11 RACF

February 1998

In this issue

update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about RACF, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all RACF users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *RACF Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

Editor

Robert Burgess

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *RACF Update*, comprising four quarterly issues, costs £185.00 in the UK; $280.00 in the USA and Canada; £191.00 in Europe; £197.00 in Australasia and Japan; and £195.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the August 1995 issue, are available separately to subscribers for £46.50 ($70.00) each including postage.

RACF Update on-line

Code from *RACF Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Resolving -unknown- categories

We use RACF security categories to allow/disallow the use of specific log-on account codes. Each account code is defined in a user's base segment as a category entry. Each category points to a member entry in class SECDATA and profile CATEGORY.

A problem has arisen where the security administration department has deleted member entries from the CATEGORY profile before removing them from associated user-ids. This leaves -unknown- entries in the user's RACF base segment. This isn't a problem (as yet), because the user can still log-on with his other account codes. However, it looks messy in our reporting and has also caused some work when cloning the affected user-ids.

HOW TO SOLVE THIS PROBLEM

In attempting to solve this problem, we found that adding the member entry back into the profile CATEGORY doesn't work! Each member entry has a 2-byte hex number associated with it. We don't reuse these numbers; each addition uses a new number, with the last-used number stored in an HWM member.

To get round this problem you should identify duff category entries. I've used a job in SG.CONSUL.CNTL member CATEGORY to identify all the categories defined. It will:

- Look for entries matching <xxx> where xxx is a decimal number. If there are no entries like this, we have clean user-id categories.
- Note down the decimal values. We need to create member entries that use these pointer numbers.
- Get your security administration department to create member entries in the CATEGORY profile.
- Use a profile name of CATxxx, where xxx is the decimal value you've just noted down.

Next we need to patch RACF to allow users to point to the new categories.

Please note: the following should be attempted only during a quiet time.

You'll need an id that has update access to the RACF database, SYS1.RACF. The utility we're going to use reserves the database and therefore makes it impossible for other users to update or even share the RACF database (ie nobody can log-on). From TSO option 6 allocate the RACF database:

```
Command ==> ALLOC F(SYSRACF) DA('SYS1.RACF') SHR
```

Start the block update utility, ICHUT300:

```
Command ==> BLKUPD
```

You should now receive a BLKUPD prompt, 'BLKUPD:'. We need to find the RBA for the CATEGORY profile BLKUPD:

```
LOCATE CLASS(SECDATA) ENTRY(CATEGORY)
```

You'll be presented with a display, starting from the first RBA in the block. Page through the display until you find an entry name of SECDATA -CATEGORY. Note the RBA address, which is the 6-byte hex value on the right hand side, eg on the Service system it's X'0000000E7000'.

Page to the end of the display until you get the BLKUPD prompt. Ensure that you do *not* hit attention because there is no attention handling support in this utility.

We now need to read in the relevent RBA with update intent:

```
BLKUPD: READ x'rba' UPDATE
```

To list the RBA:

```
BLKUPD: LIST ALL
```

This will present you with a couple of pages of output. Look for the 'CATxxx' entries. The format of the entry is 'aabbcccc CATxxx' where:

- 'aa' is X'01'.
- 'bb' is the hex value length of a member name plus two for the pointer length.

- ‘cccc’ is 2-byte pointer number followed by CATxxx, member name.

Note the offset to cccc; you’ll get the start of the offset at the left hand side and count along. We’ll need to replace the cccc value with the hex value of xxx, ie xxx = 101cccc = X'0065'.

To replace the value:

```
BLKUPD:REP x'cccc' OFFSET(X'offset') VER(x'dddd')
```

where:

- ‘cccc’ is the hex representation of xxx
- ‘dddd’ is the value that is to be overlayed.

If you make a complete hash of it, use the following:

```
BLKUPD:REREAD
```

This will ditch the in-storage copy of the RBA and replace it with the existing physical image of the record. List out the RBA again to see if the change is OK:

```
BLKUPD: LIST range(x'offset',2)
```

To save the change:

```
BLKUPD:END SAVE
```

This will write the in-storage copy of the RBA to disk.

Enter:

```
BLKUPD:END
```

To exit the block update utility and free up the RACF database:

```
Command ==> FREE F(SYSRACF)
```

Rerun the job in SG.CONSUL.CNTL member CATEGORY to list out the defined categories <xxx> should now be replaced by CATxxx.

Get your security administration department to remove category CATxxx from all associated users. A list of those affected users can be generated from JCL held in SG.CONSUL.CNTL member CATEGOR2.

After they've actioned the above, you can remove the member entry from the profile CATEGORY.

The JCL mentioned as being in dataset SG.CONSUL.CNTL member CATEGORY is shown below. JCL runs Consul/RACF to extract and display user category information.

```
//SGCSRT   JOB (,IS),'SOFTWARE SUPPORT',CLASS=A,MSGCLASS=X,
//         NOTIFY=&SYSUID
//*
//STEP0001 EXEC PGM=CNRACF
//LICENSE  DD DISP=SHR,DSN=MV.MOCONZPX.CNRSAMP(CNRLLIC)
//SYSPRINT DD SYSOUT=*
//PRINT    DD SYSOUT=J
//SYSIN    DD *
 print nopage
 n file=print
 s s=base c=user
 define cnt count
 summary category cnt
 /*
```

Calum Reid
Senior Systems Technician (UK)

An enhanced LISTUSER command – part 2

This month we conclude the REXX EXEC, which is intended to provide the user with a structured display of the information returned by the RACF LISTUSER command.

```
Keyword= 'MSCOPE='                      /* Keyword to check       */
Call Parse_It Keyword Racfline          /* Let parse              */
romscope= Parsvar                       /* Keyword value          */
If (romscope ¬= '') Then Do             /* Keyword present?       */
   Call Check_Operparm_Parms            /* Get the operparm parm  */
   romscope= romscope||Kwords           /* Build the concatenation*/
End
Keyword= 'ROUTCODE='                    /* Keyword to check       */
Call Parse_It Keyword Racfline          /* Let parse              */
rortecde= Parsvar                       /* Keyword value          */
If (rortecde ¬= '') Then Do             /* Keyword present?       */
   Call Check_Operparm_Parms            /* Get the operparm parm  */
   rortecde= rortecde||Kwords           /* Build the concatenation*/
```



```
End
Keyword= 'STORAGE='                          /* Keyword to check       */
Call Parse_It Keyword Racfline               /* Let parse              */
rostorge= Parsvar                            /* Keyword value          */
If (rostorge ¬= '') Then Do                  /* Keyword present?       */
   Call Check_Operparm_Parms                 /* Get the operparm parm  */
   rostorge= rostorge||Kwords                /* Build the concatenation*/
End
Keyword= 'UD='                               /* Keyword to check       */
Call Parse_It Keyword Racfline               /* Let parse              */
roud= Parsvar                                /* Keyword value          */
If (roud ¬= '') Then Do                      /* Keyword present?       */
   Call Check_Operparm_Parms                 /* Get the operparm parm  */
   roud= roud||Kwords                        /* Build the concatenation*/
End
Return
                                             /* Return To caller       */
Display_user_information:
ADDRESS "ISPEXEC" "TBCREATE RACFUSER NOWRITE REPLACE"
if (rc >  4) then do                         /* call okay?             */
   say 'TBCREATE error    rc = 'rc''         /* no- inform the user    */
   call Deallocate_panel_lib                 /* Deallocate             */
   exit(Ø)                                   /* let's quit             */
end

ztdmark= ' '
ADDRESS "ISPEXEC" "VPUT    (ZTDMARK) SHARED"
tabrows=  256                                /* max table rows         */
Do i = 1 to Group_count                      /* construct group info   */
   group= ' Group= '||group.i||' Auth= '||auth.i||
    ' Revoke Date= '||revdate.i
   group= group||' Resume Date= '||resdate.i
   Call Write_Table_Record                   /* Write a table record   */
   group= ' Uacc=             '||uacc.i  /* uacc                   */
   Call Write_Table_Record                   /* Write a table record   */
   group= ' Connect Owner=    '||connown.i /* connect owner          */
   group= group||'Connect Date= '||conndte.i
   Call Write_Table_Record                   /* Write a table record   */
   If (substr(connects.i,1,2) = '  ') Then
      connects.i= ' Ø'
   connects.i= left(connects.i,8,' ')    /* make it 8 bytes long   */
   group= ' No Of Connects=  '||connects.i /* connect date          */
   group= group||'   Connect Attributes= '||connattr.i
   Call Write_Table_Record                   /* Write a table record   */
   If (substr(lastconn.i,1,7) = 'UNKNOWN') Then
      lastconn.i= ' '
   group= ' Last Conn Date=   '||substr(lastconn.i,1,6)
   group= group||'    Last Conn Time= '||substr(lastconn.i,8,8)
   Call Write_Table_Record                   /* Write a table record   */
   If (i ¬= y) Then do                       /* miss for last entry?   */
```

```
      group= ' '                          /* Yes                    */
      Call Write_Table_Record             /* Write a table record   */
   end
end
ADDRESS "ISPEXEC" "TBTOP RACFUSER"       /* position to top of tab */
if (rc ¬= Ø) then do                      /* call okay?             */
   say 'TBTOP error    rc = 'rc''        /* no-inform the user     */
   call Deallocate_panel_lib              /* Deallocate             */
   exit(Ø)                                /* let's quit             */
end

Do Forever
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
   ADDRESS "ISPEXEC" "TBDISPL RACFUSER PANEL(RACFUPN2)"
   if (rc = 8) then do                    /* exit?                  */
      ADDRESS "ISPEXEC" "REMPOP"          /* remove popup           */
      return                              /* and quit               */
   end
   if (rc > 8) then do                    /* error?                 */
      say 'TBDISPL error    rc = 'rc''    /* yes-output message     */
      call Deallocate_panel_lib           /* Deallocate             */
      exit(Ø)                             /* and quit               */
   end
   If (stso = 'y' | stso = 'Y') Then do   /* display tso info       */
      If (TSO = 'No') Then do             /* TSO Segment?           */
         ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
         Racferrm= 'TSO Segment Not Available'
         ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN5)"
         ADDRESS "ISPEXEC" "REMPOP"       /* remove popup           */
         ADDRESS "ISPEXEC" "REMPOP"       /* remove popup           */
         Iterate                          /* back again             */
      end
      ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
      ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN3)"
      if (rc > 8) then do                 /* error?                 */
        say 'DISPLAY error    rc = 'rc'' /* yes- output message    */
        call Deallocate_panel_lib         /* Deallocate             */
        exit(Ø)                           /* and quit               */
      end
      ADDRESS "ISPEXEC" "REMPOP"          /* remove popup           */
   End
   If (sdfp = 'y' | sdfp = 'Y') Then do   /* display dfp info       */
      If (DFP = 'No') Then do             /* DFP Segment?           */
         ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
         Racferrm= 'DFP Segment Not Available'
         ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN5)"
         ADDRESS "ISPEXEC" "REMPOP"       /* remove popup           */
         ADDRESS "ISPEXEC" "REMPOP"       /* remove popup           */
         Iterate                          /* back again             */
      end
```

```
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
   ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN4)"
   if (rc > 8) then do                /* error?                 */
     say 'DISPLAY error    rc = 'rc'' /* yes-output message     */
     call Deallocate_panel_lib        /* Deallocate             */
     exit(Ø)                          /* and quit               */
   end
   ADDRESS "ISPEXEC" "REMPOP"         /* remove popup           */
End
If (snetview = 'y' | snetview = 'Y') Then do /* Netview Segment */
   If (NETVIEW = 'No') Then do        /* NETVIEW Segment?       */
      ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
      Racferrm= 'Netview Segment Not Available'
      ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN5)"
      ADDRESS "ISPEXEC" "REMPOP"      /* remove popup           */
      ADDRESS "ISPEXEC" "REMPOP"      /* remove popup           */
      Iterate                         /* back again             */
   end
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
   ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN6)"
   if (rc > 8) then do                /* error?                 */
     say 'DISPLAY error    rc = 'rc'' /* yes-output message     */
     call Deallocate_panel_lib        /* Deallocate             */
     exit(Ø)                          /* and quit               */
   end
   ADDRESS "ISPEXEC" "REMPOP"         /* remove popup           */
End
If (scics = 'y' | scics = 'Y') Then do /* CICS Segment          */
   If (CICS = 'No') Then do            /* CICS Seg?             */
      ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
      Racferrm= 'CICS Segment Not Available'
      ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN5)"
      ADDRESS "ISPEXEC" "REMPOP"      /* remove popup           */
      ADDRESS "ISPEXEC" "REMPOP"      /* remove popup           */
      Iterate                         /* back again             */
   end
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
   ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN7)"
   if (rc > 8) then do                /* error?                 */
     say 'DISPLAY error    rc = 'rc'' /* yes-output message     */
     call Deallocate_panel_lib        /* Deallocate             */
     exit(Ø)                          /* and quit               */
   end
   ADDRESS "ISPEXEC" "REMPOP"         /* remove popup           */
End
If (soperprm = 'y' | soperprm = 'Y') Then do /* Operparm Seg    */
   If (OPERPARM = 'No') Then do       /* No OPERPARM Segment    */
      ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
      Racferrm= 'OPERPARM Segment Not Available'
      ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN5)"
```

```
         ADDRESS "ISPEXEC" "REMPOP"      /* remove popup          */
         ADDRESS "ISPEXEC" "REMPOP"      /* remove popup          */
         Iterate                         /* back again            */
      end
      ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
      ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN8)"
      if (rc > 8) then do                /* error?                */
        say 'DISPLAY error    rc = 'rc'' /* yes-output message    */
        call Deallocate_panel_lib        /* Deallocate            */
        exit(Ø)                          /* and quit              */
      end
      ADDRESS "ISPEXEC" "REMPOP"         /* remove popup          */
   End
   ADDRESS "ISPEXEC" "REMPOP"            /* remove popup          */
End
return

Write_Table_Record:
ADDRESS "ISPEXEC" "TBADD RACFUSER        /* add the entries       */
         SAVE(group)
         MULT("TABROWS")"
if (rc ¬= Ø) then do                     /* call okay?            */
   say 'TBADD error    rc = 'rc''        /* no-inform the user    */
   call Deallocate_panel_lib             /* Deallocate            */
   exit(Ø)                               /* let's quit            */
end
Return

Get_the_userid:
ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(1)"
ADDRESS "ISPEXEC" "DISPLAY PANEL(RACFUPN1)"
if (rc = 8) then                         /* exit?                 */
   Return                                /* Return to caller      */
if (rc > 8) then do                      /* error?                */
   say 'DISPLAY error    rc = 'rc''      /* yes-output message    */
   call Deallocate_panel_lib             /* Deallocate            */
   exit(Ø)                               /* and quit              */
end

ADDRESS "ISPEXEC" "REMPOP"               /* remove popup          */
Return
Parse_It: Procedure Expose Keyword Racfline parsvar therest
parsvar= ''                              /* init                  */
Parse Var racfline (Keyword) parsvar therest
Return                                   /* Return To Caller      */

Check_Netview_Parms: Procedure Expose Therest Kwords
Kwords= ''                               /* Set To Nulls          */
Word_Count= Words(Therest)               /* Number Of Words       */
If (Word_Count= Ø) then                  /* Any words?            */
```

```
   Return                                  /* No-                    */
Do i= 1 to Word_Count                      /* Scan The Words         */
   The_Word= Word(Therest,i)               /* Pick up the next word  */
   If (The_Word = 'CONSNAME=') Then        /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'MSGRECVR=') Then        /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'CTL=') Then             /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'NGMFADMN=') Then        /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'OPCLASS=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'DOMAINS=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'IC=') Then              /* Keyword?               */
      Return                               /* yes-                   */
   Kwords= Kwords||' '||The_word           /* Build it up            */
End
Return                                     /* Return To Caller       */

Check_Cics_Parms: Procedure Expose Therest Kwords
Kwords= ''                                 /* Set To Nulls           */
Word_Count= Words(Therest)                 /* Number Of Words        */
If (Word_Count= Ø) then                    /* Any words?             */
   Return                                  /* No-                    */
Do i= 1 to Word_Count                      /* Scan The Words         */
   The_Word= Word(Therest,i)               /* Pick up the next word  */
   If (The_Word = 'OPCLASS=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'OPIDENT=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'OPPRTY=') Then          /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'TIMEOUT=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   If (The_Word = 'XRFSOFF=') Then         /* Keyword?               */
      Return                               /* yes-                   */
   Kwords= Kwords||' '||The_word           /* Build it up            */
End
Return                                     /* Return To Caller       */

Check_Operparm_Parms: Procedure Expose Therest Kwords
Kwords= ''                                 /* Set To Nulls           */
Word_Count= Words(Therest)                 /* Number Of Words        */
If (Word_Count= Ø) then                    /* Any words?             */
   Return                                  /* No-                    */
Do i= 1 to Word_Count                      /* Scan The Words         */
   The_Word= Word(Therest,i)               /* Pick up the next word  */
   If (The_Word = 'ALTGRP=') Then          /* Keyword?               */
```

```
      Return                              /* yes-                  */
   If (The_Word = 'AUTH=') Then           /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'AUTO=') Then           /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'CMDSYS=') Then         /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'DOM=') Then            /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'KEY=') Then            /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'LEVEL=') Then          /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'LOGCMDRESP=') Then     /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'MFORM=') Then          /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'MIGID=') Then          /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'MONITOR=') Then        /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'MSCOPE=') Then         /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'ROUTECDE=') Then       /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'STORAGE=') Then        /* Keyword?              */
      Return                              /* yes-                  */
   If (The_Word = 'UD=') Then             /* Keyword?              */
      Return                              /* yes-                  */
   Kwords= Kwords||' '||The_word          /* Build it up           */
End
Return                                    /* Return To Caller      */

Allocate_panel_lib:
/*.............................................................*/
/*  Allocate the Panel and Message Library                     */
/*.............................................................*/

address "ISPEXEC" "LIBDEF ISPPLIB DATASET ID('SHTSØØ1.SHL.LIB.PANELS')"
Return

Deallocate_panel_lib:
/*.............................................................*/
/*  Deallocate the Panel and Message Library                   */
/*.............................................................*/

address "ISPEXEC" "LIBDEF ISPPLIB"        /* remove allocation     */
Return                                    /* Return To Caller      */
```

RACFUPN1 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(55,7)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
# Userid: *z      #
+
+ @z                                              #
+
)INIT
.ZVARS= '(ruserid,rmess001)'
&ZCMD=   ' '
.CURSOR= ruserid
&ZWINTTL= 'RACF Userid Panel'
)REINIT
&ZCMD=    ' '
.CURSOR= ruserid
&ZWINTTL= 'RACF Userid Panel'
)PROC
)END
```

RACFUPN2 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(WHITE)
)BODY WINDOW(76,22)
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
# Userid:       *z      # Name:*z                  # Create Date:*z
#
# Owner:        *z       #Default Group:    *z       #
# Passdate:     *z     #  Password Interval:*z   #
# Revoke Date:  *z     #  Resume Date:      *z     #
# Last Acc/Date:*z       #Last Access Time: *z       #
```

```
+
# Attributes:            *z                                  #
# Class Authorizations:  *z                                        #
# Installation Data:     *z
#
# Model Name:            *z
#
# ---------------------------------------------------------------------
# |TSO Segment:  _z#   Netview Segment:  _z#   DFP Segment:      _z#|
# |CICS Segment: _z#   OPERPARM Segment: _z#                        |
# ---------------------------------------------------------------------
# Group Information:
)MODEL
@z
#
)INIT
.ZVARS= '(ruser,rname,rcrdate,rowner,rdefgrp,rpdate,rpint,rrevdate, +
rresdate,rlastdte,rlasttme,rattr,rclasaut,rinstdta,rmodnme,stso,snetview,
+
sdfp,scics,soperprm,group)'
&ZCMD=     ' '
&stso=     'n'
&snetview= 'n'
&sdfp=     'n'
&scics=    'n'
&soperprm= 'n'
&sworkatr= 'n'
.CURSOR= stso
&ZWINTTL= 'RACF User Display'
)REINIT
&ZCMD=     ' '
&stso=     'n'
&snetview= 'n'
&sdfp=     'n'
&scics=    'n'
&soperprm= 'n'
&sworkatr= 'n'
.CURSOR= stso
&ZWINTTL= 'RACF User Display'
)PROC
)END
```

RACFUPN3 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
```

```
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(64,13)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
# Account Number:      *z                                        #
# Hold Class:          *z#
# Job Class:           *z#
# Message Class:       *z#
# Procedure Name:      *z       #
# Region Size:         *z       #
# Maximum Region Size: *z       #
# Sysout Class:        *z#
# Unit:                *z       #
# Userdata:            *z       #
)INIT
.ZVARS= '(racctnum,rhclass,rjclass,rmclass,rproc,rsize,rmsize,rsclass, +
runit,rudata)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF TSO Information'
)REINIT
&ZCMD=    ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF TSO Information'
)PROC
)END
```

RACFUPN4 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(55,7)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
# Management Class:    *z       #
# Storage Class:       *z       #
# Data Class:          *z       #
# Data Appl:           *z       #
)INIT
```

```
.ZVARS= '(rmgmtcls,rstorcls,rdatacls,rdatappl)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF DFP Segment'
)REINIT
&ZCMD=    ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF DFP Segment'
)PROC
)END
```

RACFUPN5 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(WHITE)
* TYPE(TEXT)    COLOR(YELLOW)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(BLUE)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(YELLOW)
)BODY WINDOW(38,5)
%
%COMMAND ===>_ZCMD      %
%
%  @racferrm                        %
%
)INIT
&ZCMD= ' '
&ZWINTTL=  'RACF Error Message Panel'
)REINIT
&ZCMD= ' '
)PROC
)END
```

RACFUPN6 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(63,1Ø)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
```

```
+
# Consname:   *z       #
# CTL:        *z       #
# MSGRECVR:   *z       #
# NGMFADMN:   *z       #
# Domains:    *z                                              #
# IC:         *z                                              #
# Opclass:    *z                                              #
)INIT
.ZVARS= '(rconsnme,rctl,rmsgrcvr,rngmfadm,rdomains,ric,rnopclas)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF Netview Segment'
)REINIT
&ZCMD=    ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF Netview Segment'
)PROC
)END
```

RACFUPN7 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(63,1Ø)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
+?z                              #
+
# Opclass:    *z                                              #
# Opident:    *z  #
# Opprty:     *z    #
# Timeout:    *z   #
# XRFSOFF:    *z      #
)INIT
.ZVARS= '(rclstext,rcopclas,rcopidnt,rcopprty,rctimout,rcxrfsof)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF CICS Segment'
)REINIT
&ZCMD=    ' '
.CURSOR= ZCMD
```

```
&ZWINTTL= 'RACF CICS Segment'
)PROC
)END
```

RACFUPN8 ISPF PANEL

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
* TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(68,18)
+
+ COMMAND ===>_ZCMD          +    SCROLL ===>_AMT   +
+
# Altgrp:        *z       #
# Auth:          *z                                  #
# Auto:          *z  #
# CMDSYS:        *z       #
# DOM:           *z     #
# Key:           *z       #
# Level:         *z                                  #
# Log Cmdresp:   *z     #
# MFORM:         *z             #
# MSGID:         *z  #
# Monitor:       *z                        #
# MSCOPE:        *z                                                  #
# ROUTCODE:      *z                                                  #
# Storage:       *z   #
# UD:            *z  #
)INIT
.ZVARS= '(roaltgrp,roauth,roauto,rocmdsys,rodom,rokey,rolevel,rolcresp,
+
romform,romigid,romonitr,romscope,rortecde,rostorge,roud)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF OPERPARM Segment'
)REINIT
&ZCMD=    ' '
.CURSOR= ZCMD
&ZWINTTL= 'RACF OPERPARM Segment'
)PROC
)END
```

R F Perretta
Senior Systems Programmer (UK)

An ISPF interface for RACF group connects

This is an ISPF table-based interface for managing RACF groups and the users connected to those groups. Essentially, it reformats the output from the LISTGRP command and allows ISPF generated commands such as CONNECT and REMOVE to manipulate the connected users. It also allows user details to be displayed from the connected user list.

This needs no system changes. To install, just copy the REXX to a SYSPROC library, the panels to an ISPPLIB library, and the message to ISPMLIB. Add the initial REXX (RACFGRP) to a selection menu. No programs are called by the REXX, other than the IBM commands, so no exposures are introduced.

These tables can be scrolled, and entries selected from the list for further display of, or changes or additions to, the list of connected users. Some sample displays are reproduced here; however, the interface is very intuitive to anyone familar with ISPF (and particularly option 3.4).

The first display, as shown in Figure 1, requests the input of a RACF group name. The LISTGRP command is executed and the list of user-ids that it generates is then trapped and displayed as an ISPF selection

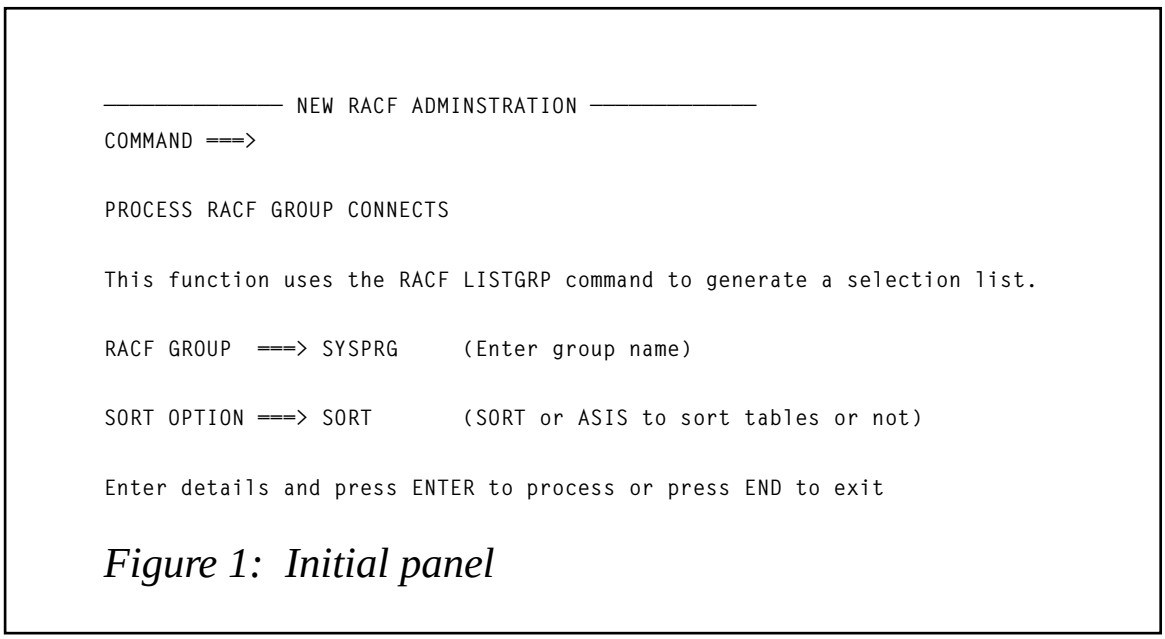

Figure 1: Initial panel

table.

Having entered the group name, pressing ‘enter’ will generate the list of user-ids connected to that group, with their attributes. An example

```
──────────── NEW RACF CONNECT DISPLAY ─── ROW 1 TO 13 OF 13
COMMAND ==>                                              SCROLL ==> CSR

GROUP  : SYSPRG       (Enter command L xx to position display to id xx)
DESC   : UNIT: SYSTEMS PROGRAMMING

SELECT : S DISP USER  C CHANGE CONNECT  R REMOVE CONNECT  A ADD NEW CONNECT

S ID       AUTH    COUNT   GROUP ATTR     REVOKE DATE      RESUME DATE
──────────────────────────────────────

  PEBW     USE     000000  NONE           NONE               NONE
  PKL      USE     000000  SPECIAL        NONE               NONE
  PAN      JOIN    000000  NONE           NONE               NONE
  PSXT     USE     000009  NONE           DECEMBER 12, 1999  NONE
  PRN      USE     000000  NONE           NONE               NONE
  Q061     USE     000000  NONE           NONE               NONE
  Q155     USE     000000  NONE           NONE               NONE
  Q625     CREATE  000000  AUDIT          NONE               NONE
  Q903     USE     000000  NONE           NONE               NONE
  Q961     USE     000000  NONE           NONE               NONE
  X071     USE     000000  NONE           NONE               NONE
  X066     USE     000000  NONE           NONE               NONE
  X335     USE     000000  NONE           NONE               NONE
****************************** BOTTOM OF DATA ********************************
```

Figure 2: Example list of user-ids connected

display of a list of user-ids connected to a group profile is shown in Figure 2.

To connect or remove users, the selection characters A or R can now be entered against existing user-ids. This will result in a CONNECT or REMOVE command being formatted and issued by the REXX

```
———————————— NEW RACF ADMINSTRATION ————————————
COMMAND ===>

ADD RACF CONNECT
This function will issue a CONNECT command.
This will add the user to group SPROGDEV with the options below:

GROUP   :   SYSPRG

ID     ===> X444          (Enter NEW user-id to be connected)

AUTH   ===> USE           (Authority level, USE/CREATE/CONNECT/JOIN)
ATTR   ===> NONE          (NONE for none)
                          (Enter any special attributes such as GRPACC)
REVOKE ===> YES           (YES or NO to revoke user connect, Y/N)
Optionally:
DATE   ===> Ø1 / 12 / 99  (DD / MM / YY to revoke access, blank if no revoke)

RESUME ===> NO            (YES or NO to resume user connect, Y/N)
Optionally:
DATE   ===>    /    /     (DD / MM / YY to resume access, blank if no resume)

Enter details and press ENTER to process or press END to exit
```

Figure 3: Adding a user to a group profile

code. Note that date, range, and attribute parameters can be issued for the connect. An example display of adding a user to a group profile is shown in Figure 3.

To display a user's details, the S characters can be entered against the user-ids. This issues a LISTUSER command with a display such as is shown in Figure 4.

Please note that changes to the RACF commands' output format can affect the trapping and reformatting very slightly. The code provided has been tested on RACF 1.8 and 1.9. It has not been tested on RACF 2.1, but any changes should be only very minor, if needed at all.

```
———————————— THE RACF USER DISPLAY ————————————
COMMAND ===>

DISPLAY RACF USERID INFORMATION

USERID         :  X625             RACF userid

NAME OF USER   :  SMITH J          Name as recorded

PASSWORD DATE  :  97.266           Date of last password change

DEFAULT GROUP  :  SYSTEMS          Default RACF group

CONNECT GROUPS :  SERVICE1  SYSPROG  DEVELOP1  IMS62  EMAIL

JOBCLASS DFT   :  A                Used by SUBMIT command if no jobcard
MSGCLASS DFT   :  X                Used by SUBMIT command if no jobcard

CURRENT ACCOUNT:  ABCDEF#    At last logon, can be changed during logon
CURRENT PROC   :  £TSOPROC   At last logon, can be changed during logon

Press END to exit
```

Figure 4: Example of LISTUSER selections

SOURCE OF RACFGRP REXX

```
/********************************************************* REXX ****/
/* RACF GROUP CONNECT RULE PROCESSING REXX                        */
/*                                                                */
/* A PANEL IS SHOWN TO REQUEST SPECIFICATION OF A GROUP NAME      */
/* THEN THE RACF LISTGRP COMMAND IS ISSUED AND THE RESULTS        */
/* TRAPPED TO PROVIDE A TABLE OF USER-IDS. OPTIONS ALLOWED        */
/* AGAINST THIS TABLE ARE ADD/CHANGE/DELETE CONNECT.              */
/* MULTIPLE SELECTIONS ARE SUPPORTED AS IS THE CONCURRENT         */
/* USE OF THIS REXX ON BOTH SIDES OF A SPLIT SCREEN.              */
/******************************************************************/
ARG 'GROUP(' RGROUP ')'

ADDRESS ISPEXEC "VGET (ZSCREEN) ASIS"
ADDRESS TSO "PROFILE MSGID"
STATUS = Ø
XS = ZSCREEN
ONCE = Ø
TBLA = 'RCFRA'XS
```

```
IF RGROUP ¬= '' THEN
   DO
      ADDRESS ISPEXEC "CONTROL NONDISPL ENTER"
      ONCE = 1
   END

ADDRESS ISPEXEC "DISPLAY PANEL(RACFGRP1)" /* REQUEST GROUP */
PNRRC = RC
DO WHILE (PNRRC = Ø)  /* WHILE NOT END PANEL */
   GRP = RGROUP
   CALL DISG
   IF ONCE = 1 THEN
      LEAVE
   ADDRESS ISPEXEC "DISPLAY PANEL(RACFGRP1)" /* REQUEST GROUP */
   PNRRC = RC
END
EXIT

/* DISPLAY GROUP CONNECTS */
DISG:
ADDRESS ISPEXEC "TBCREATE" TBLA "KEYS(ID)",
        "NAMES(AUTH CNT ATR REV RES) REPLACE NOWRITE"

FLAGU = 'OFF'
INSDATA = ' '
X = OUTTRAP('VAR.')
ADDRESS TSO "LISTGRP" GRP
X = OUTTRAP('OFF')
DO I = 1 TO VAR.Ø WHILE (FLAGU ¬= 'END')   /* SCAN LISTGRP OUTPUT */
   TEMP = VAR.I
   L = LENGTH(TEMP)
   IF INSDATA = ' ' THEN
      IF SUBSTR(TEMP,5,17) = 'INSTALLATION DATA' THEN
         DO
            INSDATA = SUBSTR(TEMP,23)
            I    = I + 1
            TEMP = VAR.I
            L    = LENGTH(TEMP)
            IF SUBSTR(TEMP,5,17) = ' ' THEN
               INSDATA = INSDATA || STRIP(TEMP)
         END

   IF FLAGU = 'ON' & L = 1 THEN
      FLAGU = 'END'
   IF FLAGU = 'ON' THEN       /* DURING CONNECT LIST PART */
      DO
         ID   = SUBWORD(TEMP,1,1)
         AUTH = SUBWORD(TEMP,2,1)
         CNT  = SUBWORD(TEMP,3,1)
         I    = I + 1
```

```
         TEMP = VAR.I
         PARSE VAR TEMP 'ATTRIBUTES=' ATR ' '
         I    = I + 1
         TEMP = VAR.I
         L    = LENGTH(TEMP)
         PARSE VAR TEMP 'REVOKE DATE=' REV 'RESUME'
         PARSE VAR TEMP 'RESUME DATE=' RES
         ADDRESS ISPEXEC "TBMOD" TBLA    /* ADD ENTRY TO TABLE */
      END

   IF SUBSTR(TEMP,5,7) = 'USER(S)' THEN
      FLAGU = 'ON'         /* START OF CONNECT LIST */
   IF SUBSTR(TEMP,1,9) = 'ICH51003I' THEN
      ID = 'INVALID'
   IF SUBSTR(TEMP,1,9) = 'ICH32002I' THEN
      ID = 'NOTALLOW'
   IF SUBSTR(TEMP,5,18) = 'NO USERS' THEN
      DO
         ID   = 'NONE'
         AUTH = 'NONE'
         RES  = 'NONE'
         REV  = 'NONE'
         CNT  = '000000'
         ATR  = ' '
         ADDRESS ISPEXEC "TBMOD" TBLA    /* ADD ENTRY TO TABLE */
      END
END

IF ID = 'INVALID' THEN      /* INVALID GROUP */
   DO
      CALL EMSG 'ERR05'
      ADDRESS ISPEXEC "TBEND" TBLA
      RETURN
   END
IF ID = 'NOTALLOW' THEN     /* NOT ALLOWED */
   DO
      CALL EMSG 'ERR06'
      ADDRESS ISPEXEC "TBEND" TBLA
      RETURN
   END

/* CONNNECT TABLE DISPLAY SECTION */
FOUND = 'NO'
OPTA  = ' '
IF RSORT = 'SORT' THEN
   ADDRESS ISPEXEC "TBSORT" TBLA "FIELDS(ID,C,A)"
ADDRESS ISPEXEC "TBTOP"   TBLA
ADDRESS ISPEXEC "TBDISPL" TBLA "PANEL(RACFGRP4)"
RETA = RC
DO WHILE (RETA < 8)  /* WHILE NOT END ON TABLEB PANEL */
```

```
   ADDRESS ISPEXEC "CONTROL DISPLAY SAVE"
   IF SUBSTR(ZCMD,1,2) = 'L ' THEN
      DO
         SCAN = SUBSTR(ZCMD,3)
         CALL LOCU TBLA, SCAN     /* POSITION TABLE */
      END

   OPTA = TRANSLATE(OPTA)         /* UPPER CASE IT */
   SELECT
      WHEN (OPTA = 'A') THEN
        CALL ADDC TBLA, ID, AUTH
      WHEN (OPTA = 'C') THEN
        CALL CHGC TBLA, ID, AUTH
      WHEN (OPTA = 'R') THEN
         CALL DELC TBLA, ID
      WHEN (OPTA = 'S') THEN
         CALL DISU ID
      OTHERWISE NOP
   END

   DO WHILE (RETA = 4)  /* PROCESS MULTI SELECTION */
      ADDRESS ISPEXEC "CONTROL DISPLAY RESTORE"
      ADDRESS ISPEXEC "TBDISPL" TBLA
      RETA = RC
      ADDRESS ISPEXEC "CONTROL DISPLAY SAVE"
      OPTA = TRANSLATE(OPTA)         /* UPPER CASE IT */
      SELECT
         WHEN (OPTA = 'A') THEN
           CALL ADDC TBLA, ID, AUTH
         WHEN (OPTA = 'C') THEN
           CALL CHGC TBLA, ID, AUTH
         WHEN (OPTA = 'R') THEN
            CALL DELC TBLA, ID
         WHEN (OPTA = 'S') THEN
            CALL DISU ID
         OTHERWISE NOP
      END
   END

   OPTA = ' '
   IF FOUND = 'NO' THEN
      DO
         ADDRESS ISPEXEC "TBTOP" TBLA
         ADDRESS ISPEXEC "TBSKIP"  TBLA "NUMBER("ZTDTOP")"
      END
   FOUND = 'NO'
   ADDRESS ISPEXEC "CONTROL DISPLAY RESTORE"
   ADDRESS ISPEXEC "TBDISPL" TBLA "PANEL(RACFGRP4)"
   RETA = RC
END
```

```
ADDRESS ISPEXEC "TBEND" TBLA
RETURN

/* LOCATE USER IN TABLE */
LOCU:
ARG TBLA, SCAN
ADDRESS ISPEXEC "TBTOP" TBLA  /* START FROM TOP ROW */
ADDRESS ISPEXEC "TBSKIP" TBLA
LRC = RC
DO WHILE (LRC = Ø)    /* LOOP THROUGH THE TABLE ROWS */
   L2 = LENGTH(SCAN)
   IF SUBSTR(ID,1,L2) = SCAN THEN
      LRC = 99
   IF LRC ¬= 99 THEN
      DO
         ADDRESS ISPEXEC "TBSKIP" TBLA
         LRC = RC
      END
END

IF LRC = 99 THEN
   FOUND = 'YES'         /* TABLE NOW POSITIONED TO ROW */
ELSE
   CALL EMSG 'ERRØ4'     /* NOT FOUND */
RETURN
/* CHANGE CONNECT OPTION */
CHGC:
ARG TBLA, ID, AUTH
IF ID = 'NONE' THEN
   RETURN
ADDRESS ISPEXEC "DISPLAY PANEL(RACFGRP2)"
IF RC > Ø THEN
   RETURN
IF ATR = 'NONE' THEN
   ATR = ' '
PARSE VAR ATR P1 'REVOKED' P2
ATR = P1 P2
PARSE VAR ATR P1 '* UPDATED *' P2
ATR = P1 P2
REV = ' '
IF RVO = 'YES' THEN
   DO
      IF REVDD ¬= ' ' THEN
         DO
            REV  = REVMM'/'REVDD'/'REVYY
            REVC = 'REVOKE('REV')'
         END
      ELSE
         REVC = 'REVOKE'
   END
```

```
ELSE
   REVC = ' '
RES = ' '
IF RSO = 'YES' THEN
   DO
      IF RESDD ¬= ' ' THEN
         DO
            RES  = RESMM'/'RESDD'/'RESYY
            RESC = 'RESUME('RES')'
         END
      ELSE
         RESC = 'RESUME'
   END
ELSE
   RESC = ' '
ADDRESS TSO 'CONNECT' ID 'GROUP('GRP') AUTH('AUTH')' ATR REVC RESC
IF RC = Ø THEN
   DO
      ATR = '* UPDATED *'
      ADDRESS ISPEXEC "TBMOD" TBLA    /* UPDATE TABLE */
      IF RSORT = 'SORT' THEN
         ADDRESS ISPEXEC "TBSORT" TBLA "FIELDS(ID,C,A)"
   END
ELSE
   CALL EMSG 'ERRØ1'      /* CONNECT FAILED */
RETURN

/* ADD CONNECT OPTION */
ADDC:
ARG TBLA, ID, AUTH
NEW = 'NO'
IF ID = 'NONE' THEN
   NEW = 'YES'
ADDRESS ISPEXEC "DISPLAY PANEL(RACFGRP3)"
IF RC > Ø THEN
   RETURN
IF ATR = 'NONE' THEN
   ATR = ' '
PARSE VAR ATR P1 'REVOKED' P2
ATR = P1 P2
PARSE VAR ATR P1 '* UPDATED *' P2
ATR = P1 P2
REV = ' '
IF RVO = 'YES' THEN
   DO
      IF REVDD ¬= ' ' THEN
         DO
            REV  = REVMM'/'REVDD'/'REVYY
            REVC = 'REVOKE('REV')'
         END
```

```
      ELSE
         REVC = 'REVOKE'
   END
ELSE
   REVC = ' '
RES = ' '
IF RSO = 'YES' THEN
   DO
      IF RESDD ¬= ' ' THEN
         DO
            RES  = RESMM'/'RESDD'/'RESYY
            RESC = 'RESUME('RES')'
         END
      ELSE
         RESC = 'RESUME'
   END
ELSE
   RESC = ' '
ADDRESS TSO 'CONNECT' ID 'GROUP('GRP') AUTH('AUTH')' ATR REVC RESC
IF RC = Ø THEN
   DO
      ATR = '* UPDATED *'
      CNT = 'ØØØØØØ'
      ADDRESS ISPEXEC "TBMOD" TBLA    /* UPDATE TABLE */
      IF NEW = 'YES' THEN
         DO
            ID = 'NONE'
            ADDRESS ISPEXEC "TBDELETE" TBLA   /* DELETE NONE ENTRY */
         END
      IF RSORT = 'SORT' THEN
         ADDRESS ISPEXEC "TBSORT" TBLA "FIELDS(ID,C,A)"
   END
ELSE
   CALL EMSG 'ERRØ2'      /* CONNECT FAILED */
RETURN
/* DELETE CONNECT OPTION */
DELC:
ARG TBLA, ID
IF ID = 'NONE' THEN
   RETURN
ADDRESS TSO 'REMOVE' ID 'GROUP('GRP')'
IF RC = Ø THEN
   ADDRESS ISPEXEC "TBDELETE" TBLA   /* DELETE FROM TABLE */
ELSE
   CALL EMSG 'ERRØ3'      /* CONNECT FAILED */
RETURN

/* DISPLAY USERID OPTION */
DISU:
ARG ID
```

```
IF ID = 'NONE' THEN
   RETURN
X = OUTTRAP('VAR.')                                /* TRAP OUTPUT */
"LISTUSER" ID "TSO"
X = OUTTRAP('OFF')
USER = GETW('USER=' VAR.1)                        /* EXTRACT DATA */
NAME = SUBSTR(VAR.1,INDEX(VAR.1,'NAME=')+5,2Ø)
DEFG = GETW('DEFAULT-GROUP=' VAR.2)
PASD = GETW('PASSDATE=' VAR.2)
GROUPS = ''                              /* PROCESS CONNECT GROUPS */
DO I = 6 TO VAR.Ø
   IF INDEX(VAR.I,'TSO INFORMATION') > Ø THEN
      LEAVE
   W = GETW('GROUP=' VAR.I)
   IF W ¬= ' ' THEN
      GROUPS = GROUPS || W || '  '
END

JOBC = GETA('JOBCLASS=' I)
MSGC = GETA('MSGCLASS=' I)
ACCT = GETA('ACCTNUM='  I)
PROC = GETA('PROC=' I)
ADDRESS ISPEXEC "DISPLAY PANEL(RACFGRP5)"      /* DISPLAY PANEL */
RETURN

/* FUNCTION, EXTRACT DATA */
GETW:
ARG KEYWORD LINE
IN = INDEX(LINE,KEYWORD)
IF IN > Ø THEN
   RET = SUBWORD(SUBSTR(LINE,IN+LENGTH(KEYWORD)),1,1)
ELSE
   RET = ' '
RETURN RET

/* FUNCTION, EXTRACT TSO  */
GETA:
ARG KEYWORD K
RET = ' '
DO J = K TO VAR.Ø WHILE (RET = ' ')
  IN = INDEX(VAR.J,KEYWORD)
  IF IN > Ø THEN
     RET = SUBWORD(SUBSTR(VAR.J,IN+LENGTH(KEYWORD)),1,1)
END
RETURN RET

/* ERROR MESSAGE PROC */
EMSG:
ARG CODE
RCFLMSG = 'REISSUE DISPLAY TO VERIFY RESULTS OF COMMAND'
```

```
SELECT
   WHEN (CODE = 'ERRØ1') THEN
      RCFSMSG = 'CONNECT CHANGE FAILED'
   WHEN (CODE = 'ERRØ2') THEN
      RCFSMSG = 'CONNECT ADD FAILED'
   WHEN (CODE = 'ERRØ3') THEN
      RCFSMSG = 'CONNECT DELETE FAILED'
   WHEN (CODE = 'ERRØ4') THEN
      RCFSMSG = 'ID WAS NOT LOCATED'
   WHEN (CODE = 'ERRØ5') THEN
      RCFSMSG = 'INVALID GROUP NAME'
   WHEN (CODE = 'ERRØ6') THEN
      RCFSMSG = 'NOT ALLOWED FOR YOU'
   OTHERWISE NOP
END

ADDRESS ISPEXEC "SETMSG MSG(RCFMØØ1)"   /* MESSAGE WITH ALARM */
RETURN
```

ISPF PANEL RACFGRP1

```
)ATTR DEFAULT(@+_)
)BODY
@———————————————@NEW RACF ADMINSTRATION@———————————————+
@COMMAND ===>_ZCMD
+
@PROCESS+RACF@GROUP+CONNECTS
+
+This function uses the RACF LISTGRP command to generate a selection
list.
+
+
@RACF GROUP @===>_RGROUP  +  (Enter group name)
+
+SORT OPTION@===>_RSORT+     (SORT or ASIS to sort tables or not)
+
+
+Enter details and press@ENTER+to process or press@END+to exit
)INIT
  .HELP = RACFGRP6
  .CURSOR = RGROUP
  IF (&RSORT = ' ')
     &RSORT = SORT
)PROC
  VER (&RGROUP,NB)
  VER (&RSORT,NB,LIST,SORT,ASIS)
  VPUT (RGROUP RSORT) PROFILE
)END
```

ISPF PANEL RACFGRP2

```
)ATTR
  # TYPE(TEXT) INTENS(LOW) SKIP(ON)
)BODY
%————————————%NEW RACF ADMINSTRATION%————————————+
%COMMAND ===>_ZCMD
+
%CHANGE+RACF CONNECT
+This function will issue a CONNECT command.
+Changing the AUTH or ATTR or dates will modify the connect options.
+
%GROUP   :   &GRP
+ID      : + &ID
+
+AUTH  %===>_AUTH    + (New authority level, USE/CREATE/CONNECT/JOIN)
+ATTR  %===>_ATR       +(NONE for none)
+                      (Enter any special attributes such as GRPACC
OPERATIONS)
+CURRENT REVOKE :%&REV               +RESUME :%&RES
+
+REVOKE%===>_RVO+          (YES or NO to revoke user connect, Y/N)
+Optionally:
+DATE  %===>_Z #/_Z #/_Z + (DD / MM / YY to revoke access, blank if no
revoke)
+
+RESUME%===>_RSO+          (YES or NO to resume user connect, Y/N)
+Optionally:
+DATE  %===>_Z #/_Z #/_Z + (DD / MM / YY to resume access, blank if no
resume)
+
+Enter details and press%ENTER+to process or press%END+to exit
)INIT
  .ZVARS = '(REVDD REVMM REVYY RESDD RESMM RESYY)'
  .HELP = RACFGRP6
  .CURSOR = AUTH
  &RVO = 'NO'
  &RSO = 'NO'
)PROC
  VER (&AUTH,NB,LIST,USE,CREATE,CONNECT,JOIN)
  VER (&ATR,NB)
  &RVO = TRANS(&RVO Y,YES N,NO *,*)
  VER (&RVO,NB,LIST,YES,NO)
  IF (&REVDD ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
     VER (&REVYY,NB,NUM)
  IF (&REVMM ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
```

```
     VER (&REVYY,NB,NUM)
  IF (&REVYY ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
     VER (&REVYY,NB,NUM)
  &RSO = TRANS(&RSO Y,YES N,NO *,*)
  VER (&RSO,NB,LIST,YES,NO)
  IF (&RESDD ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
  IF (&RESMM ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
  IF (&RESYY ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
)END
```

ISPF PANEL RACFGRP3

```
)ATTR
  # TYPE(TEXT) INTENS(LOW) SKIP(ON)
)BODY
%────────────────%NEW RACF ADMINISTRATION%────────────────+
%COMMAND ===>_ZCMD
+
%ADD+RACF CONNECT
+This function will issue a CONNECT command.
+This will add the user to group &GRP with the options below:
+
%GROUP   :   &GRP
+
+ID    %===>_ID      +     (Enter NEW user-id to be connected)
+
+AUTH  %===>_AUTH    +     (Authority level, USE/CREATE/CONNECT/JOIN)
+ATTR  %===>_ATR           +(NONE for none)
+                          (Enter any special attributes such as GRPACC)
+REVOKE%===>_RVO+          (YES or NO to revoke user connect, Y/N)
+Optionally:
+DATE  %===>_Z #/_Z #/_Z + (DD / MM / YY to revoke access, blank if no
revoke)
+
+RESUME%===>_RSO+          (YES or NO to resume user connect, Y/N)
+Optionally:
+DATE  %===>_Z #/_Z #/_Z + (DD / MM / YY to resume access, blank if no
resume)
```

```
+
+Enter details and press%ENTER+to process or press%END+to exit
)INIT
  .ZVARS = '(REVDD REVMM REVYY RESDD RESMM RESYY)'
  .HELP = RACFGRP6
  .CURSOR = ID
  &OLD = &ID
  IF (&ID = NONE)
     &ID = ' '
  IF (&AUTH = NONE)
     &AUTH = ' '
  IF (&ATR = ' ')
     &ATR = NONE
  &RVO = 'NO'
  &RSO = 'NO'
)PROC
  IF (&OLD = &ID)
     &ID = ' '
  VER (&ID,NB)
  VER (&AUTH,NB,LIST,USE,CREATE,CONNECT,JOIN)
  VER (&ATR,NB)
  &RVO = TRANS(&RVO Y,YES N,NO *,*)
  VER (&RVO,NB,LIST,YES,NO)
  IF (&REVDD ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
     VER (&REVYY,NB,NUM)
  IF (&REVMM ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
     VER (&REVYY,NB,NUM)
  IF (&REVYY ¬= ' ')
     VER (&REVDD,NB,NUM)
     VER (&REVMM,NB,NUM)
     VER (&REVYY,NB,NUM)
  &RSO = TRANS(&RSO Y,YES N,NO *,*)
  VER (&RSO,NB,LIST,YES,NO)
  IF (&RESDD ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
  IF (&RESMM ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
  IF (&RESYY ¬= ' ')
     VER (&RESDD,NB,NUM)
     VER (&RESMM,NB,NUM)
     VER (&RESYY,NB,NUM)
)END
```

ISPF PANEL RACFGRP4

```
)ATTR
  # TYPE(OUTPUT) INTENS(LOW) JUST(LEFT) COLOR(TURQUOISE)
)BODY
%─────────── NEW RACF CONNECT DISPLAY ──────────────
%COMMAND ==>_ZCMD                                          +SCROLL
==>_SAMT+
%
%GROUP  : &GRP       +(Enter command%L xx+to position display to id xx)
+DESC   : &INSDATA
%
+SELECT :%S+DISP USER %C+CHANGE CONNECT %R+REMOVE CONNECT %A+ADD NEW
CONNECT
%
+S ID       AUTH    COUNT   GROUP ATTR     REVOKE DATE        RESUME
DATE
+──────────────────────────────────────────
)MODEL
_Z#Z       #Z      #Z      #Z             #Z                 #Z
)INIT
 .HELP = RACFGRP6
 .ZVARS = '(OPTA ID AUTH CNT ATR REV RES)'
 .AUTOSEL = NO
 IF (&SAMT = ' ')
    &SAMT = &ZSCED
 &ZCMD = ' '
)PROC
  VER (&OPTB,LIST,S,s,C,c,R,r,A,a)
)END
```

ISPF PANEL RACFGRP5

```
%──────────────%THE RACF USER DISPLAY%──────────────+
%COMMAND ===>_ZCMD
+
%DISPLAY+RACF%USERID+INFORMATION
+
+USERID         :% &USER    +       RACF userid
+
+NAME OF USER   :% &NAME    +       Name as recorded
+
+PASSWORD DATE  :% &PASD    +       Date of last password change
+
+DEFAULT GROUP  :% &DEFG    +       Default RACF group
+
+CONNECT GROUPS :% &GROUPS
+JOBCLASS DFT   :% &JOBC    +      Used by SUBMIT command if no jobcard
+MSGCLASS DFT   :% &MSGC    +      Used by SUBMIT command if no jobcard
```

```
+
+CURRENT ACCOUNT:% &ACCT   + At last logon, can be changed during logon
+CURRENT PROC   :% &PROC   + At last logon, can be changed during logon
+
+Press%END+to exit
)INIT
  .HELP = RACFGRP6
)PROC
)END
```

ISPF PANEL RACFGRP5

```
%TUTORIAL ──────────%NEW RACF ADMINSTRATION%──────── TUTORIAL+
%COMMAND ===>_ZCMD
+
% PROCESS GROUP CONNECTS+
+
+ This option allows specification of a group name to list the%CONNECTS+
+ to that group. The connect list can then be maintained with further
+ selections or some information about the user can be displayed.
+
+ Note that if the user has connect status because the group is the
+ default for the user, the remove connect selection WILL NOT WORK.
+
+ The connect count and other information displayed in the connect list
+ is related to use of the GROUP by the user and not necessarily to the
+ user-id.
+
+ Multiple selections can be made and are processed serially. Note that
+ % when adding+connects you select an%existing+userid, this is then
+ used simply to supply some defaults for the add of the new entry and
+ does not affect the existing entry. Simply choose one similar to the
+ attributes required of the new one and then change the details on the
+ panel displayed.
+
+ Press%END+to return to display.
)PROC
  &ZCONT = RACFGRP6
)END
```

ISPF MESSAGE MEMBER RCFM00

```
RCFMØØØ   '&RCFSMSG'                    .ALARM=NO
'&RCFLMSG'
RCFMØØ1   '&RCFSMSG'                    .ALARM=YES
'&RCFLMSG'
```

Encrypting/decrypting datasets

Even if your personal datasets are protected by RACF, CA- ACF2, or CA-TOP SECRET, there is almost certainly someone in your organization who has the authority to edit/browse them, even if it's only the security administrator. RACF administrators usually have SPECIAL, OPERATIONS, and AUDITOR authority, which means they can do anything.

The aim of SKED80F is to provide single-key encryption/decryption for 80-byte fixed-length records. Encryption and decryption are performed on a record by record basis. SKED80F can be used to hide sensitive 80-byte MVS datasets.

OPERATION

The same key is used for both encryption of plain text and decryption of the corresponding cipher text. The key is an 80-byte (640-bit) internally-generated value, created from one to 50 input records containing any EBCDIC values. The input file will be closed after 50 records have been read.

The key is used to modify, and select from, a large table of pseudo-random data (16,000 bytes) derived from the irrational number pi. This table, and the value in the internal key field, change continually during the processing of a plain text/cipher text dataset.

The process of encryption involves these steps for each input record (decryption is the reverse):

- Transposition of bytes (shuffling).
- Rotation of groups of words.
- Transposition of bytes (shuffling).
- Byte addition using bytes of shuffled 'KEY80'.
- Byte substitution using a dynamic translate table (different for each record).

- Transposition of bytes (shuffling).
- A variable number of XORs (1–10) of the record with 80-byte fields selected from the large pseudo-random table.

All these operations depend on values in the key field (640 bits), which is changed after each record has been processed. The 16,000-byte pseudo-random table is also modified after each input record has been processed.

USAGE

Cryptographic systems are usually compromised through careless key handling. If you leave your key data lying around in a file or, worse, written on a yellow sticky note stuck to your terminal, then you deserve to be exposed. Even the strongest military-grade cryptography will be useless in such circumstances.

The best way to use SKED80F is to set up your key data in a RACF-protected dataset just before you submit the encrypt/decrypt job, then delete it immediately after the run completes. Don't use instream key data because someone with the appropriate authority could use SDSF, or a similar product, to capture your job to a dataset while it is on the input queue, and use the hex edit/browse facility to see your key data.

As far as I can determine there are no 'weak' keys for SKED80F. However, there is at least one key that is totally lame and should be avoided. This is a single 80-byte record that contains EBCDIC spaces. Try to use several key records containing a full range of EBCDIC values. Try to avoid a single key record that contains one of the classic 'bad' passwords such as your name, your child's name, or your dog's name, etc.

An ideal key comprises multiple records containing pseudo-random EBCDIC data. The problem is, how do you remember it? I use a system whereby I created a file with an innocuous name (eg TEST.DATA, MISC, etc), containing several thousand 80-byte records that were generated by running SKED80F against a member of IPO1.DOCLIB. When I want to do an encryption/decryption, I copy a subset of this information into my key file. Use your imagination

here. You could copy in records 999 to 1004, then 3330 to 3338, then change all X'FF' to X'DD' etc. Figure out a scheme that you can easily remember and is meaningful to you.

OBSERVATIONS

Cryptographers talk a lot about abstruse mathematics, usually involving Galois fields, huge primes, and other bizarre things. I am not a mathematician, and have only tried to make the transformations performed by this program as complex and non-linear as possible, consistent with acceptable performance. If there is no method of attack against SKED80F better than brute force, then it would appear to be very strong indeed. With an internal key having 2**640 possible values, a brute force attack should take quite a bit longer than the projected age of the universe. Remember, DES held out for a long time using just 56 bits for its key. The current crop of 128-bit key algorithms are supposed to be impregnable to brute force attack, even with the speed and parallelism of modern processors.

I believe that SKED80F is fairly secure. However, should you find all kinds of laughable mistakes that make breaking SKED80F a trivial exercise, I would very much appreciate some feedback. Please send e-mail to Phil-Rigby@usa.net (constructive criticism or effusive praise only, please).

Testing of SKED80F has shown:

- SKED80F is not particularly fast. On a medium-load 9021/340 system, doing no paging, it processes approximately 6,000 records per minute. However, it is designed as a batch utility, where speed is not of the essence.
- For large files (100,000 records or more), the distribution of the values X'00' through X'FF' in the cipher text closely approaches the expected median value, generally being in a range of -1.6% to +1.6% either side of it. Nobody examining a cipher text should be able to deduce anything about the plain text by analysing the frequency of occurrence of various values.
- It would appear that changing a single bit in a key results in a

massive change in the output cipher text. I have tested this by using a constant Initialization Vector (IV) value and a large key of 50 records. The IV is an 8-byte field in the program that contains an Initialisation Vector generated from the system clock.This ensures that separate encryptions of the same input text using the same key generate different cipher text outputs. Even altering the least-significant bit of the 4000-byte key produced this massive change. I believe this is what cryptographers refer to as the 'avalanche' effect.

- There is no evidence of recurring patterns of characters appearing in cipher text, even in the case of large datasets of 100,000 lines or more.

SAMPLE JCL

```
//jobname JOB (acct),'pgmrname',CLASS=c,MSGCLASS=X,NOTIFY=userid
//*
//SKED8ØF  EXEC PGM=SKED8ØF
//STEPLIB  DD   DISP=SHR,DSN=your.loadlib
//INFILE   DD   DISP=SHR,DSN=data.to.encrypt/decrypt
//OUTFILE  DD   DISP=SHR,DSN=data.from.encrypt/decrypt
//KEYFILE  DD   DISP=SHR,DSN=your.key.file
//*SYSABEND DD   SYSOUT=*
//
```

SKED80F ASSEMBLER

```
PRINT NOGEN
SKED8ØF  TITLE 'SKED8ØF - CRYPTO FOR F,8Ø DATASETS'
*————————————————————————————————*
*                                                                *
*    Module name:    SKED8ØF                                     *
*                                                                *
*    Attributes:     AMODE/RMODE 24, User key                    *
*                    Assembler parm = LOAD,NODECK,OBJECT,NORENT  *
*                    Linkedit  parm = LET,LIST,XREF,MAP          *
*                                                                *
*————————————————————————————————*
SKED8ØF  CSECT
SKED8ØF  AMODE 24
SKED8ØF  RMODE 24
*
         SAVE  (14,12)
         LR    R11,R15
```

```
         LR    R12,R11
         LA    R12,2Ø48(,R12)     Add 4K to
         LA    R12,2Ø48(,R12)       second base register
         USING SKED8ØF,R11,R12
*
         ST    R13,SAVEAREA+4
         LA    R15,SAVEAREA
         ST    R15,8(R13)
         LR    R13,R15
*-------------------------------------*
*    Open and read first record of input file to determine whether    *
*    this is an encryption or decryption run.                         *
*-------------------------------------*
         OPEN  (INFILE,INPUT,OUTFILE,OUTPUT)
         GET   INFILE,REC8Ø
         MVI   RUNIND,C'E'          Assume an encrypt run
         CLC   REC8Ø(HEADERL),HEADER
         BNE   ENCRUN
*
         MVI   RUNIND,C'D'          It's a decrypt run
         MVC   IV,REC8Ø+HEADERL     Save the Initialization Vector
         GET   INFILE,REC8Ø         and get first encrypted record
         B     KEYSETUP
*-------------------------------------*
*    Create an Initialization Vector from the system clock.           *
*-------------------------------------*
ENCRUN   EQU   *
         STCK  TODCLOCK
*-------------------------------------*
*    Set up the CBYTE field by XORing the 4 low-order bytes of the    *
*    TODCLOCK (the most volatile bytes).                              *
*-------------------------------------*
         LA    RØ3,TODCLOCK+4
         LA    RØ4,4               Number of TODCLOCK bytes
         MVI   CBYTE,Ø
TODXOR   EQU   *
         XC    CBYTE,Ø(RØ3)        XOR TODCLOCK byte into CBYTE
         LA    RØ3,1(,RØ3)         Next byte of TODCLOCK
         BCT   RØ4,TODXOR
*-------------------------------------*
*    Rotate the first, last and middle words of the TODCLOCK.         *
*-------------------------------------*
         LA    RØ2,TODCLOCK
         LA    RØ3,1                Rotate 1 word
         BAL   R14,ROTATE           Call rotate subroutine
         LA    RØ2,TODCLOCK+4       Point to last 4 bytes
         LA    RØ3,1                Rotate 1 word
         BAL   R14,ROTATE           Call rotate subroutine
         LA    RØ2,TODCLOCK+2       Point to middle 4 bytes
         LA    RØ3,1                Rotate 1 word
```

```
         BAL   R14,ROTATE           Call rotate subroutine
         LA    RØ2,TODCLOCK
         LA    RØ3,2                Rotate 2 words
         BAL   R14,ROTATE           Call rotate subroutine
*————————————————————————————————*
*    The IV value will be the rotated TODCLOCK in reverse.          *
*————————————————————————————————*
         LA    RØ2,IV
         LA    RØ3,TODCLOCK+7
         LA    RØ4,8
REVTOD   EQU   *
         MVC   Ø(1,RØ2),Ø(RØ3)      Move TODCLOCK byte to IV
         LA    RØ2,1(,RØ2)          Next byte of IV
         SH    RØ3,=H'1'            Next byte of TODCLOCK (backwards)
         BCT   RØ4,REVTOD
*————————————————————————————————*
*    Finally, rotate the two words of the IV.                       *
*————————————————————————————————*
         LA    RØ2,IV               Point to start of field
         LA    RØ3,2                Number of words to rotate
         BAL   R14,ROTATE           Call rotate subroutine
*————————————————————————————————*
*    Set up the header record containing the IV.                    *
*————————————————————————————————*
         MVI   HDRREC,C' '
         MVC   HDRREC+1(79),HDRREC
         MVC   HDRREC(HEADERL),HEADER
         MVC   HDRREC+HEADERL(L'IV),IV
*
KEYSETUP EQU   *
         MVI   CBYTE,255            Initialize the CBYTE
*————————————————————————————————*
*    Now open and process the records in the KEYFILE. There must be *
*    at least one and we will accept a maximum of 5Ø. The data from *
*    the KEYFILE will be "crunched" into a single 8Ø-byte value.    *
*    Each byte of each key record will be weighted by multiplying it*
*    with an incrementing value. After the multiplication the result*
*    will be XORed into the original byte.                          *
*————————————————————————————————*
         OPEN  (KEYFILE,INPUT)
         XC    KEY8Ø,KEY8Ø
         LA    RØ7,3                Start value for multiplier
KEYLOOP  EQU   *
         GET   KEYFILE,KEYREC
         AP    KEYCNT,=P'1'         Increment count of key records
         CP    KEYCNT,=P'5Ø'        Check for more than 5Ø
         BH    EOFKEY
         LA    RØ2,KEYREC
         LA    RØ3,8Ø
KEYLPA   EQU   *
```

```
         LA    RØ8,Ø
         IC    RØ8,Ø(RØ2)          Key record byte into RØ8
         LTR   RØ8,RØ8             If it's zero then avoid
         BZ    KEYLPB               doing the multiplication
         STH   RØ7,HALFW            RØ7 contains the multiplier
         MH    RØ8,HALFW
         ST    RØ8,FULLW
         XC    Ø(1,RØ2),FULLW
         XC    Ø(1,RØ2),FULLW+1
         XC    Ø(1,RØ2),FULLW+2
         XC    Ø(1,RØ2),FULLW+3
KEYLPB   EQU   *
         LA    RØ2,1(,RØ2)          Point to next key record byte
         LA    RØ7,1(,RØ7)          Increment the multiplier
         BCT   RØ3,KEYLPA
*
         XC    KEY8Ø,KEYREC         "Crunch" KEYREC into KEY8Ø
         BAL   R14,CBKEY8Ø          Modify CBYTE value
*-------------------------------------*
*    Rotate two segments of the key field. The segment size, in words,*
*    is determined by dividing CBYTE by 21 to get a remainder from Ø  *
*    to 2Ø. The second segment size is 2Ø minus the remainder.        *
*-------------------------------------*
         LA    RØ4,Ø
         LA    RØ5,Ø
         IC    RØ5,CBYTE
         D     RØ4,=F'21'
         ST    RØ4,REMQUOT          Save remainder
         LR    RØ3,RØ4              Use remainder as no. of words
         LA    RØ2,KEY8Ø
         BAL   R14,ROTATE
*
         LA    RØ3,2Ø
         L     R15,REMQUOT          Saved remainder
         SR    RØ3,R15              Subtract saved remainder from 2Ø
         LA    RØ2,KEY8Ø
         SLL   R15,2                Multiply saved remainder by 4
         AR    RØ2,R15               and add as offset into KEY8Ø
         BAL   R14,ROTATE
*
         B     KEYLOOP
*
EOFKEY   CLOSE KEYFILE
         CP    KEYCNT,=P'1'
         BL    EXITRC8              Error if zero KEYFILE records
*-------------------------------------*
*    Add logical the two words of IV and add the count of records     *
*    in the KEYFILE. Use the result to calculate an offset into       *
*    TAB16ØØØ of an 8Ø-byte area to be XORed with KEY8Ø.              *
```

```
*-------------------------------------*
         XC    DWORK,DWORK
         MVC   DWORK+6(2),KEYCNT    Convert count of KEYFILE records
         CVB   R02,DWORK             to binary
         LA    R04,0
         L     R05,IV
         AL    R05,IV+4
         ALR   R05,R02           Add KEYFILE rec. count
         D     R04,F15921        Remainder is in R04 (use as offset)
         LA    R05,TAB16000
         AR    R05,R04           Point into TAB16000 and
*                                 XOR the data at this location
         XC    KEY80,0(R05)        into the value in KEY80
         BAL   R14,CBKEY80       Update a CBYTE value using KEY80
*-------------------------------------*
*    Now use the contents of KEY80 and IV to select 80-byte areas     *
*    in TAB16000. The data at these locations may be used to XOR      *
*    with KEY80, depending on whether the high-order bit is set on.   *
*-------------------------------------*
         LA    R02,KEY80
         LA    R03,10
         LA    R05,0
CALCKEYW EQU   *
         X     R05,0(R02)        First word of pair
         AL    R05,4(R02)        Second word of pair
         LA    R02,8(,R02)       Next pair of KEY80 words
         BCT   R03,CALCKEYW
*
         LA    R04,0             Clear even reg. for division
         D     R04,=F'16'        Get a remainder 0 thru 15
         LA    R04,1(,R04)       Add 1 to it
         LR    R09,R04           Save as loop counter in R09
*
         TM    KEY80,X'80'       Hi-order bit set ?
         BNO   KEYSCHED          BIF no, avoid rotation
         LA    R02,KEY80
         LA    R03,20            Rotate 20 words
         BAL   R14,ROTATE
KEYSCHED EQU   *
*-------------------------------------*
*   This is the final key schedule before starting encryption or      *
*   decryption. It is performed from 1 to 16 times.                   *
*   For each iteration calculate a value WORKX and WORKY using the    *
*   first 10 and second 10 words of KEY80 plus an XOR of the rotated  *
*   IV value. Use these values to point into TAB16000 and control     *
*   the number of times consecutive pairs of words from TAB16000 are  *
*   used to calculate offsets into the table from which an 80-byte    *
*   field may, conditionally, be XORed with KEY80. If the field is    *
*   not XORed with KEY80 we will rotate KEY80 instead.                *
*-------------------------------------*
```

```
         LA    RØ2,KEY8Ø        Point to first 1Ø words
         LA    RØ3,5
         LA    RØ5,Ø
KSCALCX  EQU   *
         TM    Ø(RØ2),X'8Ø'
         BNO   KSCALCXI
         X     RØ5,Ø(RØ2)
         AL    RØ5,4(RØ2)
KSCALCXI EQU   *
         LA    RØ2,8(,RØ2)      Point to next pair of words
         BCT   RØ3,KSCALCX
*
         LA    RØ2,IV
         LA    RØ3,2
         BAL   R14,ROTATE
         LA    RØ4,Ø
         AL    RØ5,IV
         AL    RØ5,IV+4
         D     RØ4,F15921
         ST    RØ4,WORKX
*
         LA    RØ2,KEY8Ø+4Ø     Point to second 1Ø words
         LA    RØ3,5
         LA    RØ5,Ø
KSCALCY  EQU   *
         TM    Ø(RØ2),X'8Ø'
         BNO   KSCALCYI
         X     RØ5,Ø(RØ2)
         AL    RØ5,4(RØ2)
KSCALCYI EQU   *
         LA    RØ2,8(,RØ2)      Point to next pair of words
         BCT   RØ3,KSCALCY
*
         LA    RØ2,IV
         LA    RØ3,2
         BAL   R14,ROTATE
         AL    RØ5,IV
         AL    RØ5,IV+4
         LA    RØ4,Ø
         D     RØ4,=F'7'
         AH    RØ4,=H'4'
         ST    RØ4,WORKY
*
         L     RØ5,WORKX
         LA    RØ4,Ø
         D     RØ4,F15921
         LA    RØ5,TAB16ØØØ
         AR    RØ5,RØ4          Point into TAB16ØØØ
*
         L     RØ7,WORKY
```

```
         LA    RØ6,Ø
         D     RØ6,=F'7'
         AH    RØ6,=H'4'        Calculate loop counter (4 to 1Ø)
KSLOOP2  EQU   *
         L     RØ3,Ø(RØ5)       Word from TAB16ØØØ
         AL    RØ3,4(RØ5)       Add logical the next word
         XC    IV,Ø(RØ5)
         TM    4(RØ5),X'8Ø'
         BNO   KSLOOP2B
         XC    IV,4(RØ5)
KSLOOP2B EQU   *
         LA    RØ2,Ø
         D     RØ2,F15921
         LA    RØ3,TAB16ØØØ
         AR    RØ3,RØ2
         TM    Ø(RØ3),X'8Ø'
         BNO   KSLOOP2R
         XC    KEY8Ø,Ø(RØ3)
         B     KSLOOP2Z
KSLOOP2R EQU   *
         LA    RØ2,KEY8Ø
         LA    RØ3,2Ø
         BAL   R14,ROTATE
KSLOOP2Z EQU   *
         LA    RØ5,8(,RØ5)       Next pair of words in TAB16ØØØ
         BCT   RØ6,KSLOOP2
*
         BCT   RØ9,KEYSCHED
*
         CLI   RUNIND,C'D'       A decrypt run ?
         BE    CHKCONST
*-------------------------------------*
*    Write out the first and second header records. The second header *
*    will be an encryption of CONSTEXT.                               *
*-------------------------------------*
         PUT   OUTFILE,HDRREC    Write first header
         BAL   R14,SELECT1Ø      Choose 1Ø 8Ø-byte areas from TAB16ØØØ
         BAL   R14,TVECSET       Set up transposition vectors
*                                 TVEC1 and TVEC2
*-------------------------------------*
*    Create TVEC3 by shuffling TVEC1 using TVEC2.                     *
*-------------------------------------*
         LA    RØ1,TVEC1          8Ø-byte field to be shuffled
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   TVEC3,WORK8Ø       Save new transposition vector
*
         MVC   SAVREC8Ø,REC8Ø
         MVC   REC8Ø,CONSTEXT
         BAL   R14,ENCRYPT        Encrypt constant text and
```

```
         PUT   OUTFILE,REC8Ø       write it to OUTFILE then
         MVC   REC8Ø,SAVREC8Ø       restore the original data
         B     ROT16ØØØ
*
CHKCONST EQU   *
*————————————————————————————————*
*    Check that the second record of the encrypted file can be de-    *
*    crypted to reveal the data in CONSTEXT. If not, the key used for *
*    this run is invalid so exit with return code 12.                 *
*————————————————————————————————*
         BAL   R14,SELECT1Ø      Choose 1Ø 8Ø-byte areas from TAB16ØØØ
         BAL   R14,TVECSET       Set up transposition vectors
*                                 TVEC1 and TVEC2
*————————————————————————————————*
*    Create TVEC3 by shuffling TVEC1 using TVEC2.                     *
*————————————————————————————————*
         LA    RØ1,TVEC1          8Ø-byte field to be shuffled
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   TVEC3,WORK8Ø       Save new transposition vector
*
         BAL   R14,DECRYPT
         CLC   REC8Ø,CONSTEXT
         BNE   EXITRC12
         GET   INFILE,REC8Ø
*
ROT16ØØØ EQU   *
*————————————————————————————————*
*    Do a variable number of rotations (1 to 7) of TAB16ØØØ.          *
*————————————————————————————————*
         L     RØ5,KEY8Ø
         X     RØ5,KEY8Ø+4
         AL    RØ5,KEY8Ø+8
         LA    RØ4,Ø
         D     RØ4,=F'7'
         LA    RØ4,1(,RØ4)
T16LPROT EQU   *
         BAL   R14,T16KROT        Do rotate of TAB16ØØØ words
         LA    RØ2,KEY8Ø
         LA    RØ3,2Ø
         BAL   R14,ROTATE
         BCT   RØ4,T16LPROT
*
MAINLOOP EQU   *
         BAL   R14,SELECT1Ø
*————————————————————————————————*
*    Now do the encryption/decryption.                                *
*                                                                     *
*    For each data record;                                            *
*                                                                     *
```

```
*    Encryption sequence                                            *
*      1)  Transposition of record bytes using TVEC1                *
*      2)  Rotation of record segments (a variable number of        *
*          rotations of variable length word segments)              *
*      3)  Transposition of record bytes using TVEC2                *
*      4)  Shuffle KEY8Ø and add each byte to each corresponding    *
*          byte of REC8Ø, ignoring any carry                        *
*      5)  Substitution of record bytes using dynamic translation   *
*          table that is changed for each record                    *
*      6)  Transposition of record bytes using TVEC3                *
*      7)  A variable number of XORs with TAB16ØØØ data (from 1 to 1Ø)*
*                                                                   *
*    Decryption sequence                                            *
*      1)  A variable number of XORs with TAB16ØØØ data (from 1 to 1Ø)*
*      2)  Reverse transposition of record bytes using TVEC3        *
*      3)  Reverse substitution using an inverse of the translation *
*          table created in 5) above                                *
*      4)  Shuffle KEY8Ø and subtract each byte from each           *
*          corresponding byte of REC8Ø                              *
*      5)  Reverse transposition of record bytes using TVEC2        *
*      6)  Reverse rotation of record segments.                     *
*      7)  Reverse transposition of record bytes using TVEC1        *
*                                                                   *
*-------------------------------------*
         BAL   R14,TVECSET       Set up transposition vectors
*                                 TVEC1 and TVEC2
*-------------------------------------*
*    Create TVEC3 by shuffling TVEC1 using TVEC2.                   *
*-------------------------------------*
         LA    RØ1,TVEC1         8Ø-byte field to be shuffled
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   TVEC3,WORK8Ø      Save new transposition vector
*
         CLI   RUNIND,C'E'       Is it an encrypt run ?
         BNE   DECRRUN           BIF not, go and decrypt record
*
         BAL   R14,ENCRYPT
         B     PUTGET
*
DECRRUN  EQU   *
         BAL   R14,DECRYPT
*
PUTGET   PUT   OUTFILE,REC8Ø
         GET   INFILE,REC8Ø
*-------------------------------------*
*    Now modify KEY8Ø by doing a variable no. of XORs with the 1Ø   *
*    data areas previously chosen. Do an unconditional XOR with the *
*    first data area and 9 conditional XORs with the remainder. If  *
*    the first bit in the area is Ø do the XOR. When this has been  *
```

```
*    done rotate the entire KEY8Ø field.                              *
*-------------------------------------*
         XC    KEY8Ø,T16KDATA
         LA    RØ4,T16KDATA+8Ø
         LA    RØ5,9
KEY8ØXLP EQU   *
         TM    Ø(RØ4),B'10000000'
         BO    K8ØXLP1
         XC    KEY8Ø,Ø(RØ4)
K8ØXLP1  LA    RØ4,8Ø(,RØ4)
         BCT   RØ5,KEY8ØXLP
*
         LA    RØ2,KEY8Ø
         LA    RØ3,2Ø
         BAL   R14,ROTATE
*
         BAL   R14,T16KROT        Rotate all of TAB16ØØØ
         CLI   CBYTE,128
         BH    T16SHUFF
         BAL   R14,T16KROT        Rotate all of TAB16ØØØ
T16SHUFF EQU   *
*-------------------------------------*
*    Finally, use TVEC1, TVEC2 and TVEC3 to perform substitution on   *
*    3 8Ø-byte fields selected from TAB16ØØØ. Always call ESHUFFLE.   *
*-------------------------------------*
         LA    RØ4,Ø
         L     RØ5,KEY8Ø          Set up a dividend
         X     RØ5,KEY8Ø+2Ø
         AL    RØ5,KEY8Ø+24
         D     RØ4,F15921         Remainder is in RØ4 (use as offset)
         LA    RØ7,TAB16ØØØ
         AR    RØ7,RØ4            Point into TAB16ØØØ
         LR    RØ1,RØ7            8Ø-byte field to be shuffled
         LA    RØ2,TVEC1
         BAL   R14,ESHUFFLE
         MVC   Ø(8Ø,RØ7),WORK8Ø   Move shuffled field into TAB16ØØØ
*
         LA    RØ4,Ø
         L     RØ5,KEY8Ø+4        Set up a dividend
         X     RØ5,KEY8Ø+28
         AL    RØ5,KEY8Ø+32
         D     RØ4,F15921         Remainder is in RØ4 (use as offset)
         LA    RØ7,TAB16ØØØ
         AR    RØ7,RØ4            Point into TAB16ØØØ
         LR    RØ1,RØ7            8Ø-byte field to be shuffled
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   Ø(8Ø,RØ7),WORK8Ø   Move shuffled field into TAB16ØØØ
*
         LA    RØ4,Ø
```

```
         L     RØ5,KEY8Ø+8        Set up a dividend
         X     RØ5,KEY8Ø+36
         AL    RØ5,KEY8Ø+4Ø
         D     RØ4,F15921         Remainder is in RØ4 (use as offset)
         LA    RØ7,TAB16ØØØ
         AR    RØ7,RØ4            Point into TAB16ØØØ
         LR    RØ1,RØ7            8Ø-byte field to be shuffled
         LA    RØ2,TVEC3
         BAL   R14,ESHUFFLE
         MVC   Ø(8Ø,RØ7),WORK8Ø   Move shuffled field into TAB16ØØØ
*
         B     MAINLOOP
*
EOFIN    EQU   *
         BAL   R14,CLOSFILE
         LA    R15,Ø
         B     RETURN
EXITRC8  BAL   R14,CLOSFILE
         LA    R15,8
         B     RETURN
EXITRC12 BAL   R14,CLOSFILE
         LA    R15,12
RETURN   DS    ØH
         L     R13,SAVEAREA+4
         ST    R15,16(R13)
         RETURN (14,12)
*
*———————————————————————————*
*    S u b r o u t i n e s (all return on R14)                *
*———————————————————————————*
*———————————————————————————*
*    Select 1Ø 8Ø-byte areas of TAB16ØØØ. Derive an offset from  *
*    consecutive words in TAB16ØØØ.                           *
*———————————————————————————*
SELECT1Ø EQU   *
         LA    RØ8,Ø
         L     RØ9,KEY8Ø          Load first word of KEY8Ø
         AL    RØ9,KEY8Ø+76        and add logical the last word
         D     RØ8,F15921         Calculate an offset value
         LA    RØ3,TAB16ØØØ       Point to start of TAB16ØØØ
         AR    RØ3,RØ8            Remainder as offset into TAB16ØØØ
         LA    RØ4,KEY8Ø+4
         LA    RØ5,T16KDATA
         LA    RØ6,1Ø
SEL1ØLP  EQU   *
         L     RØ9,Ø(RØ3)         Load current word of TAB16ØØØ
         AL    RØ9,4(RØ3)          and add logical the next one
         AL    RØ9,Ø(RØ4)            and add logical KEY8Ø word
         LA    RØ8,Ø
         D     RØ8,F15921         Calculate an offset value
```

```
         LA    RØ7,TAB16ØØØ
         AR    RØ7,RØ8           Point into TAB16ØØØ and
         MVC   Ø(8Ø,RØ5),Ø(RØ7)   save the 8Ø bytes of data
         LA    RØ5,8Ø(,RØ5)      Next 8Ø-byte save field
         LA    RØ3,8(,RØ3)       Next pair of words in TAB16ØØØ
         LA    RØ4,4(,RØ4)       Next word in KEY8Ø
         BCT   RØ6,SEL1ØLP
*
         BR    R14
*-------------------------------------*
*    E n c r y p t i o n     P r o c e s s i n g   *
*-------------------------------------*
ENCRYPT  EQU   *
         B     ENCRYPTA            Branch around local save area
ENCRYPT_SAVEAREA DS   16F
ENCRYPTA EQU   *
         STM   RØØ,R15,ENCRYPT_SAVEAREA   Non-standard save all regs.
*-------------------------------------*
*    TVEC1 contains a list of 8Ø 1-byte offsets.                    *
*    Use these to shuffle the bytes of the input record.            *
*-------------------------------------*
         LA    RØ1,REC8Ø          8Ø-byte field to be shuffled
         LA    RØ2,TVEC1
         BAL   R14,ESHUFFLE
         MVC   REC8Ø,WORK8Ø       Move shuffled record to input rec.
*-------------------------------------*
*    Do rotate of input record.                                     *
*-------------------------------------*
         BAL   R14,REC8ØROT       Rotate record segments
*-------------------------------------*
*    TVEC2 contains a list of 8Ø 1-byte offsets.                    *
*    Use these to shuffle the bytes of the input record.            *
*-------------------------------------*
         LA    RØ1,REC8Ø          8Ø-byte field to be shuffled
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   REC8Ø,WORK8Ø       Move shuffled record to input rec.
*-------------------------------------*
*    Shuffle the KEY8Ø field then do byte addition of the result    *
*    to REC8Ø.                                                      *
*-------------------------------------*
         LA    RØ1,KEY8Ø          8Ø-byte field to be shuffled
         LA    RØ2,TVEC1
         BAL   R14,ESHUFFLE
         MVC   K8ØSHUFF,WORK8Ø
*
         CLI   TVEC2,4Ø           Only do second shuffle if first
         BH    K8ØADD              byte of TVEC2 is 4Ø or less
*
         LA    RØ1,K8ØSHUFF       8Ø-byte field to be shuffled
```

```
         LA    RØ2,TVEC2
         BAL   R14,ESHUFFLE
         MVC   K8ØSHUFF,WORK8Ø
K8ØADD   EQU   *
         LA    RØ1,REC8Ø
         LA    RØ2,K8ØSHUFF
         LA    RØ3,8Ø
K8ØADDLP EQU   *                  Byte addition loop
         BAL   R14,BYTEADD
         LA    RØ1,1(,RØ1)
         LA    RØ2,1(,RØ2)
         BCT   RØ3,K8ØADDLP
*-------------------------------------*
*    Set up a dynamic translation table using data from TAB16ØØØ.     *
*    Then use it to translate the input record.                       *
*-------------------------------------*
         BAL   R14,SETUPR1Ø
         XC    TRTAB,TRTAB
         XC    TRTUSED,TRTUSED
         LA    RØ2,TRTAB
         LA    RØ3,256
TRTLP1   EQU   *
         LA    RØ4,TRTUSED
         LA    RØ5,Ø
         IC    RØ5,Ø(R1Ø)
         AR    RØ4,RØ5
         TM    Ø(RØ4),X'8Ø'
         BNO   TRTLP1A
         BAL   R14,INCRR1Ø
         B     TRTLP1
TRTLP1A  EQU   *
         MVC   Ø(1,RØ2),Ø(R1Ø)
         OI    Ø(RØ4),X'8Ø'
         LA    RØ2,1(,RØ2)
         BAL   R14,INCRR1Ø
         BCT   RØ3,TRTLP1
*
         TR    REC8Ø,TRTAB         Translate the input record
*-------------------------------------*
*    Shuffle REC8Ø using TVEC3.                                       *
*-------------------------------------*
         LA    RØ1,REC8Ø           8Ø-byte field to be shuffled
         LA    RØ2,TVEC3
         BAL   R14,ESHUFFLE
         MVC   REC8Ø,WORK8Ø
*
         BAL   R14,VARIXOR         Variable no. of XORs of REC8Ø
         LM    RØØ,R15,ENCRYPT_SAVEAREA   Restore all regs.
         BR    R14
*
```

```
*-------------------------------------*
*    D e c r y p t i o n     P r o c e s s i n g   *
*-------------------------------------*
DECRYPT  EQU   *
         B     DECRYPTA           Branch around local save area
DECRYPT_SAVEAREA DS   16F
DECRYPTA EQU   *
         STM   R00,R15,DECRYPT_SAVEAREA   Non-standard save all regs.
*
         BAL   R14,VARIXOR        Variable no. of XORs of REC80
*
         LA    R01,REC80          80-byte field to be shuffled
         LA    R02,TVEC3
         BAL   R14,DSHUFFLE
         MVC   REC80,WORK80       Move shuffled record to input rec.
*-------------------------------------*
*    Set up a dynamic translation table using data from TAB16000.     *
*    Then use it to translate the input record. This table will be    *
*    an inverse of the equivalent table created during encryption.    *
*-------------------------------------*
         BAL   R14,SETUPR10
         XC    TRTAB,TRTAB
         XC    TRTUSED,TRTUSED
         LA    R03,256
         LA    R07,0
TRTLP2   EQU   *
         LA    R02,TRTUSED
         LA    R05,0
         IC    R05,0(R10)
         AR    R02,R05
         TM    0(R02),X'80'
         BNO   TRTLP2A
         BAL   R14,INCRR10
         B     TRTLP2
TRTLP2A  EQU   *
         LA    R04,TRTAB
         LA    R05,0
         IC    R05,0(R10)
         AR    R04,R05
         STC   R07,0(R04)
         OI    0(R02),X'80'
         LA    R07,1(,R07)
         BAL   R14,INCRR10
         BCT   R03,TRTLP2
*
         TR    REC80,TRTAB        Translate the input record
*-------------------------------------*
*    Shuffle the KEY80 field then do byte subtraction of the result   *
*    from REC80.                                                      *
*-------------------------------------*
```

```
         LA    R01,KEY80          80-byte field to be shuffled
         LA    R02,TVEC1
         BAL   R14,ESHUFFLE
         MVC   K80SHUFF,WORK80
*
         CLI   TVEC2,40           Only do second shuffle if first
         BH    K80SUB              byte of TVEC2 is 40 or less
*
         LA    R01,K80SHUFF       80-byte field to be shuffled
         LA    R02,TVEC2
         BAL   R14,ESHUFFLE
         MVC   K80SHUFF,WORK80
K80SUB   EQU   *
         LA    R01,REC80
         LA    R02,K80SHUFF
         LA    R03,80
K80SUBLP EQU   *
         BAL   R14,BYTESUB
         LA    R01,1(,R01)
         LA    R02,1(,R02)
         BCT   R03,K80SUBLP
*-------------------------------------*
*    TVEC2 contains a list of 80 1-byte offsets. Use these to         *
*    shuffle the bytes of the input record in the inverse way to      *
*    shuffling performed during the encryption process.               *
*-------------------------------------*
         LA    R01,REC80          80-byte field to be shuffled
         LA    R02,TVEC2
         BAL   R14,DSHUFFLE
         MVC   REC80,WORK80       Move shuffled record to input rec.
*-------------------------------------*
*    Do rotate of input record.                                       *
*-------------------------------------*
         BAL   R14,REC80ROT       Rotate record segments
*-------------------------------------*
*    TVEC1 contains a list of 80 1-byte offsets. Use these to         *
*    shuffle the bytes of the input record in the inverse way to      *
*    shuffling performed during the encryption process.               *
*-------------------------------------*
         LA    R01,REC80          80-byte field to be shuffled
         LA    R02,TVEC1
         BAL   R14,DSHUFFLE
         MVC   REC80,WORK80       Move shuffled record to input rec.
         LM    R00,R15,DECRYPT_SAVEAREA   Restore all regs.
         BR    R14
*
ROTATE   EQU   *
*-------------------------------------*
*    ROTATE will perform a rotation of a field that is a multiple of  *
*    4 bytes in length, in 4-byte increments of size.                 *
```

```
*                                                                     *
*    The direction of rotation is determined by the high-order bit    *
*    of the CBYTE field. If "Ø" the rotation is to the left, else to  *
*    the right. NOTE: If the area being rotated is in REC8Ø then the  *
*    direction of rotation depends on whether this is an encryption   *
*    or decryption run, since this operation has to be reversible.    *
*                                                                     *
*    Bits 1 and 2 of CBYTE are ignored.                               *
*                                                                     *
*    Bits 3 through 7 of CBYTE are the number of bits to be shifted   *
*    (from Ø to 31). If all these bits are zero the routine is        *
*    exited without any rotation taking place.                        *
*                                                                     *
*    On entry RØ2 must point to the start of the field to be          *
*    rotated and RØ3 should contain the number of words to be         *
*    rotated.                                                         *
*                                                                     *
*    If the word count is zero, or the number of bits                 *
*    to be shifted is zero, the subroutine is immediately exited.     *
*                                                                     *
*    Registers 4 through 9 are used as work registers and will be     *
*    preserved, then restored on exit from this subroutine.           *
*                                                                     *
*-------------------------------------*
         LTR   RØ3,RØ3                Test the word count value
         BZR   R14                    Exit if word count is zero
*
         MVC   WORKBYTE,CBYTE
         NI    WORKBYTE,B'ØØØ11111'   Only lower 5 bits required for
         CLI   WORKBYTE,Ø             Check low-order 5 bits
         BNH   ROTEXCB                 and exit if zero
*
         STM   RØ4,RØ9,ROT_REGS       Save work registers
         LR    RØ4,RØ2                Copy area address to RØ4
         LR    RØ5,RØ3                Copy word count to RØ5
         XC    ROT_WORK,ROT_WORK      Clear work word
*
         CLI   REC8ØIND,1             Are we rotating REC8Ø ?
         BNE   ROTNORM
         CLI   RUNIND,C'E'            Encrypt run ?
         BE    ROTNORM
         TM    CBYTE,B'1ØØØØØØØ'      Check high-order bit
         BO    ROTLEFT                If set, go to rotate left
         B     ROTRIGHT
*
ROTNORM  EQU   *
         TM    CBYTE,B'1ØØØØØØØ'      Check high-order bit
         BO    ROTRIGHT               If set, go to rotate right
*-------------------------------------*
*    Rotate area to the left. RØ4=>area, RØ5 contains word count.     *
```

```
*-------------------------------------*
ROTLEFT  EQU   *
         MVC   ROTLSLDL+3(1),WORKBYTE Set up shift bits
         BCTR  RØ3,Ø                  Reduce word count by 1 and
         SLL   RØ3,2                   multiply by 4
         AR    RØ4,RØ3                Point to last word
         ST    RØ4,LASTWORD           Save this address
ROTLLOOP EQU   *
         LA    RØ8,Ø                  Clear the even-numbered reg.
         L     RØ9,Ø(RØ4)             Load word to be shifted
ROTLSLDL SLDL  RØ8,Ø
         ST    RØ9,Ø(RØ4)
         XC    Ø(4,RØ4),ROT_WORK
         ST    RØ8,ROT_WORK
         SH    RØ4,=H'4'              Next word back to be shifted
         BCT   RØ5,ROTLLOOP           Loop around
*
         L     RØ4,LASTWORD
         XC    Ø(4,RØ4),ROT_WORK
*
         B     ROTEXIT
*
ROTRIGHT EQU   *
         MVC   ROTRSRDL+3(1),WORKBYTE Set up shift bits
*-------------------------------------*
*    Rotate area to the right. RØ4=>area, RØ5 contains word count.    *
*-------------------------------------*
ROTRLOOP EQU   *
         LA    RØ9,Ø                  Clear the odd-numbered reg.
         L     RØ8,Ø(RØ4)             Load word to be shifted
ROTRSRDL SRDL  RØ8,Ø
         ST    RØ8,Ø(RØ4)
         XC    Ø(4,RØ4),ROT_WORK
         ST    RØ9,ROT_WORK
         LA    RØ4,4(,RØ4)            Next word to be shifted
         BCT   RØ5,ROTRLOOP           Loop around
*
         XC    Ø(4,RØ2),ROT_WORK
*
ROTEXIT  LM    RØ4,RØ9,ROT_REGS       Restore work registers
ROTEXCB  XC    CBYTE,Ø(RØ2)           Modify CBYTE
         XC    CBYTE,1(RØ2)           Modify CBYTE
         XC    CBYTE,2(RØ2)           Modify CBYTE
         XC    CBYTE,3(RØ2)           Modify CBYTE
         BR    R14
*-------------------------------------*
*    Byte addition (ignore any carry)                                 *
*    Register 1 points to target byte                                 *
*    Register 2 points to source byte                                 *
*-------------------------------------*
```

```
BYTEADD  EQU   *
         LA    RØ5,Ø
         IC    RØ5,Ø(RØ1)
         MVI   HALFW,Ø
         MVC   HALFW+1(1),Ø(RØ2)
         AH    RØ5,HALFW
         STC   RØ5,Ø(RØ1)
*
         BR    R14
*-------------------------------------*
*    Byte subtraction                                          *
*    Register 1 points to target byte                          *
*    Register 2 points to source byte                          *
*-------------------------------------*
BYTESUB  EQU   *
         MVI   HALFW,1
         MVC   HALFW+1(1),Ø(RØ1)
         LH    RØ5,HALFW
         MVI   HALFW,Ø
         MVC   HALFW+1(1),Ø(RØ2)
         SH    RØ5,HALFW
         STC   RØ5,Ø(RØ1)
*
         BR    R14
*-------------------------------------*
*    Byte shuffle of input record - Encryption phase           *
*    Register 1 must point to the 8Ø-byte field to be shuffled.*
*    Register 2 must point to TVEC1,TVEC2 or TVEC3.            *
*-------------------------------------*
ESHUFFLE EQU   *
         LA    RØ3,8Ø
         LA    RØ4,WORK8Ø
SHUFFLEE EQU   *
         LA    RØ5,Ø
         IC    RØ5,Ø(RØ2)         Offset byte from TVECn
         LR    RØ6,RØ1            Start of 8Ø-byte record
         AR    RØ6,RØ5            Point to record byte and
         MVC   Ø(1,RØ4),Ø(RØ6)     move it to WORK8Ø byte
         LA    RØ2,1(,RØ2)        Point to next offset byte
         LA    RØ4,1(,RØ4)        Point to next WORK8Ø byte
         BCT   RØ3,SHUFFLEE
*
         BR    R14
*-------------------------------------*
*    Byte shuffle of input record - Decryption phase           *
*    Register 1 must point to the 8Ø-byte field to be shuffled.*
*    Register 2 must point to TVEC1,TVEC2 or TVEC3.            *
*-------------------------------------*
DSHUFFLE EQU   *
         LA    RØ3,8Ø
```

```
         LR    RØ4,RØ1            Start of 8Ø-byte record
SHUFFLED EQU   *
         LA    RØ5,Ø
         IC    RØ5,Ø(RØ2)         Offset byte from TVECn
         LA    RØ6,WORK8Ø
         AR    RØ6,RØ5            Point to output record byte and
         MVC   Ø(1,RØ6),Ø(RØ4)     move REC8Ø byte to it
         LA    RØ2,1(,RØ2)        Point to next offset byte
         LA    RØ4,1(,RØ4)        Point to next REC8Ø byte
         BCT   RØ3,SHUFFLED
*
         BR    R14
*-------------------------------------*
*    Update a CBYTE value from the XOR of all the bytes of KEY8Ø.     *
*-------------------------------------*
CBKEY8Ø  EQU   *
         LA    RØ4,8Ø
         LA    RØ5,KEY8Ø
CBLOOP   EQU   *
         XC    CBYTE,Ø(RØ5)
         LA    RØ5,1(,RØ5)
         BCT   RØ4,CBLOOP
*
         BR    R14
*-------------------------------------*
*    Rotate the word segments of REC8Ø.                               *
*-------------------------------------*
REC8ØROT EQU   *
         ST    R14,SAVER14
         MVC   SAVEBYTE,CBYTE   Preserve CBYTE value
         MVI   REC8ØIND,1       Tell ROTATE subroutine it's REC8Ø
*                                that we want to rotate
         LA    RØ5,Ø
         LA    RØ7,KEY8Ø
         LA    RØ8,1Ø
R8ØLOOP  EQU   *                Create dividend in RØ5
         X     RØ5,Ø(RØ7)
         AL    RØ5,4(RØ7)
         LA    RØ7,8(,RØ7)
         BCT   RØ8,R8ØLOOP
*
         LR    R15,RØ5          Save dividend value
         LA    RØ4,Ø
         LA    RØ1,21           Divisor in RØ1
         DR    RØ4,RØ1
         STM   RØ4,RØ5,REMQUOT  Save remainder and quotient
         LA    RØ6,KEY8Ø
         LA    RØ2,REC8Ø
         LR    RØ9,RØ2          Start of REC8Ø into RØ9
R8ØROTLP EQU   *
```

```
         L     RØ3,REMQUOT      Use remainder as number of words
         LTR   RØ3,RØ3          As soon as we get a zero
         BZ    R8ØLAST           remainder then exit
         LR    RØ8,RØ3
         SLL   RØ8,2            Multiply number of words by 4 to
*                                create an offset into REC8Ø
         MVC   CBYTE,Ø(RØ6)     Use KEY8Ø byte value as CBYTE
         BAL   R14,ROTATE
*
         S     RØ1,REMQUOT      Subtract remainder from original
*                                divisor to create new divisor
         CH    RØ1,=H'1'        Check for one
         BNH   R8ØEXIT          BIF not high, exit
*
         AR    RØ9,RØ8          Add offset to next word to be
         LR    RØ2,RØ9           processed
         LA    RØ4,Ø
         LR    RØ5,R15          Original dividend plus
         AL    RØ5,REMQUOT       remainder plus
         AL    RØ5,REMQUOT+4      quotient for new dividend
         DR    RØ4,RØ1
         STM   RØ4,RØ5,REMQUOT  Save remainder and quotient
         LA    RØ6,1(,RØ6)      Point to next byte of KEY8Ø
         B     R8ØROTLP
*
R8ØLAST  EQU   *
         LR    RØ3,RØ1          Divisor minus 1 is number of words for
         BCTR  RØ3,Ø             the final rotation
         MVC   CBYTE,Ø(RØ6)     Use KEY8Ø byte value as CBYTE
         BAL   R14,ROTATE
R8ØEXIT  EQU   *
         MVI   REC8ØIND,Ø
         MVC   CBYTE,SAVEBYTE   Restore CBYTE value
         L     R14,SAVER14
*
         BR    R14
*-------------------------------------*
*    Set up an initial value in R1Ø pointing into TAB16ØØØ.          *
*    Use the first 1Ø words of KEY8Ø to calculate an offset.          *
*-------------------------------------*
SETUPR1Ø EQU   *
         L     RØ5,KEY8Ø
         X     RØ5,KEY8Ø+4
         AL    RØ5,KEY8Ø+8
         X     RØ5,KEY8Ø+12
         AL    RØ5,KEY8Ø+16
         X     RØ5,KEY8Ø+2Ø
         AL    RØ5,KEY8Ø+24
         X     RØ5,KEY8Ø+28
         AL    RØ5,KEY8Ø+32
```

```
         X     R05,KEY80+36
*
         LA    R04,0
         D     R04,F15921
         LA    R10,TAB16000
         AR    R10,R04
*
         BR    R14
*-------------------------------------*
*    Set up 2 transposition vectors using TAB16000 data. Register 10  *
*    is used to provide a pointer that is always within the table.    *
*-------------------------------------*
TVECSET  EQU   *
         L     R05,KEY80+40
         X     R05,KEY80+44
         AL    R05,KEY80+48
         X     R05,KEY80+52
         AL    R05,KEY80+56
*
         LA    R04,0
         D     R04,F15921            Calculate an offset
         LA    R10,TAB16000
         AR    R10,R04               Point into TAB16000
*
         XC    TVEC1,TVEC1
         XC    TVEC2,TVEC2
         XC    USED80,USED80
         LA    R02,TVEC1
         LA    R09,2
TRNLOOP  EQU   *
         LA    R03,80
TRNLP1   EQU   *
         MVC   OFFBYTE,0(R10)    Save offset byte value
         CLI   OFFBYTE,79        Check value is 79 or less for
         BNH   TRNLP1A            use as offset into 80-byte record
*
         LA    R04,0             At this point OFFBYTE is greater
         LA    R05,0              than 79 so divide it by 80 and
         IC    R05,OFFBYTE         use the remainder as
         D     R04,=F'80'           a new OFFBYTE value
         STC   R04,OFFBYTE
```

Editor's note: This article will be continued in the next issue.

Philip Rigby (UK)

The RACF marketplace

The RACF marketplace *is devoted to keeping our readers informed of products and services that are offered for the RACF environment. These might consist of software utilities, text books, education classes, etc – if it helps the RACF administrator, we shall endeavour to tell him or her about it.*

You should be aware that, while it is our usual policy not to publish articles with a marketing flavour, this is not true for The RACF marketplace*, although we shall do our best to avoid some of the excesses to which sales people are often prone. The information published here is taken directly from material provided by the suppliers of RACF-related products and services themselves. Xephon accepts no payment from the suppliers of products featured on this page.*

* * *

Software AG has announced its EntireX middleware for the OS/390, integrating DCOM within its own Message Broker technology, and supporting RACF via the SAF interface. There are plans for the product to provide a single sign-on function, making it possible for users to log-on only once in order to reach various applications on other platforms. The Message Broker is also accessible from Web environments, such as ActiveX Controls, and from Java class libraries.

Existing mainframe applications can be enhanced with DCOM interfaces to appear like components within an enterprise-wide component structure.

The EntireX Message Broker includes functions such as the dynamic start of servers under CICS, support for protocols such as LU6.2 and TCP/IP, and the wrapping of existing applications. There's also a Web and security interface, as well as a message-handling interface for MQSeries, along with a system developer's kit and a run-time component.

For further information contact:
Software AG of North America, 11190 Sunrise Valley Drive, Reston, VA 22091, USA.
Tel: (703) 860 5050.
Software AG (UK), Charter Court, 74-78 Victoria Street, St Albans, AL1 3XH, UK.
Tel: (01727) 844 455.

* * *

IBM has enhanced its IMS Performance Analyser so it now includes reports on RACF usage, OSAM sequential buffer statistics, and external subsystem connection statistics. There will be an ISPF CUA user interface for report requests.

The new interface provides command support for experienced ISPF users, customization of the on-line interface, on-line facilities to create, maintain, and submit report requests, then manipulate averages and expectations for management exception reporting in spreadsheet format.

New report requests are based on defaults or modelled on existing reports, with visible defaults, and JCL is automatically generated, but can be modified before submission.

As for the reports, they'll be easier to read, with more consistency between log and monitor reporting.

For further information contact your local IBM representative.

* * *