livecode

(Nearly) Unbreakable KeyFile Encryption

The following script can be used to encode/decode text. First, you need to generate a file of random numbers (I'll call it the "keyFile") which must be saved on each computer where you want to send or receive encrypted data. You only do this once and then save the same file to each computer. In this sample, I generate a keyFile of 10,000 random numbers. I made no provisions in the script for exceeding that amount. This file will let you encrypt 3,333 characters.

The way the encryption works is simple. You simply get the ASCII number for each character and add a random number to it. If the resulting number is greater than 999, I drop the first digit to keep all code numbers at 3 charaters. I add the first char back when I decrypt whenever the decryption would result in a negative ASCII number.

In this example, you can create a field called "myText" where you type the message to be encrypted. After running encodeMyText you can copy the encoded text and paste it into your message for transmission. At the receiving end, it is important that *just* the coded message is pasted back into the field for decryption. This is because, the first part of the message contains the starting place for using the random number file to encode/decode the data.

By incrementing the starting number each time to the place where you last left off, the same numbers in the keyFile are never used again. This is defined as a "one time pad." The encryption is unbreakable (even by NSA). To maintain the intergrity of the code, you might want to have a different random number file for each direction you will be sending encrypted messages. For example, if Bob sends a message to Jane, he might wish to encode it with keyFile A. When Jane receives the message, she will decode it because she also has a copy of keyFile A on her computer. However, if Jane wants to send a message back to Bob, she will encode it using keyFile B. Bob also has a copy of keyFile B so he can decode her message. By each user having their own keyFile for encryption, the chance of re-using numbers from one file is eliminated. (For instance, if Bob and Jane both wrote messages to each other at the same time using the same keyFile, there would be two messages in existence using the same numbers since the start numbers were not incremented sequentially.) In my sample, I used an "A" to indicate the start of the encrypted message. If Jane encrypts her message using keyFile B, she could indicate this by using a "B" instead of an "A." Your decryption handler could then note this change in order to decode the message using the proper keyFile.

If anyone is interested in reading about codes and cyphers, I recommend an excellent book by David Kahn called "The Codebreakers." There is also an excellent recent book called "The Code Book" by Simon Singh.

Hope this helps.

Philip Chumbley ----------

```
# myNumberFile is the file of random numbers which constitutes my keyFile
# myPointerFile stores the starting point for the next encryption of text

on generateNumbers
  # This handler will generate a keyFile containing 10,000 random numbers
  put empty into myNum
  repeat 10000
    put random(10) - 1 after myNum
  end repeat
  open file myNumberFile for write
  write myNum to file myNumberFile
  close file myNumberFile

  # Set the pointer to the start of the file
  put 1 into tStart
  open file myPointerFile for write
  write tStart to file myPointerFile
  close file myPointerFile
  beep
end generateNumbers

on encodeMyText
  # Takes text from field "myText" and encodes it and puts it back into the field
  put field myText into tText

  # Get the starting point
  open file myPointerFile for read
  read from file myPointerFile until eof
  close file myPointerFile
  put it into tStart
  put tStart into tStart0

  # Read in the keyFile (your random file)
  open file myNumberFile for read
  read from file myNumberFile until eof
  close file myNumberFile
  put it into myKey

  # Encrypt the text
  put empty into cipherText    # cipherText is the encyrpted message
  repeat for each char c in tText
    put char tStart to (tStart + 2) of myKey into cNum
```

```
      add 3 to tStart
      put charToNum(c) + cNum into charCodeNumber
      if the number of chars in charCodeNumber = 4 then delete char 1 of charCodeNumber   # Save only the last three digits
      if the number of chars in charCodeNumber = 1 then put "00" & charCodeNumber into charCodeNumber
      if the number of chars in charCodeNumber = 2 then put "0" & charCodeNumber into charCodeNumber
      put space & charCodeNumber after cipherText
    end repeat
    # You must pass the starting point along with the cipherText
    put tStart0 & "A" & cipherText into field "myText"

    # Save the new starting point
    open file myPointerFile for write
    write tStart to file myPointerFile
    close file myPointerFile
  end encodeMyText

  on decodeMyText
    # Takes text from field 'myText" and decodes it and puts it back into field "myText"
    put field "myText" into tText

    # Read in the key
    open file myNumberFile for read
    read from file myNumberFile until eof
    close file myNumberFile
    put it into myKey

    # Get the starting point
    get offset("A",tText)
    put char 1 to (it - 1) of tText into tStart
    delete char 1 to it of tText

    # Decode the text
    put empty into plainText
    repeat for each word i in tText
      put i into ii
      put char tStart to (tStart + 2) of myKey into cNum
      add 3 to tStart
      if cNum > ii then add 1000 to ii
      put numToChar(ii - cNum) after plainText
    end repeat
    put plainText into field "myText"


    # Save the new starting point
    open file myPointerFile for write
    write tStart to file myPointerFile
    close file myPointerFile
  end decodeMyText
```

Posted 4/9/2003 by Philip Chumbley to the MetaCard List (*See the complete post/thread*)

Print this tip

News and Rumors Products Services Developer Resources Contact STS About STS