**SOURCE CODE:**

;--- constant definitions ---;

null equ 0x00

cr equ 0x0d

lf equ 0x0a

tab equ 0x09

keyBytesRAMaddress equ 0x22 ;symbolic constant for base address of encryption key in RAM

txtRAMaddress equ 0x30 ;symbolic constant for base address of encryption key in RAM

;--- variable definitions ---;

keyvalIndex equ 0xe0 ;variable to index the keyval constant array

keylength equ 0x20 ;variable to track length o key

txtlength equ 0x21 ;variable to track length o key

charIndex equ 0xe0 ;alias for accumulator

choice equ 0x7f ;variable to store selected operation

jmp main ;jump past interrupt vector table

org 0x0030 ;put main program at rom location 0x0030

main:

;---------------- Initialization/configuration ----------------;

mov tmod, #0x20 ;config timer 1 mode 2

mov scon, #0x50 ;config serial 8-data, 1 start, 1 stop, no parity

mov th1, #0xFD ;9600 baud

setb tr1 ;start timer 1 to enable serial communication

mov r0, #keyBytesRAMaddress ;initialize RAM pointer

mov dptr, #keyVal ;initialize ROM pointer

mov keyvalIndex, #0x00 ;initialize keyvalIndex

call LoadKeyFromROM

mov dptr, #PromptPT ;set pointer to the read the string

call WriteString

call BufferText

call RotationEncrypt

call WriteBufferedText

call WaitForEnterKey

call BufferText

call RotationDecrypt

call WriteBufferedText

jmp terminate

;--- load key from ROM ---;

;receives ROM location of key array in dptr before it is called

;loads bytes from a constant array of key values into RAM

;returns nothing

LoadKeyFromROM:

mov r0, #keyBytesRAMaddress-1 ;initialize RAM pointer

mov keyvalIndex, #0x00 ;initialize accumulator

GetNextKeyByteFromROM:

inc r0 ;increment RAM pointer

push keyvalIndex ;preserve keyvalIndex variable

movc a, @a+dptr ;load byte of key into accumulator

notNull:

mov @r0, a ;put byte of key into RAM

pop keyvalIndex ;restore keyvalIndex variable

inc keyvalIndex ;increment keyvalIndex

cjne @r0, #0x00, GetNextKeyByteFromROM ;check for null terminating character

LoadDone:

mov @r0,#0x00 ;append null char to string

ret

;------- Buffer Text -------

;receives no parameters

;reads a series of TXT bytes from serial Rx

;writes the bytes to RAM at location indicated by keyBytesRAMaddress

;returns length of key in the keyLength variable

BufferText:

mov r1, #txtRAMaddress ;initialize pointer

WaitForTXTChar:

jnb ri, $ ;wait to receive char

call getChar ;char received, get it

mov @r1, a ;store character into RAM

inc r1 ;increment pointer

cjne a, #0x0D, WaitForTXTChar ;DEBUG: check for Enter char for debug

ret

;------ Rotation Encrypt ------

;receives no parameters

;encrypts the plain text contained in RAM

;returns nothing

RotationEncrypt:

mov r0, #keyBytesRAMaddress ;re-initialize key pointer

mov r1, #txtRAMaddress ;re-initialize txt pointer

RotationEncryptNextChar:

mov a, @r0 ;initialize rotate loop count

mov r6, a ;must be passed to a before r6

mov a, @r1 ;get char from plain text

rotateEncrypt:

rr a

djnz r6, rotateEncrypt

mov @r1, a ;write encrypted character bac to RAM

inc r0 ;point to next keyByte

cjne @r0, #0x00, dontResetRotationEncryptionKeyPtr

mov r0, #keyBytesRAMaddress ;re-initialize key pointer

dontResetRotationEncryptionKeyPtr:

inc r1 ;point to next plain text char

cjne @r1, #0x00, RotationEncryptNextChar

ret ;end of string reached

;------ Rotation Decrypt ------

;receives no parameters

;encrypts the plain text contained in RAM

;returns nothing

RotationDecrypt:

mov r0, #keyBytesRAMaddress ;re-initialize key pointer

mov r1, #txtRAMaddress ;re-initialize txt pointer

RotationDecryptNextChar:

mov a, @r0 ;initialize rotate loop count

mov r6, a ;must be passed to a before r6

mov a, @r1 ;get char from cipher text

rotateDecrypt:

rl a

djnz r6, rotateDecrypt

mov @r1, a ;write decrypted character back to RAM

inc r0 ;point to next keyByte

cjne @r0, #0x00, dontResetRotationDecryptionKeyPtr

mov r0, #keyBytesRAMaddress ;re-initialize key pointer

dontResetRotationDecryptionKeyPtr:

inc r1 ;point to next cipher text char

cjne @r1, #0x00, RotationDecryptNextChar

ret ;end of string reached

;------ WriteBufferedText -----

;receives address of buffered text in r1

;sends buffered text serially using writechar

;returns nothing

WriteBufferedText:

mov r1, #txtRAMaddress ;re-initialize txt pointer

NextBufChar:

mov a, @r1

call writeChar

inc r1

cjne @r1, #null, NextBufChar

mov a, @r1

call writeChar

ret

;----- Wait For Enter Key -----

;receives no parameters

;loops until keyboard Enter key press is detected

;returns nothing

WaitForEnterKey:

jnb ri, $

call getChar

cjne a, #0x0d, WaitForEnterKey

ret

;----------- getChar ----------;

;subroutine receives nothing before it is called

;writes the character to the serial console

;returns a byte in the accumulator

getChar:

mov a, sbuf ;get serial data (char)

clr ri ;acknowledge data received

ret ;return from subroutine call

;----------- writeChar ----------;

;receives byte or character

;reads a character that has been received serially

;returns the c

writeChar:

mov sbuf, a ;send data (char) serially

jnb ti, $ ;wait until data is sent

clr ti ;acknowledge data has been sent

ret ;return from subroutine call

;----- WriteString -----

;receives address of string in dptr

;sends string serially using writeChar

;returns nothing

WriteString:

mov charIndex, #0x00

NextChar:

push charIndex

movc a, @a+dptr

cjne a, #null, notNullChar

pop charIndex

ret

notNullChar:

call writeChar

pop charIndex

inc charIndex

jmp NextChar

terminate:

mov a, #0x00 ;load null character into accumulator

call writechar ;append the null character to text output

sjmp $ ;halt

org 0x200

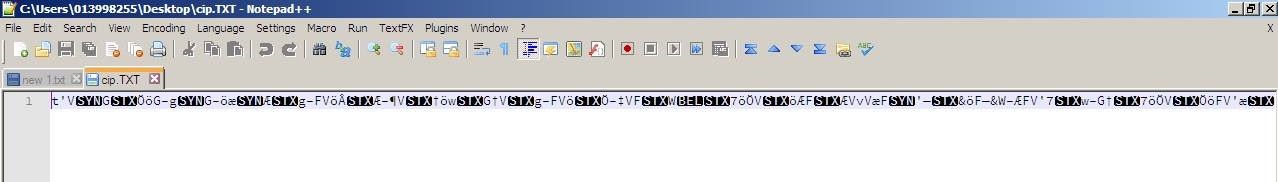
keyVal: db 0x34, 0x00

PromptPT: db "Begin the capture and send the plain.txt file", cr, lf

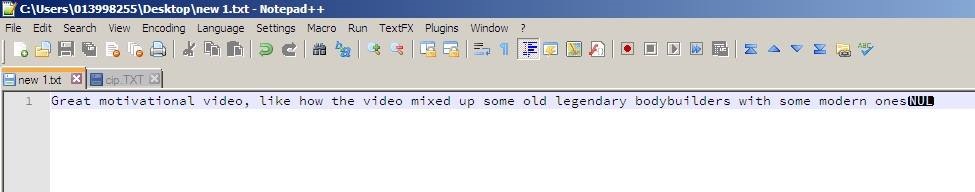
db "Stop the capture once the cipher text is displayed.", cr, lf, null

END

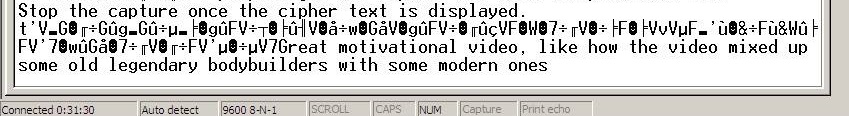
**SCREENSHOT OF CIPHER TEXT FILE:**



**SCREENSHOT OF DECRYPTED PLAIN TEXT FILE:**



**SCREENSHOT OF DECRYPTED TEXT MESSAGE IN HYPERTERMINAL:**



**MANUAL DECRYPTION:**

