SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

01001O

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER: 205

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Reference Manual".

(1) Program Number (to be filled in by SPLA)..... 360D-01.0.010

(2) System Type (machine)..... 360/370

(3) Search Key..... Password protection
irreversible security transformation
computer resource authorization
encryption

(4) Programming Systems/Languages..... standard 360/ASSEMBLER

(5) Author's Name and Address..... H. D. Knoble
214 Computer Building
University Park, PA 16802

(6) Direct Technical Inquiries to Name & Address (if different than Author) H. D. Knoble
214 Computer Building
The Pennsylvania State University
University Park, PA 16802

(7) Title of Program..... One-Way Enciphering Algorithm for Password Protection

(8) Submitter's Installation Membership Code..... PSU

(9) Submitter's Own Program Identification and Suffix(Optional).. ---

(10) Primary Subject Code..... Security

(11) Minimum System Requirements S/360 2K bytes memory

(12) New or Revision Code (if revision, show prior Program Number in Item 1) 10/76

(13) Year Completed..... 1976

(14) Date of Submittal..... 10/76

(15) Documentation (number of original pages submitted)..... 10 pages*

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

Revised 4/74

* The implemented computer program has been copyrighted by the author in an effort to assess its usability. Your cooperation in requesting permission to distribute the program across computer installations will therefore be appreciated.

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

Subroutine PURDY is a re-enterable system utility program which evaluates a family of mathematically sound, one-way enciphering functions with known properties. The algorithm is implemented here to enable 8-character passwords to be irreversibly enciphered for security applications (computer resource authorizations). Unlike many existing methods used for several current operating system security applications (e.g. MVS passwords), this method does not rely on keeping the algorithm or list of enciphered keys secret; this is true because no known algorithm exists to invert the enciphering function, and even if one were discovered, deciphering a key would still require, on the average, many years of CPU time on modern, high-speed equipment. Because the family enciphering functions upon which this routine is based has essentially an infinite number of parameterizations, this implementation allows computer resource authorization to be independent and unique across applications.

15 tape files on distribution tape.

(Please attach additional pages if necessary).....Total pages attached 0

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date X Herman D. Knoble

(18) Signature of Installation Addressee X Herman D. Knoble

Installing the One-way Encryption Subprogram, PURDY

The distribution tape, VOL=SER=ENCRYP, is standard labeled and contains five data sets, each with DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200) as follows:

LABEL=1,DSN=SOURCE, contents = assembler source code for the encryption routine, PURDY.

LABEL=2,DSN=KOMP, contents = assembler routine KOMP needed for TESTDECK.

LABEL=3,DSN=TESTDECK, contents = FORTRAN test deck.

LABEL=4,DSN=OBJECT, contents = object code corresponding to DSN=SOURCE.

LABEL=5,DSN=TESTDATA, contents = test data needed for TESTDECK.

I) The system may be installed using any 360/370 assembler which will batch source modules. For example, using Waterloo's ASSEMBLER (G) the routines may be installed as follows:

```
// EXEC ASMGCL,PARM.ASM='BATCH',PARM.LKED='XREF,RENT'
//ASM.SYSIN DD UNIT=2400,VOL=SER=ENCRYP,DSN=SOURCE,DISP=(OLD,PASS)
//LKED.SYSLMOD DD DSN=SYS1.SYSTEM.LIBRARY(PURDY),DISP=(OLD,KEEP),
// VOL=,UNIT=
```

If no such assembler is available, the file SOURCE may be punched and appropriate JCL inserted to enable multiple assemblies followed by a linkedit, similar to that illustrated above. An alternative procedure would be to linkedit the object modules on file OBJECT as follows:

```
// EXEC LKED,PARM.LKED='XREF,RENT'
//SYSLMOD DD DSN=SYS1.SYSTEM.LIBRARY(PURDY),DISP=(OLD,KEEP)
//SYSIN DD DSN=OBJECT,UNIT=2400,VOL=SER=ENCRYP,LABEL=4
```

II) Once the routine has been installed, it may be tested as follows: (page 6 of the documentation shows correct output)

```
// EXEC ASMFC
//ASM.SYSGO DD DSN=&&LOADSET,DISP=(,PASS),SPACE=(CYL,2)
//ASM.SYSIN DD UNIT=2400,VOL=SER=ENCRYP,LABEL=2,DSN=KOMP,
// DISP=(OLD,PASS)
// EXEC FORTGCLG
//FORT.SYSIN DD UNIT=2400,VOL=SER=ENCRYP,LABEL=3,DSN=TESTDECK,
// DISP=(OLD,PASS)
//LKED.SYSLIB DD
// DD DSN=SYS1.SYSTEM.LIBRARY,DISP=SHR
//GO.FT05F001 DD UNIT=2400,VOL=SER=ENCRYP,LABEL=5,DSN=TESTDATA
```

An Efficient One-way Enciphering Algorithm for Password Authorization

by H. D. Knoble (PSU)
The Pennsylvania State University Computation Center
Computer Building, University Park, Pa. 16802. 814-863-0422

Session S405
Tuesday, August 17, 1976 - 3:30 pm

INTRODUCTION

The purpose of this paper is to introduce and document a re-enterable system utility program which evaluates a family of mathematically sound one-way enciphering functions[1] with known theoretical deciphering times. The supporting algorithm[2] is implemented here to enable 8-character passwords to be irreversibly enciphered for security applications on an IBM 360/370 host computer system. Unlike some existing methods which rely on keeping the algorithm and list of enciphered keys secret, this scheme, while remaining simple in principle, defys any known method of mathematical breakin even when these resources are openly available. Because the family of enciphering functions upon which it is based has essentially an infinite number of parameterizations, this implementation allows computer resource authorization to be independent and unique across applications.

USE

At the expense of a few more program parameters and a loop Subroutine Purdy may easily be coded to evaluate a more general f(x) mod P than equation (1). This has not been done in the distributed version for reasons of execution efficiency. Therefore this section is presented in two parts: (I) Use of the Distributed Program; and (II) Modifications to Evaluate Other Classes of Enciphering Functions.

I) Use of the Distributed Program

Once the subprogram is made available to the calling program by placing it in an appropriate library, linkage is effected via a standard OS/VS calling sequence:

```
LA   13,SAVEAREA            point 13 at your save area
CALL PURDY,(X,FX,PARMWORK)
LTR  15,15                  test the condition code
BZ   NORMAL                 zero is normal
```

where:

X is given as the address of an 8-character password to be enciphered;

FX is the address at which the 8-character enciphered password corresponding to f(X) in equation (1) is returned.

PARMWORK is given as a 200 word array as follows:

PARMWORK+00 - NP, the precision or number of computer words per password; this must be equal to the integer 2.

PARMWORK+04 - A, the value of the prime offset constant in the equation, $P=2^{64}-A$, chosen to make P prime; the value of A therefore must be a positive integer in the interval $[59, 2^{31}-1]$.

PARMWORK+08 - Q, the number of bits in an 8-character (2-word) password; this must be equal to the integer 64.

PARMWORK+12 - N, the polynomial degree corresponding to equation (1); N should be chosen as a non-prime large positive integer (say larger than 10^5).

```
*BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
*---------BEGIN POLYNOMIAL EVALUATION.
                                        .
                                        .
     The input password is stored at the symbolic address, X, and
        the final result must be stored in the multiprecision
                integer addressed by the symbol, FX.
                                        .
                                        .
*----------END POLYNOMIAL EVALUATION.
*EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
```

These statements between the delimiting comments are simply re-enterable calls to ADDP, MULTP, and EXPP ordered to effect the desired function evaluation. For example, if N=2*N1-1, the following sequence will evaluate equation (1) after appropriate factoring as:

$$(2)\quad f(x) = x^{N1-1}(x(x^{N1-1}+A_1))+x(A_4+x(A_3+A_2*x)) \bmod P$$

```
*BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
*----------------BEGIN POLNOMIAL EVALUATION.
*
*         FACTOR X**N+A1*X**N1=X**(N1-1)*(X*(X**(N1-1)+A1))
*         SINCE IN THIS CASE N WAS CHOSEN SUCH THAT N=2*N1 - 1.
*
          RENTCALL EXPP,(X,N1M1,T1)
          RENTCALL ADDP,(T1,A1,T2)
          RENTCALL MULTP,(X,T2,T2)
          RENTCALL MULTP,(T1,T2,T2)
*
*         FACTOR A2*X**3+A3*X**2+A4*X = X*(A4+X*(A3+A2*X)).
*
          RENTCALL MULTP,(A2,X,T1)
          RENTCALL ADDP,(T1,A3,T1)
          RENTCALL MULTP,(X,T1,T1)
          RENTCALL ADDP,(T1,A4,T1)
          RENTCALL MULTP,(X,T1,T1)
*
*         ADD BOTH FACTORS AND THE CONSTANT TERM.
*
          RENTCALL ADDP,(T1,T2,T2)
          RENTCALL ADDP,(T2,A5,FX)
*
*----------------END POLYNOMIAL EVALUATION.
*EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
```

This identity in N, N1 if applied to the distributed version of Subroutine PURDY, would enable a call to EXPP to be eliminated thus reducing running time by about 30% (from 7 ms to 5 ms on an IBM 370/168). Refering to the assembler listing note that the PARMWORK symbols T1 and T2 are used as temporaries, and those corresponding to the polynomial parameters N, N1, N1-1, P, X, etc. are initialized before control passes to statements between the delimiting comments.

Note that in general $f(x)=f(P+x)$ for $x=0,1,2,\ldots,(A-1)$, $f(0)$=the constant term (A_5 in this example), and $f(1)=A_1+A_2+A_3+\ldots \bmod P$; $f(1)=2^{64}-571$ for this example. The test deck distributed with Subroutine PURDY uses the polynomial parameters of this example to evaluate equation(1).

METHOD AND DISCUSSION

I) Computing Time Advantage

The algorithm used in Subroutine PURDY to evaluate polynomials $f(x) \bmod P$ is described in [2]. It essentially capitalizes on a special case of Knuth's[3] theorems 4.3.1-A and -B. The running time advantage of this algorithm over general multiprecision integer division to compute $X*Y \bmod P$ grows as the square of the precision; The advantage for $X+Y \bmod P$ is substantially greater. In particular, if the routines ADDP and MULTP would utilize a general multiprecision division routine, then the execution time for the EXAMPLE function would increase from about 5 ms to 9 ms on an IBM 370/168.

II) Enciphering Function Properties

The polynomial enciphering functions were suggested by Purdy[1]. He has developed breakin time formulae and shows that breakin time by trial and error is far shorter than by mathematical methods. The mathematical soundness of Purdy's enciphering functions is that computing their inverse entails the programming of a congruential polynomial root solving algorithm which is yet undiscovered. Even if such a program existed, Purdy has shown that several years of high-speed computer time would be necessary to utilize it to break a password. Trial and error breakin is dependent on the password length, the number of used passwords, and the degeneracy of the enciphering function. Assuming 10000 used passwords the expected trial and error breakin time for 64-bit passwords utilizing Subroutine PURDY to evaluate the EXAMPLE function is about two months on a 370/168 computer. While it has been suggested that this time may be reduced by future computing technology[4], it is also true that typical operating system security as well as most computer room security offer much less expensive breakin paths.

While such a facility is useful in principle, it is somewhat lacking from the viewpoint that any potential subsystem user scrupulously or inadvertently may be able to password protect his neighbors' unprotected data sets with only knowledge of its name (and perhaps a logon sequence). Assigning unique "initial passwords" to read/write protect every data set may not only be operationally tedious, but can burden every subsytem user with repetitiously specifying passwords on data sets that are naturally public, e.g. a mailbox, or on data sets that offer minimal security risks, e.g. a scratch tape. A scheme involving multiple passwords or reserved data set names to be treated as special cases add to system overhead and maintenance. This hypothetical "password paranoia" can be avoided in part by introduction of a third authorization level, namely "no protection." That is, by assigning a password to such a data set once, but making its authorization status "unprotected", the owner renders it thereafter publically readable and writable without authorization, but retains the key to change the authorization status.

CAPABILITIES AND LIMITATIONS

I) General.

This version of Subroutine Purdy supports 64-bit passwords (NP=2, Q=64). The algorithm in general requires that Q be an exact multiple, NP>1, of the number of bits in an integer machine word. Extending the algorithm for larger passwords requires higher precision primitives[3] and correspondingly larger array sizes for multiprecision integers (ie. vectors of length NP, NP+1, and 2*NP instead of 2,3,4). For degeneracy to be computable the parameter A for $P=2^Q-A$ must be chosen to make P prime; the primality of P is not checked by the program. Some valid values for A to force P prime for various values of Q may be found in Knuth[3].

III) Timing Considerations

The routines are listed below with the approximate number of machine cycles required for linkage and execution (see [2]).

Program Name	Machine Cycles
PURDY(as distributed)	84840
ADDP	440
MULTP	1160
EXPP	39000

INFORMATION REQUEST

In order to assess usability of Subroutine PURDY, you are kindly requested to fill out, fold, staple and return this sheet to the author (address on reverse).

Name, Affiliation, Title and Address:

_____

_____

_____

_____

_____

****

Comments: