472                        7627895604318853644982
56396586700921005620232075                        6932405818144837290724
31793389528338023813334854                        1841633239251091579983
98468009576829261296784216                        1317611914414171638987
33854709116385431149656179                        1311015838054378196506
59187600123095407694810056                        7832069254948890462148
95533925106449825680916679                        3870469698138039562730
736541379290635736866429749299865167108634169524573743481537756372523128
538854421079616671320540393555231650944898288146109564081351730552442180
773637965313952576642461997025629410184473469524723743867716362412963888
365115849098132013855522919175817112856376709480564225603895608635137965
```

Fig.12. RGB Sets

***Screenshots of Decryption Module of the Developed Prototype***

**A)** Screen below showing an appropriate Image Key is loaded and the data source is selected. Now by pressing the *Decrypt* button (after decryption the caption has changed to "Select File"), decryption has been completed.



D:\TestData_Encrypted

Shoot at Balrog whenever he is on the ground. When he either flies or jumps at you, dodge his attack appropriately and just continuously pound him with your weapons. Unload your Missiles at him, and after you run out switch to the Polar Star to finish him off. The Fireball isn't recommended in this fight because that annoying ditch in the middle of the room severely hampers the weapon's overall effectiveness.

> Advanced Strategy <

When I said that the Fireball was useless in this fight, I lied. As soon as the battle starts, get up close to Balrog, then as he lifts off, hitch a ride on top of him. You won't get contact damage, and you're at a prime position to use

Click Once on Each of the Different Colours before Starting Decryption

Fig.13. Decrypted Information

The prototype has been developed in a Windows 10 machine with Visual Basic 6.0. The rapid nature of the development environment has been the key factor for its selection as the development platform. The performance analysis have been carried out using *VB Watch 2*.

## REFERENCES

[1] Caeser, Kennedy (2008). Security Engineering: A Guide to Building Dependable Distributed Systems, Second Edition, Wiley, pp. 74.

[2] Kahn, David (1996). The Codebreakers, Rev. Edition, Scribner, pp. 90-107.

[3] A Brief History of Cryptography. Cypher Research Laboratories, Retrieved: 18 March 2017.

[4] Schneier, Bruce (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition, Wiley, pp. 461-466.

[5] Voborn ḱ, Petr. Migration of the Perfect Cipher to the Current Computing Environment. In: WSEAS Transactions on Information Science and Applications, 2014.

[6] Bauer, Friedrich (1998). Decrypted Secrets: Methods and Maxims of Cryptology, Fourth Edition, Springer, pp. 25-26.

[7] Robshaw, Knudsen (2011). The Block Cipher Companion, Springer, pp. 3-67.

[8] Wu, Hongjun. Cryptanalysis and Design of Stream Ciphers. Thesis Paper, Katholieke Universiteit Leuven, 2008.

[9] Schneier, Ferguson (2003). Practical Cryptography, First Edition, Wiley, pp. 83-95.

[10] Devi, Kiran. A Review on Visual Cryptography Schemes. In: Journal of Global Research in Computer Science, V: 3, N: 6, 2012.

[11] Mani. K, Viswambari. M,"Enhancing the Security in Cryptosystems Based on Magic Rectangle", International Journal of Computer Network and Information Security(IJCNIS), Vol.9, No.4, pp. 37-47, 2017.DOI: 10.5815/ijcnis.2017.04.05

[12] Oladeji P. Akomolafe, Matthew O. Abodunrin,"A Hybrid Cryptographic Model for Data Storage in Mobile Cloud Computing", International Journal of Computer Network and Information Security(IJCNIS), Vol.9, No.6, pp. 53-60, 2017.DOI: 10.5815/ijcnis.2017.06.06

[13] Assche, Gilles Van (2006). Quantum Cryptography and Secret-Key Distillation, First Edition, Cambridge University Press, pp. 01-03.

[14] Kahn, David (1996). The Cryptography for Image Processing and Security (Theory, Methods and Application), 2014, Springer, pp. 23-25.

[15] M.I.Khalil,"Real-Time Encryption/Decryption of Audio Signal", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.2, pp.25-31, 2016.DOI: 10.5815/ijcnis.2016.02.03

[16] Figure 1-6 have been sourced from the World Wide Web.

## Author's Profiles

**Asif Karim** holds BSc. (Hons.) from University of East London, UK, and is currently pursuing his MSc. in Charles Darwin University, Australia. He worked in IT industry for nearly 6 years and in academia over 1 year. His research interest lies in the fields of Cryptography and Predictive Modelling. Asif loves to visit and see new places as well as sharing ideas with like-minded peers, groups and individuals.

**How to cite this paper:** Asif Karim,"Multi-layer Masking of Character Data with a Visual Image Key", International Journal of Computer Network and Information Security(IJCNIS), Vol.9, No.10, pp.41-49, 2017.DOI: 10.5815/ijcnis.2017.10.05