****

COMPUTER SUPPLEMENT #22

****

In this issue:

SOLVING CRYPTORYTHMS — PARROT presents a brute force method implemented in the C language.

LOCATING CRIBS IN PATS — G4EGG has a BASIC program to interactively work out Patristocrats.

BREAKTHROUGH '32 — DAEDALUS reports on a solution to a minor problem with C.A. Deavors `ENIGMA.BAS` program.

AN INTRODUCTION TO PUBLIC KEY CRYPTOGRAPHY — André Kesteloot walks us through the basics.

TEA AND SYMPATHY — DAEDALUS reviews The Electronic Alveary, a computerized aid for wordsmiths.

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 #21 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, $45. Contains most ACA-related material. Available from the Editor.

TO OBTAIN THESE MATERIALS

`Write to:` `Or via Electronic Mail:`

`Dan Veeneman` `dan@decode.com`

`PO Box 2442`

`Columbia, Maryland 21045-1442, USA.`

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.

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.

From the Keyboard

Dan Veeneman

First, an apology for the extremely long delay since the last issue, and thanks for your patience. The contents of the *Computer Supplement* are primarily from contributions by the Krewe, and things have been sparse for a while, which is curious since encryption has become very topical lately.

The explosion of the Internet and the desire for secure electronic commerce is driving the development of protected Web connections and cryptographic payment schemes. Web browsers developed by Netscape and Microsoft, to name two, come with such connection capabilities built-in, and companies like CyberCash are working to ensure payments transferred over the Internet are secure and reliable. The Internet also allows the dissemination of information regarding product weaknesses as programmers and hackers report the results of their poking and probing. Currently Microsoft Windows NT and Cisco routers are getting a closer examination than their creators would like.

The Internet has also made possible the brute force solution to difficult cryptographic challenges by distributing the processing load across thousands of volunteered machines. The keyspace to be searched is broken into small chunks and dispersed to otherwise idle machines, allowing selected ciphertext to be attacked en-masse across the planet. For instance, in June a message encrypted using the Data Encryption Standard (DES) was cracked by the coordinated efforts of tens of thousands of machines, each trying a portion of the 72 quadrillion possible keys. Other cryptosystems using relatively short keys are also vulnerable to this type of attack.

Here in the United States the development and use of cryptography has been brought into question by the Director of the Federal Bureau of Investigation. Historically, export of strong codes has been regulated by the federal government, but domestic use was left unhindered. At a Senate subcommittee hearing in September FBI Director Louis Freeh called for limits on the domestic use of cryptography and wanted a legal requirement for every cryptographic package developed, sold, and used within the United States to contain a "backdoor" that would allow law enforcement agents to immediately decrypt any message or file. More frightening was the warm reception this Orwellian plan received from some Senators, and the *Secure Public Networks Act* authored by John McCain (R-AZ) and Bob Kerrey (D-NE) that would implement such a plan.

On a more positive note, Eric Blossom is now marketing to the public a secure telephone that uses triple-DES encryption and Diffie-Hellman key exchange. Researchers at Silicon Graphics, Inc. are using Lava Lamps to generate cryptographically secure random numbers.

This issue contains a variety of articles gathered over the past year or so. By all means, if there's something you're working on and you'd like to let others know about it, write it up and send it in ! It doesn't have to be long and involved, sometimes a short note for inclusion in "What the Other Guy is Doing" is enough to motivate others to action or garner additional help for your project.

Again, I apologize for the long delay between issues. Thank you for your patience, and please send in those articles!

Good Solving,

Solving Cryptorythms — A Brute Force Approach

PARROT

Trying every number one by one is a perfect job for an idiot, yes a computer. Consider *CM JF95* C-SP-1 by **HANO**. The problem reads:

> Square roots. EAGRFMRDA gives root DEGUA. DEGUA gives root AUS. AUS gives root (GGID). There are only 6 different digits we need to keep track if we use: (AUS)*(AUS)=DEGUA.

I use Turbo C version 2.0, but I think this program should work on most versions of C with only minor modifications. It is important to use `(long int)` because `(int)` is only defined up to `32,768`.

The crux of this program is that you assign each letter of the cryptarythm a variable, and the computer tries each number one a time and compares it according a given condition. That condition is the if statement.

```
/* cry1.c */
#include <stdio.h>
#include <conio.h>
main()
{
long int d,e,g,u,a,s;
clrscr();
         for (d=1;d<10;d++){
           for (e=1;e<10;e++){
             for (g=1;g<10;g++){
               for (u =0;u<10;u++){
                 for (s=0;s<10;s++){
                   for (a=1;a<10;a++){
if(a*100+u*10+s)*(a*100+u*10+s)= = (d*10000+e*1000+g*100+u*10+a))
{
printf("d=%ld e=%ld g=%ld u=%ld s=%ld a=%ld\n",d,e,g,u,s,a);
} } } } } } } }
```

The output was: `d=1 e=0 g=2 u=0 s=1 a=1 d=2 e=8 g=5 u=6 s=9 a=1`

The first answer comes from the multiplication by zeros. The second output translates to $169x169 = 28561$. That makes `EAGRFMRDA` $= 815,730,721$ according to my calculator, and the rest is easy.

This problem was nice because I could isolate six digits with a mathematical expression to give me a result. Sure I could run ten `for()` loops on any cryptorythm and wait a few hours (many hours) for a result. But my XT has better things to do and there are ways to speed things up.

But when you look at *CM JF95* C-10 by **ZIP**, there is no way to avoid looking at all ten digits. The problem is: Subtractions `BROOK-TROUT=GAAEA`, `HOOK-TORE=UAGO`. Here is the listing of the C program that breaks it.

```
/* cry2.c */
#include <stdio.h>
#include <conio.h>
main()
{
long int h,o,k,t,r,e,u,a,g,b;
int c1,c2,c3,c4;
clrscr();

/* part 1 */
      for (h=0;h<10;h++){
        for (k=0;k<10;k++){
          for (t=0;t<10;t++){
            for (b=0;b<10;b++){
              for (r=0;r<10;r++){
                for (e=0;e<10;e++){
/* part 2 */
c1=0;c2=0;c3=0;
o=k-e;    if(o<0) {o=o+10; c1=1;}
g=o-r-c1; if(g<0) {g=g+10; c2=1;}
a=o-o-c2; if(a<0) {a=a+10; c3=1;}
u=h-t-c3; if(u<0)  u=a+10;
/* part 3 */
if((b*10000+r*1000+o*100+o*10+k)-(t*10000+r*1000+o*100+u*10+t)
           = =(g*10000+a*1000+a*100+e*10+a)){
if((h*1000+o*100+o*10+k)-(t*1000+o*100+r*10+e)
           = =(u*1000+a*100+g*10+o)){
/* part 4 */
if( h!=o && o!=k && k!=t && t!=r && r!=e && e!=u && u!=a && a!=g && g!=b
         && h!=k && o!=t && k!=r && t!=e && r!=u && e!=a && u!=g && a!=b
                 && h!=t && o!=r && k!=e && t!=u && r!=a && e!=g && u!=b
                         && h!=r && o!=e && k!=u && t!=a && r!=g && e!=b
                                 && h!=e && o!=u && k!=a && t!=g && r!=b
                                         && h!=u && o!=a && k!=g && t!=b
                                                 && h!=a && o!=g && k!=b
                                                         && h!=g && o!=b
                                                                 && h!=b){
/* part 5 */
printf("%ld %ld %ld %ld %ld %ld %ld %ld %ld %ld\n",h,o,k,t,r,e,u,a,g,b);

 } } } } } } } } } }
```

The output was: `5 2 0 1 7 8 3 9 4 6`, which corresponds to the variables at the end of the `printf` statement. I could have put identifiers on the output, like the previous program. This program has six `for()` loops and four expressions I was able to use to speed things up.

```
 HOOK
-TORE
UAGO
```

part 2: `o=k-e;  if(o<0) o=o+10; c1=1;`

This line comes from the right side of the above equation. `c1` becomes one if it is necessary to "borrow" from the tens column. Thus if `E` was greater than `K`, `c1` will subtract a 1 from the next expression: `g = o - r - c1; if(g<0) g=g+10; c2=1;` and `c2` will act the same as `c1`.

Using expressions greatly speed up the program, because the more `for()` loops in part 1, the slower it will take to get an answer. My XT will takes 12 minutes to run six `for()` loops. Seven for loops should take 10 times longer or 120 minutes, or two hours. Eight `for()` loops... 20 hours. And nine would take days. Yes, a faster way is called for.

Part 4 simply insures that no result will print out if one variable is equal to another. I usually run the program without this part, and include it only if there is a need to limit the amount of output. Often, if only ten lines or so are printed out, I can weed out the wrongs ones by sight.

Part 3 is the problem as given in the *Cryptogram*. The result will only print out if the `if()` expression is true. The two sides of the equations are separated by the logical `==` double equal sign. In C, you need a double equal sign in a logical expression, and a single equal sign if you are assigning a value to a variable.

The last example is from *CM JF95* C-Sp-2 by **ACHAMP**. It is unidecimal addition. `BERLIN + GERMANY = TRAILER`.

```
/* cry3.c */
#include <stdio.h>
#include <conio.h>
main()
{
long int b=0,e=0,r=0,l=0,i=0,n=0,g=0,m=0,a=0,y=0,t=0;
int c1;
clrscr();
for (l=0;l<11;l++){
 for (i=0;i<11;i++){
  for (n=0;n<11;n++){
   for (m=0;m<11;m++){
    for (a=0;a<11;a++){
     for (y=0;y<11;y++){
      for (g=1;g<11;g++){
       for (b=1;b<11;b++){
c1=0;
r=n+y;    if(r>=11) {r=r-11;c1=1;}
e=n+i+c1; if(e>=11) e=e-11;
t=g+1;    if(t>=11) t=t-11;

if(b!=e && e!=r && r!=l && l!=i && i!=n && n!=g && g!=m && m!=a && a!=y && y!=t
        && b!=r && e!=l && r!=i && l!=n && i!=g && n!=m && g!=a && m!=y && a!=t
                && b!=l && e!=i && r!=n && l!=g && i!=m && n!=a && g!=y && m!=t
                        && b!=i && e!=n && r!=g && l!=m && i!=a && n!=y && g!=t
                                && b!=n && e!=g && r!=m && l!=a && i!=y && n!=t
                                        && b!=g && e!=m && r!=a && l!=y && i!=t
                                                && b!=m && e!=a && r!=y && l!=t
```

```
                                                    && b!=a && e!=y && r!=t
                                                             && b!=y && e!=t
                                                                      && b!=t){
/* part 5 */
          if((b*161051+e*14641+r*1331+l*121+i*11+n)+
   (g*1771561+e*161051+r*14641+m*1331+a*121+n*11+y)
 ==(t*1771561+r*161051+a*14641+i*1331+l*121+e*11+r)){

printf("%ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld \n",b,e,r,l,i,n,g,m,a,y,t);

} } } } } } } } } } }
```

The `if` statement labeled part 5 is why I'm showing this program. Since it is an unidecimal program, I had to convert the numbers over to base eleven. Thus the tens positions was multiplied by `11`, the 100's by `11 x 11` or `121` and so on. It works and the output was `5 8 3 6 7 1 9 4 0 2 10`.

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. For postal items, send material to

   `Dan Veeneman`
   `PO Box 2442`
   `Columbia, Maryland, 21045-1442`
   `USA.`

Cribloc, a programme to help locate cribs in PATs

G4EGG

Solving PATs for me, after a frequency count and probable vowel check, means locating the crib or probable word, and then building on this known information. Both extending partial words and adding to the substitute alphabets are used. Having once missed the correct fit, I decided to let the computer find them for me. That's how this programme started, and it worked quite well. Then, like Topsy, it just growed!

Yes, I know. That's not the way to write software. But it works this time. It was a simple step to evaluate or score each fit, and so consider them in best fit order. The score was calculated on the basis of letter frequency expected from crib, and that found in a ciphertext fragment (square of difference). This worked better.

After that, displaying the substitute letters under the ciphertext to see other full/part words and vowel grouping helped rule out unlikely positions. An improvement indeed.

Naturally, being able to add more letters to this information followed, resulting in the current version of the programme. There is a built in example, so let's work through it before describing the routines in the listing.

Running the programme gives the screen:

```
CRIBLOC, a programme to help locate a crib in a PAT

Enter cypher text, in UPPER CASE, (ENTER for example, 'F' if data on disc) ?
```

So, just press ENTER for the example. For type, enter 1. (or K1, the example is a K1) ENTER again finds the crib, which is `ENERGY`.

A row of periods shows that something is going on, and finally ciphertext and entered data are shown as:

```
                CRIBLOC, a programme to help locate a crib in a PAT

  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
  6  5  7  9  2  4  5 12  4  2  4  0  3  1  0  6  0  4  0  2  0  3  9  2  8  1
  Total characters 99 Type K1    Crib ENERGY          There are 3 fit(s).

Ciphertext is:
RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZPCYJ
  en   e               r e   e   y re      n   n        renergy n  e       ng

AKHVADVWMHYHRNBDAHCYP
  e      ene  r  e n

  Positions with best fit of ENERGY:
             Order         Value        Position        Sequence
               1             7            59           HYHBJT

     ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
     R     E G         Y    N


        (Q)uit this crib, (N)ext pos. (R)e try, or (*A)dd a letter
```

There is a frequency count, reminders of type and crib, and the number of fits found. In this case there are three positions where the word pattern for ENERGY will go. The best one, ie. that with lowest score, is shown above. It has a very low score, 7, indicating a good choice, so lets stay with it. (Order is from best as 1, up to number found or limit of 18. Value is score from square of differences.)

The first line of plaintext ends with `...ng`
Surely the first guess must be `C` ciphertext, is `i` plaintext?
To enter this, at the prompt type A (or just ENTER), and then `C` for ciphertext followed by `I` for plaintext (either case). The substitution alphabets for the ciphertext and plaintext, respectively, are then correctly shown, as:

```
CT$  ABCDEFGHIJKLMNOPQRSTUVWXYZ
PT$   ri    e g         y    n
```

from which it may be deduced that the `CT$` = `...HIJ...` and `PT$` = `...e.g...`, needs `I` ciphertext for `f` plaintext? Again at the prompt, ENTER, `I`, `f`

Screen then shows Ciphertext as:

```
RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZPCYJ
  enf  e  i       f    rie   e   y re     n  in   i  f renergyin  e   f  ing

AKHVADVWMHYHRNBDAHCYP
  e      ene  r  ein
```

See end of the ciphertext. `P` ciphertext is `s` plaintext?

And just before plaintext word energy, `D` ciphertext gives `o` plaintext? Entering these values results in the following screen:

```
                CRIBLOC, a programme to help locate a crib in a PAT

  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
  6  5  7  9  2  4  5 12  4  2  4  0  3  1  0  6  0  4  0  2  0  3  9  2  8  1
  Total characters 99 Type K1    Crib ENERGY          There are 3 fit(s).

Ciphertext is:
RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZPCYJ
  enf  e  i       of   ories  e o y re  s o n  ino  i sforenergyins e  of sing

AKHVADVWMHYHRNBDAHCYP
  e  o   ene  ro eins
```

```
  Positions with best fit of ENERGY:
             Order         Value        Position        Sequence
               1             7            59           HYHBJT

  CT$  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  PT$   rio   efg     s   y    n                CT letter D    The PT o


         (Q)uit this crib, (N)ext pos. (R)e try, or (*A)dd a letter
```

There are now a number of possibilities. Last comes `instead`? So we now have: plaintext word `protiens`? After `for energy`

```
                CRIBLOC, a programme to help locate a crib in a PAT

  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
  6  5  7  9  2  4  5 12  4  2  4  0  3  1  0  6  0  4  0  2  0  3  9  2  8  1
  Total characters 99 Type K1    Crib ENERGY          There are 3 fit(s).

Ciphertext is:
RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZPCYJ
  enfa ed it a a  of a oriest e ody rea sdo na inoa idsforenergyinsteadofusing

AKHVADVWMHYHRNBDAHCYP
t e to a ene proteins

  Positions with best fit of ENERGY:
             Order         Value        Position        Sequence
               1             7            59           HYHBJT

  CT$  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  PT$  trio  defg   p s   y  a nu               CT letter G    The PT D


         (Q)uit this crib, (N)ext pos. (R)e try, or (*A)dd a letter
```

The addition of `h`, `w`, `c`, `l`, `k` and `b` now follow, giving the solution:

```
                CRIBLOC, a programme to help locate a crib in a PAT

  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
  6  5  7  9  2  4  5 12  4  2  4  0  3  1  0  6  0  4  0  2  0  3  9  2  8  1
  Total characters 99 Type K1    Crib ENERGY          There are 3 fit(s).

Ciphertext is:
RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZPCYJ
whenfacedwithalackofcaloriesthebodybreaksdownaminoacidsforenergyinsteadofusing

AKHVADVWMHYHRNBDAHCYP
```

```
themtomakenewproteins

  Positions with best fit of ENERGY:
             Order         Value        Position        Sequence
               1             7            59           HYHBJT

  CT$  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  PT$  triobcdefghjkpqsvwxyzmalnu                CT letter O    The PT Q


         (Q)uit this crib, (N)ext pos. (R)e try, or (*A)dd a letter
```

The prompt line allows:
`Q` to get back to base, run again, or try another crib/probable word.
`N` moves on to the next fit position (not necessary in this example!).
`R` resets the same crib position, if wrong assumptions are made.
`A` allows adding new substitute letters.

Now comments on the programme listing:

The programme is written in QuickBasic, but line numbers have been maintained for users of other dialects. Some syntax may also need amendment for other flavours of BASIC, like the `UCASE$(` and `LCASE$(` functions, and the `IF`, `THEN`, `ELSE`, and `END IF` construct. Each section of the programme starts with a line numbered *nnn9*, (which may be omitted), indicating what that part does.

The programme can be useful with K3 and K4 types, but adding letters to plaintext alphabet is less reliable. As with all my programmes, if on disc the ciphertext is in a file with extension `.CT$` Line `1540` limits the fits tried to 18, even if more are found. It can be increased if required. In use, the highest number actually needed has been 11. Finally, when finished, I call my `MENU` programme. Line `2400` will need amending to your own exit method.

Duplicate use of letters is not checked. But entering a second substitution overwrites the earlier one, and `R` at prompt allows a fresh start with same crib location. Use is suggested for PAT's numbered 4 or 5 to about 10. With luck, even P-11 and P-12! The programme has been used successfully with simple XENO's, but the scoring of best fits is not accurate. Try more of them!

```
99   '   A PROG. TO HELP LOCATE A CRIB IN A PAT. v 4.1
100  ' Set up variables, etc.
110  COLOR 4, 7: AB$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ": BA$ = SPACE$(26)
120  DIM F(26), A(26), B(20), R(20), T(255), Z(20)
130  CX$ = "RKHYIWFHGRCAKWXWFMDIFWXDBCHPAKHEDGTEBHWMPGDRYWVCYDWFCGPIDBHYHBJTCYPAHWGDIZ
     PCYJAKHVADVWMHYHRNBDAHCYP"
140  HD$ = SPACE$(17) + "CRIBLOC, a programme to help locate a crib in a PAT" + CHR$(13)
150  BX$ = "ENERGY"  ' PT begins "INSPIRATION FOR CONS ...."
     'NEXT' is last word, 11th fit.
160  K2$ = "  PT$  ": K1$ = "  CT$  ": S$ = "ETAONIRSHLDCUPFMWYBGVKQXJZ"
170  F$ = " 13 9 8 8 7 7 7 6 6 4 4 3 3 3 3 2 2 2 1 1 1 0 0 0 0 0 ": GOTO 600

389 ' The subs start here
```

```
399 ' Make pattern of crib/crib length fragment of CT
400  J = 1: TT$ = SPACE$(C)
410  FOR K = 1 TO C - 1: K$ = RIGHT$(STR$(K), 1): A$ = MID$(CP$, K, 1)
420    IF INSTR(D$, A$) THEN 460
430    A = INSTR(J + K, CP$, A$): IF A = 0 THEN GOTO 460
440    MID$(TT$, K, 1) = K$: MID$(TT$, A, 1) = K$: J = A: D$ = D$ + A$
450    IF A + 1 < C THEN 430
460    J = 1
470  NEXT: PRINT "."; : D$ = ""
480  RETURN

599  ' Prog. starts here!
600 CLS : PRINT HD$  ' Get CT and type (K1 or K2)
610 PRINT " Enter cypher text, in UPPER CASE, (ENTER for example, 'F' if data on disc)"
620 INPUT CT$: CT$ = UCASE$(CT$): IF CT$ = "" THEN CT$ = CX$: GOTO 670
630 IF CT$ <> "F" THEN 670
640 PRINT : INPUT "Name of file holding data "; N$: I = INSTR(N$, ".")
650 IF I > 0 THEN N$ = LEFT$(N$, I - 1)
660 N$ = N$ + ".CT$": OPEN "I", #1, N$: INPUT #1, CT$: CLOSE #1
670 I = INSTR(CT$, " ")
    :IF I > 0 THEN CT$ = LEFT$(CT$, I - 1) + MID$(CT$, I + 1): GOTO 670
680 PRINT : PRINT CT$: PRINT : INPUT "     Type K(1) or K(*2) ", TP$
690 IF INSTR(TP$, "1") THEN TP$ = "K1": SWAP K1$, K2$ ELSE TP$ = "K2"

799 ' Calc freq. dist of CT
800 L = LEN(CT$): PT$ = SPACE$(L)
810 FOR I = 1 TO L
820   N = ASC(MID$(CT$, I, 1)) - 64: F(N) = F(N) + 1: A(N) = A(N) + 100 / L
830 NEXT

899 ' Display CT and freq. dist
900 CLS : PRINT HD$: PRINT "  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R
    S  T  U  V  W  X  Y  Z"
910 FOR TT = 1 TO 26: PRINT USING "###"; F(TT); : NEXT: PRINT
920 PRINT "  Total characters"; L; "Type "; TP$: PRINT
    : PRINT "Ciphertext is:": CL2 = CSRLIN
930 IF L > 78 THEN PRINT LEFT$(CT$, 78) ELSE PRINT CT$: GOTO 1000
940 PRINT : PRINT : PRINT MID$(CT$, 79)

999 ' Get crib
1000 PRINT : PRINT : CL = CSRLIN
1010 LINE INPUT "  Enter crib: "; CB$: CB$ = UCASE$(CB$)
1020 C = LEN(CB$): IF C < 2 THEN CB$ = BX$: C = LEN(CB$)
1030 LOCATE 5, 34: PRINT "Crib "; CB$
1040 LOCATE CL, 15: PRINT CB$;

1099 ' Check letters in crib, & find freq. of 'em. DD shows up patt. words
1100 FOR I = 1 TO C: A$ = MID$(CB$, I, 1): A = INSTR(S$, A$)
1110   IF A = 0 THEN PRINT A$; " NOT FOUND IN S$": STOP
1120   B(I) = VAL(MID$(F$, A * 2, 2))
1130   A = INSTR(I + 1, CB$, A$): IF A > 0 THEN DD = DD + 1
1140 NEXT
1150 IF DD > 0 THEN      '  If crib is pattern word,
```

```
1160   CP$ = CB$: GOSUB 400
1170   TC$ = TT$
1180 ELSE                ' and if not,
1190   TC$ = SPACE$(C)
1200 END IF

1299  ' step through CT$ for self subst. and pattern mis-match
1300 FOR I = 1 TO L - C + 1: CP$ = MID$(CT$, I, C)
1310   FOR U = 1 TO C ' Kill self substitution
1320     IF MID$(CB$, U, 1) = MID$(CP$, U, 1) THEN T(I) = 999: U = 99
1330   NEXT U: IF U > 90 THEN GOTO 1530

1399   ' patterns match?
1400   GOSUB 400: IF TC$ <> TT$ THEN T(I) = 999: GOTO 1530
1499   ' calc a value of freq dist, CT letters/crib normals
1500   FOR J = 1 TO C
1510     T(I) = T(I) + (A(ASC(MID$(CT$, I + J - 1, 1)) - 64) - B(J)) ^ 2:
1520   NEXT
1530 NEXT I        ' Get value of fit, and count 'em
1540 FOR K = 1 TO 18: H = 999:
1550   FOR I = 1 TO L - C + 1
1560     IF H > T(I) THEN H = T(I): R(K) = I: CC = CC + 1
1570   NEXT: Z(K) = T(R(K)): T(R(K)) = 999: PRINT ",";
1580 NEXT
1590 LOCATE 14, 1: PRINT SPACE$(240)
1600 LOCATE 5, 55: PRINT "There are"; CC; "fit(s).  ";
1610 IF CC = 0 THEN PRINT : GOTO 2330
1620 LOCATE 14, 1
1630 PRINT "  Positions with best fit of "; CB$; ":"
     : PRINT "             Order         Value        Position        Sequence"
1640 CL = CSRLIN
1650 FOR I = 1 TO 18
1660   IF R(I) = 0 OR Z(I) = 999 THEN
1670      I = 19: PRINT "No more."
1680   ELSE
1690     BB$ = MID$(CT$, R(I), C)
1700     LOCATE CL, 3: PRINT "  ", I, INT(Z(I)), R(I), BB$
1710     FOR J = 1 TO C: K = 1
1720        A$ = MID$(BB$, J, 1)
1730        JJ = INSTR(K, CT$, A$)
1740        IF JJ > 0 THEN K = JJ + 1
            : MID$(PT$, JJ, 1) = LCASE$(MID$(CB$, J, 1)): GOTO 1730
1750     NEXT
1760     LOCATE CL2 + 1, 1
1770     IF L > 78 THEN PRINT LEFT$(PT$, 78) ELSE PRINT PT$: GOTO 1790
1780     PRINT : PRINT : PRINT MID$(PT$, 79)
1790   END IF

1899    ' Put crib into PT$
1900   FOR II = 1 TO C
1910     A1$ = MID$(CB$, II, 1): A2$ = MID$(BB$, II, 1)
1920     IF TP$ = "K1" THEN SWAP A1$, A2$
1930     III = INSTR(AB$, A1$): MID$(BA$, III, 1) = LCASE$(A2$)
```

```
1940   NEXT II
1950   LOCATE 18, 1: PRINT "     " + AB$: PRINT "     " + UCASE$(BA$) + "  "
1960   LOCATE 22, 65: PRINT "    ": LOCATE 22, 10

2090   INPUT "(Q)uit this crib, (N)ext pos. (R)e try, or (*A)dd a letter ", Z$
2100   IF UCASE$(Z$) = "N" THEN GOTO 2310
2110   IF UCASE$(Z$) = "Q" THEN I = 999: GOTO 2310
2120   IF UCASE$(Z$) = "R" THEN I = I - 1: GOTO 2310
2130   LOCATE 19, 40: PRINT SPACE$(38)
2140   LOCATE 19, 50: INPUT "CT letter ", C$: C$ = UCASE$(C$): IF C$ = "" THEN GOTO 2310
2150   LOCATE 19, 65: INPUT "The PT ", P$: P$ = UCASE$(P$)
2160   IF TP$ = "K2" THEN
2170      AA = ASC(P$) - 64: IF AA < 1 THEN C$ = " "
2180      MID$(BA$, AA, 1) = LCASE$(C$)
2190      LOCATE 18, 1: PRINT K1$ + UCASE$(BA$): PRINT K2$ + LCASE$(AB$) + "  "
2200   ELSE
2210      AA = ASC(C$) - 64: IF AA < 1 THEN P$ = " "
2220      MID$(BA$, AA, 1) = LCASE$(P$)
2230      LOCATE 18, 1: PRINT K2$ + UCASE$(AB$): PRINT K1$ + LCASE$(BA$) + "  "
2240   END IF

2259    'Put entered letters into PT and alphabets
2260   IR = 0
2270   A = INSTR(IR + 1, CT$, C$): IF A = 0 THEN GOTO 2290
2280   MID$(PT$, A, 1) = LCASE$(P$): IR = A: GOTO 2270
2290   LOCATE CL2 + 1, 1: IF L > 78 THEN PRINT LEFT$(PT$, 78) ELSE PRINT PT$: GOTO 1960
2300   PRINT : PRINT : PRINT MID$(PT$, 79):  GOTO 1960
2310   PT$ = SPACE$(L): BA$ = SPACE$(26)
2320 NEXT
2330 LOCATE 22, 10: PRINT SPACE$(65)
2340 LOCATE 22, 10: INPUT "Back to (M)enu, (R)un again, (F)resh crib"; Z$
     : Z$ = UCASE$(Z$)
2350 IF Z$ = "R" THEN RUN
2360 IF Z$ = "F" THEN
2370   DD = 0: CC = 0: FOR I = 1 TO L - C: T(I) = 0: NEXT
2380   FOR I = 1 TO 15: Z(I) = 0: R(I) = 0: NEXT: GOTO 900
2390 END IF
2400 IF Z$ = "M" THEN RUN "MENU.BAS" ELSE GOTO 2330
```

PGP REMINDER

For those of you wanting to get started with Pretty Good Privacy (PGP), **MCTAYLOR** (Michael C. Taylor) reports that PGP 5.x and PGP 2.6.x may be found on the Web at:

• non-commerical in US/Canada
`http://web.mit.edu/network/pgp.html`

• commerical in US/Canada
`http://www.pgp.com/`

• outside US (but including Canada)
`http://www.pgpi.com/`

Shakespeare on the Web

Patrick Leary (RETREAD)

ACA members may be interested in a new crypto site. Discussed is the authorship of Shakespeare's Works which I attribute to Francis Bacon. Among other things it includes a copy of my book, *The Second Cryptographic Shakespeare*, a pamphlet summary and a BASIC crypto program. The Uniform Resource Locator (URL) is:

`http://fly.hiwaay.net/~paul/outline.html`

Paul Dupuy, who continues work on the page, found my book in the library and did this on his own. He is a graduate student in computer science and later found me on the Internet.

Baltimore Sun series on the NSA

RagyR

On two recent occasions (MA96 Cryptogram and April 96 Cryptologia) , MEROKE (Louis Kruh) has reviewed the Baltimore Sun's reprint of their series of articles on the National Security Agency. The series sounded interesting enough that I attempted to obtain copies to make available to the Krewe at discount. Alas, that will not happen as the Sun didn't cooperate. But, I did uncover some additional information that may be of interest.

The reprint can be ordered from SunSource, the Sun's reprint distribution office, by phone at (800) 829-8000, Ext. 6800. Be warned, though, that its price has gone up to $6.95. This still seems reasonable and I have seen it advertised by another bookseller for twice that amount.

Also, an excerpt from the series can be found on the web at:
`http://www.sunstore.com/sunsource/`

CORRECTION TO QUAG 1&2

Wilfred Higginson

The Quag 1&2 programme in the current issue contains a bug. Not much, the programme works O.K., but the number of fits reported after a change of crib is incorrect. The amendment is just to add `M=0` after the `RESTORE` in line `2690`, so that it reads :

```
2960 IF Z$="R"...ELSE RESTORE 2700:M=0:CLS...
```

Hope that doesn't cause too much trouble? Also, there is confusion here between CIPHERTEXT and CLEARTEXT. Don't know how that arose, but no doubt the Krewe will sort it out?

BREAKTHROUGH '32 - A minor problem solved!

David Hamer (DAEDALUS)

In CS19 the editor relayed my call for assistance with *BREAKTHROUGH '32; The Polish Solution of the Enigma*. As a result I received the solution to a minor problem I had encountered with C. A. Deavours' `ENIGMA.BAS` program, from **ODAL** and a further communication on the subject from **BINO**.

I had begun work on the *BREAKTHROUGH '32* problems a couple of years ago but came to an abrupt halt on page 13 when the expected `FAST ROTOR STEPS ONLY (Y OR N)?` didn't appear on schedule. Some time passed and I found myself back in the United States; the `ACA-L` mailing list was established on the Internet and I posted a cry for help. [**Ed:** information on the ACA-L list may be found on page 24. DMV]

I had no response directly but after my request was repeated in CS19 I got a letter from Heinz Ulbricht (**ODAL**) in Braunschweig — who had encountered the problem earlier — in which he suggested adding lines 145 and 775 to my copy of Deavours' program, thus:

```
140 MID$(R$(I), Z, 1) = CHR$(J + 64): NEXT J: NEXT I
145 INPUT "FAST ROTOR STEPS ONLY (Y OR N)"; SP$
150 PRINT "ROTOR WIRINGS:"

770 REM MOVE ROTORS
775 IF SP$ = "Y" THEN RP(1) = (RP(1) + 1) MOD 26: P(1) = (P(1) + 1)
    MOD 26: P(7) = P(1): RETURN
780 MOVE2 = 0: MOVE3 = 0
```

This worked fine and I was able to proceed, literally, 'by-the-book'.

Additional discussion with other readers/users of the Aegean Park Press text (#51) suggested that Wayne Barker had included the "shorter" version of Deavours' program with some, if not all, copies of the book. My sampling on this is small, but out of the four users with whom I corresponded, all had encountered the same difficulty. A telephone conversation with Professor Deavours revealed that there are several versions of `ENIGMA.BAS` in circulation. The "normal" version does not contain the extra lines; presumably because one does not usually want to prevent the slower rotors from stepping. Somehow this "normal" version was included with *BREAKTHROUGH '32*, rather than the "special" version required by the exercises. In fact, the additional lines are included in the version of `ENIGMA.BAS` that Deavours distributed on request after the publication of *The Turing Bombe: Was it Enough ?* in *CRYPTOLOGIA, Vol.XIV, No.4, (1990)*.

Interestingly, the two versions are date-stamped only one day apart; 12/03/87 and 12/04/87 !

A caution: Most of the `.BAS` programs associated with *BREAKTHROUGH '32* and the *CRYPTOLOGIA* article are not saved in ASCII format and so cannot be read, compiled or otherwise manipulated with the MS-QuickBASIC (or similar) editor/compiler without first being converted. Conversion to ASCII involves the use of the now-almost-defunct GW-BASIC/BASICA etc. interpreter, first to list the `.BAS` file and then to save it with the `,a` option.

Moral: Don't throw out your old "line" BASIC software !

WHAT THE OTHER GUY IS DOING

DAEDALUS (David Hamer) has "more-or-less" retired from his aviation career but keeps his hand in by occasionally piloting a business jet for a small company based in his home state of New Jersey. He has resumed his interest in things mathematical and is involved in the development and marketing of computer software related, primarily, to system security. He also teaches, on a part-time basis, at a local college where he is striving, with questionable effect, to stay one step ahead of his students in calculus !

CERE E. US (Ed Magelky) has been working in the C language, and in working on "recursive reduction" has invented several techniques and algorithms to greatly speed up the reduction process.

LANAKI (Randall K. Nichols) has a two volumes of cryptographic material available from RAGYR's Classical Crypto Books and from Aegean Park Press. Volumes I and II of Nichols' Classical Cryptography Course are the summation of a electronic-mail course given to more than 450 students by Mr. Nichols.

LANAKI also reports that he has accepted a position as Technology Director — Cryptography for the National Computer Security Association (NCSA) in Carlisle, Pennsylvania. He will be in charge of both the Cryptography Lab and staff as well as the Commercial Certification process for Cryptography Products Consortium members (CPC) from both United States and Europe. In his spare time, LANAKI will continue to edit the ARISTOCRATS column for ACA and complete his *Advanced Cryptography and Commercial Computer Security* text for McGraw-Hill.

Also, for ENIGMA buffs, LANAKI suggests `http://www.enigma-co.com/resource.html`

SI SI (Clayton Pierce) has assembled a 28 page monograph entitled *Hobby Cryptography For Classical Ciphers*, to serve as a replacement for the cryptography in chapter eight of the classic *ACA & YOU*.

John K. Taber reports that the NSA has declassified reams of cryptographic stuff under their "Opendoor" program. An index of released material is available on the NSA web site, `http://www.nsa.gov:8080/programs/opendoor/` Apparently, you should search the index for items you are interested in, then go to National Archives and Records Administration (NARA) for the document(s). Has anyone done this?

MCTAYLOR (Michael C. Taylor) asks: Has anyone been able to duplicate some of the the documents available via OpenDoor? In particular the Zimmerman Telegram, Turing's treatise on the Engima, anything specific to analysis of ciphers and codes, and my personal interest, Canadian references.

Being outside USA, and in particular not near Maryland, it is little more difficult for me to check it out myself. I am uncertain, but it is possible that the material may be in the public domain, as I presume the NSA is considered a US government organization. This would facilitate the duplication and redistribution of this information to ACA members.

FUTHARK (Joe Palakanis) is up and running with DOS 6.22 and QBasic on his 166 MHz Pentium. He reports that QBasic is available on the Windows 95 CD-ROM distribution, and can be located using the "Find" function, at which point it may be rolled onto the hard drive. For that tip he thanks new member James A. Parsly, who it turns out has a program called `CRYPTOSOLVE` which does a fair job solving Aristos. It is available at `http://www.public.usit.net/jparsly/`.

FUTHARK also notes that Buckmaster Publishing Company at `http://www.buck.com/cryp.html` sells a reprint of TM-11-485, *Advanced Military Cryptography*, a War Department Technical Manual originally published in 1944, for $19.95.

MEROKE (Louis Kruh) adds: For my review of *Advanced Military Cryptography* from Buckmaster Publishing, see CM, J/A '93, p. 15.

CLASSICAL CRYPTO BOOKS

If you've been looking for a particular book related to cryptography, or just wanted a "one-stop-shop" bookstore for your code-breaking hobby, get in touch with RAGYR (Gary Rasmussen). RAGYR has a list of more than 150 titles covering such topics as SIGINT history, recreations, classical cryptology, modern and advanced cryptology, traffic analysis, and various references. He also carries books on intelligence, foreign language, has books for beginners, and has a number of fiction titles.

From his catalog:

> Specializing in Traditional Ciphers, Codes, and Signals Intelligence
>
> Classical Crypto Books was started in 1995 by Gary Rasmussen as a service for other members of the American Cryptogram Association. Its objective is to provide a convenient place for the Krewe (as ACA members are known) to obtain new (unused) books on traditional codes, ciphers, and signals intelligence at discount prices. Orders are welcomed from members and non-members alike. However, members receive discounts which are unavailable to non-members.

```
CLASSICAL CRYPTO BOOKS
PO Box 1013
Londonderry, NH 03053-1013
USA
```

Send electronic mail inquiries to `RagyR@aol.com`

CRYPTOSYSTEMS JOURNAL

After more than two years of "spare time" effort, Tony Patti has produced Volume 4 of his *Cryptosystems Journal*. This monumental effort, comprising almost 300 pages of text and color illustrations, covers a number of software and hardware issues related to cryptography, as well as book reviews and sanguine commentary.

One section is dedicated to a discussion of randomness, with a variety of quoted sources and detailed "hands-on" projects for creating a "random" number generator out of digital hardware. Another section describes the generation of realistic 3-D computer graphics and fractal images, complete with references and illustrations. Other sections cover software implementations of various cryptosystems, book reviews, and there's even a crossword puzzle!

Tony Patti may be reached at:

```
485 Middle Holland Road
Holland, Pennsylvania  18966
USA
(215) 579-9888
```

Tony is also reachable via electronic mail, `crypto@compuserve.com`, and has a World Wide Web page set up at `http://ourworld.compuserve.com/homepages/crypto`.

An Introduction To Public Key Cryptography

André Kesteloot

Anyone who reads the Usenet groups on the Internet from time to time will have noticed that many postings now include a PGP key. PGP (Pretty Good Privacy) allies the strength of Public Key Cryptography (PKC) with the speed of another encryption algorithm known as IDEA. This article will attempt to offer an introduction to Public Key Cryptography, along with a few simple BASIC programs that can be run on any personal computer for demonstration purposes.

Background

Transferring information from point A to point B, while hiding its true meaning, has long been a cherished human endeavor. Cryptography was probably created the day after the first alphabet was invented, and since then, many clever schemes have been generated, only to be subsequently defeated. The traditional methods always relied on the two parties having a common reference. One side would encode his or her message, using the common reference, and forward it to the intended recipient, who would then somehow look up the reference, and decode the message. (During the Civil War, for instance, it was not uncommon for people to own a Bible, and provided that both parties had agreed in advance to use a given chapter of a given book of the same edition of the Bible, then the message could be enciphered using just the page number, the line number and the position of the desired word on that line.)

This being a fairly slow process, better machines were eventually invented and the introduction of computers certainly made things go faster, but essentially, it was still the same old problem: how to generate a better "key" and, equally troublesome, how to pass it to the intended recipient so that both the sender and the recipient (and no one else) have access to it. Traditionally, if Mrs. A wanted to send a encrypted love letter to Mr. B, but was being watched 24 hours a day, the perennial problem was for her to pass the key to Mr. B. They would have to come in contact at least once (either face to face, by mail, by phone, etc.) for them to exchange keys or to agree to some common reference. Since that exchange can either be observed, compromised or spoofed, the whole security of the future epistolary relationship between Mrs. A and Mr. B resided (a) in the strength of the encryption algorithm chosen, and (b) in the integrity and/or secrecy of the key exchange process.

Then came True Change with the introduction of Public Key Cryptography (I would venture to say that PKC will be seen, in retrospect, as one of the most momentous developments in technology during the past 20 years, not only because of the radically new approaches it offers to old problems, but also because of the sheer number of aspects of our lives it will quietly impact). I won't go into the history of PKC (the reader may wish to reference *Contemporary Cryptology*, an excellent anthology published by the IEEE in 1992) but the basic revolutionary idea which came to its creators in 1978 was to challenge the established axiom that the same key had to be available at both ends for encryption and decryption to be able to take place. There are many different forms of PKC but the best known is probably the RSA system, named after its creators, Rivest, Shamir and Alderman. It uses one key for encryption (called the Public key because it can actually be made public) and another (Private or secret) key for decryption. The concept is based on the use of one-way, or trapdoor, mathematical functions. These functions are mathematical operations trivial to perform in one direction, but extremely arduous in the reverse direction. For instance, to raise 23 to the third power, i.e. 23^3, we simply multiply $23 \times 23 \times 23$ and instantly obtain 12167. In the reverse direction though, extracting the cubic root of 12167 is not quite so easy !

The old concept of cryptography usually evoked the idea of conspirators meeting in dark alleys for nefarious purposes; in fact, every step of the exchange was somehow shrouded in secrecy. The new PKC system changed that too, and even the way an exchange is described in technical literature was modified: the token protagonists named in Public Key Cryptography scientific articles are no longer faceless entities, but have become plausible human beings, usually referred to as "Bob" and "Alice". Because this is all quite revolutionary, and yet impacts seriously on our everyday life, allow me to describe, step by, step, a typical PKC transaction. (I have added Vader-the-Villain, a.k.a. Darth Vader, as the imperishable enemy, forever trying to intercept and understand the exchange of messages between Alice and Bob).

The New Process

Let us suppose that Alice wants to send an encrypted message to Bob. Bob has a public key which he has posted at the bottom of his e-mail messages, his Usenet postings, etc. Alice looks up Bob's public key in a publicly available directory. She uses it to encrypt her message and then posts this encrypted message in a public place. (This could be the local newspaper classified ads, or a banking network. Although the latter is certainly more private than the former, we will assume that it could be under attack by rogues, thieves and assorted cutthroats.) Now Bob retrieves the message, and decrypts it using his private (secret) key. This is the only step in the whole transaction that needs to be protected and kept confidential. Note that the public key and the private key are related to one another but that, if the one-way mathematical function is properly chosen, the private key cannot be derived from the public one. (Or, perhaps more accurately, can only be derived using exorbitant amounts of computing power, not usually available to your everyday villian in a convenient time frame. Note that if Bob has published several public keys, and Alice uses a new key for each transaction, then Vader-the-Villian is faced with even more serious problems!)

Key generation

The problem is to generate two keys, one private and the other, public which are related to, but cannot easily be derived from, each other. There are many ways to generate such keys, and here is but one example.

First select two large **prime numbers** p and q, each possibly 100 digits long. Their product,

$$n = p \times q \qquad \text{(Eqn. 1)}$$

will be Bob's 200 digit long **public** encryption key. (There are certain restrictions in the choice of appropriate values for p and q, which are beyond the scope of this overview, and the interested reader should look up the references listed at the end of this article). This is the key he will make available in some public place.

The **private** or secret decryption key d is now obtained by Bob by calculating

$$d = [2(p-1) \times (q-1) + 1]/3 \qquad \text{(Eqn. 2)}$$

Remember that the world-at-large, (including Alice and Vader-the-Villain) only knows Bob's public key n, but if n is a 200-digit number, it is very difficult, if not practically impossible to retrieve, with the sole knowledge of n, the values of the two prime numbers p and q.

Now to encrypt a message we can, for instance, take each letter or digit and convert it to its ASCII equivalent. Several of these ASCII values can now be grouped and encrypted by cubing them modulo n. This will be the encrypted group C which will be transmitted by Alice to Bob. Incidentally, a modulo operation is performed by dividing the number in question by the modulus, and posting only the remainder. Hence $4 \times 4 \times 4 \bmod 11 = 9$ because $64/11 = 5$ with a remainder of 9. Note that $15 \times 15 \times 15 \bmod 11 = 9$; $26^3 \bmod 11 = 9$; $37^3 \bmod 11 = 9$; $43^3 \bmod 11 = 9$, etc., which will make the job of Vader-the-Villain even more difficult, as the only information Alice sends to Bob is C, the remainder of the modulo operation.

To decrypt the message C, Bob must know his public encryption key n and his private/secret decryption key d and must calculate the result of $[C^d \bmod n]$; that is, Bob must raise C to the d power and then modulo n the result.

An Example

Let us consider a practical example: in the January 1983 issue of *Byte Magazine*, John Smith published a very interesting article describing the implementation of a simplified form of the RSA algorithm. I have simplified it even further, and will present now two short BASIC programs you can use to demonstrate to yourself the concepts of PKC. To make my example both intelligible and manageable, I will use small prime number (but please remember that the real system is only secure because it uses huge, 100-digit long prime numbers as factors.)

For the purpose of this simple demonstration, my two "large" prime numbers will be

$$\begin{aligned}
 & p &=& \ 11 \\
 & q &=& \ 17 \\
perEq.1, \quad & n &=& \ p \times q \\
 & &=& \ 11 \times 17 \\
 & &=& \ 187 \\
perEq.2, \quad & d &=& \ [2(p-1) \times (q-1) + 1]/3 \\
 & &=& \ [20 \times 16 + 1]/3 \\
 & &=& \ 321/3 \\
 & &=& \ 107
\end{aligned}$$

To recapitulate:

Bob's first prime number	$p = 11$
Bob's second prime number	$q = 17$
Bob's public encryption key	$n = 187$
Bob's private decryption key	$d = 107$

The only thing Alice knows is that Bob's public key is 187, and that she must cube her clear-text group and then "modulo 187" it. The result C will be transmined to Bob, whose traffic, we will assume, can be, somehow, intercepted by Vader-the-Villain. Vader will thus know not only C which he has just intercepted, but also $n = 187$ since this is Bob's public key, readily available to everyone.

Remember that, on the other hand, Bob knows both Alice's message as well as his own public and private keys. To decrypt Alice's message C, Bob will have to calculate $C^d \bmod n$.

Let's suppose that Alice wants to send Bob the letter "A" (her initial) whose ASCII equivalent is 65. Using Bob's public key $n = 187$, she calculates

$$\begin{aligned} C &= 65^3 \bmod 187 \\ &\quad (65 \times 65 \times 65) \bmod 187 \\ &\quad 274625 \bmod 187 \\ &\quad 109 \end{aligned}$$

and she sends 109 to Bob.

Bob must now simply calculate $109^{107} \bmod 187$! *Hmmmmm, say... what?* How specifically does Bob calculate the 107th power of 109? I thought you would never ask! There is a simple method to do so, described in some detail in the Byte 1983 article, and known as the "Russian Peasant's method." It is implemented in the short BASIC program listed below. Anyway, amazingly enough, $109^{107} \bmod 187 = 65$, the number Alice originally wanted Bob to receive. Please note that Vader cannot derive the value of 65 from 187 and 109, the only two numbers he knows. Try it for yourself, remembering that the clear text you want to encrypt should be smaller than the public key you use.

BASIC Programs

Here is a 6-line encryption program `ENCRYPT.BAS`

```
100 CLS'THIS IS A SIMPLE RSA ENCRYPTION PROGRAM, SEE BYTE JAN 83
110 INPUT "PLEASE ENTER THE PUBLIC KEY OF YOUR CORRESPONDENT";N
120 INPUT "NOW ENTER THE NUMBER YOU WANT TO ENCRYPT (0=END)";A
130 A1=A*A*A: A1=A1-INT(A1/N)*N'        CUBE OF (A) THEN MODULO N
140 PRINT "THE ENCRYPTED BLOCK IS"A1:PRINT
150 IF A=0 THEN END ELSE GOTO 120'      END, OR PLAY IT AGAIN SAM
```

Now for the corresponding decryption program `DECRYPT.BAS`

```
100 CLS'      this program decrypts rsa messages, see byte jan83
110 DEFDBL C,D,M,N'     DOUBLE PRECISION
120 INPUT "PLEASE ENTER YOUR PUBLIC KEY";N
130 INPUT "AND NOW YOUR PRIVATE KEY";D
140 INPUT "AND NOW THE RECEIVED CRYPTOGRAM BLOCK (TO EXIT, ENTER 0)";C
150 IF C=0 THEN END'    EXIT THIS PROGRAM
160 GOSUB 190'  START DECRYPTING
170 PRINT "AND THE CLEAR TEXT IS";M:PRINT
180 GOTO 140'   AND AGAIN
190 D1=D: M=1'  RUSSIAN PEASANT METHOD
200 IF D1/2=INT(D1/2) GOTO 220' SKIP IF D1 = EVEN
210 M=M*C: M=M-INT(M/N)*N'      M = (M*C) MODULO N
220 C=C*C: C=C-INT(C/N)*N'      C = (C*C) MODULO N
230 D1=INT(D1/2): IF D1>0 GOTO 200
240 RETURN
```

How does this magick work? Let me now explain why and how the problem of efficiently raising a huge number to a huge power can best be tackled by a computer. My approach is slightly different from the Russian Peasant Method originally described in the Byte article, and since I live in Northern Virginia, I decided to name it the Northern Virginian Peasant Algorithm).

The Northern Virginia Peasant Algorithm

Firstly, computers use the binary system, in which several operations can be performed with particular efficacy:

1. checking whether a number is even or odd is determined by looking at the least significant bit; if that bit = 0, the number is even, and if that bit = 1, the number is odd.
2. multiplying a number by 2, which simply means shifting everything one bit to the left and making the new least-significant bit = 0.
3. dividing an even number by 2, by removing the least significant bit and shifting everything one bit to the right.
4. checking whether a number is > 0, by checking the most significant bit.

Secondly, before we tackle the problem of raising a number X to a power Y, let us first examine how a computer can best be used to multiply efficiently a number X by another number Y, i.e., $X \times Y$. Let us open three registers `x`, `y`, and `z` and load them with the discrete values X, Y and 0 respectively.

Since we haven't modified anything, it is certainly true that

$$X \times Y = (\mathtt{x} \times \mathtt{y}) + \mathtt{z} \qquad \text{(Eqn. 3)}$$

Now for our process:

1. First check that $Y > 0$, then
2. if Y is even, we can replace the value in register `y` by `y`/2 and the value in register `x` by 2`x`.

   Again, we haven't changed the original relationship since we now have $(2\mathtt{x} \times \mathtt{y}/2) + \mathtt{z}$ which still satisfies equation 3.
3. if Y is odd, then we can replace the value in register `y` by $(\mathtt{y}-1)$ and the value in register `z` by $(\mathtt{z}+\mathtt{x})$.

   Equation 3 now becomes
   $$\begin{aligned} X \times Y &= \mathtt{x} \times (\mathtt{y}-1) + (\mathtt{z}+\mathtt{x}) \\ &= (\mathtt{x} \times \mathtt{y}) - \mathtt{x} + \mathtt{z} + \mathtt{x} \\ &= (\mathtt{x} \times \mathtt{y}) + \mathtt{z} \end{aligned}$$
   and our original relation still holds.
4. Now go back to (1) above and check again if Y is > 0. Keep performing the steps just described until $Y = 0$, at which time, stop! The result of our operation is the value of register `z`.

Example: to multiply 5 by 6, we will set the three registers `x`, `y` and `z` as follows: $\mathtt{x} = 5, \mathtt{y} = 6, \mathtt{z} = 0$.

```
Is y=6>0 ? yes
Is 6 even ? yes, then replace y=6 by (6/2)=3 and replace x=5 by (5*2)=10
Is y=3>0 ? yes
Is y=3 even ? no, then replace y=3 by y=2 and z=0 by z=(0+10)
Is y=2>0 ? yes
Is y=2 even ? yes, then replace y=2 by y(2/2)=1 and x=10 by x=(10*2)=20
Is y=1 even ? no, then replace y=1 by y=0 and z=10 by z=z+x=(10+20)=30
Is y=0>0? no, then stop!
Result = (5*6) = value of register z = 30 (Answer)
```

Here is a short BASIC program I wrote to demonstrate the above Russian Peasant Method:

```
100 CLS:PRINT "MULTI.BAS, THE RUSSIAN PEASANT METHOD"
110 PRINT "TO MULTIPLY A NUMBER X BY ANOTHER NUMBER Y."
```

```
120 PRINT: INPUT "WHAT IS YOUR FIRST NUMBER (0=EXIT)";X
130 IF X=0 THEN END
140 INPUT "WHAT IS YOUR SECOND NUMBER ";Y
150 Z=0
160 IF Y>0 GOTO 170 ELSE GOTO 200
170 IF Y/2=INT(Y/2) GOTO 190'   IS Y EVEN?
180 Y=Y-1: Z=X+Z: GOTO 160'     IF Y IS ODD
190 X=X*2: Y=Y/2: GOTO 160'     IF Y IS EVEN
200 PRINT "THE ANSWER IS:"Z:PRINT:GOTO 120
```

Finally, here is the way to raise a number X to the Yth power. Again, let us create three registers `x`, `y`, and `z` and load them with the values X, Y and Z. (Note that now $Z = 1$ and not 0 as in the above multiplication method.) The relation between these three values is $X^Y \times Z$.

```
Is Y>0 ? Then Y is either even or odd.
If Y is even, replace Y by Y/2,
Now replace X by X*2
Leave Z unmodified, and note that equation 4 still holds,
as X^Y = (X^2)^Y/2*Z = (X^Y)*Z
Is Y odd? Then replace Y by Y-1
Leave X unmodified
Now replace Z by Z*X
Note that equation 4 still holds, i.e.,
X^Y=X^(Y-1)*ZX=(X^Y)*(X^-1)*ZX=(X^Y)*Z
And go back to the first step above until Y=0
```

Example: to calculate 2^3, $\mathtt{x} = 2, \mathtt{y} = 3, \mathtt{z} = 1$

```
is y=3 > 0 ? yes
is y=3 even ? no, then replace y=3 by (y-1)=2 and replace z=1 by z=2
is y=2 even ? yes, then replace y=2 by y=2/2=1 and replace x=2 by x^2=4
is y=1 even ? no, then replace y=1 by y=1-1=0 and replace z=2 by z=2x=2*4=8
is y=0 > 0 ? no, then stop!
Result = (2^3) = value of register Z = 8 (Answer)
```

I have modified the short BASIC program above to demonstrate what I have called the Northern Virginian Peasant Method:

```
100 CLS:PRINT "EXPON.BAS IS THE NORTHERN VIRGINIAN PEASANT"
110 PRINT"METHOD USED TO RAISE A NUMBER X TO A POWER Y."
120 PRINT:INPUT "WHAT IS YOUR FIRST NUMBER (0=EXIT)";X
130 IF X=0 THEN END
140 INPUT "NOW ENTER THE EXPONENT";Y
150 Z=1
160 IF Y>0 GOTO 170 ELSE GOTO 200
170 IF Y/2=INT(Y/2) GOTO 190'   IS Y EVEN?
180 Y=Y-1: Z=Z*X: GOTO 160'     IF Y IS ODD
190 X=X*X: Y=Y/2: GOTO 160'     IF Y IS EVEN
200 PRINT "THE ANSWER IS:'Z:PRINT:GOTO 120
```

We have already established that the reason why these methods are efficient is due to the fact that they exploit the properties of binary arithmetic. One only needs to perform as many steps N as the power of 2 required to cover Y, i.e., $N = log_2 Y$. Hence, since we always preserve the relationship in Eqn. 3 or Eqn. 4, these are not mathematical issues, but simply implementation ones.

Bibliography

• Beutelspacher, Albrecht. *Cryptology*, The Mathematical Association of America, 1994.

• Halsall, Fred. *Data Communications, Computer Networks and Open Systems*, Addison Wesley, London, 1992. pp.596-599.

• Kahn, David. *The Code Breakers*, Macmillan, New York, 1967.

• Simmons, Gustavus, (Editor). *Contemporary Cryptology*, IEEE Press, NY, 1992.

• Smith, John. Public Key Cryptography, *Byte Magazine*, January 1983, pp. 198-216.

• Zimmermann, Philip R. *The Official PGP User's Guide*, The MIT Press, Cambridge, Mass. 1995.

ACA CD-ROM Compilation

The ACA CD-ROM compliation has been updated and now contains more than 500 megabytes of cryptography-related material, all uncompressed and ready for use:

• The entire contents of the Crypto Drop Box, totalling more than 300 megabytes of programs, text files, and graphics gathered from the Krewe and other sources. This includes more than 170 megabytes of word lists, pattern lists, and dictionaries; Enigma and Hagelin machine simulators; implementations of PGP, TEX, and some programs from Bruce Schneier's book *Applied Cryptography*; statistics and analysis programs; and a number of other files;

• The entire contents of the ACA Bulletin Board system (Decode), including programs for running secure audio and protecting hard drives;

• A variety of dictionaries and word lists gathered from the Internet, split into two dozen directories for different languages and categories;

• Grady Ward's Moby Lexicon, including Moby Words, Moby Hyphenator, Moby Part-of-Speech, Moby Pronunciator, Moby Thesaurus, and Moby Language. According to the compiler,

> This includes SCRABBLE$^{(R)}$ lists, the most comprehensive rhyming dictionary in English, and a huge pronouncing dictionary only dwarfed by the million-entry thesaurus.

• *Jargon*, the Hacker's Dictionary containing relevant and historical entries for computer- and programming-related subjects totalling 1.2 megabytes;

• Turing emulators and a tutorial;

• Cryptanalysis programs and files, including crackers for Word, WordPerfect, and PKZip; and Postscript files describing various attacks on modern cryptosystems;

The CD-ROM comes in IBM-PC readable format and is available from:

```
Dan Veeneman
Post Office Box 2442
Columbia, Maryland  21045-1442
USA
```

Inquiries to `dan@decode.com`.

Tea and Sympathy

David Hamer (DAEDALUS)

`TEA` — The Electronic Alveary — and its companion program, `SYMPATHY`, were written by British crossword constructor Ross Beresford, of Bryson Ltd., to aid in the solution (`TEA`) and creation (`SYMPATHY`) of crossword puzzles of varying styles. While the latter may generate little or no excitement among cryptology enthusiasts, the former, together with its associated large (about 200,000 words/phrases) wordlist very well might !

`TEA` comes in two flavours - MS-DOS and MS-Windows. While they both operate in similar fashion, the Windows versions are much more powerful. The DOS version will no longer be supported or developed by Bryson and is freeware. The Windows 3.x and Windows 95 versions, (both v1.20, dated November 2, 1996) which are shareware and offer support to registered users, hold the promise of future development. The following description of the program will be limited to the Windows versions.

The program files come in compressed format as `TEAW1216.ZIP` (1419037 bytes) for Windows 3.x; `TEAW1232.ZIP` (1413955 bytes) for Windows 95. Each must be uncompressed into a temporary directory on the hard drive. After decompression you will have, in a little over 2 Megabytes:

```
ANSI.TSM              ANSIACA.TSM
ANSIACCE.TSM          ANSIALL.TSM
ANSICASE.TSM          BALTIC .TSM
CENTEURO.TSM          DEFAULT.TSM
ORDERING.WRI          README
TEA.EXE (Win 3.X)     TEA32.EXE (Win 95)
TEA.INI               TEA.HLP
TEAW12.DOC            TSBUILD.EXE
TSBUILD.HLP           TURKISH.TSM
UKACD14.DOC           WORDS.TSD
```

`README` gives detailed procedures for setting up `TEA` in both the Windows 3.x and Windows 95 environments.

The included default wordlist is the UK Advanced Cryptics Dictionary (v1.4, dated Feb. 11, 1996) which, in the words of `UKACD14.DOC`...

> ....original basis .....was a word list of some 500,000 entries..... reduced to a list of around 190,000 words by eliminating entries that aren't generally allowable in UK advanced cryptic [crossword] puzzles.

`TEA`'s usage is fairly straightforward and well supported by the help files. For example: entering `e.e..y` will produce `energy` and several other words with this pattern. Wild cards are acceptable. Anagramming is done by preceding the subject word or phrase with `;`: for example, `;diverse` produces `derives`, `revised` and `deviser`. With appropriately configured input, `TEA` will answer such questions as: "what words end in GRY?"; "what words include a Q, an X and a Z?"; "what words include all five vowels consecutively?"; and more!

The `UKACD` contains, in addition to pattern words, a large number of "pattern phrases" and the best way to illustrate this is by example. If you enter `;pigslitteran` — a concealed but anagrammed clue from a recent Times Crossword — `TEA` produces `ear-splitting`, which fits the clue perfectly! Note that `;pigslitter..` would produce the same result! Many other common (British) phrases are included. Word/phrase lengths from 2 to 31 are included.

A major feature of `TEA` is its ability to work with computerised dictionaries (I use the Concise Oxford Dictionary) also running under Windows 3.x or '95. The procedure here is to start the dictionary and minimise it in Windows. Then run `TEA` — with any of its search options — to look for the words under investigation. With the dictionary 'running' `TEA` has a "lookup" option so that you may mark a word and get its definition directly from the dictionary; then use any print, quotation or thesaurus options normally available with that dictionary. This feature is, of course, unavailable when using the DOS version.

Sources:

• `TEA` is on the Crypto Drop Box [**Ed:** See the next article for details on the CDB. DMV] as `TEAW1216.ZIP` and `TEAW1232.ZIP` — the Windows versions; and `TEA101.ZIP` — the MS-DOS version. Shareware registration costs about US$30.

• Ross Beresford may be found at:
e-mail: `ross@bryson.demon.co.uk`
URL: `http://www.compulink.co.uk/~bryson`
or by snail-mail:

```
Bryson Ltd.
10 Wagtail Close
Twyford
Reading RG10 9ED
United Kingdom
```

• The Concise Oxford Dictionary is part of *The Oxford Compendium*, a Compact Disc ROM which contains the *Concise Oxford Dictionary*, *The Oxford Thesaurus*, *The Oxford Dictionary of Quotations*; and *The Oxford Dictionary of Modern Quotations*, and is available for about US$60 from

```
Lotus Development Corporation
300 River Park Drive
North Reading, MA 01864
```

The Concise Oxford Dictionary alone is available on diskette for considerably less.

ACA AND THE INTERNET

There are a number of resources on the Internet of interest to members of the ACA. Besides the innumerable web pages devoted to cryptography, the following items are directly related to the ACA:

• The Crypto Drop Box (CBD) is a repository for programs, files, and other helpful crypto aids. It resides on a server at the University of North Dakota and is hosted there thanks to NORTH DECODER (Jerry Metzger). It may be accessed at the following Uniform Resource Locator (URL): `http://www.und.nodak.edu/org/crypto/crypto`

Questions about the CDB may be directed to `metzger@sage.und.nodak.edu`.

• DABASAP (Greg Griffin) is the contact point for the ACA Mailing List. A single electronic mail message sent to the list will be distributed to all of the list subscribers, providing a convenient means for keeping in touch and making collective use of the Krewe's knowledge and experience. To subscribe, send an e-mail message to DABASAP at `vlad@holonet.net` asking for current instructions.

About This Issue

This issue was produced on an IBM-compatible 486DX/66 under MS-DOS. The `vi` editor supplied with the Mortice Kern Systems (MKS) toolkit was used to enter and edit the text of a LaTeX-formatted file. EmTeX processed the `.TEX` file, with the result printed on a Hewlett-Packard LaserJet III*p*.

After five years and several thousand impressions my faithful LaserJet III*p* had to go in for service recently for a replacement fuser. I'm still not sure if it was old age or the aftereffects of a transparency that got wrapped around the fuser a few months ago, but $200 an an interminable delay later I'm able to print again.

I've also added an Iomega ZIP drive to the collection of computer equipment, joining the HP 4020i CD-ROM Writer as a means of transporting larger quantities of data. This was a big help in transporting the large download data sets from the Crypto Drop Box.
