****

COMPUTER
SUPPLEMENT #21

****

In this issue:

THE QUAGMIRES, 1 AND 2 — G4EGG provides a program to aid in the solution of Quagmire ciphers.

ENIGMA 95 — ESSAYONS and LANAKI introduce a QBASIC program to simulate an Engima machine, with enhancements.

dBASE ADDITIONS — Ken Madl gives some additions and shortcuts to PARROT's dBASE routines.

AN INTRODUCTION TO MODERN CRYPTOLOGY — G. H. Foot has written an excellent introduction to modern crypto issues.

A TRIP TO THE NSA MUSEUM — BROADSWORD relates the details of a recent trip to NSA's public museum.

ENIGMA CONTEST — DAEDALUS explains how he solved a publishing house contest.

CURRENT PGP VERSIONS — A guide to the various versions and modifications of a popular cryptographic program.

BASIC PEEKS, POKES AND SUBROUTINES — If you're using BASIC, this list may help you perform minor miracles.

Plus: News and notes for computerists interested in cryptography, and cryptographers interested in computers.

Published in association with the American Cryptogram Association

INTRODUCTORY MATERIAL

The ACA and Your Computer (1p). Background on the ACA for computerists. (As printed in *ACA and You*, 1988 edition; [Also on Issue Disk #11]

Using Your Home Computer (1p). Ciphering at the ACA level with a computer. (As printed in *ACA and You*, 1988 edition).

Frequently Asked Questions (approx. 20p) with answers, from the Usenet newsgroup `sci.crypt`.

REFERENCE MATERIAL

BASICBUGS - Bugs and errors in `GW-BASIC` (1p). [Also on Issue Disk #11].
BBSFILES - List of filenames and descriptions of cryptographic files available on the ACA BBS (files also available on disk via mail).

BIBLIOG — A bibliography of computer magazine articles and books dealing with cryptography. (Updated August 89). [available on Issue Disk #11].

CRYPTOSUB - Complete listing of Cryptographic Substitution Program as published by PHOENIX in sections in *The Cryptogram* 1983–1985. (With updates from CS #2,3). [available on Issue Disk #3].

DISKEX - A list of programs and reference data available on disk in various formats (Apple—Atari—TRS80—Commodore—IBM—Mac). Revised March 1990.

ERRATA sheet and program index for Caxton Foster's *Cryptanalysis for Microcomputers* (3p). (Reprint from CS #5,6,7 and 9) [disk available from TATTERS with revised programs].

BACK ISSUES

$2.50 per copy. All back issues from #1 to #20 are available from the Editor.

ISSUE DISKS AND CD-ROM

$5 per disk; specify issue(s), format and density required. All issues presently fit on two IBM High Density 3.5 inch (1.44M) floppy disks, archived with PKZIP. For other disk formats, ask. Disks contain programs and data discussed in the issue. Programs are generally BASIC or Pascal, and almost all executables are for IBM PC–compatible computers. Issue text in LaTeX format is available for issues 16 to current. CD-ROM in MS-DOS format also available, containing most ACA-related material. Available from the Editor.

TO OBTAIN THESE MATERIALS

`Write to:`
`Dan Veeneman`
`PO Box 2442`
`Columbia, Maryland`
`21045-2442, USA.`

`Or via Electronic Mail:`
`dan@decode.com`

Allow 6–8 weeks for delivery. No charge for hard copies, but contributions to postage appreciated. Disk charge $5 per disk; specify format and density required. ACA Issue Disks and additional crypto material resides on `Decode`, the ACA Bulletin Board system, `+1 410 730 6734`, available 24 hours a day, 7 days a week, 300/1200/2400/9600/14400/28800 baud, 8 bits, No Parity, 1 stop bit. All callers welcome.

SUBSCRIPTION

Subscriptions are open to paid-up members of the American Cryptogram Association at the rate of US$2.50 per issue. Contact the Editor for non-member rates. Published three times a year or as submitted material warrants. Write to `Dan Veeneman, PO Box 2442, Columbia, MD, 21045-2442, USA.` Make checks payable to Dan Veeneman. UK subscription requests may be sent to G4EGG.

CHECK YOUR SUBSCRIPTION EXPIRATION by looking at the `Last Issue =` number on your address label. You have paid for issues up to and including this number.

The Quagmires, 1 and 2

G4EGG

The Quagmire ciphers, (*Practical Cryptanalysis*, page 24) types 1 and 2, are similar, and mainly require the interchange of plaintext and cleartext alphabets in the solution method given in the above book. The following programme was developed as an aid in solving the type 1, but the few changes necessary to allow use with type 2 have been added.

The programme details are:

Lines 10 - 120
General set-up and setting of constants.

Lines 900 - 1000
Notes on the use of the programme, displayed on screen when run, and selection of type, i.e. 1 or 2.

Lines 1100 - 1850
Get cleartext and crib from disk, keyboard, or example. This is just my way of doing it, and not necessarily the best!

Lines 1890 - 1990
Convert crib to pattern, and drag through cleartext to find fits. Note fit position(s).

Lines 2100 - 2140
Prepare cleartext for printing on screen in columns of period width.

Lines 2200 - 2210
Select which fit to try. If only one, go straight into initial display and crib fitting, plus other determined letters.

Lines 2250 - 2520
Initial display and crib fitting. Note that the screen size limits amount of text that can be displayed. The programme gives up to the first 18 lines of period widths of text, so the last few lines may be omitted from the screen.

Lines 2600 - 2660
Select next action. Do alphabet cross references, get more cleartext/plaintext pairs, or start again with new crib position.

Sub routines: **Lines 200 - 350**
Change sections of text to patterns for cross checking.

Lines 400 - 500
Put plaintext corresponding to cleartext into plaintext block

Lines 600 - 800
Test letter spacing in alphabets, and transfer similar spacing to other alphabets where letters fit. Also put newly determined letters in plaintext block.

Lines 400, 725, 1000, 1010 1120, 1180, 1840, 2400 and 2420 contain the code to direct type 1 or type 2.

The programme is written in QuickBasic but to make it more compatible with other dialects line numbering has been maintained. Two functions of QuickBasic that have been used may not be available in other dialects. They are `UCASE$()` which converts a string to upper case, and `LCASE$()` that does the opposite. The "classic" method alternative for `UCASE$()` is:

```
 nnn A$="qweRTYuio": for Z = 1 to LEN(A$): MID$(A$,Z,1)=CHR$(ASC(MID$(A$,Z,1)
     AND 223)): NEXT

For LCASE$() MID$( part of the above becomes:
 MID$(A$,Z,1)=CHR$(ASC(MID$(A$,Z,1) AND 223) - 32)
```

To use the programme with the built in example: When RUN, comments on the use are given on screen, and a request to select TYPE 1 or TYPE 2 Quagmire. The screen will show:

```
  An aid to solving the QuagmireI and QuagmireII ciphers.     (G4EGG)


  If input data is from a file, the file name extension must be .CT$
  (It is not necessary to enter the extension when giving file name.)

  Period may be from 4 to 9.

  Examination of both PT columns and alphabet grid should indicate more
  PT letters. These are added by entering the PT col. number or the grid
  row number, (both are the same) and then CT letter and PT letter.

  Entering 99 instead of a col. will cross reference the known letters.
  This may be slow, especially when there are a lot of letters in!

  Enter >99 to restart/new crib position, and 0 to end.

                Quag (*1) or Quag (2) ?
```

Enter 1. Input of cleartext, crib and period are then requested. RETURN key lets programme supply the required data. Screen then shows:

```
  An aid to solving the QuagmireI and QuagmireII ciphers.     (G4EGG)

   Enter ciphertext: ('ENTER' for example, 'F' if data on disc)?
RGEESEDIZMYUXIZUXBRJSPLIXWYDSFWHDARDSTDGQYUFQUWWPJFGQOBXFJXEQGXXQKXLRFLBAERUIUW
FOXMFFWSXLJUSKXESOJHARVTQUXUTDZVFOCPOFMUTVVJXVTCFZVHUCDIZUUFRSTX



   Enter crib: ? EARTHMILESFROMTHE

   And period: ? 8
.....................................................................
.....................
```

The periods are run to show that something is going on, crib dragging is not very fast! In this case, only one position matches, so the programme goes straight into the next stage, the display of cleartext, the plaintext grid, and the alphabets:

```
    An aid to solving the QuagmireI and QuagmireII ciphers.     (G4EGG)

RGEESEDI      1 l..h..i. 1
ZMYUXIZU      2 ..ro.... 2          ct=   ABCDEFGHIJKLMNOPQRSTUVWXYZ
XBRJSPLI      3 ........ 3          pt  1 .....o.........h.l.......... 1
XWYDSFWH      4 ..r...r. 4          pt  2 .........m................... 2
DARDSTDG      5 ......i. 5          pt  3 ...f.i.................tr.... 3
QYUFQUWW      6 ....eart 6          pt  4 ....h.l.............o........ 4
PJFGQOBX      7 hmilesfr 7          pt  5 ................e............ 5
FJXEQGXX      8 omthe..r 8          pt  6 ..............s.....a........ 6
QKXLRFLB      9 ..t..... 9          pt  7 .f.i.................tr...... 7
AERUIUWU     10 ...o.ar. 10         pt  8 ..f.i.................tr..... 8
FOXMFFWS     11 o.t...r. 11               ABCDEFGHIJKLMNOPQRSTUVWXYZ
XLJUSKXE     12 ...o...i 12
SOJHARVT     13 ......t. 13
QUXUTDZV     14 ..to.... 14             Col. No.
FOCPOFMU     15 o....... 15
TVVJXVTC     16 .......f 16
FZVHUCDI     17 o.....i. 17
ZUUFRSTX     18 .......r 18
```

Now for the guess-work, (or intelligent intuition!) Line 1 of the alphabets shows `...h.l....` and line 3 has `..f.i...` It is reasonable to assume that these two will not marry, as `i`, `j`, and/or `k` must fit into one space (`..h.l..`) So, `g` must fill the `..f.i..` gap.

To do this, enter `3` for Col. No. `e` for cleartext letter and `g` for plaintext letter. Row 1 of cleartext block begins `l..h.i` Try `light`? Col. No. 2, cleartext letter `g`, plaintext letter `i`, etc. for cols 3 and 5. Then `99` in Col. No. to add this new information to all alphabets and plaintext grid. Screen will then be as:

```
    An aid to solving the QuagmireI and QuagmireII ciphers.     (G4EGG)

RGEESEDI      1 light.i. 1
ZMYUXIZU      2 ..ro...e 2          ct=   ABCDEFGHIJKLMNOPQRSTUVWXYZ
XBRJSPLI      3 ....t... 3          pt  1 .....o.........h.l.......... 1
XWYDSFWH      4 .er.t.rm 4          pt  2 ....fgi..m............e.tr... 2
```

```
DARDSTDG      5 ....t.i. 5
QYUFQUWW      6 .t..eart 6
PJFGQOBX      7 hmilesfr 7
FJXEQGXX      8 omthe..r 8
QKXLRFLB      9 ..t..... 9
AERUIUWU     10 .f.o.are 10
FOXMFFWS     11 o.t...r. 11
XLJUSKXE     12 ...ot..i 12
SOJHARVT     13 ....i.t. 13
QUXUTDZV     14 ..tor... 14
FOCPOFMU     15 o......e 15
TVVJXVTC     16 ..e...ef 16
FZVHUCDI     17 ore...i. 17
ZUUFRSTX     18 ......er 18
```

```
pt  3 ...fgi..m...........e.tr.... 3
pt  4 ....h.l.............o........ 4
pt  5 i..m...........e.tr....fg... 5
pt  6 ..............s.....a........ 6
pt  7 .fgi..m...........e.tr...... 7
pt  8 ..fgi..m...........e.tr..... 8
      ABCDEFGHIJKLMNOPQRSTUVWXYZ


   Col. No.
   CT letter S
   PT letter T
```

Now, adding to the `light` in line 1, for `lightning`, and line 16/17 suggests `before` will give enough to solve as the keyword is revealed.

Run again, and select type 2. ENTER allows the built in example to be used, and all screens and actions are as for type 1. With this example, four fits are found. The correct one is No. 2 To help get you into things quickly, the plaintext begins `acatusesits`.

Be careful in inputting the letters, it's cleartext first, then plaintext. In the alphabet grids they are reversed for type 1 and 2.

```
10 ' *** QUAGMIRE-I AID *** MAY 1994    by G4EGG
100 CLEAR : KEY OFF: COLOR 4, 7, 2: CLS : M = 0: P(1) = 8: P(2) = 9
110 D$="12345":AZ$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":T$(1)="pt=":T$(2)="ct="
120 HD$="An aid to solving the QuagmireI and QuagmireII ciphers.    (G4EGG)":
        GOTO 900
198 '        subs start here
199 ' make patterns of text
200 FOR I = 1 TO P: P$(I) = ""
210 FOR J = 0 TO CB - 1 STEP P
220 P$(I) = P$(I) + MID$(C$, J + I, 1)
230 NEXT
240 NEXT
250 FOR I = 1 TO P: TW = 1: TC = 1
260 FOR J = 1 TO LEN(P$(I))
270 A$ = MID$(P$(I), J, 1)
280 B = INSTR(P$(I), A$)
290 IF B > 0 THEN MID$(P$(I), B, 1) = MID$(D$, TW, 1): GOTO 280
300 TW = TW + 1
310 NEXT
320 NEXT: PRINT ".";
330 CBB$ = ""
340 FOR I = 1 TO P: CBB$ = CBB$ + P$(I): NEXT
350 RETURN
```

```
390 ' put letters in alphabets grid
400 IF TYP <> 2 THEN
405   A = INSTR(AZ$, CTL$): IF A = 0 THEN 500
410   MID$(RL$(CL), A, 1) = LCASE$(PTL$)
412 ELSE
416   A = INSTR(AZ$, PTL$): IF A = 0 THEN 500
418   MID$(RL$(CL), A, 1) = LCASE$(CTL$)
420 END IF
430 LOCATE 4 + CL, 49: PRINT RL$(CL)

440 ' put them into PT block
450 FOR L = 1 TO LEN(CC$(CL)): IF L > 19 THEN L = 99: GOTO 490
460 IF MID$(CC$(CL), L, 1) = CTL$ THEN
470   LOCATE 2 + L, 21 + CL: PRINT LCASE$(PTL$)
480 END IF
490 NEXT
500 RETURN

590 '      cross ref. alphabets
600 FOR RM = 1 TO P: LOCATE 4 + RM, 45: PRINT ">"
610   IF RM > 1 THEN LOCATE 3 + RM, 45: PRINT " "
620   FOR RS = 1 TO P: IF RS = RM THEN GOTO 760
630     CL = RS: FOR II = 1 TO 26
640     A$=MID$(RL$(RM),II,1):IF A$="." OR INSTR(RL$(RS),A$)=0 THEN 750
650     ARM = II: ARS = INSTR(RL$(RS), A$): DIF = II - ARS
660     FOR JJ = 1 TO 26: S$ = MID$(RL$(RM), JJ, 1)
670       IF S$ = "." OR INSTR(RL$(RS), S$) > 0 THEN 740
680       SS = JJ - DIF
690       IF SS < 1 THEN SS = SS + 26: GOTO 690
700       IF SS > 26 THEN SS = SS - 26: GOTO 700
710       MID$(RL$(RS), SS, 1) = S$
720       PTL$ = UCASE$(S$): CTL$ = CHR$(64 + SS)
725       IF TYP = 2 THEN SWAP PTL$, CTL$
730       GOSUB 400
740     NEXT
750   NEXT
760 NEXT
770 IF CK = 1 THEN CK = 0: LOCATE 22, 9: PRINT SPACE$(30): GOTO 790
780 NEXT
790 LOCATE 4 + P, 45: PRINT " ": LOCATE 16, 45: PRINT "            "
800 RETURN

890 ' start with notes on programme
900 PRINT : PRINT TAB(10); HD$: PRINT
910 PRINT : PRINT "       If input data is from a file, the file name
    extension must be .CT$"
920 PRINT "       (It is not necessary to enter the extension when giving
    file name.)"
930 PRINT : PRINT "       Period may be from 4 to 9."
940 PRINT : PRINT "       Examination of both PT columns and alphabet grid
    should indicate more"
950 PRINT "       PT letters. These are added by entering the PT col. number
    or the grid"
```

```
960 PRINT "       row number, (both are the same) and then CT letter and PT
    letter."
970 PRINT : PRINT "       Entering 99 instead of a col. will cross reference
    the known letters."
980 PRINT "       This may be slow, especially when there are a lot of
    letters in!"
990 PRINT : PRINT "       Enter >99 to restart/new crib position, and 0 to
    end."
1000 PRINT : PRINT TAB(18); : INPUT "      Quag (*1) or Quag (2) "; TYP
1010 IF TYP <> 2 THEN TYP = 1

1090 ' start here!
1100 CLS : PRINT : PRINT TAB(10); HD$: PRINT
1110 INPUT "    Enter ciphertext: ('ENTER' for example, 'F' if data on
disc)"; CT$
1120 IF CT$ = "" THEN READ CT$: IF TYP = 2 THEN READ CT$ ELSE READ TT$
1125 PRINT CT$
1130 IF CT$ <> "F" AND CT$ <> "f" THEN 1160
1140 PRINT : INPUT "Name of file holding data "; N$: I = INSTR(N$, "."):
IF I > 0 THEN N$ = LEFT$(N$, I - 1)
1150 N$ = N$ + ".CT$": OPEN "I", #1, N$: INPUT #1, CT$: CLOSE #1: PRINT CT$
1160 I = INSTR(CT$, " "): IF I > 0 THEN CT$ = LEFT$(CT$, I - 1) + MID$(CT$,
I + 1): GOTO 1160
1170 LOCATE 10, 5: INPUT "Enter crib: "; CB$: IF CB$ = "" THEN READ CB$
1180 IF TYP = 2 THEN READ CB$
1790 I = INSTR(CB$, " "): IF I > 0 THEN CB$ = LEFT$(CB$, I - 1) + MID$(CB$,
I + 1): GOTO 1790
1800 LOCATE 10, 19: CB$ = UCASE$(CB$): PRINT CB$
1810 CT = LEN(CT$): CB = LEN(CB$)
1820 PRINT : INPUT "    And period: "; P
1840 IF P < 4 OR P > 9 THEN P = P(TYP)
1850 LOCATE 12, 18: PRINT "          ": LOCATE 12, 18: PRINT P

1890 ' find pattern of crib
1900 C$ = CB$: GOSUB 200: CPAT$ = CBB$

1940 ' step through CT$, and check pattern of parts (=length of crib)
1950 FOR N = 1 TO CT - CB
1960 C$ = MID$(CT$, N, CB)
1970 GOSUB 200
1980 IF CBB$ = CPAT$ THEN M = M + 1: F(M) = N: F$(M) = C$
1990 NEXT

2090 'make columns of CT and grid for PT
2100 FOR I = 1 TO P
2110 FOR J = I TO CT STEP P
2120 CC$(I) = CC$(I) + MID$(CT$, J, 1)
2130 NEXT
2140 NEXT

2190 ' if more than one fit, which to try?
2200 IF M = 1 THEN MM = 1: GOTO 2250
2210 LOCATE 22,10:PRINT "There are";M;"positions that fit.";:INPUT "Which";MM
```

```
2240 ' print columns of CT and grid for PT
2250 CLS : PRINT TAB(10); HD$
2260 LOCATE 3, 1: K = 0
2265 IF CT / P > 19 THEN NR = 19 ELSE NR = INT(CT / P + .9)
2270 FOR I = 1 TO NR: K = K + 1: PRINT "     ";
2280 FOR J = 1 TO P
2290 PRINT MID$(CC$(J), I, 1);
2300 NEXT: LOCATE , 19: PRINT USING "##"; K; : PRINT " "; STRING$(P, "."); K
2310 NEXT

2390 ' print grid for alphabets
2400 LOCATE 4, 43: PRINT T$(3 - TYP); "   "; AZ$
2410 FOR I = 1 TO P: RL$(I) = STRING$(26, ".")
2420 LOCATE 4 + I, 43: PRINT T$(TYP); I; "   "; RL$(I); I
2430 NEXT: LOCATE 4 + I, 49: PRINT AZ$

2440 ' put crib into PT block and alphabet grids
2450 Y1 = INT(F(MM) / P): X1 = F(MM) - Y1 * P
2460 FOR I1 = 0 TO CB - 1
2470 CL = (X1 + I1) MOD P: IF CL = 0 THEN CL = P
2480 CTL$ = MID$(F$(MM), I1 + 1, 1)
2490 PTL$ = MID$(CB$, I1 + 1, 1)
2500 GOSUB 400
2510 NEXT
2520 CL = 99: GOTO 2620

2590 ' get new letters (from keyboard)
2600 LOCATE 16, 56: PRINT "     ": LOCATE 16, 47: INPUT "Col. No.  ", CL
2610 IF CL = 0 THEN 2670
2620 IF CL = 99 THEN GOSUB 600: GOTO 2600
2630 IF CL > 99 THEN GOTO 2200
2635 IF CL > P THEN 2600
2640 LOCATE 17, 47: INPUT "CT letter ", CTL$: CTL$ = UCASE$(CTL$)
2650 LOCATE 18, 47: INPUT "PT letter ", PTL$: PTL$ = UCASE$(PTL$)
2660 GOSUB 400: GOTO 2600
2670 LOCATE 22, 30: INPUT "Done. (N*)ew crib, (R)un again, or (M)enu"; Z$
2680 IF Z$ = "M" OR Z$ = "m" THEN RUN "MENU.BAS"
2690 IF Z$ = "R" OR Z$ = "r" THEN RUN ELSE RESTORE 2700: CLS : GOTO 1170

2699 ' the examples
2700 DATA "RGEESEDIZMYUXIZUXBRJSPLIXWYDSFWHDARDSTDGQYUFQUWWPJFGQOBXFJXEQGXX
           QKXLRFLBAERUIUWUFOXMFFWSXLJUSKXESOJHARVTQUXUTDZVFOCPOFMUTVVJXVTC
           FZVHUCDIZUUFRSTX"
2750 DATA "HOZUQJGOIEQUDAJNKUWSJROKGLNSJWGWUYZEJEBTPDTONHHGUIJVECMZWURITEGQ
           SAATOKOKMPPOFKAODMEGAMJTLEYSWMHUGPCVMBMEKJKEYCHODEFJGMSAAJIGIIWE
           UGLKJYDHJXMYIXYFFFPTYLG"
2800 DATA "EARTHMILESFROMTHE"
2850 DATA "MINEIFASPACEISTOOSM"
```

ENIGMA 95

Clarence E. Tyner Jr.
Randall K. Nichols

A simulation of an enhanced Enigma Cipher Machine on a standard personal computer.

ABSTRACT

An exploration into the possibilities of what can be done with the operating methods of the Enigma on the personal computer. The same concept of employing keyboard input, a plugboard, rotors (both normal and reflecting), Uhr box and visual output are used, but are expanded by using 100-position rotors that intermittently rotate a prime amount after each input, allowing the number of rotors to vary from 1 to 12, in front or backwards orientation, top permit any keyboard character (including spaces) to be encrypted, and to simultaneously display cipher and clear text for editing. A rotating Character Set converts single-character input into 2-digit numbers for processing and superencipherment of numeric output into alpha bigrams is possible. Regular rotors, Reversing rotors, Character Sets and Superencipherment Tables are provided in sets of 100 for extensive variety. Visual monitor display and paper printout are employed and other controls are provided. It is a "what if" speculation that shows what could have been possible if the technology had been available.

Everyone is familiar with the Enigma Cipher Machine and the way it operates. However, the more you learn about it and read about the cryptanalysis that overcame it in World War II, the more you wonder if it could be improved without becoming impossibly complicated. The personal computer provides a means to improve the concepts that made the original Enigma work, and it can make it work much better.

This project started as a simulation of the original Enigma. The pathway of the electric circuit caused by pressing a key is easy to understand. It goes from the keyboard through the plugboard to the rotors, is reflected from the reversing rotor, back through the rotors, through the plugboard and finally to a lamp that lights under a round window with an alphabet on it. At least one rotor will rotate during the pressing of the key and the pathway through the rotors will change from what it was previously. The internal wiring of the rotors is random and the cumulative circuit offset combinations produce an extensive number of substitution alphabets. The plugboard adds to this, as did the Uhr box.

Aside from administrative and operator errors, the weaknesses of the enigma were as follows:

1. The internal wiring of the rotors was fixed. It never changed except for a few specialized purposes. While the mathematical possibilities were astronomical, only a small portion of them were utilized probably because of manufacturing, cost and logistics considerations.

2. There were only eight rotors in a set and only 3 or 4 could be used at a time.

3. The rotors rotated only very restricted basis. One moved one position each time. The second moved only after the first had moved 1 to 26 positions. The 3rd moved only after the 2nd had moved 1 to 26 positions. There were notches on the rotors to accomplish this and the rotors could be set so that the movements occurred at different times, but movement of two rotors was infrequent, and movement of all rotors was limited and somewhat predictable.

4. The reversing (reflecting) rotor did not move, nor could it be moved (except on the earlier models).

5. A subtle weakness was that a given letter could never be encrypted as itself.

6. It was expensive and labor-intensive both to manufacture and to operate. Once it had been determined how to simulate the rotation of rotors and to simulate the transfer of the electrical current between rotors correctly, a major problem was solved. Then it was necessary to determine how to keep the internal wiring connections unchanged during rotation. This was followed by developing a method of selecting and installing the rotors at a given position and then how to rotate them to an initial setting. Having an old Model D Enigma (3 rotor) so that it was possible to determine what the outcome should be was helpful.

Creation of rotors presented a challenge in establishing the internal wiring and in making a set from which to choose three. Edward H. Hebern used the Interval Method of wiring his rotors, so it was decided to use that approach. For those who are not familiar with it, it involves determining the positional difference (interval) between points connected on opposite faces of the rotor. For a 26 (A - Z) position rotor, the intervals range from 0 to 25, with each interval being used only once. But the geometry of the problem prevents one interval from being used and requires one interval to be used twice. All intervals are measured in the same direction. For example, a connection from point A on one face to point C on the other has an interval of 2 (assuming opposite positions are identified with the same letter).

I don't know how Mr. Hebern did it, but it is a job perfectly suited for a computer. At any rate, "wiring" a rotor using the Interval Method can be very tedious because it involves a lot of trial and error if done manually (or, as it turned out, by computer). It would be interesting to know if there is a simple algorithm that would produce a more secure encryption. After trying to do it manually (by diagramming on paper), programs were written to do it for both regular and reversing rotors. The programs also produce a file on a floppy disk to simulate a set of rotors and print the results for record purposes. Each rotor had to be unique from all others so use of random numbers was involved.

The plugboard was programmed so that it was possible to enter the 2-point (from – to) sets that were to be connected. Multiple sets could be created, just as it is possible to have multiple cable connections on a mechanical Enigma. A file of plugboards is not needed because the variance within fixed fields is derived from the connections, and to allow numbers of connections to be varied. It was necessary though to provide for editing to insure that each position was used only once (as in real life).

At this point, the idea of expanding the Enigma came into being in the form of introducing variability between the keyboard and the plugboard such as the Uhr Box does. It was decided to make the Enigma process the data in numerical form and expand it from a 26 to a 100 character format. This numerical format (00 – 99) has the disadvantage of doubling the length of a message, but it has certain advantages. In addition to handling alphabetic letters, it can also:

- Allow upper/lower cases, numbers, symbols, punctuations, and spaces to be encrypted.
- Better conceal the language and individual characters being transmitted.
- Eliminate the problem of a letter not being encrypted as itself.
- Allow a longer period between repetitions.
- Permit superencipherment.
- Provide 100-position rotors and plugboard which are more difficult to analyze.

- Facilitate masking control elements in messages. (e.g., rotor settings, etc.)

This format required a method of converting input into 2-digit form. It was done by creating what are called "Character Sets". These are randomly organized sets of 100 characters (upper and lowercase) that appear on the keyboard. The entire 100 positions are not used and the unused are filled with a seldom-used accent mark. One hundred sets are available in a file on floppy disk. The sets are used in both encryption and decryption to convert from and back to cleartext.

Using 100 as a common feature, brought into use the digits `00` – `99` to identify rotors, sets, tables and plugboard positions. Sets of these components have 100 of each (`00` means "100").

The next feature was to provide for the unique rotation or non-rotation (movement of each rotor is randomly intermittent) of each regular and the reversing rotor after each input. The Character Set also rotates so that doubles (like `oo` in `book`) are converted differently. Rotation is by a prime amount to 100 (2 and 5 are not used). Editing prevents using other numbers. An additional feature was to provide a Rotor Display similar to the windows on the Enigma. This is primarily informational but has proven to be helpful in de-bugging the program. . . and it does provide a sense of rotor movement.

Another idea was borrowed from Mr. Hebern. That was the ability to "insert" rotors into the machine either forwards or backwards which doubles the number of rotors in a given set. It was also possible to provide for a variable number of rotors. An arbitrary limit of 12 was chosen but it would be possible to have more (though that might be considered overkill). The important thing here is that it would be possible to employ from 1 to 12 rotors (from a set of 100), depending on the security desired. The rotor display automatically adjusts to the selected number.

The next feature that was added was the ability to optionally superencipher the resulting numeric ciphertext. This involves replacing a 2-digit numeric cipher with a 2-character alphabetic bigram (e.g., `36` to `HK`). It also permits each numeric cipher to be represented by one of 6 or 7 bigrams (e.g., `36` could be `HK`, `UM`, `RY`, `AU`, `ZM` or `BI`). The 7th bigram appears only for selected numerics because the 676 (26 x 26) possible bigrams are evenly distributed amongst the 100 numerics. In addition, the use of a given bigram in a set for each numeric is incremented sequentially so using this example, the numeric `36` would be converted to `HK` the first time it appears, to `UM` the second time, etc. The first selection can start at any of the first 6 positions and it cycles around to position 1 when position 6 or 7 is used. A SuperEnciphering Table (Figure 18) accomplishes this and there is a matching SuperDeciphering Table (Figure 19) to reverse it.

Text input requires no use of the `[ENTER]` key and the computer buffer handles rapid input so that the entry of clear or cipher text is faster than that of the original Enigma. Input is displayed on the monitor and the resulting cipher/clear text is displayed immediately below so that it is possible to visually check it. If an error occurs, a simple procedure allows you to correct it without having to re-type everything. A screenful of data consists of 6 sets of double lines (one input, one output) double spaced with the sets separated by a dotted line for clarity. There are 27 inputs per line for a total of 162. When the 159th - 161st are entered, a beep sounds to alert you to the approaching end of a screen. This allows you to make a final check of the input for errors (and easily correct them) before entering the 162nd which triggers printing that screenful to paper. During the printing you can start entering the next screenful. A limit of 1943 inputs (12 screenfuls less 1) was arbitrarily chosen for demonstration purposes (more would be possible, depending on memory available). This limit can be easily set to a shorter value to

control message length to make cryptanalysis more difficult.

Printing is considered essential for the purpose of having a record of what was sent and how it was encrypted or decrypted (e.g., was the cleartext entered correctly and was the machine correctly set). It also eliminates the need for a second person to transcribe the output. Following are four exhibits that are examples of the printouts that can be produced:

A Encryption into numeric form
B Decryption of Exhibit A
C Encryption in Superenciphered Form
D Decryption of Exhibit C

Each exhibit is divided into the following parts:

The Heading

This indicates whether it is encryption or decryption, and the date and time that the settings were entered . This does not change for repeated use of the settings for two or more consecutive messages. To enter a new date/time group or change the internal settings, the program must be completely restarted (See A1, B1, C1 or D1).

The Internal Control Settings

This indicates the number of plugboard connections used, the specific plugboard connections, the number of rotors used, the specific rotor numbers in the position sequence and then each rotors orientation (frontwards or backwards). The reversing rotor number is indicated. Next, the unique rotation value for each rotor and the reversing rotor are shown, followed by the character set number and its rotation value. These constitute the internal settings that would be specified by the Signal Operating Instructions (SOI). All of these settings generate an Internal Checksum which is used to verify that the settings have been correctly entered.

This checksum is printed. If it does not agree with that provided in the SOI, then all the settings must be re-entered by restarting the program. Intermittent rotation of each rotor is a function of the installed rotors and previous entries and does not have to be specified.

The External Control Settings

This lists the settings that the operator selects and enters for the specific message. They consist of the Initial Settings of each rotor and optionally the Superencipherment Table number if it is used. These settings add to the Internal Checksum and produce an External Checksum in the form of a 2-digit number (mod-100 of the total sum) that is sent with the message. The superencipherment table counter setting is NOT included and is NOT sent because the recipient does not have to know it (See A1, B1, C1, D1).

The Input/Output Message Text

This duplicates that which appears on the monitor screen and is provided primarily for a message audit (to insure that the message was entered correctly). Each "line" has 27 inputs with the 27 outputs below. Twenty-seven was used to provide legibility on an 80-column screen. Six such "lines" are possible for each screenful (See A1, B1, C1 or D1).

The Message Control Data

A count of the input characters (message length) is provided for both superenciphered and non-superenciphered messages. However, only non-superenciphered (numeric ciphertext) messages have the following additional data provided:

1. A Hash Total which is a Mod-100 sum of the numeric cipher text (See A1, B1).

2. A set of Column Check Totals which is the Mod-100 sum of each of the 27 columns of cipher text. This is followed by a non-mod total of the columns (See A2, A3, B2).

3. A total of Row Check Totals which is the Mod-100 sum of each row of cipher text. This is followed by a non-mod total of the rows (See A2, A3, B2).

The purpose of providing column and row totals is to be able to locate transmission garbles. They would be sent only if requested. Variances in any given column and row would locate the error by intersection.

The Message in Transmission Form

This is what would be sent and would contain only the External Control Settings (rotor settings, superencipherment table number and external checksum) , the date and time group, the message ciphertext and the character count. The External Control Settings would be disguised by a simple manual superencipherment that would be administrative and outside the operation of the Enigma 95 (i.e., prescribed by the SOI). (See A3, C2). If it is decryption, the cleartext message is presented with normal horizontal spacing and vertically double spaced for convenient reading (See B3, D2).

Optional Message Analysis

This is simply a count of input and output characters. This can be skipped and was provided only to assist any system analysis. (See A4 and C3).

This completes the printing.

Next displayed on the monitor is an option to re-use the Internal Control Settings for another message (it was assumed that these would remain in effect for a period of time as was the case for the Enigma). If this is not selected, the program ends.

COMPUTING REQUIREMENTS

The Enigma 95 is a program written in Microsoft QBasic. This was done so that it could be run on any standard MS DOS computer using MS DOS 5 or higher (QBasic is bundled with MS-DOS) thereby eliminating the need for a specialized computer.

It fits onto a 3.5 inch floppy disk, together with the necessary data files that constitute the Regular Rotors Set, Reversing Rotors Set, Character Sets and Superencipherment Tables. It is possible to also have on the same disk, the programs that create these files and the necessary documentation (`.DOC`) text files for each one. This makes the Enigma 95 very portable, very inexpensive and very easy to replicate.

Any computer that will run MS DOS QBasic is suitable for the Enigma 95. A color monitor is preferred but not essential. A printer is very useful, but could be eliminated if one is willing to copy output manually from the monitor screen (as the original Enigma required).

There is provided a program that produces a graphic representation of the circuit path through the Enigma 95 and a program to produce pseudo-random numbers to use in programs that produce the rotor disks. Also included are programs to analyze the Enigma 95.

EPILOGUE

The Enigma rotor operation principle has probably been long superseded by much more sophisticated methods of encryption that are faster and more secure, but it will remain interesting for a long time to amateurs such as myself. It is something that is understandable and before the advent of the computer, resulted in some beautiful machines.

The Enigma 95 is not one now, but I believe that it could be “translated” into a handsome electro-mechanical device. It is something to dream about.

The only absolutely secure cipher is the One Time Pad and it has the disadvantage of requiring copies to be destroyed after one use. The Enigma 95 is an attempt to approach this holy Grail of cryptography by providing an almost unlimited supply of enhanced (both in size and method of rotation) Rotors, Character Sets, Superencipherment Tables and a lengthened Plugboard. While I cannot prove it mathematically or otherwise, I suspect that

the ability to use almost unlimited expendable sets of all possible combinations of these for very limited periods (throw away feature) such as is possible in the Enigma 95, would strengthen any cipher considerably by preventing the accumulation of sufficient material on which to base an in-depth cryptanalysis. Any comments would appreciated.

AT THE CRYPTO DROP BOX

The disk accompanying this article contains ENIGMA 95 and the necessary supporting files needed in its operation. Also included are program files to create them and to analyze and test its operation. `DOC` files are included for each file to explain them. Start with `CRYPTO.1ST`, then read `ENIGMA95.DOC` and study `ENIGMA95.FLO` to gain an understanding of Enigma 95 before running it. The list of files is:

```
CRYPTO.1ST       :  An outline of the files that constitute Enigma 95 system
ENIGMA95.DOC     :  Detailed documentation pertaining to ENIGMA95
ENIGMA95.FLO     :  A flowchart of the ENIGMA95 operation
ENIGMA95.BAS  *  :  ENIGMA95

ROTORS.DAT       :  Set of 100 Regular Rotors
REVROTRS.DAT     :  Set of 100 Reversing Rotors
CHARS.DAT        :  Set of 100 Character Sets
CODE.DAT         :  Set of 100 Super Encipherment Tables

CRYPTO05.BAS  *  :  Random Numbers Generator for CRYPTO27 & CRYPTO34
CRYPTO27.BAS  *  :  Regular Rotor Creation using the Interval Method
CRYPTO28.BAS  *  :  Super Encipherment Tables Creation
CRYPTO30.BAS  *  :  Character Set Creation
CRYPTO34.BAS  *  :  Reversing Rotor Creation

CRYPTO43.BAS  *  :  ENIGMA95 Cipher Machine Data Paths Demonstrator
CRYPTO45.BAS  *  :  Rotors Matching Analysis
CRYPTO47.BAS  *  :  Check of Rotor Files for Errors
CRYPTO48.BAS  *  :  Analysis of Cleartext vs. Ciphertext
CRYPTO49.BAS  *  :  Rotor Intermittent Movement Test
CRYPTO51.BAS     :  Plugboard Combinations

ENIGMA95.WRI      :  The article about Enigma 95.
                     (Created using Windows 3.1 Write)

               * = Has a matching .DOC file

      The .1st , .DOC and .FLO files are DOS files
            The .BAS and .DAT files are QBASIC or QUICKBASIC  files
            The .WRI file is a WINDOWS 3.1 Write file
```

AUTHOR BIOGRAPHIES

Clarence E. Tyner Jr., 69, is retired from the Army as a Major in the Corps of Engineers in which he served in topographical and engineer technical intelligence services. He is also a retired Certified Public Accountant, having specialized in Internal Auditing for a large county-wide public school system. His interest in cryptography began in World War II with the M-209 and is a collector of both machines, materials and publications that are about cryptography.

11322 Carrollwood Drive
Tampa, Florida 33618, USA.

Randall K. Nichols, 52, has served as the President (1994-1996) and Vice President (1992-1994) of the American Cryptogram Association (ACA), which since its formation in 1929, has been devoted to the pursuit of primarily classical and recreational cryptography. Mr. Nichols is the Aristocrats' Department Editor for ACA's bimonthly publication *The Cryptogram*. Mr. Nichols also works as Cryptology Section Leader for the National Computer Security Association, (NCSA) Compuserve Forum. Mr. Nichols is currently teaching one of the first electronic courses in classical cryptography on the Internet. One hundred forty nine (149) students are participating worldwide. He is considered an expert (as well as an author/editor) in the field of Classical Cryptography. Mr. Nichols is the author of *Classical Cryptography*, a book to be published in 1997.

When not glued to the computer screen thinking up ways to torture his students, Mr. Nichols has another life as a senior manager with a Fortune 100 Company in charge of implementing a massive ISO 9000 standards project for his company. He has previously served as Manager of Raw Materials Inventory, Marine Operations, Transportation, and Computer Applications Departments. Mr. Nichols has 30 years foreign and domestic project management experience in a wide variety of leadership roles in the engineering, construction, and chemicals industries.

Mr. Nichols holds a BSCHE degree from Tulane University, New Orleans, LA. (1967), a MBA from University of Houston, Houston, TX (1970) and a MSCHE from Texas A&M University, Kingsville, TX (1991).

In 1995, Randy was awarded a 2nd Degree Black Belt in Tae Kwon Do (Korean Karate) by the Moo Duk Kwan International and the American Korean Tae Kwon Do Associations. He teaches Tae Kwon Do Self-Defense and Rape-Defense courses in Corpus Christi, Texas.

5953 Long Creek Drive
Corpus Christi, Texas 78414, USA

ACA CD-ROM Available

There is now a CD-ROM containing ACA and crypto-related materials available from the Editor.

Through the wonders of Recordable CD-ROM, the following items are available in MS-DOS format:

- All issue disks, expanded and ready-to-run.
- LaTeX-formatted *Computer Supplement* issues, from 16 to 21.
- Various word lists and pattern dictionaries.
- Various utilities for IBM-PC compatibles.
- Cryptographic items of historical interest.
- Random number generation and testing.
- General computer security documents and software.
- Various unprotects and password guessers.

---

dBASE Additions

Kenneth Madl

[Ed: Ken Madl includes here some enhancements to **PARROT**'s dBASE routines from *Computer Supplement* #20. DMV]

[Following] are copies of three different methods to achieve the same result, but without having to type the replacement routine 20 times. I thought initially that I could simply use the command `APPE FROM Dict FOR Count = LEN(TRIM(Word))`, but I discovered that dBASE seems to append the record, then based on the `FOR` condition, decide whether or not to accept it. Thus, every record was accepted into `Dict3`, since words larger than three characters were truncated to three. I could only get around that limitation by making the field length one character larger than the length of the word.

I tried adding an extra field to the `Dict` database that contained the length of the word, and then used that field in the `FOR` condition. However, I discovered that the fields used in the `FOR` expression must reside in the structure of both databases.

After giving up on the `SET RELATION` command (the program would not append from an open database), I believe that the `APPE3.PRG` program is the easiest and quickest way to approach the problem. (I have not compared the times for the three methods, but I suspect that `APPEND` and `COPY` would each take less than the hour it took you originally.) After each database is created, the `MODI STRU` command is used to match the length of the `Word` field to the name of the file, i.e. `Dict13` would have a field length of 13, and so on.

```
* APPE1.PRG
* Append records using the APPEND command
* Length of WORD must be 1 larger than Dict name, e.g. Dict3 is length 4

CREATE Dict
APPE FROM WORDS.TXT SDF
STORE 3 TO Count
DO WHILE Count < 23
   FileName = "Dict" + LTRIM(STR(Count,2))
   CREATE &FileName
   APPE FROM Dict for SUBS(Word,Count)=" " .AND. SUBS(Word, Count - 1) # " "
   STORE Count + 1 TO Count
ENDDO
RETU
```

```
* APPE2.PRG
* Append records using a loop
* Length of WORD is the same as Dict name, e.g. Dict3 is length 3
```

```
SET TALK OFF
CREATE Dict
APPE FROM WORDS.TXT SDF
STORE 3 TO Count
DO WHILE Count < 23
   FileName = "DICT" + LTRIM(STR(Count,2))
   CREATE &FileName
   STORE Count + 1 TO Count
ENDDO

USE Dict
STORE 1 TO Count
DO WHILE .NOT. EOF()
   NextWrd = TRIM(Word)
   WordLen = LEN(NextWrd)
   NewFile = "DICT" + LTRIM(STR(WordLen,2))
   USE &NewFile
   APPE BLANK
   REPL Word WITH NextWrd
   STORE Count + 1 TO Count
   USE Dict
   GO Count
ENDDO
RETU
```

```
* APPE3.PRG
* Append records using the COPY command
* Length of Word is reduced after records are entered

CREATE Dict
APPE FROM WORDS.TXT SDF
STORE 3 TO Count
DO WHILE Count < 23
   FileName = "DICT" + LTRIM(STR(Count,2))
   COPY TO &FileName FOR LEN(TRIM(Word)) = Count
   USE &FileName
   MODI STRU                              && Reduce size of "Word" field
   USE Dict
   STORE Count + 1 TO Count
ENDDO
RETU

DEL *.BAK (from DOS command line)
```

An Introduction to Modern Cryptology

Copyright G. H. Foot April 1996

Cryptography is the art of disguising a communication (of any nature and transmitted by any means) in order that the information conveyed cannot be understood by anyone except the person for whom it is intended. For this to be possible, a secret understanding of some kind (a *Key*) must be arranged in advance between the parties concerned and must never be revealed to any other person. The use of the Key enables the communication to be encrypted at the source and decrypted after its arrival.

Commonly the process involved in encrypting a textual message is to manipulate the characters so that the words become unrecognizable. A great variety of methods have been adopted through the ages from the simple replacement of each character with the third character in the alphabet beyond it as used by Julius Caesar to intricate arrangements of characters in columns, rows and groups which have to be rearranged by the recipient before the message can be understood.

Machines for cryptographic purposes came into prominence during the Great Wars of this century for the protection of military and political intelligence: The story of their successes — and sometimes of their failures because of brilliant counter-intelligence operations — has been related extensively.

More recently all other methods have been eclipsed by the introduction of modern computers which can manipulate numbers (representing characters) of a size and complexity far beyond anything which could be handled manually. But computers have also been enlisted for the assistance of *Cryptoanalysts* (those people who exercise ingenuity and cunning to unravel the secrets of cryptography without the assistance or the knowledge of the *Cryptographers*) so that the battle of wits and skills between the opposing crafts continues more intensively than ever — with the rapid growth of computing power being pressed into the service of each of the rival factions as one tries to outperform the other.

Entering the fray also are the Mathematicians because the secrets of encryption are nowadays based on transformations in number theory of a most advanced mathematical nature — advances which have become feasible only because of the enormously increased computing power which is available.

Any cryptosystem which has been validated for extensive use will have been subjected previously to intensive examination for potential weaknesses by experts in this field. It is always a requirement that it must be possible to publish full details of the nature of the crypto system and the method in which it operates without diminishing the security it provides. The cryptosystem must be secure against all attacks in circumstances in which the Key is unknown.

It has also to be assumed in assessing the merit of a cryptosystem that an eavesdropper is able to intercept encrypted messages during transmission, that he is able to obtain specimens of the plaintext (which is a message before encryption) and compare it with the *ciphertext* (which is the message after encryption), that he can generate plaintext and ciphertext with Keys he may choose in his attempts at cryptoanalysis and that he has as much time as he wishes for this purpose.

Frequent changes of the Key provide much additional security. Nevertheless, it is never possible to prove beyond doubt that a cryptosystem is totally secure or to be absolutely certain that a weakness in a cryptosystem does not exist so that a short-cut to decryption without knowledge of the Key can be discovered. There is one exception — the so-called *One-Time Pad* — which will be described later but which, unfortunately, is cumbersome and has limited application in practice.

Historically and until very recent times, the same Key was required for encryption and decryption so that unless the parties concerned had the opportunity to meet and pass Keys directly from one to another, the secure conveyance of Keys between them has always been — and remains — a serious problem. In circumstances in which the highest possible security is required, a courier is employed for the transmission of the Keys — but this is a slow and expensive process. In any case, no method is completely proof against malicious ingenuity, bad faith, corruption and like measures which it must be anticipated an enemy will employ energetically.

An important development occurred in the 1970's when it was shown that different Keys could be employed for encryption and decryption. Moreover, one of the Keys could be published without prejudicing security if the other Key were kept secret. The Key which could be published was called the *Public Key* and the system became known as *Public Key Cryptography*. The secret Key is known as the *Private Key*. With Public Key Cryptography it is possible to utilise a Public Key for secure communication with the owner of that Key without the need for any prior contact. Only the person in possession of the Private Key is able to decrypt the ciphertext and to recover the message.

Moreover, if double encryption is performed it becomes possible to establish that the communication came from a particular person and from nobody else — both the origin of the message and its content can be authenticated because a *Digital Signature* has been attached to the message.

The most successful of the Public Key Cryptosystems is that known as *RSA* (named from its inventors Rivest, Shamir and Adleman), the mathematical basis of which is the great difficulty of factoring the product of two large prime numbers — a problem studied by mathematicians for centuries without finding an easy solution. Numbers which are hundreds of digits in length are used and considerable computing capacity is required.

However, there are reservations concerning Public Key Cryptography in practice. The distribution and especially the authentication of a large number of Public Keys throughout the world is a severe problem — even more severe is the problem of cancelling a Public Key if it has been distributed extensively but is no longer valid. The loss of a Private Key can be announced falsely in order to repudiate a contractual liability and other mispractices can arise — an illustration that a sound technical solution is insufficient to ensure that a cryptosystem is suitable for general use even if it may be excellent in a specialized application.

One other practical problem confronts RSA. The computing power necessary for its application is considerable so that even if the computers available are adequate in capacity the speed of operation may be unacceptably slow. A class of hybrid systems has therefore arisen in which RSA is employed to establish initial contact between the parties but subsequently the main body of the message is transmitted with a second cryptosystem which functions more rapidly because it requires less computing effort.

A prominent example of a hybrid cryptosystem is *PGP* (standing for Pretty Good Privacy) which is a combination of RSA with *IDEA* (a cryptosystem of Swiss origin). PGP was introduced in the USA by Mr. Philip Zimmermann and soon gained recognition throughout the world. To understand the background to this development, it is necessary to provide an explanation of the discussions and controversies which relate to cryptography at the present time.

Until recently, the practice and preserve of cryptography was largely the prerogative of governments, each desiring to conduct their political and military communications in private whilst learning as much as possible of the affairs and intentions of other governments. For the proper regulation of domestic matters,

each government prohibited the transmission of messages in codes and ciphers of any type which an agency of the government could not decrypt.

Into this well-ordered world burst a flood of electronic communications passing over worldwide electronic networks which had little respect for national frontiers but which conveyed a vast amount of private, commercial, banking, financial and other traffic which required privacy for competing business interests, which needed protection from fraud and financial loss, and which demanded noninterefernce from government.

A new situation arises inasmuch as modern computers commonly employed in home and office have become so powerful that they have the potential to make use of strong cryptosystems which are difficult or virtually impossible for government agencies to break. Governments are loath to yield their overriding power to read every communication transmitted by electronic means. They declare that their right to do so is necessary for the detection of criminals and terrorists and for the suppression of unsocial practices. The debate continues and is especially intense in the USA where cryptosystems can be classified as munitions, the export of which is banned.

In this contentious atmosphere, Philip Zimmermann introduced his PGP, a cryptosystem providing high grade security. In some manner knowledge of PGP passed out of the USA (an illustration of the impossibility of confining any system within national frontiers in modern times) and Zimmermann was in peril of being charged with exporting munitions. That threat is now lifted but nothing else is resolved.

In the meantime, the US government has proposed a plan for escrow cryptography known as *Clipper*. The idea is that the US government would permit the use of a secure high-grade cryptosystem (designed by the US government) if it were confined in chips (supplied by the US Government) each chip containing a unique code which is identifiable. Using this code and with the sanction of a US court order, the US Government could recover any Key from compulsory escrow with an approved US government agency to allow the appropriate US government agency surreptitiously to decrypt all messages being transmitted with the chip under supervision. Fierce objections to the plan have arisen.

Other countries have attempted to impose a complete ban on cryptography other than for their own government communications. This is unlikely to be practical — some forms of cryptography are essential for banking and similar purposes and are already in extensive use.

The arrival of the Internet and its employment for an enormous and still rapidly growing number of diverse purposes is another reason why cryptography is required to safeguard the privacy of communications transmitted electronically: But this is a contentious and difficult area in which to introduce any agreed measures of control as it is predominately international in character.

It has not been specifically indicated that messages in the context of this article includes data, graphics, sound and any and every kind of information which can be transmitted from place to place. Moreover it includes such techniques as *steganography* (for example, hiding dramatic text within a picture of an innocuous scene) — but space does not allow diversions to discuss such topics.

Interestingly, there is one, and only one, crypto system which is unbreakable. This is the one time pad which combines and conceals the message to be transmitted with a series of numbers which are entirely random — random means that it is entirely impossible to predict the next number from a knowledge of all previous numbers in the series. But this cryptosystem requires pads of random numbers to be prepared and to be available in advance to both the sender and receiver of a message —

moreover there is the important limitation to general use that each pad can be used once and once only. Nevertheless, be prepared for new developments in this as in every other direction.

About the Author

Returning to the UK on retirement from the position of communications engineer with a US company, I became interested in home computers as a hobby.

An extension of this hobby has been an interest in modern cryptography and its application as an aid to privacy in private and commercial transactions via electronic media.

I have a (joint) UK Patent for an invention in this field and I am engaged (with colleagues) in the development of a secure cryptosystem which has a sound theoretical basis and also novel features.

My formal qualifications include Senior Member of the IEEE in the US and Fellow of the IEE in the UK.

E-Mail: `georgefoot@oxted.demon.co.uk`
Web Page: `http://www.cybervillage.co.uk/personal/gfoot/`

NOTES TO AUTHORS

The *Computer Supplement* is intended as a forum to publish articles on the cryptographic applications of computers. We are always looking for submissions, but we ask potential authors to bear in mind:

1. Many readers are new to ciphers; please include a brief description of the cipher in question.
2. Many readers are new to computers; explain **why** you are using a computer as well as how.
3. Include the output of a typical run. If possible, build in an example for the reader to check the operation. Indicate how long it took to obtain this result.
4. Include a full description of how the program works, and back it up with comments in the listing.
5. Include a table of variables, either separately or as a part of the listing.
6. If at all possible, please submit *everything* in electronic form, either on a disk (any IBM format), uploaded to the ACA BBS, or electronically mailed to `dan@decode.com`. This makes it much easier for us to typeset.
7. Send material for publication to Dan Veeneman, PO Box 2442, Columbia, Maryland, 21045–2442, USA.

A TRIP TO THE NSA MUSEUM

BROADSWORD (David W. Cuccia)

Greetings everyone!

I want to share with you all some details about a recent visit I was lucky enough to make to the NSA Museum.

First: The NSA Museum is located on Fort Meade, in Maryland, right next to the NSA headquarters.

To get there: take I-95 (north or south, depending on where you live relative to Maryland) and take the exit for Route 32 east in Maryland. The exit will say Fort Meade, and Columbia. On Route 32 east, go past route 295. There are signs on the road that point to the Museum: you want to look on your left for a Shell gas station. Immediately after the Shell station, on your left, is Colony 7 Road. Take this road (which brings you to the gas station you just saw), and follow it right up to the NSA museum. Its a little building, surrounded with an anchor link fence topped with barbed wire!

Open Weekdays (except holidays)	:	9am - 3 pm
Open Saturdays	:	10am - 2pm
Phone	:	301-688-5849

The curator's name is Jack Ingram. I went with two friends of mine.

One of our party, a friend named Doug Stanton, had recovered some code books from a German U-boat that was sunk off the coast of Rhode Island in the last days of WWII. After 6 months of treatment, the books came out in very readable condition. Jack looked over these code books for some minutes, and speculated on the possibility that an Enigma machine may still be on the U-boat. This U-boat was sunk by depth charges, so the crew never had a chance to surface and ditch the Enigma machine. Unless they shot it out a torpedo tube, it might still be on board (buried under 3 feet of silt, and who knows what else?).

Jack estimates that if the Enigma is found, and if it has a printer attached, it could be worth somewhere in the $40K range. It would be an interesting find.

After my friend showed the codebooks to Jack, Jack showed us the NSA's newest acquisition: the BOMBE machine. This machine was built by the Allies to speed up the cracking of Enigma coded messages. Apparently, three Polish cryptanalysts were discussing over lunch how to speed up the decryption process, and they came up with this machine. When they went to name the machine, they selected the name of their dessert: bombay ice cream, or BOMBE. Anyway, this particular machine was in the Smithsonian, and they agreed to "give" it to the NSA (I don't know the terms of agreement). The NSA still had to get some of the pieces from the Smithsonian, so regrettably, they did not have the whole machine there when I visited.

This was quite a large machine: 5,000 lbs, about 6 feet high, 8 feet long, and 2 feet wide. It performed a brute force crack on the messages. Some of the rotors it used spun so fast that they cracked on a regular basis: this machine required a fair amount of maintenance.

After that, I (and my friends) went around the rest of the exhibits: They had Enigma machines, field rotor sets, T-series Engimas (that were sold to the commercial public before WWII), an Enigma with a printer (recovered from another U-boat). They also had the Tunny, used by German high command, the Purple Machine built by Friedman and his crew to crack the Japanese purple code, part of a purple machine recovered from the Japanese embassy in Berlin, a Jade machine, a Hagelin Machine, a Russian coding machine, some Venona documents, a rare book collection (started by Friedman) which contained one book from the 16th century, a 300 ter-

abyte tape drive (about the size of a walk-in closet, complete with a robot arm to retrieve tapes), one of the first Cray machines used by the NSA (they made it into a bench which you can sit on and relax), several models of secure phones/FAX machines and some of the more recently used field coding machines.

After taking a few pictures, I returned to the conference room where my two friends were measuring rotors from two rotor sets. I helped buzz out a reflector rotor. The friend who has all the mappings is away on a business trip, and when he returns, he will find me gone on a vacation trip. When I touch base with him again, I will get the mappings and post them for your enjoyment.

We also used an Enigma machine which is on display for the public to use. Its really something to actually use the very machine that was regarded as the top secret device of the German command only 50 short years ago. Very impressive.

Overall, I was very impressed with the museum: it has a very nice collection, that is well kept, and well presented. The curator (and the other staff) were all very helpful, and *very* knowledgeable about the theory and the history of these machines. I recommend a visit.

ZIMMERMANN CHARGES DROPPED

Phil Zimmermann, the author of the Pretty Good Privacy encryption program.
In 1991, the Federal government convened a Grand Jury and began and investigation into possible export violations for making PGP available online.
In January 1996, Phil posted the following message:

> My lead defense lawyer, Phil Dubois, received a fax this morning from the Assistant US Attorney in Northern District of California, William Keane. The letter informed us that I "will not be prosecuted in connection with the posting to USENET in June 1991 of the encryption program Pretty Good Privacy. The investigation is closed."
>
> This brings to a close a criminal investigation that has spanned the last three years. I'd like to thank all the people who helped us in this case, especially all the donors to my legal defense fund. Apparently, the money was well-spent. And I'd like to thank my very capable defense team: Phil Dubois, Ken Bass, Eben Moglen, Curt Karnow, Tom Nolan, and Bob Corn-Revere. Most of the time they spent on the case was pro-bono. I'd also like to thank Joe Burton, counsel for the co-defendant.
>
> There are many others I can thank, but I don't have the presence of mind to list them all here at this moment. The medium of email cannot express how I feel about this turn of events.

ENIGMA – The Random House Contests

David Hamer (DAEDALUS)

Early in October 1995, Random House published *ENIGMA*, a novel by Robert Harris based upon a series of actual events which took place in and around Bletchley Park and Cambridge University in the early days of World War Two. As a promotion, the publisher offered a complimentary copy of the book to the first fifty or so solvers of a very simple substitution cipher, presented on the Random House pages of the World Wide Web.

This first cipher, a Patristocrat, was easily decrypted by a number of ACA members who, in very short order, received copies of the book together with an indication that Random House would shortly announce a more difficult cryptographic challenge: this time with cash prizes!

Later in October the new challenge appeared on the Random House World Wide Web site. The cut-off date for entries was to have been December 15, 1995 but this date was extended to January 3, 1996. Thirteen prizes were awarded, four of these going to ACA members (alphabetically): Frank Dezzi (**SIMRAM**), Jim Gillogly (**SCRYER**), David Hamer (**DAEDALUS**), and Bill Sutton (**PHOENIX**). The text of the cipher: `QXQF VFLR TXLG VLWD PRUA` (Too short for statistics - so, it could be anything!)

My personal story begins on the day I received the free copy of *ENIGMA* — well before the second contest was announced. The above ciphertext appeared as part of the author's dedication at the beginning of the book. Out of curiosity, I tried simple substitution, transposition, a few other things (including an Enigma simulator program) and of course Caesar. I seemed to get no positive result, so I left it alone.

When the ciphertext reappeared as the subject of the contest I reapplied my efforts to its solution. Vigenere seemed the most likely candidate but after a couple of days of this and a number of other blind alleys, I was still no further ahead. After all, I thought, I had tried all of the "easy" tracks already. Simultaneously, I had been reading the book — dwelling heavily on the first chapter where, according to the publisher, the clue to the solution lay. If so it was not evident to me.

However, in one of the book's later chapters the central character has trouble deciphering a number of Enigma messages until he realises that the output is not in German, as expected, but is a list of Polish names. I had gone to bed that evening still thinking about the contest cipher. At about six in the morning I awoke with the thought: "It's not in English!". Rushing to the computer I tried the simplest thing — Caesar! And there it was: in Latin. I had had the answer for several weeks, and had overlooked it.

All that remained was to get a confirmation of the translation into colloquial English. Any Latin that I had been exposed to had gone in the forty-plus years since my schooldays. Who did I know with a knowledge of Latin and who could I comfortably call at 6:30 a.m.? The answer to this double-barreled question was obvious — I telephoned the local Catholic church! The job was now completed but for the transmission of the result to the Random House WWW site.

Well, it was a lot of fun — and I learned never to overlook the obvious; at least I think I did! And for those who missed the solution:

```
          QXQF VFLR TXLG VLWD PRUA
          nunc scio quid sita morx
Latin:    nunc scio quid sit amor x
English:  now I know what love is
```

WHAT THE OTHER GUY IS DOING

LANAKI (Randy Nichols) asks:
One of the NCSAFORUM subscribers poised an interesting question about the *Bell, Book and Candle Cipher* supposedly used during WWI and WWII...

I have searched my books and records and found *Bell*, *Bell Labs*, *Book ciphers*, *BBC and radio messages*, even found a movie with Jimmy Stewart and Tippy Hedron? in it regarding a witch... Followed that lead, took the movie out, looked for sorcery codes. Then I pulled out the big guns.. called **PHOENIX** and **MEROKE**. **PHOENIX** had a vague notion but nothing came to mind.

I am stumped. Okay, Krewe, has anyone heard of this variant, and if so, when and where?

G4EGG (Wilfred Higginson) is using CRYPTODYCT with dBASE IV and some .PRG. Any interest from members of the Krewe to share dBASE routines and files?

RETREAD (Penn Leary) continues to pursue historical cryptographic matters, including ciphers in the works of Shakespeare. Contact him for printed materials and software, including the 41-page pamphlet *Are there Ciphers in Shakespeare?*

FAT DRAGON (Daniel Killoran) has scanned most of his 35-year collection of *The Cryptogram* into his computer, in a form suitable for optical character recognition (the format used by Caere's OmniPage Pro). If this would be of any use to the Krewe, they are welcome to it, although it might be a little difficult to send (over 400 megabytes).

BOATTAIL (Patrick J. Larkin) has upgraded to a Pentium 60MHz from his old 386-16. He has also upgraded his monitor to a Sony 15-inch SVGA. His is now running Windows for Workgroups 3.11, and is planning on upgrading further by adding a CD-ROM and Windows 95 later on this year. He hasn't done any programming for a long time, but still uses his crypto programs to work on each issue of the *Cryptogram*. He also continues to subscribe to the ACA-L electronic mailing list.

COLD DUCK (L. Rucinski) is interested in locating a copy of Caxton Foster's *Cryptanalysis for Microcomputers*. He is unable to locate a copy, or even a telephone number or address for Hayden Books. Any help would be appreciated.

FAT DRAGON (Daniel Killoran) has, for his own use and convenience, scanned most of his 35-year collection of *The Cryptogram* into his computer, in a form suitable for optical character recognition (the format used by Caere's OmniPage Pro). If this would be of any use to the Krewe, you are welcome to it, although it weighs in at over 400 megabytes.

DAEDALUS (David Hamer) has moved again — but this time only fourteen miles laterally and a few hundred feet vertically. He has added an aged 386DX to his collection of hardware which will be dedicated to performing those large calculations which would otherwise occupy valuable time on his "serious" machine.

Richard Brisson (hagelin@magi.com), a student in **LANAKI**'s crypto class, has some questions about a NEMA Machine and some about the ENIGMA:

Here are (all) the details I have on the NEMA (don't think it was made by Hagelin unless the company below was a subsidiary of Hagelin).

The NEMA was made in 1947 by Aschmann & Scheller A.G. Zurich; it contains 10 rotors of which rightmost is coded red while all others are black; it was intended as a post-war Enigma-like commercial device; and the manual has both German and French instructions.

I would be most appreciative of any further details with regards to the NEMA such as actual purchasers/users (the Swiss?) and the company.

Now on to the other topic of the ENIGMA. My questions are the following:

- The main letter designator which we find on the German military Enigma's such as A (most 3-rotors) and M (mostly Kriegsmarine) are related to the types of users?
- The three letter code which precedes the serial number (e.g. "aye" or "jla" which is on mine) represents the manufacturer?
- Sources for the bulbs which have to be 12mm screw-type, hemispherical and 3.5 volts...

WARTHOG (Walt Howe) writes:

Now that I've established contact, I guess I should (re)introduce myself. I joined the ACA in 1960 or 1961 while I was attending the Army's old 21-week MOS 981 Cryptanalytic Specialist Course at Fort Devens at the US Army Security Agency Training Center & School. I was a frequent completer during the 60's and since I stayed on as an Cryptanalytic Instructor at Ft. Devens, first in uniform, and then as a Civil Servant, I steered many military members to the ACA, particularly those who loved the challenges of the theory and the solutions. Somewhere through the years, I dropped my ACA subscription.

I remained as a Signals Intelligence trainer, training developer and training manager at the USASATC&S, later renamed as the US Army Intelligence School, Fort Devens. Along the way, the 981 was renamed as the 98B MOS and then folded into the 98C Traffic Analyst MOS.

I was delighted to have the chance to write the Army's Field Manual FM 34-40-2 Basic Cryptanalysis, published in 1990. The manual, as all Army manuals, acknowledges no authorship, but you can find my name in there anyway, concealed in Chapter 14, if you happen to have a copy.

I retired from Civil Service in 1992 as the School began transferring to merge with the Intelligence School at Fort Huachuca, AZ.

Since then, I have been self-employed in various ways as an Internet consultant, author, trainer, and speaker. I like to say that I have been surfing the nets since 1960. See my home page at either `http://www.delphi.com/walthowe` or `http://www.tiac.net/users/walthowe/`.

Oh, yes. My old NOM was **WART HOG**. Anyone remember me? I contributed some nasty cryptograms back then. I hope to do so again.

ACA COMPUTER BULLETIN BOARD UPDATE

All members of Krewe are welcome to use the ACA bulletin board system, `Decode`, for electronic mail to the Internet. It is available 24 hours a day at +1 410 730 6734. Each user will automatically gain an Internet address of the form < *user* >`@decode.com`, and may correspond via e-mail to members of the Krewe and other Internet users.

The system subscribes to a number of electronic mailing lists, including Cypherpunks, Microsoft's CryptoAPI project, and Cloak-and-Dagger.

The `FILES` section also contains various ACA and cryptographic-related files and programs, as well as an assortment of other topics.

Current PGP Versions

Ståle Schumacher

PGP exists in several different versions. If you are not sure which version is the right one for you, you may find help here. The following is a list of the most popular PGP versions available today:

PGP 2.3a

This is the "classic" PGP version, and until recently, this was the version generally used by PGP users all over the world. You may still use PGP 2.3a if you want to, but you may experience problems when trying to process messages and keys generated with PGP 2.6 and later versions, or when using keys that are larger than 1280 bits (the maximum size is now 2048 bits). PGP 2.3a is presumably illegal to use within the USA because of patent restrictions.

PGP 2.6ui

This is an unofficial, "hacked" version of PGP 2.3a, which aimed at correcting the incompatibility problems introduced by MIT PGP 2.6. Please observe that PGP 2.6ui is not a "true" 2.6 version as it is based on the source code for PGP 2.3a, and as such does not include the improvements and bug fixes found in the newer versions. PGP 2.6ui was published by mathew in the UK, but is no longer supported.

PGP 2.62ui

Tony Lezard in the UK based this version of PGP on mathew's 2.6ui, but tried to bring it up-to-date with the latest PGP 2.6x improvements (bigger keys, bug-fixes, new command options etc.).

MIT PGP 2.6.2

This is the latest official version of PGP, released by MIT and adapted (some would say mangled) for use in the USA:

1. It creates messages that cannot be read by PGP versions prior to 2.5.

2. It uses the RSAREF encryption library, making it slightly slower on most platforms. Furthermore, it does not understand the old signature format used by PGP 2.2 and earlier versions.

PGP 2.6.2 is illegal to export from USA, but once exported anyone may use it freely. This version corrects a number of bugs found in PGP 2.6 and 2.6.1. If you are a US citizen living in the US, this is probably the PGP version you want. MIT PGP 2.6.2 can be downloaded here.

PGP 2.6.3i

This is the latest international version of PGP, based on the source code for MIT PGP 2.6.2 and modified for international use. PGP 2.6.3i is published by Ståle Schumacher in Norway, and differs from MIT PGP 2.6.2 in the following ways:

- It does not use the RSAREF encryption library

- It is 100% compatible with all other PGP 2.x versions

- It corrects a number of bugs present in PGP 2.6.2(i)

- It compiles "out of the box" for many new platforms

- It adds some new features without breaking compatibility with earlier versions

PGP 2.6.3i is the most flexible, up-to-date version of PGP available today. PGP 2.6.3i is probably illegal to use within the USA, so if you are a US citizen, you should use MIT PGP 2.6.2 or PGP 2.6.3 instead.

PGP 2.6.3

If you compile the source code for PGP 2.6.3i using the `-DUSA` option and linking with RSAREF (`rsaglue2.c`) instead of MPILIB (`rsaglue1.c`), you will get a version that identifies itself as PGP 2.6.3. It contains all the same bug-fixes and improvements as PGP 2.6.3i, but it will be slightly slower, and the "legal kludge" cannot be disabled. PGP 2.6.3 is not an official PGP version, but is still perfectly legal to use inside the USA. It is only available as source code, and not (yet) as precompiled binaries.

ViaCrypt PGP 2.7.1

ViaCrypt PGP is a commercial version of PGP available in the US and Canada only. Phil Zimmermann says that no compromises in the cryptographic strength of PGP were made for ViaCrypt's version of PGP. The ViaCrypt PGP package includes program disks (executables only, no source code), a user manual, and an individual user license. The current release is available for MS-DOS, Macintosh and UNIX. There is a special version available which interfaces to CompuServe's CIM. Prices start at $100 for the DOS version.

To purchase ViaCrypt PGP or to find out more about it, you can contact:

```
ViaCrypt
2104 W. Peoria Avenue
Phoenix, AZ  85029
USA

Phone: 602-944-0773
Fax: 602-943-2601
Credit card orders 800-536-2664
(0800-1700 MST, Mon-Fri)
E-mail: <info@viacrypt.com>
```

NSA Releases Crypto Documents

The United States National Security Agency has declassified and sent to the National Archives more than 1.3 million pages of historic material gleaned by US cryptographers. The documents cover a stretch from before World War I to the end of World War II. The collection will be available starting April 4 at the National Archives, the federal agency that collects and catalogs official US documents. Among the material being released are details of the "Codetalkers," Navajo Indians who used their native tribal language to confuse US wartime enemies while communicating to US forces.

For those with World Wide Web access, a list of the documents is available on the NSA web page under *Project OPENDOOR*. The Uniform Resource Locator (URL) is `http://www.nsa.gov:8080/`.

For those without Web access, **S-LOST** (Fred Kolbrener) has uploaded a copy of the listing of the documents on file at the National Archives which NSA declassified. The file is called `NARA-NSA.ZIP` and is located in the in the directory `/crypto/general.crypt.info` in the Crypto Drop Box (`sage.und.nodak.edu`). The file size is about 125Kb and the contents of the `ZIP` file expand to more than 400 kilobytes of ASCII listings.

Basic Peeks, Pokes and Subroutines

Mike Todd

This list of memory addresses contains information about the status of the PC and/or locations which may be modified to change the way the PC reacts to its world. The programs to use this information have been compiled by a large number of people. Some of it may be gleaned from the IBM Technical Reference Manual and the `BASIC` manual. Some of it exists only because someone worked very hard to find a way to make something happen. Some of it exists because someone made a mistake and was presented with a new capability because of it. No matter who found it or how the information was found this document provides some really useful information for `BASIC` programmers. It even provides good information for other programmers who will access the PC memory directly.

BIOS LOCATIONS

By specifying a `DEF SEG=&H40` in any `BASIC` program, it is possible to reference the following vectors (fields) in the ROM BIOS area by using a `PEEK` function and the following offsets from the current segment as defined by the `DEF SEG` statement.

- **Offset** `&H0`
  RS232 Addresses on your IBM PC. This will allow you to tell how many (up to four) async cards are attached, if any.

- `&H8`
  Printer Addresses on your IBM PC. This will tell you what printer addresses, and how many (up to four) exist. Each is addressed by a two byte Hex value.

- `&H10`
  Equipment Flag. This field describes the setting of the options switches. It describes what optional devices are attached to the system. The following lists the bit-significance of this field:

  - **Bit 0** — Indicates that there are diskette drives on the system. `0` = No diskettes, `1` = 1 or more diskettes.
  - **Bit 1** — 8087. `0` = Not installed, `1` = Installed.
  - **Bit 2,3** — Planar RAM Size

PC-1	XT and PC-2
00=16K	64K
10=32K	128K
01=48K	192K
11=64K	256K

  - **Bit 4,5** — Initial Video Mode. `00` = Unused, `10`=40x25 Color, `01`=80x25 Color, `11`=80x25 Mono or both.
  - **Bit 6,7** — Number of Diskette Drives (bit 0 = 1 in all cases). `00`=1, `10`=2, `01`=3, `11`=4.
  - **Bit 8** — Unused.
  - **Bit 9,10,11** — Number of RS232 Cards attached.
  - **Bit 12** — Game I/O Attached.
  - **Bit 13** — Not used.
  - **Bit 14,15** — Number of printers attached.

- **&H13** — Memory Size in K bytes.

- **&H15** — I/O RAM Size in K bytes.

- **&H17** — Keyboard Flag – the following lists the masks set to describe current keyboard status:

  Byte 1

  - `&H80` - Insert state active
  - `&H40` - Caps Lock State Has been toggled
  - `&H20` - Num Lock State has been toggled

  - `&H10` - Scroll Lock State has been toggled
  - `&H08` - Alternate Shift key depressed
  - `&H04` - Control Shift key depressed
  - `&H02` - Left Shift key depressed
  - `&H01` - Right Shift key depressed

  Byte 2

  - `&H80` - Insert Key is depressed
  - `&H40` - Caps Lock Key is depressed
  - `&H20` - Num Lock Key is depressed
  - `&H10` - Scroll Lock key is depressed
  - `&H08` - Suspend key has been toggled

- `&H49` — Current CRT mode
  - `&H00` - 40x25 BW
  - `&H01` - 40x25 Color
  - `&H02` - 80x25 BW
  - `&H03` - 80x25 Color
  - `&H04` - 320x200 Color
  - `&H05` - 320x200 BW
  - `&H06` - 640x200 BW
  - `&H07` - 80x25 B&W Card – specialized use, used internally by the video routines.
- `&H4A` — Number of CRT columns.
- `&H50` — Cursor Position (one of eight).
- `&H60` — Current cursor mode.
- `&H6C` — Low word of Timer count.
- `&H6E` — High word of Timer count.
- `&H71` — `&H07` - Break key depressed.
- `&HFA6E` — Beginning of character regen memory.
- `&HFF53` — Print Screen [`PRTSC`] routine address.

Following are some BASIC statements and subroutines showing how to use the above information plus additional functions.

KEYBOARD:

- To disable entire keyboard:
```
DEF SEG=64:  OUT 97,204
```
- To re-enable keyboard:
```
DEF SEG=64:  OUT 97,76
```
- Clear line buffer:
```
DEF SEG: POKE 106,0
```
- Clear Keyboard Buffer:
```
DEF SEG=0:  POKE 1050,PEEK(1052)
```
- Turn on NUM LOCK:
```
DEG SEG=&H40:
POKE &H17,PEEK(&H17) OR 32
```
- Turn off NUM LOCK:
```
DEG SEG=&H40:
POKE &H17,PEEK(&H17) AND 223
```
- Turn on CAPS LOCK:
```
DEG SEG=&H40:
POKE &H17,PEEK(&H17) OR 64
```
- Turn on CAPS LOCK:
```
DEG SEG=&H40:
POKE &H17,PEEK(&H17) AND 171
```

Restore Function Keys to Default values:

```
10 DEF SEG = &HFACE
20 K = 1
30 I = 13
40 T$ = STRING$(13,32): J = 1
50 T1 = PEEK(I):IF T1 < 0 THEN MID$(T$,J,1) = CHR$(T1):J = J + 1:
   I = I + 1 : GOTO 50
60 KEY K,LEFT$(T$,J-1):IF K <10 THEN K = K + 1: I = I + 1: GOTO 40 :
   ELSE KEY ON
```

Disable and re-enable CTRL+BREAK:

```
100 ' Subroutine to save old CTRL+BREAK address and set new to IRET
110 DIM OLD%(4)
120 DEF SEG=0
130 FOR I=&H6C TO &H6F
140   OLD%(I-&H6C)=PEEK(I)  ' The array OLD%() must remain available
150 NEXT
160 ' establish new CTRL+BREAK address  (point to IRET)
170 POKE &H6C,&H53
180 POKE &H6D,&HFF
190 POKE &H6E,&H0
200 POKE &H6F,&HF0
210 DEF SEG: RETURN ' CTRL+BREAK will return to the program
220 ' Subroutine to reset old CTRL+BREAK address
230 DEF SEG=0
240 FOR I=&H6C TO &H6F
250   POKE I,OLD%(I-&H6C)
260 NEXT
270 DEF SEG: RETURN ' CTRL+BREAK will interupt the program
```

SCREEN: Determine Monitor type:

```
10 DEF SEG = 0
20 MONITOR.TYPE = PEEK(&H410) AND &H40
30 IF MONITOR.TYPE = 1 THEN PRINT "40 X 25 Color"
40 IF MONITOR.TYPE = 32 THEN PRINT "80 X 25 Color"
50 IF MONITOR.TYPE = 48 THEN PRINT "Monochrome"
60 IF MONITOR.TYPE = 64 THEN PRINT "Both"
```

Switch displays on a system with both monochrome and color/graphics:

```
10 'Subroutine to switch to monochrome adaptor
20 DEF SEG=0
30 POKE &H410,(PEEK(&H410) OR &H30)
40 SCREEN 0           ' restore screen to text mode
50 WIDTH 80           ' restore screen width to 80
60 LOCATE ,,1,12,13   'restore cursor to "normal"
70 RETURN
80 ' Subroutine to switch to color adaptor.  Change
90 ' SCREEN and WIDTH statements for different modes
100 DEF SEG=0
110 POKE &H410,(PEEK(&H410) AND &HCF) OR &H10
120 SCREEN 1,0,0,0    ' Medium resolution graphics, color,
                        active page 0, visual page 0
130 WIDTH 40          ' WIDTH 80 would force the screen to clear and
                        set to SCREEN 2 for high resolution
140 LOCATE ,,1,6,7    ' restore cursor to "normal" for color screen
150 RETURN
```

Subroutine to save and restore a screen image:

```
100 DEF SEG = &HB800                        'Save screen image
110 INPUT "Enter name of file";FILENAME$  'for color/graphics
120 BSAVE FILENAME$,0,&H4000              'display in text mode
130 RETURN
1000 INPUT "Enter name of file";FILENAME$ 'Restore image
1010 CLS
1020 DEF SEG = &HB800                       'Change DEF SEG=&HB800
1030 BLOAD FILENAME$                        'to &HB000 for monochrome
```

Set split screen scroll window starting on line X ending on line Y:

```
DEF SEG: POKE 91,X: POKE 92,Y  'Sets up "window"
LOCATE X,C  'Place cursor "in" the scroll window at line X column C
```

Set window width:

```
DEF SEG: POKE 41,X  'Set window width to X
```

Set 16 background colors:

```
While WIDTH 40: OUT &H3D8,8
While WIDTH 80: OUT &H3D8,9
```

Use other than palate color 3 for characters in medium resolution

```
DEF SEG: POKE &H4E,COLOR '(COLOR may be 1, 2, or 3)
```

DISKETTE DRIVES:

Read drive switches:

```
DEF SEG = 0: NUMBER.OF.DRIVES% = PEEK(&H410) AND &HC0
```

Current disk information:

```
DEF SEG=64
PEEK(69) --- Tracks
PEEK(70) --- Heads
PEEK(71) --- Sectors
PEEK(72) --- Bytes per sector
```

PRINTER:

Printer Status (works with IBM Dot Matrix and Epson printers)

```
DEF SEG=64
A=PEEK(8)+256*PEEK(9)
B=(INT(A+1) AND 248) XOR 72
IF (B AND 128)<>128 THEN PRINTER.STATUS="OFF LINE"
ELSE PRINTER.STATUS="ON LINE"
```

Initialize Printer:

```
DEF SEG=64
A=PEEK(8)+256*PEEK(9)
DEF SEG: OUT A+2,8
OUT A+2,12
```

MEMORY:

Determine amount of memory installed (Only works for greater than 48k):

```
DEF SEG = 0: MEMORY% = PEEK(&H413)+(256*PEEK(&H414))
```

or, put another way:

```
DEF SEG=0
((PEEK(1040) and 12) + 4 ) * 4} --- Memory on Mother-board
PEEK(1045) + 256 * PEEK(1046)} --- Expansion memory (add on)
PEEK(1043) + 256 * PEEK(1044)} --- Total memory
```

GAME ADAPTER: Determine if game adapter exists:

```
DEF SEG = 0: GAME.ADAPTER% = PEEK(&H411) AND &H10
IF GAME.ADAPTER% = 0 THEN GAME.ADAPTER$="No"
     ELSE GAME.ADAPTER$="Yes --Installed"
```

MISCELLANEOUS:

To unprotect a BASIC program that was saved with `",P`. First you must create a file to overlay the `,P` setting. From the DOS prompt start up `BASICA` or `BASIC` and enter the BASIC command `BSAVE "UN.P",1124,1`. This will create a file on your default drive named `UN.P`.

Next `LOAD` your program that had been saved using `",P`. If it was named `MYPROG.BAS` the BASIC command would be `LOAD "MYPROG`.

Now to use the `UN.P` file to overlay the protection setting use the command `BLOAD "UN.P",1124`.

You may now use the `LIST`, `EDIT` and `SAVE` commands as usual.

INTERRUPTS:

Following is a program by Richard Tremmel to print all the BASIC interrupts. It is set to write the interrupt locations to a printer:

```
100 REM INT-LIST IBM-PC Software Interrupt Listing by Richard L. Tremmel
110 DEF SEG=0
120 DIM NAM$(255)
130 DEF FNJUSTIFY$(STRG$,LNGTH) = RIGHT$("0000"+STRG$,LNGTH)
140 TITLE$ = "INT ADDR   VECTOR        FUNCTION      "
142 REM
144 FOR I = 0 TO 39 : READ NAM$(I) : NEXT I
146 FOR I = 40 TO 63 : NAM$(I) = "Reserved for DOS" : NEXT I
148 NAM$(64) = "Revector Diskette" : NAM$(65) = "Fixed Disk Parms"
150 FOR I = 66 TO 95 : NAM$(I) = "Reserved" : NEXT I
152 FOR I = 96 TO 103 : NAM$(I) = "User Interrupts" : NEXT I
```

```
154 FOR I = 104 TO 127 : NAM$(I) = "Not Used" : NEXT I
156 FOR I = 128 TO 133 : NAM$(I) = "Reserved by BASIC" : NEXT I
158 FOR I = 134 TO 240 : NAM$(I) = "BASIC Interpreter" : NEXT I
160 FOR I = 241 TO 255 : NAM$(I) = "Not Used" : NEXT I
165 REM
170 OPEN "LPT1:" FOR OUTPUT AS #1 : PRINT #1,""
180 FOR PAGE = 1 TO 2
190   PRINT #1,CHR$(12); TITLE$; " "; TITLE$
200   FOR K = (PAGE-1)*128 TO (PAGE-1)*128+64-1
210     I = K    : ADDRESS = I*4 : GOSUB 280 : PRINT #1," ";
220     I = K+64 : ADDRESS = I*4 : GOSUB 280 : PRINT #1,""
230   NEXT K
240 NEXT PAGE
250 CLOSE #1
260 SYSTEM
270 REM
280 PRINT #1," "; FNJUSTIFY$(HEX$(I),2);
290 PRINT #1," "; FNJUSTIFY$(HEX$(ADDRESS),4);
300 PRINT #1," "; FNJUSTIFY$(HEX$(PEEK(ADDRESS+2)+PEEK(ADDRESS+3)*256),4);
310 PRINT #1,":"; FNJUSTIFY$(HEX$(PEEK(ADDRESS+0)+PEEK(ADDRESS+1)*256),4);
320 PRINT #1," "; LEFT$(NAM$(I)+SPACE$(20),20);
330 RETURN
340 REM

1000 DATA "Divide by Zero"
1001 DATA "Single Step"
1002 DATA "Nonmaskable"
1003 DATA "Breakpoint"
1004 DATA "Overflow"
1005 DATA "Print Screen"
1006 DATA "Reserved"
1007 DATA "Reserved"
1008 DATA "Time of Day"
1009 DATA "Keyboard"
1010 DATA "Reserved"
1011 DATA "Communications"
1012 DATA "Communications"
1013 DATA "Disk"
1014 DATA "Diskette"
1015 DATA "Printer"
1016 DATA "Video"
1017 DATA "Equipment Check"
1018 DATA "Memory"
1019 DATA "Diskette/Disk"
1020 DATA "Communications"
1021 DATA "Cassette"
1022 DATA "Keyboard"
1023 DATA "Printer"
1024 DATA "Resident BASIC"
1025 DATA "Bootstrap"
1026 DATA "Time of Day"
1027 DATA "Keyboard Break"
1028 DATA "Timer Tick"
1029 DATA "Video Parameters"
1030 DATA "Diskette Parameters"
1031 DATA "Video Graphics Exten"
1032 DATA "DOS Program Term"
1033 DATA "DOS Function Call"
1034 DATA "DOS Terminate Addr"
1035 DATA "DOS Ctrl Break Exit"
1036 DATA "DOS Fatal Error"
1037 DATA "DOS Abs Disk Read"
1038 DATA "DOS Abs Disk Write"
1039 DATA "DOS Terminate & Fix"
1256 END 'of program.
```