****

COMPUTER SUPPLEMENT #17

****

In this issue:

CAESAR SOLUTIONS — KARL has a QuickBASIC program to display Caesar solutions.

BACONIAN CYPHER AID REVISITED — G4EGG has some extensions to THE DOC's program.

PRETTY GOOD PRIVACY — A public key encryption system is now available as freeware, including source code.

AN AID IN FINDING VIGENERE KEYWORDS — G4EGG has a program to help with Vigenere cyphers.

THE POLLUX CIPHER — BOATTAIL has a Pascal program to solve Pollux ciphers by trial and error.

BEGINNER'S GUIDE TO THE ENVIRONMENT — An introduction to the MS-DOS environment.

CONS WITH CRYPTO — DAEDALUS gives a step-by-step method of creating cons using BITSIFTER's program `CRYPTO`.

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

Published for 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 GWBASIC (1p). [Also on Issue Disk #11].
BIBLIOG — A bibliography of computer magazine articles and books dealing with cryptography (2p). (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 prior to 13 have been exhausted, and are awaiting reprinting. Contact the Editor for current availability.

ISSUE DISKS

$5 per disk; specify issue(s), format and density required. All issues are presently available on two IBM High Density 1.2M disks, archived with PKZIP. For other disk formats, ask. Disk One — Issues 1 - 10; Disk Two — issues 11 to current. Disks contain ONLY 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 TEX format is available for issues 16 to current. Available from the Editor.

TO OBTAIN THESE MATERIALS

Write to:	Or via Electronic Mail:
Dan Veeneman PO Box 2442 Columbia, Maryland 21045-2442, USA.	dan%decode.UUCP@uunet.uu.net or uunet!anagld!decode!dan

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 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.

CAESAR SOLUTIONS

KARL

Microsoft's QuickBASIC is a joy to use after years of working with the old versions of standard BASIC, and its capability for compilation as DOS-executable files is dessert. The `CAESAR03.BAS` source code herewith is one of my first attempts to wean myself away from old BASIC; its speed of execution was almost unbelievable when I first ran it, compared to its old-BASIC counterpart. After compiling it to the DOS-EXE version, it was so fast as to appear instantaneous.

A line-by-line study of the source-code with its attendant annotation affords a step-by-step description of program execution, due to Quick-BASIC's top-down construction. The DESCRIPTION and LIMITATIONS paragraphs explain the program in general.

Note that between the `Begin: END` lines I have inserted a `REM` symbol before the `RUN "CRYPTO.BAS"` line. This line would normally return the action to the MENU program, from which I can call any one of 22 programs (so far) of my working computer-crypto tools, as included in my book, *COMPUTER CRYPTOLOGY: Beyond Decoder Rings* (Prentice-Hall, 1988). Because QuickBASIC will run both old BASIC versions as well as the new, I can continue to use the MENU-driven toolset while rewriting the programs one by one in the new format.

In QuickBASIC, sub-procedures are not equivalent to sub-routines as in old BASIC. They stand alone, and their variables are immune from values used on other parts of the program, unless special indications to make the usable elsewhere are included. Subroutines may be used in both the main module and the sub-procedures, although the line may not be crossed — that is, a subroutine appearing in a sub-procedure cannot be called from the main module, for instance.

Program action is controlled step-by-step by the lines between `Begin: END` (the fourth to ninth lines of the program proper). The first command is `GOSUB OPENER`. This transfers action to line 10, a subroutine almost like old BASIC, in that, when its action is completed, command is transferred back to the line following `GOSUB OPENER`.

The `OPENER:` line activates the screen and printer as "devices," in the same manner as files are opened. The next set of lines continue the opening action by presenting screen information in the form of a short page of introductory text each time the program is activated. The final line of this subroutine clears the screen and ends the subroutine action. Action now returns to the next line between `Begin: END` — `CALL Processor(Prnt\$())`.

Sub-procedure `CALL Processor(Prnt$())` becomes the commanding procedure. Its main loop, `WHILE Key$ <> "^" WEND`, will "play" over and over until you enter a caret at the prompt, at which instant command reverts to the next line between `Begin: END`, which is `GOSUB CLOSER`. The closer does what the name implies.

Since we are not yet ready to close, the first line within the action loop of the sub-procedure calls for screen-print of the legend, or prompts, via the `Legend: RETURN` subroutine within the procedure. This is another interesting but not often used feature of Quick-BASIC; a subroutine is permissible within a sub-procedure. Note that physically the subroutine is the last few lines of the program (not a general requirement), though they are NOT the final lines executed.

The `DO LOOP` sequence is next, with its first action being a sub- `DO LOOP` that repeats until a key is touched. The `SELECT CASE` / `END SELECT` sequence acts in much the same way as multiple `IF - THEN - ELSE` sequences. Note

that the action items at each `CASE` are the symbols indicated in the prompt line of screen including a space-bar input.

The screen is split vertically following the keyboard input of a letter. An endless "belt" of A to Z's is generated. Note the line-by-line `REM`'d annotations leading to the splitter, which uses the `MOD` function limiting the character printing to 39 items. Another `DO LOOP` sequence kicks in, fully annotated for analysis. QuickBASIC uses a `LOCATE x,y` instead of the old BASIC `PRINT @`.

The parameters for `LOCATE` are set within the `FOR-NEXT` loop, altered as required by keyboard input within the overriding `DO-LOOP` sequence and the X1 - X2 values.

If the "?" is entered, the `SELECT CASE / END SELECT` function calls the Sub-procedure `HPrnt(Prnt$())`. This is not often used since the program is primarily for revealing one or two words, as in *The Cryptogram*, Vol. LVIII, No. 4, page 22, Patristocrats cribs-within-parentheses. I entered the ciphertext from the cover of this issue on the off-chance that it might be a Caesar substitution cipher, and sure enough, it is. The first seven words are revealed as the last horizontal line, Block 2, of the screen-print of this program, herewith. The input appears as the top line, Block 1. Spaces were arbitrarily entered at the end of each word for clarity; ordinarily the cipher- and plaintext would appear as solid lines.

```
DPNF FBSMZ BOE TUBZ MBUF TQFOE ZPVS     QCAS SOFZM OBR GHOM ZOHS GDSBR MCIF
EQOG GCTNA CPF UVCA NCVG URGPF AQWT     RDBT TPGAN PCS HIPN APIT HETCS NDJG
FRPH HDUOB DQG VWDB ODWH VSHQG BRXU     SECU UQHBO ODT IJQO BQJU IFUDT OEKH
GSQI IEVPC ERH WXEC PEXI WTIRH CSYV     TFDV VRICP REU JKRP CRKV JGVEU PFLI
HTRJ JFWQD FSI XYFD QFYJ XUJSI DTZW     UGEW WSJDQ SFV KLSQ DSLW KHWFV QGMJ
IUSK KGXRE GTJ YZGE RGZK YVKTJ EUAX     VHFX XTKER TGW LMTR ETMX LIXGW RHNK
JVTL LHYSF HUK ZAHF SHAL ZWLUK FVBY     WIGY YULFS UHX MNUS FUNY MJYHX SIOL
KWUM MIZTG IVL ABIG TIBM AXMVL GWCZ     XJHZ ZVMGT VIY NOVT GVOZ NKZIY TJPM
LXVN NJAUH JWM BCJH UJCN BYNWM HXDA     YKIA AWNHU WJZ OPWU HWPA OLAJZ UKQN
MYWO OKBVI KXN CDKI VKDO CZOXN IYEB     ZLJB BXOIV XKA PQXV IXQB PMBKA VLRO
NZXP PLCWJ LYO DELJ WLEP DAPYO JZFC     AMKC CYPJW YLB QRYW JYRC QNCLB WMSP
OAYQ QMDXK MZP EFMK XMFQ EBQZP KAGD     BNLD DZQKX ZMC RSZX KZSD RODMC XNTQ
PBZR RNEYL NAQ FGNL YNGR FCRAQ LBHE     COME EARLY AND STAY LATE SPEND YOUR

  < ^ > to Quit, < ? > to print,  < # > to clear screen,   Space-Bar OK
```

Output of CAESAR03.BAS, with *Cryptogram* ciphertext on the upper-right line and a solution on the bottom-right.

CAESAR03.BAS

```
REM     filespec: CAESAR03.BAS                  a CRYPTOLOGICAL PROGRAM

'DESCRIPTION:
'        This program has been adapted from one of the programs in the book,
'   "COMPUTER CRYPTOLOGY: Beyond Decoder Rings," by Karl Andreassen, Prentice
'   Hall, 1989."  The program was originally prepared in TRSDOS BASIC, a
'   proprietary version of MicroSoft BASIC.  Karl Andreassen is a pseudonym
'   of Waldo T. Boyd, and this program and the book is copyright in the
'   latter family name.
'        The program decrypts simple substitution-type ciphers and presents
'   25 "solutions."  One of these solutions wo;; be in plain language, called
'   "plaintext" by cryptologists.  It will likewise accept plain language input
'   at the keyboard, converting it to 25 Caesar versions of straight
'   substitution ciphertext, of infinite length.

'LIMITATIONS:
'        Presentation is in vertical columns.  Because most screens are used
'   with 24 horizontal lines while the alphabet contains 26 alpha characters,
'   the screen is divided into two blocks of 13 characters each, which per-
'   mits full-screen viewing of the entire alphabet.  The printer function
'   presents the original and 25 lines full length.
'        The program and its output is elementary since it is not intended
'   for more than a few minutes' use by cryptanalysts seeking to test short
'   phrases or keywords during work with more elaborate programs.  Due to
'   the double block format, upon reaching the half-screen position the
'   screen replaces older letters with those more recent.  Thus, if working
'   with longer text, frequent print-outs should be made, and the screen
'   cleared after each printing.

'ASSUMPTIONS:
'        A mono-screen or color screen with mono capability is available.
'   Although a print-out function is provided, it is not likely that
'   it will be often used in view of the elementary nature of this program.

                              '**********************

DECLARE SUB Processor (Prnt$())
DECLARE SUB HPrnt (Prnt$())

DIM Prnt$(26, 80)

Begin:
   GOSUB OPENER
   CALL Processor(Prnt$())
   GOSUB CLOSER
   RUN "CRYPTO.BAS"
END

OPENER:
   CLS
   OPEN "SCRN:" FOR OUTPUT AS #1
   OPEN "LPT1:" FOR OUTPUT AS #2
```

```
   LOCATE 2, 1
   PRINT #1,
   PRINT #1, TAB(25); STRING$(30, "=")
   FOR x = 1 TO 5
      PRINT #1, TAB(25); "||"; TAB(53); "||"
   NEXT x
   LOCATE 5, 33
   PRINT #1, "QUICK TEST FOR"
   LOCATE 6, 38
   PRINT #1, "CRIBS"
   LOCATE 7, 31
   PRINT #1, "by Karl Andreassen"
   LOCATE 9, 25
   PRINT #1, TAB(25); STRING$(30, "=")
   PRINT #1,
   PRINT #1, TAB(12); "This program will print letters on-screen in two vertical"
   PRINT #1, TAB(12); "descending orders.  Input from keyboard will be either"
   PRINT #1, TAB(12); "plain language or ciphertext.  If the ciphertext has been"
   PRINT #1, TAB(12); "created by straight substitution without added complication"
   PRINT #1, TAB(12); "the plain language will appear on a subsequent horiz. line."
   PRINT #1,
   PRINT #1, TAB(12); "ENTER LETTERS ONLY:  NO PUNCTUATION PERMITTED IN TEXT."
   PRINT #1,
   PRINT #1, TAB(12); "(Touch <ENTER> to begin..... .... .... ....all subsequent"
   PRINT #1, TAB(12); "letters will appear with alphabetic extensions."
   PRINT #1,
   INPUT "", du$
   CLS
RETURN

CLOSER:
   CLOSE #1
   CLOSE #2
RETURN

SUB HPrnt (Prnt$())
   FOR Row = 1 TO 13                        '==Lprint first block
      FOR Col = 1 TO 40
         PRINT #2, Prnt$(Row, Col);
      NEXT Col
      PRINT #2,
   NEXT Row
   FOR Row = 1 TO 13                        '==Followed by 2nd block
      FOR Col = 41 TO 80
         PRINT #2, Prnt$(Row, Col);
      NEXT Col
      PRINT #2,
   NEXT Row
   PRINT #2,                                '==Single space between printings

END SUB
```

```
SUB Processor (Prnt$())
'
WHILE Key$ <> "^"

   GOSUB Legend                  '==Screen-print the option legend

   DO                            '==Loop while non-alpha input
      DO                         '==Keyboard input loop
         Key$ = INKEY$
      LOOP WHILE Key$ = ""

      SELECT CASE Key$
         CASE " "                '==Space-bar is special case, handle...
            EXIT DO              '==...outside the loop
         CASE "^"
            EXIT SUB             '==End of program, close devices
         CASE "?"
            CALL HPrnt(Prnt$())  '==Printer call
         CASE "#"
            CLS                  '==Hot restart with clear screen
            GOSUB Legend         '==Refresh the footnote
            X3 = 1               '==Reset the determinants
            P1 = 1
         CASE ELSE
            Key$ = UCASE$(Key$)  '==Continue program if alpha charac.
         END SELECT

   LOOP WHILE ASC(Key$) < 65 OR ASC(Key$) > 90

   Nr = ASC(Key$) - 66           '==Convert incoming letters to ASCII
   X3 = 1                        '==Begin upper-left Block 1
   P1 = P1 + 1                   '==Prep for vertical screen printing
   P = P1                        '==Store P1 before shift to Block 2
   P1 = P1 MOD (39)              '==Split the screen horizontally

   DO
      SELECT CASE X3
      CASE 1                     '==Column 1 loop params.
         X1 = 1                  '==Upper-left corner of screen block 1
         X2 = 13                 '==Lower-left corner of screen block 1
      CASE 2                     '==Column 2 loop params.
         X1 = 14                 '==Upper-left corner of screen block 2
         X2 = 26                 '==Lower-left corner of screen block 2
         P = P + 40 + P1         '==Block designator
      END SELECT

      FOR x = X1 TO X2           '==Obtain 1/2 alphabet builder
         C = Nr + x              '==Get progressive alpha params.
         M = C MOD (26) + 1      '==Endless A to Z beltline

         P3 = INT(P / 80)        '==Row determinant for screen location
         P4 = INT(P - P3 * 80) + 1 '==Column   "    "     "       "
         P3 = P3 + 1             '==Next row
```

```
         LOCATE P3, P4          '==Row, column cursor locator

         IF Key$ = " " THEN     '==A space is a special case
            PRINT #1, " ";
            Prnt$(P3, P4) = " " '==Load hardprint array with space
         ELSE
            PRINT #1, CHR$(M + 64); " ";  '==Print column of consec. letters
            Prnt$(P3, P4) = CHR$(M + 64)  '==Load hardprint array with letter
         END IF

         P = P + 80             '==Prep for new line
      NEXT x

      P = 0                     '==Reset print position flag
      X3 = X3 + 1               '==Xfer action to  Block 2
      P = P + 1

   LOOP WHILE X3 < 3            '==Start over at the top
WEND

Legend:                         '==User convenience footnote
   LOCATE 20, 1
   PRINT #1, TAB(9); "< ^ > to Quit,  < ? > to print,  ";
   PRINT #1, " < # > to clear screen,   Space-Bar OK "
RETURN

END SUB
```

PRETTY GOOD PRIVACY

Pretty Good Privacy version 2.1 has been released. The public domain version of the public key software is available from several archive sources.

From the documentation:

PGP (Pretty Good Privacy) version 2.1 — RSA public-key encryption freeware for MS–DOS, protects E-mail. Lets you communicate securely with people you've never met, with no secure channels needed for prior exchange of keys. Well featured and fast ! Excellent user documentation.
PGP has sophisticated key management, an RSA/conventional hybrid encryption scheme, message digests for digital signatures, data compression before encryption, and good ergonomic design. Source code is free.

BACONIAN CYPHER AID REVISITED

G4EGG

THE DOC's programme and article (CS #7, page 12) concludes with an invitation to extend the programme. The change to allow interactive use is trivial, and one version is given herewith. (The programme differs from the original in line numbering and some cosmetic detail. Also whilst making the mods., several example cons. were used, and so the included example is also different from that chosen by THE DOC.) In use, selecting the default example gives the screen:

```
                          BACON CYPHER AID         by M. Dale.
                          ----------------
Enter UPPER case, letters only. (F) if data on disc

BRAIDPROXYFAIRSGRAINJOKERDROVESTOVEMOVEDFROZEBOUTSZILCHSHARPSNOWYTROOPBONESDECA
YBROWNVEINSSTONEDONORGIVENWORSTSQUADFLAMEMOOSEBAYOUINEPTSOUSEDROSSZEBRA

Enter crib in UPPER case without spaces:
GOOD
.............................

ABCDEFGHIJKLMNOPQRSTUVWXYZ   ( 1 )
baa    bb  b aab aabb b ba

BRAID PROXY FAIRS GRAIN JOKER DROVE STOVE MOVED FROZE BOUTS ZILCH SHARP SNOWY
aabb  baa b abbaa  abba  a  a  aa   aba    a     aaa  aabba abbab abbab aaabb
             N                                        G     O     O     D
TROOP BONES DECAY BROWN VEINS STONE DONOR GIVEN WORST SQUAD FLAME MOOSE BAYOU
baaab aaa a   abb aaaba   baa abaa   aaaa ab  a baaab a bb  abb    aaa  abbab
  S                C                             S                      O
INEPT SOUSE DROSS ZEBRA
ba bb aaba   aaaa a aab

(D)one, (L)ook more, (N)ew crib, (A)dd to crib: ?
```

It is as well to "Look more", and find all possible positions of the crib. The most likely is noted, and the selection "New crib" used to restart. In this case, there is only one position for GOOD. Adding to crib is not restricted to extending the existing letters. Any word or letter in any position may be used. Now look at group 27. This must be a "Z" or "U/V". "U" is most likely, so "Add to crib" the "U" and get:

```
Enter crib in UPPER case without spaces: U
.............................

ABCDEFGHIJKLMNOPQRSTUVWXYZ   ( 1 )
baa    bb  b aab aabb bbba

BRAID PROXY FAIRS GRAIN JOKER DROVE STOVE MOVED FROZE BOUTS ZILCH SHARP SNOWY
```

```
aabb  baabb abbaa  abba  a  a  aa   aba    a     aaa  aabba abbab abbab aaabb
        U     N                                       G     O     O     D
TROOP BONES DECAY BROWN VEINS STONE DONOR GIVEN WORST SQUAD FLAME MOOSE BAYOU
baaab aaa a   abb aaaba   baa abaa   aaaa ab  a baaab a bb  abb    aaa  abbab
  S                C                              S                       O
INEPT SOUSE DROSS ZEBRA
ba bb aaba   aaaa a aab

(D)one, (L)ook more, (N)ew crib, (A)dd to crib: ?
```

The "U" has appeared in the second group; it was 27 that was required. So just select "Look more", and all is well. Further examination of the groups offer several possibilities, but to illustrate the next point, chose group 18. This must be "E" or "W". Again the vowel is more likely, so add "E" ? Well, no. "E" is a common letter, and may well fit in several places. To avoid a sequence of "Look more"'s, enter a group of letters, some existing, to identify the required group. Here CEI will identify the spot. The screen is now as:

```
Enter crib in UPPER case without spaces: CEI

.............................

ABCDEFGHIJKLMNOPQRSTUVWXYZ   ( 1 )
baaba  bb  b aab aabbab ba

BRAID PROXY FAIRS GRAIN JOKER DROVE STOVE MOVED FROZE BOUTS ZILCH SHARP SNOWY
aabbb baa b abbaa  abba  a aa baaaa abaaa  aaab  aaaa aabba abbab abbab aaabb
  H           N                R     I                G     O     O     D
TROOP BONES DECAY BROWN VEINS STONE DONOR GIVEN WORST SQUAD FLAME MOOSE BAYOU
baaab aaaaa baabb aaaba aabaa abaaa baaaa abaaa baaab a bbb abb a  aaaa abbab
  S     A     U     C     E     I     R     I     S                       O
INEPT SOUSE DROSS ZEBRA
baabb aabaa baaaa aaaab
  U     E     R     B
(D)one, (L)ook more, (N)ew crib, (A)dd to crib: ? A
```

(Don't forget, a U can be a V) And so on. (Group 23 may be next to complete, and then 24 and 25) Several cons. have been tried, with complete success. Good luck with all sols. How did it work out?

BACONAID.BAS

```
100 '   BACON/AID by Martin Dale, Oct 86
110 DATA "BRAIDPROXYFAIRSGRAINJOKERDROVESTOVEMOVEDFROZEBOUTSZILCHSHARPSNOWY
         TROOPBONESDECAYBROWNVEINSSTONEDONORGIVENWORSTSQUADFLAMEMOOSEBAYOU
         INEPTSOUSEDROSSZEBRA","GOOD"
120 SP$(0)=SPACE$(78):SP$(1)=SP$(0):AB$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":W=0
130 T$=CHR$(12)+SPACE$(29)+"BACON CYPHER AID         by M. Dale."
    :U$=SPACE$(29)+STRING$(16,"-")
140 K=0:PRINT T$:PRINT U$:PRINT "Enter UPPER case, letters only. (F) if data on disc"
150 LINE INPUT CT$:IF CT$="" THEN READ CT$:PRINT CT$
160 IF CT$<>"F" THEN 190
170 PRINT:INPUT "Name of file holding data ";N$:I=INSTR(N$,".")
    :IF I>0 THEN N$=LEFT$(N$,I-1)
180 N$=N$+".CT$":OPEN "I",#1,N$:INPUT #1,CT$:CLOSE #1:PRINT CT$
190 I=0
200 I=I+1:J=ASC(MID$(CT$,I,1)) AND 223
    :IF J<65 OR J>90 THEN CT$=LEFT$(CT$,I-1)+MID$(CT$,I+1)
210 IF I<LEN(CT$) THEN 200
220 L=LEN(CT$):IF L<>INT(L/5)*5 THEN PRINT "ERROR in entry. Try again."
    :FOR I=1 TO 2000:NEXT:GOTO 140
230 C=0:LOCATE 7,1:PRINT SP$(0)+SP$(0)+SP$(0):LOCATE 7,5
    :PRINT "Enter crib in UPPER case without spaces: ";
240 LINE INPUT TP$:IF TP$="" THEN READ TP$:PRINT TP$
250 PRINT SPACE$(34):TL=LEN(TP$)*5:PT$=""
260 FOR N=1 TO TL/5:CH=ASC(MID$(TP$,N,1))-65:IF CH>9 THEN CH=CH-1
    :IF CH>20 THEN CH=CH-1
270  FOR K=4 TO 0 STEP -1:IF (CH AND 2^(K))>0 THEN PT$=PT$+"b" ELSE PT$=PT$+"a"
280 NEXT:NEXT
290 FOR M=1 TO L-TL+1 STEP 5:AL$=SP$(W):PN$=MID$(CT$,M,TL):PF=1
300  FOR K=1 TO TL:LT$=MID$(PT$,K,1):PS=ASC(MID$(PN$,K,1))-64
310   IF MID$(AL$,PS,1)=" " THEN AL$=LEFT$(AL$,PS-1)+LT$+RIGHT$(AL$,LEN(AL$)-PS)
320   IF MID$(AL$,PS,1)<>LT$ THEN PF=0
330  NEXT:IF PF=0 THEN 570 ELSE PB=1
340  BC$="":PRINT SP$(0):LOCATE 9,1:PRINT SP$(0):LOCATE 9,1
350  FOR Y=1 TO L STEP 5:  PRINT ".";:P$=MID$(CT$,Y,5):CH$=""
360   FOR Z=1 TO 5:P=ASC(MID$(P$,Z,1))-64:CH$=CH$+MID$(AL$,P,1):NEXT
370   IF LEFT$(CH$,2)="bb" THEN PB=0:Y=L:GOTO 400
380   IF LEFT$(CH$,2)=" b" THEN MID$(CH$,1,1)="a"
390   BC$=BC$+CH$
400  NEXT:IF PB=0 THEN 570
410  LOCATE 10,1:PRINT SP$(0):C=C+1:PRINT AB$;TAB(30);"("C")":PRINT AL$:PRINT
420  FOR I=1 TO L STEP 65:CP$=MID$(CT$,I,65)
430   FOR J=1 TO LEN(CP$) STEP 5:PRINT MID$(CP$,J,5)" ";:NEXT:PRINT
440   B$=MID$(BC$,I,65)
450   FOR J=1 TO LEN(B$) STEP 5:PRINT MID$(B$,J,5)" ";:NEXT:PRINT
460   FOR J=1 TO LEN(B$) STEP 5:BN$=MID$(B$,J,5):CR=0
470    IF INSTR(BN$," ")>0 THEN PL$=" ":GOTO 510
480    FOR K=1 TO 5:IF MID$(BN$,K,1)="b" THEN CR=CR+2^(5-K)
490    NEXT
500    CR=CR+65:IF CR>73 THEN CR=CR+1:IF CR>85 THEN CR=CR+1
510    PL$=CHR$(CR):PRINT "  "PL$"   ";
520  NEXT:PRINT:NEXT
530  LOCATE 23,3:PRINT "(D)one, (L)ook more, (N)ew crib, (A)dd to crib: ";
```

```
     :INPUT Q$:IF Q$="D" THEN RUN "MENU.BAS"
540  LOCATE 23,1:PRINT SP$(0):LOCATE 7,1:PRINT SP$(0):LOCATE 7,5
550  IF Q$="A" THEN SP$(1)=AL$:W=1:Z=1:M=260:GOTO 570
560  IF Q$="N" THEN W=0:Z=1:M=260:GOTO 570
570 NEXT:IF Z=1 THEN Z=0:GOTO 230
580 LOCATE 22,1:PRINT "Possible matches found ="C
590 INPUT " (D)one, (R)un new cypher, or (N)ew crib:";Q$:IF Q$="R" THEN RUN
600 IF Q$="D" THEN RUN "MENU"
610 IF Q$="N" THEN PRINT T$:PRINT U$:W=0:LOCATE 7,5:RESTORE:READ Z$:GOTO 230
620 LOCATE 23,1:GOTO 590
```

1993 ACA CONFERENCE

LANAKI

[**Ed**: I received this announcement from LANAKI just prior to press time. Make your plans now ! **DMV**]

Planning/Execution for the 1993 ACA convention in New Orleans is well under way.

Please announce that the ACA conference has been confirmed at the **Airport Ramada Inn** on August 13–15, 1993 and 4 different hotels are available as backups with varying rates. New Orleans has something for everyone. The Chamber of Commerce is helping LANAKI and KNOAI plan extracurricular events for spouse attendees. Licensed/bonded babysitting will be available for those members with children. Tours to the Zoo, Riverboat trips around the city, NASA space shuttle visit are just three of the attractions being offered for this ACA blowout.

Continental Airlines has been designated by the EB as the Official Airline of the ACA for this Convention. The toll free number for reservations under a very special ACA discount agreement is 1-800-468-7022. KREWE that fly to New Orleans *from anywhere in the world*, booked via Continential, will receive discounts on all tickets purchased ranging from 40% in the USA to 30% on Australia, New Zealand, Japan and Philippines International flights. All passengers will receive 1250 bonus miles on their frequent flyer program. Discounts from Europe, South America, Canada, Mexico, and the Caribbean. So this is the year to take your vacation and go to the ACA convention ! LANAKI negotiated this extraordinary agreement and has details. Plan to come ! Give him an early indication so that he can tailor convention activities to meet your needs. If you have wanted to come to the USA but thought it was too expensive, this year presents the best opportunity at the lowest cost.

A call for papers and presentations is officially extended. Depending on how many papers are received, LANAKI will bind or Xerox™ at no cost a full set of conference papers for each member of the KREWE who is interested. Some papers will be submitted to *Cryptologia* for publication. Papers must be double-spaced, typed and in English. If you can not be at the convention, write the paper, submit to LANAKI and he will have it presented professionally on your behalf.

AN AID IN FINDING VIGENERE KEYWORDS

G4EGG

The Vigenere cypher is a periodic, and described in ELCY, page 108 and Practical Cryptanalysis, Vol. V, page 4. After finding the period, the next step is to determine the keyword, and the accompanying programme helps in this task. It is not too exhaustive to test every possible substitution, and score the results to find the best fit. High frequency letters increase and low frequency ones decrease the score. Then the highest rated are used to pick out the keyword. The programme is described below.

The first lines set up an example and the variables to be used. Lines 140 to 180 are just one way to get the input of cypher text. Spaces are removed by line 190. Line 200 gets the previously determined period, P. It is suggested that 13 is the maximum value, but a 14 has appeared, so change to 14 maximum ? Some nested loops then:

1. split the cypher text into groups, by period, each Pth letter in a group,
2. test each group (column) against an alphabet,
3. step through the alphabets for successive testing,
4. and keep a score of 'good' fits and 'bad' letters.

   Line 310 and onward selects the best or highest scoring 5 substitutions and prints them out in a block so that the keyword may be inferred. A typical run would show screens as:

```
The VIGENERE cypher - an aid in finding the keyword from cyphertext and period.

      Enter text in UPPER case, letters only, no spaces,
                        (Just 'F' if data is on disc)

Enter cypher text:?
```

At this stage, ENTER selects the default example, and gives the screen:

```
The VIGENERE cypher - an aid in finding the keyword from cyphertext and period.


      Enter text in UPPER case, letters only, no spaces,
                        (Just 'F' if data is on disc)

Enter cypher text:?
PRGRFESJXHRBVWECPEFTCECEEXEBYWAOYTAMOCIANEDWELIAOERTEEQUUNITIZWHCIQSETOMHFNGRI
SDXZYWQKRFESSLBGRENTPNZVSSRTEZLIOVXIBRVDYIZYNNLOLEOGOA  (period 9)

Period (13 max)?
```

The period is entered. The maximum value of 13 may be exceeded if required. In this case, the period is 9, so enter 9 and the programme starts the tests. Just to show that something is going on, a series of full stops (periods !) are printed. These can be omitted, of course. On completion, the final screen is:

```
The VIGENERE cypher - an aid in finding the keyword from cyphertext and period.

      Enter text in UPPER case, letters only, no spaces,
                      (Just 'F' if data is on disc)

Enter cypher text:?
PRGRFESJXHRBVWECPEFTCECEEXEBYWAOYTAMOCIANEDWELIAOERTEEQUUNITIZWHCIQSETOMHFNGRI
SDXZYWQKRFESSLBGRENTPNZVSSRTEZLIOVXIBRVDYIZYNNLOLEOGOA  (period 9)

Period (13 max)? 9
 9 ........................ 8 ........................ 7 .................
......... 6 ........................ 5 ........................ 4 ........
.................. 3 ........................ 2 ........................ 1
 ........................

        HRINKAALE
        ZLOZARBWL
        DAUCEELXX
        UQVORNFMM
        AUBGFQPRK

(B)ack to MENU, or (R)un again? b
```

It can be seen that in this case, the keyword DRINKABLE is worth a try! Selected from 3rd, 1st, 1st, 1st, 1st, 1st, 2nd, 1st and 1st row.

There is a bonus with this programme listing. To modify it for other types of periodic substitution cyphers is easy, just change a few lines ! For example, a BEAUFORT. Change lines 240 and 250 to

```
240 FOR J=1 TO C:A=ASC(MID$(C$,J,1))-65:A=91+K-A
250 IF A>90 THEN A=A-26
```

For a PORTA, change line 230 as well, and add line 255:

```
230 FOR K=0 TO 12
240 FOR J=1 TO C:A=ASC(MID$(C$,J,1)):IF A<78 THEN D=1 ELSE D=-1
250 A=A(13+K)*D:IF A>90 THEN A=A-13
255 IF A<65 THEN A=A+13
```

And that's not all ! The variables `G$` and `B$` may be changed to accommodate other languages. The PORTA version was successful with a recent XENO in CM.

And of course all could be combined into one programme, with an array for the different language `G$(n)`'s and `B$(n)`'s, and a `ON - GOTO` for the type. However the difficulty of building in default examples, probably 30 or even 40 would be required, means that the listing would be over complicated, and unnecessary. Credence is given to any possible user !

Good luck, and good solving !

VIGKEY.BAS

```
100 REM 'VIGKEY'-An aid in finding VIGENERE keyword
110 DATA "PRGRFESJXHRBVWECPEFTCECEEXEBYWAOYTAMOCIANEDWELIAOERTEEQUUNITIZWHCIQ
         SETOMHFNGRISDXZYWQKRFESSLBGRENTPNZVSSRTEZLIOVXIBRVDYIZYNNLOLEOGOA"
120 DIM G(26),B(26),GG(5,13),KK(5,13)
    :TLE$="The VIGENERE cypher - an aid in finding the keyword from cyphertext and period."
    :CLS:PRINT TLE$:PRINT:PRINT
130 G$="AEHINORST":B$="JKQVWXZ"
140 PRINT "      Enter text in UPPER case, letters only, no spaces,"
150 PRINT TAB(25) "(Just 'F' if data is on disc)":PRINT
    :INPUT "Enter cypher text:";CT$:IF CT$="" THEN READ CT$
    :PRINT CT$;"  (period 9)":GOTO 190
160 PRINT:IF CT$="F" OR CT$="f" THEN PRINT:INPUT "Name of file holding data";N$
    :I=INSTR(N$,"."):IF I>0 THEN N$=LEFT$(N$,I-1)
170 IF LEN(CT$)>1 THEN 190
180 N$=N$+".CT$":PRINT:OPEN "I",#1,N$:INPUT #1,CT$:CLOSE #1
190 I=INSTR(CT$," "):IF I>0 THEN CT$=LEFT$(CT$,I-1)+MID$(CT$,I+1):GOTO 190
200 LC=LEN(CT$):PRINT:INPUT "Period (13 max)";P
210 FOR N=1 TO P:C$="":PRINT P+1-N;
220 FOR J=N TO LC STEP P:C$=C$+MID$(CT$,J,1):NEXT:C=LEN(C$)
230 FOR K=0 TO 25
240 FOR J=1 TO C:A=ASC(MID$(C$,J,1))-K
250 IF A<65 THEN A=A+26
260 C1$=CHR$(A)
270 IF INSTR(G$,C1$)>0 THEN G(K)=G(K)+1
280 IF INSTR(B$,C1$)>0 THEN B(K)=B(K)+1
290 NEXT:PRINT ".";
300 NEXT
310 FOR H=0 TO 4
320 GG(H,N)=G(0)-B(0):KK(H,N)=0:FOR Z=1 TO 26
    :IF GG(H,N)<(G(Z)-B(Z)) THEN GG(H,N)=G(Z)-B(Z):KK(H,N)=Z
330 NEXT
340 G(KK(H,N))=0:NEXT
350 FOR K=0 TO 25:G(K)=0:B(K)=0:NEXT
360 NEXT:PRINT:PRINT
370 FOR H=0 TO 4
380 FOR Z=1 TO P:PRINT CHR$(KK(H,Z)+65);:NEXT:PRINT
390 NEXT
400 PRINT:PRINT "(B)ack to MENU, or (R)un again";:INPUT Z$
410 IF Z$="B" OR Z$="b" THEN RUN "MENU.BAS"
420 RUN
```

THE POLLUX CIPHER

BOATTAIL

The Pollux cipher is based on the Morse Code. The plaintext is first encoded in Morse and then the dots, dashes and separators (x's) are enciphered by digits. For example, the phrase "home run" first becomes `....x---x--x.xx.-.x..-x-.` Then, using 1, 2, and 3 as separators, even digits as dots and odd digits as dashes, the phrase becomes "08461 75929 93013 85426 65198". The cipher is inefficient, expanding seven letters into twenty-five digits but solving it is a very intriguing puzzle.

To solve a Pollux, the first step must be finding the separators. Since there must be three (or four, in a variation) separators, the possibilities run from 012 to 789, 120 sets in all. There can never be three separators in a row. Since no morse string has more than four components, dots or dashes, separators can't be more than four spaces apart. By testing each possible set with these criteria, all but the one, correct set will usually be eliminated.

After filling in the separators, the dots and dashes are discovered by trial and error. A single character between two separators must be either a dot, for E, or a dash, for T. A three letter string with one, four and one morse characters is quite probably "THE" (`-x....x.`).

This is precisely how the Pollux program functions. The computer performs the elimination tests and stores the correct set in array `sepsol`. If there is only one correct set, it is automatically transferred to the operating set array `opsep`. If there is more than one possible set, the user selects the operating set. After the computer substitutes the separators in the morse array `text[morse]`, the cipher is displayed and the solver chooses dot and dash equivalents. As the morse strings are filled, the computer decodes them into letter equivalents in `text[plain]`.

The program uses the standard cipher library CIPHLIB with the addition of the new routine `morsedecode`. A test cipher is included in the program, for testing any future modifications. [**Ed**: Contact BOATTAIL for the most up-to-date version of the CIPHLIB library. **DMV**]

POLLUX.PAS

```
program POLLUX;  {determines separator digits, 3 or 4, by trial & error.
                  puts separators into morse text, solves plain by trial
                   & error}
                   {By BOATTAIL, July 9, 1991}
        USES Crt,Printer,Ciphlib; {uses morsedecode procedure}
        CONST
        Maxlen=600;
        test1='027345107549163784967245107451308194173571409485170891532492';
        test2='683782590467328691425803674950835261913645792314987609312564';
        test3='079813628546192857291674301985602635230714512983927645826947/';
        {PLAIN= Researchers have found a genetic abnormality that could...}
        {E-6 MJ91 ZYZZ}
        itc='Is This Correct?';
        TYPE
        line = (num,morse,plain);
        VAR
        test           :String;  {test cipher text}
        exitflag       :Boolean;
        selnum         :Integer; {menu selection}
        intext         :txtarr; {cipher input array}
        text           :array[line] of txtarr;
        clast          :Integer; {last element of cipher arrays}
        cnum           :intarr; {cipher digits in numbers}
        sepflag        :Boolean; {have separators been computed yet?}
        flag           :Boolean; {general purpose flag}
        solnum         :Byte; {how many sets of separators found?}
        numsep         :Integer; {number of separators, 3 or 4}
        sepsol         :array[0..10,0..3] of Byte; {sep solutions}
        opsep          :array[0..3] of Byte; {chosen separators}
        ddx            :array[0..9] of Char; {equivalents, dash, dot or x}
procedure ciphfill;
    var    x    :byte;
    begin
    for x:=0 to clast do begin
        text[num,x]:=intext[x];cnum[x]:=Ord(intext[x])-48;end;
    FillChar(ddx,Sizeof(ddx),' '); {clear dot-dash array to blanks}
    FillChar(text[plain],Sizeof(text[plain]),' '); {clear plaintext}
    end;
procedure sepfill(n:Byte); {put chosen separators into dot-dash array}
    var   x    :byte;
    begin
    for x:=0 to numsep-1 do begin opsep[x]:=sepsol[n,x];ddx[opsep[x]]:='x';end;
    end;
procedure morsefill; {fill morse text from dot-dash equivalents}
    var   x,y   :Byte;
    begin
    FillChar(text[morse],Sizeof(text[morse]),' '); {clear morse text}
    for x:=0 to 9 do
        for y:=0 to clast do text[morse,y]:=ddx[cnum[y]];
    end;
procedure displayline(ln:line); {display cipher, morse, or plain}
    const   wid=80;
```

```
    var  f,g,h,k,t   :Byte;  endflag  :Boolean;
    begin
    g:=0;h:=wid-1;endflag:=false;f:=Ord(ln);t:=0;
    repeat
       if h>clast then begin h:=clast;endflag:=true;end;
       GotoXY(1,1+f+(4*t));Textcolor(f+3);
       for k:=g to h do Write(text[ln,k]);
       Inc(t);Inc(g,wid);Inc(h,wid);
   until endflag=true;
   end;
procedure dispall; {display cipher, morse & plain}
    var  ln :line;
    begin
    Textmode(CO80);ClrScr;for ln:=num to plain do displayline(ln);
    end;
procedure choosesep; {which set of computed separators do you want to try?}
    var   w   :Integer;  flg :Boolean;
    begin
    if solnum=0 then begin
           message('Only One Possible Set',Cyan,9,20,false);any_key;end
        else begin
           digit_in('Which Separator Set? ',Blue,1,20,w);Write(w);
           yes_no(itc,White,1,24,flg);
           if flg=true then sepfill(w);
           end;
    end;
procedure compsep3; {deduce three separators by trying all combinations}
    var    b,i,j,k,n,sc,nsc :Byte;
    label  650,660,670;
    begin
    b:=0;
    for i:=0 to 7 do
        for j:= i+1 to 8 do
            for k :=j+1 to 9 do begin
              GotoXY(18,12);Write(i,' ',j,' ',k);sc:=0;nsc:=0;
              for n:=0 to clast do begin
                  if (cnum[n]=i) or (cnum[n]=j) or (cnum[n]=k) then goto 650;
                  if nsc=4 then goto 670 else begin Inc(nsc);sc:=0;goto 660;
                      end;
             650: if sc=2 then goto 670 else begin Inc(sc);nsc:=0;end;
             660: end; {of for n}
             sepsol[b,0]:=i;sepsol[b,1]:=j;sepsol[b,2]:=k;Inc(b);
        670: end; {of for k}
    solnum:=b-1; {number of separator solutions}
    sepflag:=true;
    end;
procedure compsep4; {deduce four separators by trial}
    var    b,h,i,j,k,n,sc,nsc,x   :Byte;
    label  750,760,770;
    begin
    b:=0;
    for h:=0 to 6 do
        for i:=h+1 to 7 do
```

```
            for j:=i+1 to 8 do
                for k:=j+1 to 9 do begin
                    for n:=0 to clast do begin
                       x:=cnum[n];
                       if (x=h) or (x=i) or (x=j) or (x=k) then goto 750;
                       if nsc=4 then goto 770 else begin
                            Inc(nsc);sc:=0;goto 760; end;
                  750: if sc=2 then goto 770 else begin Inc(sc);nsc:=0;end;
                  760: end; {of for n}
                  sepsol[b,0]:=h;sepsol[b,1]:=i;sepsol[b,2]:=j;sepsol[b,3]:=k;
                  Inc(b);
             770: end; {of for k}
    solnum:=b-1;sepflag:=true;
    end;
procedure displaysep; {display deduced separator set(s) on screen}
    var    f,g    :Byte;
    begin
    ClrScr;Textcolor(LightMagenta);GotoXY(1,6);Write('Separators:');
    for f:=0 to solnum do begin
        GotoXY(15,6+f);Textcolor(Cyan);Write(f,'  ');Textcolor(LightGreen);
        for g:=0 to numsep-1 do Write(sepsol[f,g],' ');end;
    end;
procedure decode; {change morse text to plain}
    var   x :Byte; strflag,blankflag :Boolean; xstr :Str5;
    begin
    FillChar(text[plain],Sizeof(text[plain]),' ');
    strflag:=false;blankflag:=false;xstr:='';
    for x:=0 to clast do
        case text[morse,x] of
        'x' : begin
              if (strflag=true) and (blankflag=false) then begin
                  morsedecode(xstr,text[plain,x-1]);
                  strflag:=false;
                  end;
              blankflag:=false;xstr:='';
              end;
    '.','-' : begin xstr:=xstr+text[morse,x];
                    strflag:=true;end;
        ' ' : begin strflag:=false;blankflag:=true;xstr:='';end;
        end;
    end;
procedure instruct; {display instructions while using substitution screen}
    begin
    GotoXY(1,22);Textcolor(White);Write('0123456789');Textcolor(LightCyan);
    GotoXY(1,24);Writeln('Use right and left arrows to move cursor');
    Write('Insert dot, dash, or blank below digit; Escape for Main Menu');
    Textcolor(Red);
    end;
procedure substitute; {interactive, trial & error solving on screen}
    var   x,indx   :byte;  a,b  :Char; outflag :Boolean;
    begin
    ClrScr;dispall;instruct;indx:=0;outflag:=false;
    repeat
```

```
        GotoXY(1,23);for x:=0 to 9 do Write(ddx[x]);GotoXY(indx+1,23);
        b:=Readkey;
        case b of
        ' ','.','-' : begin ddx[indx]:=b;morsefill;decode;
                            displayline(morse);displayline(plain);end;
        #0  : begin  a:=Readkey;
              case a of
                  #77 : indx:=(indx+11) mod 10; {left arrow}
                  #75 : indx:=(indx+9) mod 10; {right arrow}
                  end;
              end;
        #27  : outflag:=true; {escape key}
        end;
    until outflag=true;
    end;
procedure hardcopy; {printout cipher, morse, plain, dot-dash}
    const  dwon=#0#27#87#49#13; {NUL,Esc,W1,CR}
           dwoff=#27#87#48#13;  {Esc,W0,CR}
           wid=40; {40 chars. per printed line}
    var  f,g,h  :Byte; out :Boolean; ln :line;
    begin
    Write(Lst,dwon); {double width printing on, CR}
    g:=0;h:=wid;out:=false;
    repeat
        if h>clast then begin h:=clast;out:=true;end;
        for ln:=num to plain do begin
           for f:=g to h do Write(Lst,text[ln,f]);
           Write(Lst,#10#13); {LF,LF,CR}
           end;
       Inc(g,wid);Inc(h,wid);Write(Lst,#10); {blank line between rows}
    until out=true;
    Writeln(Lst,'0123456789');
    for f:=0 to 9 do Write(Lst,ddx[f]);
    Write(Lst,dwoff,#13#12); {double width printing off,CR,Form feed}
    end;
begin         {main body of program}
    test:=test1+test2+test3; {test cipher assembled}
    sepflag:=false;exitflag:=false;
    repeat
        Textmode(CO40); {40 column color}
        ClrScr;Textcolor(LightBlue);GotoXY(10,1);flag:=false;
        Writeln('POLLUX CRYPTANALYSIS'^J^J);
        Writeln('(1) Enter Cipher Digits'^J);
        Writeln('(2) Find Separators'^J);
        Writeln('(3) Display Cipher'^J);
        Writeln('(4) Change Separators'^J);
        Writeln('(5) Substitute in Cipher'^J);
        Writeln('(6) '^J);
        Writeln('(7) Hardcopy'^J);
        Writeln('(0) Exit to DOS'^J);
        digit_in('Select by Number: ',White,10,24,selnum);
        case selnum of
             1 : begin
```

```
                cipherin(test,Maxlen,clast,intext);
                yes_no(itc,White,1,24,flag);
                if flag=true then begin
                    squeeze(['0'..'9'],clast,intext);
                    ciphfill;
                    end;
                end;
            2 : if sepflag=false then begin
                    ClrSCr;
                    number_in('How Many Separators? ',Red,7,12,3,4,numsep);
                    if numsep=4 then compsep4 else compsep3;
                    sepfill(0);morsefill;displaysep;any_key;
                    end;
            3 : begin dispall;any_key;end;
            4 : begin displaysep;choosesep;morsefill;end;
            5 : substitute;
            6 : ;
            7 : hardcopy;
            8 : ;
            9 : ;
            0 : begin ClrScr;yes_no('Quit This Program?',Red,7,12,exitflag);
                end;
        end; {of case statement}
    until exitflag=true;
    Textmode(CO80);Textcolor(White); {80 column color}
end. {of program }
```

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 the 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) or uploaded to the ACA BBS. This makes it much easier for us to typeset.
7. Send material for publication to Dan Veeneman, PO Box 7, Burlington, IL, 60109–0007, USA.

MORSE DECODE LIBRARY PROCEDURE

BOATTAIL

This is a simple routine that changes the Morse Code into its equivalent letters. If string `mstr` is found in the array `morse`, its corresponding letter is taken from array `plain` and output as the character variable `pl`. The routine also recognizes the Morse digits. If `mstr` is not a valid Morse code, `pl` is returned as `Chr(0)`, or `NUL`. This procedure was very useful in my program `POLLUX.PAS` and I will use it in `FRACMORS.PAS`, my next cryptanalysis project.

MORSE.PAS

```
procedure morsedecode(mstr:str5; var pl:Char);
    const
    plain :array[0..35] of Char = ('e','t','a','o','n','i','r','s','h',
              'l','d','c','u','p','f','m','w','y','b','g','v','k','q','x',
              'j','z','0','1','2','3','4','5','6','7','8','9');
    morse :array[0..35] of str5 = ('.','-','.-','---','-.','..','.-.','...',
          '....','.-..','-..','-.-.','..-','.--.','..-.','--','.--','-.--',
          '-...','--.','...-','-.-','--.-','-..-','.---','--..','-----','.----',
          '..---','...--','....-','.....','-....','--...','---..','----.');
    var  x   :Byte;
    begin
    x:=0;pl:=#32;
    while (pl=#32) and (x<=35) do begin
        if mstr=morse[x] then pl:=plain[x];
        Inc(x);
        end;
    end;
```

WHAT THE OTHER GUY IS DOING

GRAPE JUICE (**Tom Cage**) has sent me a set of high density disks with his current pattern dictionary: 324,016 words at present, and growing ! If anyone would like more info, or a copy, contact him or the Editor.

Bruno Lienard sold his Atari ST and bought a Compaq Contura 3/25. He is currently working on a French pattern-word list. His first version has about 200,000 words, but the second will have closer to 400,000. These lists are available free to the Krewe, just send two 720K formatted disks (3.5 inch only) and a self-addressed mailer. He is also working on German and English lists, and has a program that helps to find pattern words. It accepts regular expressions and works under Windows 3. The C source code is included, but the comments and documentation are in French !

BOATTAIL (**Patrick Larkin**) is still using his Datatrain 386SX-16, now with 4Mb memory and a SoundBlaster board. Anyone doing any Cons. with music as the message ? His latest projects are still in Turbo PASCAL, improving his `ROUTE.BAS` program from 1986. It now presents the ciphertext on the screen in all possible encipherment blocks, allowing a solution by simple inspection. **BOATTAIL** is also occasionally on CompuServe.

Robert Matthews is looking for QuickBASIC programs to aid him in the solutions for Pollux, Morbit, Redefence, twin bifid, digrafid, two-square, four-square, Grandpre, Null, portax, slidefair, tridigital and trisquare. If anyone has any BASIC versions that could help, let him or the Editor know ! By the time you read this, **Robert** should be in Acapulco with his wife and his laptop, enjoying the sunshine !

Decis Thierry sent in a program for the Atari ST that breaks substitution ciphers. He states: "All you have to do is type in the ciphertext and wait one or two hours." The ciphertext must have at least 100 letters, and it currently only works for French crypts. He is currently searching other texts to produce an English-language version.

ABOUT THIS ISSUE

This issue was produced using Donald Knuth's amazing TEX typesetting program, with help from various style files, especially **multicol.sty** and **fullpage.sty**. After comments from readers, I've switched from a 12pt to an 11pt font. This should get more information onto a single page, and reduce the impression of "wasted space."

The **.tex** file that produced this issue is about 67,000 bytes, and is edited with a simple ASCII text editor (MKS Toolkit's **vi**). It was printed on a Hewlett-Packard LaserJet IIIP. The pcTEX version I have runs fine on my IBM–PC 8086 clone in 640K of memory !

BEGINNER'S GUIDE TO THE ENVIRONMENT

Jac Goudsmit

An interesting part of the MS-DOS operating system is the "environment". This is an area in memory where you can store a number of system settings (used by various programs) as strings (text) containing an identifier, followed by an equals sign ("=") and the value to which the identifier has been assigned. The identifier names are stored as upper case strings, but the value can be in upper case, lower case, or mixed. Almost any string value is possible.

In everyday use, the most important variables are:

PROMPT This is the text that the command interpreter puts on the screen to signal you that you can enter a DOS command (like DIR). If you don't set your prompt with the PROMPT command, it will default to ng. $n is interpreted as the current disk letter; $g is a greater-than sign (">").

Most people however put something like PROMPT pg in their AUTOEXEC.BAT, so they can also see the current directory ($p). But you can also set your prompt to something else (in the early days you could scare people by setting the prompt to ENTER YOUR PASSWORD).

PATH This is a list of directories that DOS searches through when you type a command. The default is nothing, which means to only search the current directory.

If you type a command like EDIT (which is not an internal command like DIR), DOS will first search the current directory for a file named EDIT.COM. If that doesn't exist, it tries EDIT.EXE and if that doesn't exist, it tries EDIT.BAT. If none of them is found in the current directory, it starts looking for them in the directories listed in the PATH environment.

You should set your PATH variable with the PATH command. If you have a hard disk, there is probably a PATH command in your AUTOEXEC.BAT file. An example is

```
PATH C:\;C:\DOS;C:\UTILS
```

In this example, DOS will always search for external commands in the sequence <current dir>, C:\, C:\DOS, C:\UTILS, before it gives up with the "Bad command or file name" message.

COMSPEC This variable should ALWAYS be present in the environment. It is put there automatically when you start the computer. The COMSPEC variable is used by the command interpreter (COMMAND.COM) to indicate the position from where it was loaded. This is usually something like A:\COMMAND.COM or C:\COMMAND.COM.

The reason why it needs to know this is that COMMAND.COM is loaded into memory in two parts called the resident portion and the transient portion. The resident portion stays in memory as long as DOS runs. The transient portion can be overwritten by programs that need the memory. When such a program finishes, the resident portion will detect that the transient portion is missing, and will attempt to re-load it. For that it needs to know where to look and that's where COMSPEC comes in. If the transient portion can't find the file that's in COMPSPEC or if something is wrong with it, it will print the message "Cannot load COMMAND.COM" on the screen. Depending on your particular setup, you will asked to insert a diskette with the file, or you will see "System halted" (after which you need to press Ctrl-Alt-Del).

SET You can see what's in your environment any time by typing the command SET. This gives a list of environment variables and their values.

You can also change an environment variable by typing SET *variablename* = *value*.

This is also an alternative way to change the PATH and PROMPT settings,

e.g. `SET PROMPT=$p$g` is the same as `PROMPT $p$g`. To delete a variable from the environment, assign the *variablename* to nothing, i.e. type `SET` *variablename* `=`

There is no way to use the value of an environment variable in a regular DOS command. But in batch files, you can use an environment variable by putting the identifier between percent signs ("`%`"). An example called `ADDPATH.BAT` follows below.

```
ECHO OFF
REM ADDPATH adds a directory to your path
REM The original value for the path variable is stored as ORGPATH
SET ORGPATH=%PATH%
PATH %PATH%;%1
```

Notice that unlike batch file parameters (`%1`, `%2`, `%3`, etc), an environment variable must be *between* percent signs, not just following one. Referring to environment variables in batch files with percents was undocumented until (I think) version 3.3 of DOS. Futhermore, a bug in DOS 3.00 prevents this use of the environment (it crashes the system if I remember correctly).

A lot of programs nowadays make use of the environment to retrieve their default options or directory. DOS 5.0 introduced an extremely (though still not enough) extended `DIR` command that can generate sorted directory lists, or lists of hidden files only. For example, if you prefer your directories to always be listed in alphabetical order, you can do a `DIR /ON` every time you want a directory list, or you can use the `DIRCMD` environment variable: `SET DIRCMD=/ON`

The environment is of limited space. Different DOS versions use different amounts of default environment space. But you can always change the environment space to a bigger setting by putting a `SHELL` directive in your `CONFIG.SYS`. This directive is used to specify a different command interpreter. The default is `\COMMAND.COM`. If you want to change the environment space setting, insert a line with something like

```
SHELL=C:\COMMAND.COM /P /E:512
```

in your `CONFIG.SYS` for DOS versions earlier than 5.0, or

```
SHELL=C:\COMMAND.COM C:\ /P /E:512
```

for DOS 5.0 (or above).

The `/P` is for Permanent. This is to make sure you don't accidentally `EXIT` from the first-loaded command processor (this would crash the system). The value after `/E:` indicates the number of bytes that are allocated for the environment for DOS versions 3.2 (again if my memory serves me right) and up, or the number of paragraphs for lower versions (a paragraph is 16 bytes). Again, the `/E:` parameter was not document in the early DOS versions. A good value for normal use is 512 bytes. If you are on a network, add another 256 bytes or so. If you are a programmer, you will probably need even more. If you ever see the message "Out of environment space", this means the value is too low. Try to keep it as low as possible without getting this message.

One last remark: the `COMMAND.COM` that is loaded first contains the "master environment". If you "shell to DOS" from some program, you only have access to a copy of that master environment. As soon as you exit back to the program with the `EXIT` command from DOS, any new settings would be lost. For example, if you would start WordPerfect and then type `Ctrl-F1,1` to shell to DOS, you can change the `PATH` and all other environment settings, but the old settings will be restored as soon as you type `EXIT` to return to WP.

CONS WITH CRYPTO

David Hamer/DAEDALUS

Prompted by a request from LANAKI, I recently set out to create a few Aristocrat cons. Being fundamentally lazy and wanting the computer to do most of the work I began by looking through my diskettes for software that might do the job. I settled on BITSIFTER's program, CRYPTO.

For some considerable time this was my program of choice for help with the solution of Aristocrats (and to some extent Pats). CRYPTO can usually solve — almost automatically — the first 17 to 20 Aristocrats in a given issue of the Cm and with a little intuitive assistance usually results in a clean sweep (or at least a 23 or 24). More recently, having acquired a math co-processor I have forsaken CRYPTO for SY S. ABEND's program CRYPT on my desktop machine but, when on the road with my laptop — no co-processor — I use CRYPTO.

I will not attempt to describe the normal use of CRYPTO here - for details refer to CS2:p6. A procedure for the construction of Aristocrats and Patristocrats follows:

First load a copy of the file CRYPTO.EXE into a subdirectory or onto a diskette. This is the only CRYPTO file you'll need. The absence of the huge word list files will not prevent the core program from running and will in fact speed up the somewhat non-standard use to which we are about to put it. Omit, too, all the other files which are normally included with CRYPTO. Include in the subdirectory (or diskette) the PC Magazine file PRN2FILE.COM. If you don't have this it can be found on Compuserve's PC-MAGNET or on many local bulletin boards. [**Ed**: It can also be found on the ACA BBS, and is included in the Issue Disk. **DMV**]

Next write a batch file (GO.BAT) that reads something like:

```
echo off
prn2file cons.asc
crypto
```

— and you're ready to go!

Type GO and the TSR program PRN2FILE will load followed by CRYPTO. The result will be a blank screen ! Now type in a plaintext alphabet in the upper left hand corner, thus:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
..........................
```

Using the Home and cursor keys put the cursor on the dotted line and enter the ciphertext alphabet and keyword (example = cipher):

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
vwxyzcipherabdfgjklmnoqstu
```

(Example is for a K2 cipher. For a K1, enter the keyword with the plaintext alphabet: for K3/K4, with both alphabets).

Press Shift-F5 and the Plaintext/Ciphertext alphabets will be displayed in the lower section of the screen. (Figure #1).

Press the Home key and the cursor will return to the upper left corner of the screen. With the space bar, clear the plaintext alphabet - the ciphertext alphabet will disappear simultaneously but the relationship between the alphabets is retained by CRYPTO !

Now type in the desired plaintext - in either Aristocrat or Patristocrat form - and the ciphertext appears below it. Plaintext is in upper case, ciphertext in lower. This is built into CRYPTO and cannot be changed. Errors in typing may be corrected by deleting and inserting but CRYPTO behaves differently in this respect from a regular text editor — a little trial and error will be necessary ! After entering the text press F4 and the lower display will change — about the only thing of interest to our present purpose is the letter count. (Figure #2). Press Shift-F5 to get a display

of the two alphabets and the location of the keyword for K1 and K2. (Figure #3). For K3 and K4 see my later comment.

Press `Shift-Prtsc` and `PRN2FILE` does its stuff and copies the screen to the file `CONS.ASC` which may be edited, later, with any text editor/word processor. As further cons are created they are added to `CONS.ASC` so you may do any number you care to before stopping to edit the text file.

The display on the lower part of the screen clearly shows the relationship between the alphabets and the keywords for K1 and K2 ciphers. For K3 and K4 this relationship gets a bit confusing and it is difficult to place each keyword. Accordingly, I type this information as a separate block when editing the text. If anyone knows a way of getting `CRYPTO` to display the K3 and K4 keywords in their proper relation to the alphabets I'd like to hear about it, as I've never been able to figure it out !

So now you have no excuse for not sending in cons to the Aristo and Pat editors. Go to it !

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
vwxyzcipherabdfgjklmnoqstu
```

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
v w x y z c i p h e r a b d f g j k l m n o q s t u
L M F N J O P I G Q R S T U V H W K X Y Z A B C D E

      0
```

Figure #1

```
WHEN A MAN IS TIRED OF *LONDON, HE IS TIRED OF LIFE.
qpzd v bvd hl mhkzy fc *afdyfd, pz hl mhkzy fc ahcz.



 12 9 8 8 7 7 6 6 5   4 3 3 3 2 2 2 2 2   2 2 1.5.2.2.1.0  lowf=2
( E T A O N I S R H)( L D C U P F M W Y)( B G V K Q X J Z) ltrs=38
( E I N O D F A H L)( R S T M W B C G J)( K P Q U V X Y Z)
  5 5 4 4 3 3 2 2 2   2 2 2 1 1 0 0 0 0   0 0 0 0 0 0 0 0

      0                                 570        >0
```

Figure #2

```
WHEN A MAN IS TIRED OF *LONDON, HE IS TIRED OF LIFE.
qpzd v bvd hl mhkzy fc *afdyfd, pz hl mhkzy fc ahcz.



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
v w x y z c i p h e r a b d f g j k l m n o q s t u
L M F N J O P I G Q R S T U V H W K X Y Z A B C D E

      0                                 570        >0
```

Figure #3

PUBLIC KEY DOCUMENT AVAILABLE

A good introductory document on cryptography is a 162 page publication put out by the US National Institute of Standards and Technology called Public-Key Cryptography (special publication 800–2) by James Nechvatal, April 1991. The text is not an in-depth examination of these topics but will serve as a good introduction. This document should be just right for the beginner who is looking for background information on RSA and associated subjects. [**Ed**: This document is available in electronic form on the ACA BBS, in directory `/public/crypto` as file `800-2.ZIP`. **DMV**]

Contents:

BREAKING A CRYPTOSYSTEM

Paul C. Leyland (pcl@ox.ac.uk)

[**Ed**: The following article was recently posted to the Usenet newsgroup **sci.crypt**. At the risk of boring our **sci.crypt** readers, it is reproduced here as an interesting example of cryptanalysis. **DMV**].

Last September, Gerben Wierda posted the following:

> I am reading "The Code Breakers". Many people talk about cyphers in a theoretical sense, but the test is in the cryptanalysis. So here is the cyphertext:
>
> AOBUI EBDEL ATNGI AGVWL SSNDD DBALL LTERS SAMST TDSSA CIANE EENBB XPRSD
> ADABT ATETE HUEHX IPARA NCCSS MASTO TNOOP LSNAT ATEBE BESNG OAGIV WLSAS
> SSRAN ADMMA ENOEO LESST HASSW LLABA FLLUY EYETE AETAN HNCCR DABLT YYOAO
> NNCNN EAEDU EIAID ITHHA DBLNG EGFFA THHLS TATUU ALTRR FAFVE AEORO EIUAY
> YWRDD ASSBE UNDOA RYYAE
>
> And my question is this: Apart from talking theory here. Are there people who can solve this very simple system?
>
> If you can solve this you will know why it cannot be used (hint, hint)

I solved it but no-one else has admitted to doing so, according to Gerben. Several others were interested and discussed it with me by email.

In the hope that this might be of interest to psychologists and cryptanalysts, here's how I did it. I hope that I'm not spoiling people's fun, but you have had nearly 5 months to work on it.

First, the hint told me that the system is simple and effectively useless, so it is likely to be a waste of time investigating Enigmas etc.

I counted letter and digram statistics. Very similar to English, with A and E swapped in frequency. Too many repeated letters, especially SS. First hypothesis: may be it's a transposition. Not English, though. Email to the author yielded two hints: it is *not* a standard system (so simple transpositions are out) and the plaintext is English.

Next step was to look for repeats in the ciphertext. There are only two of length greater than 3, namely VWLS. The similarity of this to the word "vowels" was instantly noticed, with vowels omitted (self-referential) but this was thought to be a co-incidence. Second mistake. (The first was in exploring transposition ciphers).

I then re-formatted the ciphertext as one line (here split for ease of viewing)

```
AOBUIEBDELATNGIAGVWLSSNDDDBALLLTERSSAMSTTDSSACIANEEENBBXPRSDADABTATETEHU
EHXIPARANCCSSMASTOTNOOPLSNATATEBEBESNGOAGIVWLSASSSRANADMMAENOEOLESSTHASS
WLLABAFLLUYEYETEAETANHNCCRDABLTYYOAONNCNNEAEDUEIAIDITHHADBLNGEGFFATHHLST
ATUUALTRRFAFVEAEOROEIUAYYWRDDASSBEUNDOARYYAE
```

I had a look at it and fancied that could see suggestions of English words. For instance, `VWLS` already mentioned; `LTERS` might be "letters"; `NOEOLESS` might be "nonetheless"; `NCCRABLTY` might be "incredibility" or "readability" and so on.

If this was significant, then the vowels are seriously misplaced. Hypothesis: the vowels have been removed from the plaintext and reinserted (pseudo-)randomly. Test this by looking at the vowel-less ciphertext.

```
BBDLTNGGVWLSSNDDDBLLLTRSSMSTTDSSCNNBBXPRSDDBTTTHHXPRNCCSSMSTTNPLSNTTBBSN
GGVWLSSSSRNDMMNLSSTHSSWLLBFLLYYTTNHNCCRDBLTYYNNCNNDDTHHDBLNGGFFTHHLSTTLT
RRFFVRYYWRDDSSBNDRYY
```

Hmm. Now, perhaps `DLTNG` is "deleting"; `DBL` is "double"; `XPRSD` is "expressed"; `BNDRYY` is "boundary". Note that each of these, and the previous cases, *all* end in a doubled letter. Insert a break after each double letter to get:

```
BB DLTNGG VWLSS NDD DBLL LTRSS MSTT DSS CNN BB XPRSDD BTT THH XPRNCC
SS MSTT NPLSNTT BB SNGG VWLSS SS RNDMM NLSS THSS WLL BFLL YY TT
NHNCC RDBLTYY NN CNN DD THH DBLNGG FF THH LSTT LTRR FF VRYY WRDD SS BNDRYY
```

It's now pretty obvious, to me at least, that the plaintext is

```
"? deleting vowels and double letters most ?? can ? expressed but the
experience ? most unpleasant.  ? ??? vowels ? random nulls this will
??? ? ? enhance readability ? can ? the doubling ? the last letter ?
every word ? boundary."
```

The `?` indicate a known consonant, with unknown vowels to be filled in. From context, we get:

```
"By deleting vowels and double letters most ideas can be expressed but
the experience is most unpleasant.  By using vowels as random nulls,
this will baffle you.  To enhance readability, one can do the doubling
of the last letter of every word's boundary."
```

This is the solution I sent off. It had one mistake: the penultimate "the" should have read "a". As consonentless words disappear, they must be replaced with some other flag; "th" was chosen, a bad choice in my opinion, but it doesn't spoil the intelligibility of the message.

Some of the reasons why this is a poor ciphersystem:

It is easy to break. Took me about three hours work with a ciphertext-only atack and only a few subtle hints. I used only the very simplest of tools to do it and could easily have used pencil and paper only.

There is no secret information (key) other than the method.

It is not unambigously reversible. For instance, "baffle" in the solution above could be replaced by "befool". Although this particular example is of an unusual word with essentially the same meaning as the true solution, other examples needn't be so benign. (Consider THEIR OTHER THREE).

To me, a great part of the pleasure in breaking this system was observing the blind-alleys and contorted reasoning of the other would-be solvers. Further, those with a greater knowledge of cryptanalysis than I found it difficult

to shake of pre-conceptions of how to solve cryptograms based on prior experience.

Finally, would anyone with a knowledge of Hebrew, Arabic and other languages commonly written without vowels care to comment?

ACA BBS UPDATE

The ACA bulletin board system, Apres, now has a new 120 megabyte hard drive to replace the two 40 meg drives. One drive failed just after the first of the year, preventing callers from downloading files. This problem has been fixed, and the system is now available for both electronic mail (local and to the Internet via UUCP) and files, 24 hours a day at +1 708 639 8853.

There are two main directories for cryptographic programs: `/public/aca`, containing ACA-related programs and files, including all issue disks; and `/public/crypto`, containing general cryptographic programs, files, and documents. Here is a sample directory listing of some of the files available in the `/public/aca` directory:

```
asolver.zip     48384   04-May-91   IBM Aristocrat solver
casupprt.txt    24388   01-Oct-90   Cryptanalysis program by Walt Howe
csindex.dat      9445   11-Jan-91   Computer Supplement Index 1 - 13
enigmas.lzh     32542   27-Apr-91   Amiga Graphic Enigma Simulator
fatcat.zip     241248   31-Mar-91   FATCAT routines from CS Issue #14
index212.zip     8077   01-Mar-91   Index to Issue disks 1 to 12
issue1.zip      19176   23-Feb-90   Computer Supplement Issue Disk One
issue2.zip     108051   23-Feb-90   Computer Supplement Issue Disk Two
issue3.zip     144218   23-Feb-90   Computer Supplement Issue Disk Three
    .           .         .                      .
    .           .         .                      .
issue14.zip    511818   31-Mar-91   Computer Supplement Issue Disk Fourteen
issue15.zip    196027   17-Aug-92   Computer Supplement Issue Disk Fifteen
issue16.zip     14071   17-Aug-92   Computer Supplement Issue Disk Sixteen
krewe.zip       24024   03-Feb-91   ABACUS' cipher solving aid program.
lauer.pak       40200   07-Jan-91   Lauer's programs for IBM
nonpat.zip     189125   30-Dec-91   Non-pattern single column
rails11.zip      3331   12-Feb-91   Update on Fleetfoots program for IBM
surrmmx.zip     76760   31-Mar-91   Minimax words 9-16 letters ex. Dan Surr
useful.zip      73239   25-Feb-90   Useful routines to aid in solving
varidict.zip   414724   30-Dec-91   50000-plus words in 26 categories
wl11_34.zip    900742   30-Dec-91   Word lists, lengths 11 to 34
wl1_10.zip     906956   30-Dec-91   Word lists, lengths 1 to 10
```
