THE BASICS OF TEXT ENCRYPTION & DECRYPTION

by
Àrpàd Incze

Abstract: People mean different things when they talk about cryptography. Children play with toy ciphers and secret languages. However, these have little to do with real security and strong encryption. Strong encryption is the kind of encryption that can be used to protect information of real value against organized criminals, multinational corporations, and major governments. Strong encryption used to be only military business; however, in the information society it has become one of the central tools for maintaining privacy and confidentiality. [1]

THE BEGINNING. CAESAR CIPHERS

Julius Caesar is supposed to have used secret codes known today as Caesar ciphers that consist in replacing a letter from another letter at a certain distance in the alphabet. In the simplest, A is replaced with B, B is replaced with C, and so on, up to Z, which is replaced with A. This is called a rotate-one Caesar cipher because it rotates the alphabet one place. A rotate-two cipher replaces A with C, B with D, and so on up to Z, which is replaced by B. [1]

The following line is this line encrypted with a rotate one cipher.
Uif gpmmpxjoh mjof jt uijt mjof fodszqufe xjui b spubuf pof dzqifs[1].

Note that in this “real life” example, the letters in the above have been changed, but spaces and punctuation marks have not. Also, capitalization has been preserved.

Considering the fact that in the computational world the letters are numbered (ASCII character code or HEX code) it is very simply to write a program which uses the Caesar ciphers. Let’s see the basic form of it:

```
CLS
INPUT "Enter the text to encrypt "; s$   * Entering the text ant he rotation step
INPUT “Enter the rotation step value <10”; pas
FOR t = 1 TO LEN(s$)                     * the encryption cycle
    c$ = c$ + CHR$(ASC(MID$(s$, t, 1)) + pas)
NEXT t↵
```

```
↳PRINT "The encrypted text is: " :PRINT c$
FOR t = 1 TO LEN(c$)     * decryption cycle of the text
    dec$ = dec$ + CHR$(ASC(MID$(c$, t, 1)) - pas)
NEXT t
PRINT "The decrypted text is ": PRINT dec$
END
```

A little better way to encrypt texts is to use a table of correspondences where for each letter in the alphabet we have a randomly chosen but unique correspondent letter like in the following table:

ENCRYPTION		DECRYPTION	
Letter	Exchanged with	Letter	Exchanged with
A	Z	Z	A
B	X	X	B
C	C	C	C
D	V	V	D
E	B	B	E
F	N	N	F
G	M	M	G
…	-----	- - - - -	…

In this case the word **DECADE** will transform in **VBCZVB**

This kind of Caesar ciphers and table of correspondences , can be easily broken by taking advantage of the known letter frequencies of the language of the encrypted text, because a certain letter will be always replaced with the same correspondent letter. E.g. in the English language the most frequent letter is **e.**

Here is the next step in encryption which consist in improving the encryption by using ciphers were the same letters are replaced with different letters each time we use them(well almost each time). The following example shows how to do this in a weary simply way:

```
CLS
 INPUT "Enter the text to encrypt"; s$
  FOR  t = 1 TO LEN(s$)
    tt = t                                    * this variable will give us the step
    IF tt > 13 THEN tt = INT(t / 13)
    c$ = c$ + CHR$(ASC(MID$(s$, t, 1)) + tt)
  NEXT t PRINT "The encrypted text is: " :PRINT c$
   FOR t = 1 TO LEN(c$)
    tt = t
 IF tt > 13 THEN tt = INT(t / 13)
```

```
    dec$ = dec$ + CHR$(ASC(MID$(c$, t, 1)) - tt)
    NEXT t
PRINT "The decrypted text is  "; dec$
END
```

The program above works this way: reads the letters one by one, replaces the current letter with a symbol at a “distance” in ASCII code corresponding to the position of the letter in the text(e.g. the first letter is replaced with the first letter ASCII code+1, the second letter is replaced with its ASCII code+2, the 3rd with its ASCII code + 3 and so one..)

Here is an output example: **Text to encode => Ug{x%zv(nxdpef .** You can see the fact that for the same letters e,t we have got different letters each time in the encode text.

This method eliminates the possibility of being broken with code breaking methods using letter frequency methods.

A much better approach, and we still talk about derivations from Caesar ciphers, is to use random steps for the rotation of each letter. We can do these using constants in the random generator engines for random seed. This way, each time we generate a series of numbers with the same random seed we will get the same numbers, so by using the same seed in the random function for booth the encryption ant the decryption the problem is solved. Because the random number generator once started, we can not reinitialize the random engine ,for implementing this method ,we wrote two different programmes one for encoding the text and putting it in a file and a second program which reads the encoded text from the file and after decoding it prints the result on the screen. Attention !! The two programs must be started separately, from different sessions. Here are the source codes for this method:

```
CLS
OPEN "code.txt" FOR OUTPUT AS #1 open the file code.txt to write in the coded text
codat$ = ""                      initialization of the string variable
INPUT "Enter the text to encrypt"; text$
RANDOMIZE( 3 )                   starting the random number generator with a certain seed
FOR t = 1 TO LEN(text$)          starting encoding the text
  pas = INT(1 + 10 * RND)        the randomly generated step
  codat$ = codat$ + CHR$(ASC(MID$(text$, t, 1)) + pas)      exchanging the letters
NEXT t
PRINT #1, codat$                 printing the encoded text in to  the code.txt file on the disk
CLOSE                            closing the file
END
```

And next the decoding program :

```
CLS
```

OPEN "code.txt" FOR INPUT AS #1 ↵ Opens the file that contains the encoded
↳ **INPUT #1, codat$** read the text from the file in a string variable
decodat$ = "" initialization of the output string variable
RANDOMIZE(3) !!! The same random seed to initialize de number generator
FOR t = 1 TO LEN(codat$) Starting decoding the text
pas = INT(1 + 10 * RND) the randomly generated step
decodat$ = decodat$ + CHR$(ASC(MID$(codat$, t, 1)) - pas) exchanging the letters
NEXT t
PRINT decodat$ outputting the result to the screen
CLOSE closing the file
END

This version of encoding program will give the following result in an output example: Input text **Text to encode** output text: **Vn y){u'hpmsgh**

This is a pretty good method considering that we can use more than 60000 different seed numbers, so even if someone knows our method they will have to try a relatively big number of possibilities. Also there is a possibility to use different seed number for encoding different text by writing only 1 new line in our program and modifying another one, in both programmes (encoding and decoding) :

INPUT "Enter the random number initialization seed ";ssd read the seed number
RANDOMIZE(ssd)

Here I described just a few examples of encryption methods . The military and commercial encoding, encrypting methods and programmes are much more sophisticated and use the advantage of strong mathematical fundaments and theories.

But no matter how good is an encryption method as long as it use a certain method to encrypt, just knowing how the encryption is done its enough for a codbraker to decrypt a message. So even a method as shown above can be useful as long as the "enemies" doesn't know how we encrypt our text.

Bibliography

[1] Douglas W. Jones Data Compression and Encryption Algorithms . University of Iowa

Author

Arpad Incze "1 Decembrie 1918" University of Alba Iulia. Romania, Department of Mathematics and Computer Science, glider@personal.ro