

Re: Encryption algorithm

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2004-01/0317.html>

From: Peter E.C. Dashwood (*dashwood_at_enternet.co.nz*)

Date: 01/07/04

Date: Thu, 8 Jan 2004 10:40:08 +1300

"Vik Mehta" <aul1231@yahoo.com> wrote in message
news:2df8b8ca.0401070700.67c3ec95@posting.google.com...
> *I am looking for a simple encryption algorithm to encrypt a 13*
> *character alphanumeric string in a COBOL program. Any help will be*
> *appreciated.*
>

Presumably you want the encryption as a COBOL Subroutine?

You will need a COBOL compiler that supports the XOR function.

(Microfus and Fujitsu do.)

"Here's one I prepared earlier..." (MUCH earlier...over 6 years ago, in fact. It works extremely well despite being simple, and is ideal for short strings that need encryption before writing to a database or file (like passwords).)

Remove anything in square brackets "[...]" before compiling.

IDENTIFICATION DIVISION.

PROGRAM-ID. 'HERACrpt'.

*AUTHOR. Peter E. C. Dashwood.

* This program encrypts/decrypts an 8 byte password.

*

*DATE_WRITTEN. August 1997.

ENVIRONMENT DIVISION.

configuration section.

source-computer. IBM-PC.

object-computer. IBM-PC.

*----- DATA DIVISION -----

DATA DIVISION.

WORKING-STORAGE SECTION.

01 encryption-key pic x(8) value 'ANYTHING'.

[You can set this key to be any character string you like. My applications do not use "ANYTHING"... It must be the same length as the field you wish to

comp.lang.cobol: Re: Encryption algorithm

encrypt, so in your case you need to make it 13 bytes and set a value in it.
(I suggest: ANYTHINGULIKE while you are testing it...<G> This encryption
key can be any combination of characters you like. You don't have to
remember it, but you should make sure that the source code of your
encryption module is not freely available, for obvious reasons. (You could
save it as an encrypted WORD document <G>)]

```
01 xor-return-code pic s9(4) comp-5.
01 hex-table.
  12 filler pic x value '0'.
  12 filler pic x value '1'.
  12 filler pic x value '2'.
  12 filler pic x value '3'.
  12 filler pic x value '4'.
  12 filler pic x value '5'.
  12 filler pic x value '6'.
  12 filler pic x value '7'.
  12 filler pic x value '8'.
  12 filler pic x value '9'.
  12 filler pic x value 'A'.
  12 filler pic x value 'B'.
  12 filler pic x value 'C'.
  12 filler pic x value 'D'.
  12 filler pic x value 'E'.
  12 filler pic x value 'F'.
01 filler redefines hex-table.
  12 hex-values pic x occurs 16
      indexed by hv-x1.

01 ws-byte-stuff.
  12 filler pic x.
  12 lod-byte pic x.
01 ws-binary redefines ws-byte-stuff pic s9(4) comp.

01 quotient pic 99.
01 rem pic 99.

01 num pic s9(5) comp-5.

01 two-bytes.
  12 hod-hex pic x.
  12 lod-hex pic x.

01 work-string.
  12 ws-char pic x occurs 16
      indexed by ws-x1.
01 xor-len pic s9(9) comp-5.
```

LINKAGE SECTION.

comp.lang.cobol: Re: Encryption algorithm

[The module takes a single parameter that includes a return code, an indication whether to ENCRYPT or DECRYPT (there's not much point encrypting stuff if you can't decrypt it...) and a passed string that will be 26 bytes in your case (It is twice the length of the string you want encrypted, as it must return each encrypted character as a hex byte.i.e. 2 bytes per byte of source)]

```
01 encryption-block.  
  12 eb-return pic x.  
    88 eb-ok value zero.  
    88 eb-bad-action value '1'.  
    88 eb-bad-char value '2'.  
  12 eb-action pic x.  
    88 HeraEncrypting value 'E'.  
    88 HeraDecrypting value 'D'.  
  12 eb-string pic x(16).
```

PROCEDURE DIVISION USING Encryption-block.

MAIN SECTION.

a000.

```
  move zero to xor-len  
  inspect eb-string  
    tallying xor-len for  
    characters BEFORE space
```

```
  evaluate TRUE  
    when HeraEncrypting  
      perform xorit  
      perform asc2hex  
    when HeraDecrypting  
      perform hex2asc  
      perform xorit  
    when other  
      set eb-bad-action to TRUE  
  end-evaluate
```

a999.

```
  exit program.
```

*-----

xorit section.

x000.

```
  CALL "CBL_XOR"  
    USING encryption-key  
      eb-string  
      BY VALUE xor-len  
      RETURNING xor-return-code  
  END-CALL  
  if xor-return-code = zero  
    set eb-ok to TRUE  
  else
```

comp.lang.cobol: Re: Encryption algorithm

```
    set eb-bad-char to TRUE
  end-if
.
x999.
  exit.
*-----
asc2hex section.
ah000.
  move eb-string to work-string
  move spaces to eb-string
  perform
    varying ws-x1
      from 1
      by 1
      until ws-x1 > xor-len
        move low-values to ws-byte-stuff
        move ws-char (ws-x1) to lod-byte
        divide 16 into ws-binary
          giving quotient
          remainder rem
        add 1 to quotient
        add 1 to rem
        move hex-values (quotient) to hod-hex
        move hex-values (rem) to lod-hex
        string
          eb-string
          delimited by space
          two-bytes
          delimited by size
          into eb-string
        end-string
      end-perform
.
ah999.
  exit.
*-----
hex2asc section.
ha000.
  move eb-string to work-string
  move spaces to eb-string
  perform
    varying ws-x1
      from 1
      by 1
      until ws-x1 > xor-len
        move low-values to ws-byte-stuff
        set hv-x1 to 1
        search hex-values
          at end set eb-bad-char to true
          go to ha999
```

comp.lang.cobol: Re: Encryption algorithm

```
      when hex-values (hv-x1) = ws-char (ws-x1)
        set num to hv-x1
        subtract 1 from num
        multiply num by 16 giving ws-binary
      end-search
      set ws-x1 up by 1
      set hv-x1 to 1
      search hex-values
        at end set eb-bad-char to true
        go to ha999
        when hex-values (hv-x1) = ws-char (ws-x1)
          set num to hv-x1
          subtract 1 from num
          add num to ws-binary
        end-search
      string
        eb-string
          delimited by space
        lod-byte
          delimited by size
          into eb-string
      end-string
    end-perform
    divide xor-len by 2 giving xor-len
  .
ha999.
  exit.

*----- END OF PROGRAM 'HERAcpt' -----
```

> *Thanks,*
> *Vik*

You're welcome...

Pete.