CRYPT

File Encryption Program

and

CIPHER

File Encryption TACL Macro

07. June 2001

greenHouse
Software & Consulting
Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg

Document History

2nd Edition:	11Oct1999	Release CRYPT100
3rd Edition:	07Jun2001	Release CRYPT200

Please Comment

If you have questions or problems concerning the content of this document, please let me know! Send your comments to:

GreenHouse Software & Consulting
Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg
Germany

Phone	+49 (0)2363 72566
Fax	+49 (0)2363 66106
Mobile	+49 (0)172 23 18248
E-Mail:	Info@GreenHouse.de
Internet:	www.GreenHouse.de
PGP fingerprint:	3A32 D90A D125 5418 1150 2484 6629 2DD2

1. Introduction

Why should one encrypt data?

"Just because you're paranoid doesn't mean they aren't out to get you"
-- anonymous

2. CRYPT - File Encryption Program

CRYPT is a program which encrypts and decrypts files using the Data Encryption Algorithm (DEA a.k.a. DES).

2.1. INVOCATION

`CRYPT` for help

```
CRYPT /IN <infile>, OUT <outfile>/ [[$]<phrase>[!]]
```

where	`<phrase>`	= 5 to 80 character phrase (blanks allowed)
	`$`	= Shared Access
	`!`	= Batch Mode (from TACL, etc.)

`<infile>` may NOT = `<outfile>` and `<outfile>` CANNOT exist.

WARNING

1) If you forget the phrase used to encrypt a file, there is no computationally feasible way to decrypt the file.

2) If your user id changes, you will be unable to decrypt files unless they were encrypted with the $ option.

WARNING

2.2. OPTIONS

2.2.1. $ - Shared Access

If you wish to send an encrypted file to another user, use the $<phrase> option, indicating shared access. Since a file encrypted with the $ option is slightly less secure, recipients of such a file are advised to decrypt the file and reencrypt it with a new phrase. There is, however, no indication in the encrypted file that it was encrypted with the $ option, thus preventing the unscrupulous from searching for such files on a system and doing a dictionary attack.

No provision is made for transmitting the phrase to the ultimate destination. This may be done by phone, secure mail, etc.

Using the $ modifier simply means, that the crypt key, derived from the pass phrase, is NOT ‘bound’ to the users ID.

2.2.2. ! - Batch Mode

This mode is provided to enable CRYPT to be invoked from TACL macros, etc. "!" signals CRYPT to not recheck the phrase.

2.3. *INSTALLATION*

CRYPT must be named CRYPT and reside on $SYSTEM.SYSTEM.
The security vector should be “OOAO” or equivalent.
The owner should be a trusted person.

2.4. *FILE TYPES*

CRYPT encrypts any file type to an Entry-Sequenced file of code 7172.

2.5. *SECURITY SETTINGS*

Security settings are replicated. i.e. the security setting of the output file is identical to that of the input file. This is true for GUARDIAN as well as SAFEGUARD access control lists.
Reminder: In case the SAFEGUARD ACL of the IN file is PERSISTENT set to ON, then the newly created OUT file gets a PERSISTENT ACL!

2.6. *FILE ACCESS RESTRICTIONS*

Access restrictions to files are enforced by Guardian or Safeguard, whichever is active. Thus you can encrypt any file for which you have read access.

2.7. *Performance*

A K122 (32 MB memory, 1 GB disks on MFC), running the accelerated version of CRYPT, has a throughput of one MB in approximately 35 seconds.

A S7000 (128 MB memory, 8 GB disks on Servernet), running the accelerated version of CRYPT, needs about 11 seconds to process one MB.

The length of the pass phrase does not change these numbers.

2.8. *Runtime Parameters*

CRYPT does **NOT** run in a HIGH PIN, but is does accept HighRequesters.

3. CIPHER - File Encryption TACL Macro

CIPHER is an example of a TACL macro to drive CRYPT to encrypt/decrypt a file in place.
It comes along with CRYPT.

3.1. INVOCATION

`CIPHER` for help

```
CIPHER <file> [[$]<phrase>]
```

3.2. BEWARE

Since there are a few more security problems with TACL macros than there are with programs, one should be prudent and copy CIPHER into one's default volume or clearly understand the implications of one's PMSEARCHLIST.
(Beware of gifts bearing Greeks)

3.3. THOUGHTS FOR THE USER

It would be easy to build similar macros to do such things as:
- reencrypt all encrypted files in a subvolume
- encrypt all unencrypted files in a subvolume
- decrypt all encrypted files in a subvolume
- encrypt all edit type files in a subvolume
- all of the above with a template

3.4. CIPHER TACL Macro:

```
?TACL Macro
=====================================================================
==                                                                 ==
==              CRYPT (200) - T7172G06 - (07Jun2001)               ==
==    Copyright (c) GreenHouse Software & Consulting 1999,2000     ==
==                                                                 ==
=====================================================================
==                            CIPHER                               ==
==                                                                 ==
==    CIPHER <file> [[$]<phrase>]                                  ==
==                                                                 ==
==    An example TACL macro to encrypt/decrypt a file in place.    ==
==                                                                 ==
== Note: There are security problems with this macro because       ==
==       the user cannot be sure that there are no Trojan Horses.  ==
==       (Beware of gifts bearing Greeks).                         ==
==                                                                 ==
==       To be safe, a copy should also reside in the user's       ==
==       default subvolume to cover PMSEARCHLIST variations.       ==
=====================================================================
#frame
#push infile     == name of file to encrypt/decrypt
#push outfile    == output file for crypt
#push phrase     == phrase
#push phrase_sz  == length(phrase) in bytes
#push pcheck     == phrase check (used if encrypting)
#push pcheck_sz  == length(pcheck) in bytes
=====================================================================
[#def get_phrase_sz delta |body|
   Gphrase$      == get phrase
   ZJ -1D        == get rid of EOL
   ZUphrase_sz$  == save length of phrase in phrase_sz
]{end_def}
=====================================================================
[#def get_pcheck_sz delta |body|
   Gpcheck$      == get pcheck
   ZJ -1D        == get rid of EOL
   ZUpcheck_sz$  == save length of pcheck in pcheck_sz
]{end_def}
=====================================================================
[#def check_phrase delta |body| {results in 0 (false) or 1 (true)}
   Gphrase$      == get the phrase (expected)
   ZJ -1D        == get rid of EOL
   ZFUl$         == save length of phrase
   Gpcheck$      == append the phrase check string to end of buffer
   0J            == start at beginning of buffer
   I1$           == assume result will be true
   Ql$ <         == compare 1st half to 2nd half
     1+Ql$A FUc$ == save 1st char of actual in 'c'
     A-Qc$ ?N    == if 1st of expected <> 1st of actual,
       0J I0$    ==   set false
       0@;       ==   & exit compare loop
       '         == end of single character compare
     FOc$        == get rid of the temporary variable
     1C          == point to next character in expected
     >           == end of compare loop
   1,ZK          == result is in the first character
   FOl$          == get rid of the temporary variable
]{end_def}
====================================================================
#set outfile cf
[#loop |while| [#fileinfo/existence/ [outfile]] |do|
   #set outfile [outfile]x
   [#if [#match cfxxxxxx [outfile]] |then|
      #output Unable to generate outfile name
      #return
   ]
]
#set infile %1%
[#if [#emptyv infile] |then|
   #output
   #output CRYPT (200) - T7172G06 - (07Jun2001)");
   #output Copyright (c) GreenHouse Software & Consulting
1999,2001");
   #output
   #output *******************************************************
   #output *  No file specified.                                  *
   #output *  Form is:  cipher filename <<$>phrase>               *
   #output *    where   <> = optional parameters                  *
   #output *             $ = shared access                        *
   #output ********************* WARNINGS ************************
   #output *                                                      *
   #output *  If you forget the phrase used to encrypt a file,  *
   #output *  there is no computationally feasible way to       *
   #output *  decrypt the file.                                 *
   #output *                                                      *
   #output *  If your user id changes, you will be unable to    *
   #output *  decrypt files unless they were encrypted with     *
   #output *  the $ option.                                     *
   #output *                                                      *
   #output ********************* WARNINGS ************************
|else|
   [#if [#fileinfo/existence/ [infile]] |then|
      #set phrase %2% %3% %4% %5% %6% %7% %8% %9% %10%
      [#if [#emptyv phrase] |then|
         #set phrase [#input /noecho/ Enter Phrase: ]
      ]
      [#if [#compute 7172 <> [#fileinfo/code/ [infile]]] |then|
         #set pcheck [#input /noecho/ Reenter Phrase: ]
         sink [#delta /commands get_phrase_sz/]
         sink [#delta /commands get_pcheck_sz/]
         [#if [#compute [phrase_sz] <> [pcheck_sz]
         or [#delta /commands check_phrase/] = 0] |then|
            #output Phrase does not match
         |else|
            CRYPT /in [infile], out [outfile]/ [phrase]!
            [#if [#fileinfo/existence/ [outfile]] |then|
               sink [#purge [infile]]
               RENAME [outfile],[infile]
               #output [infile] encrypted
            |else|
               #output Encryption Failure
            ]
         ]
      |else|
         CRYPT /in [infile], out [outfile]/ [phrase]!
         [#if [#fileinfo/existence/ [outfile]] |then|
            sink [#purge [infile]]
            RENAME [outfile],[infile]
            #output [infile] decrypted
         |else|
            #output Decryption Failure
         ]
      ]
   |else|
      #output [infile] does not exist
   ]
]
#unframe
```