A new approach to revealing the Byrne machine.

Introduction.

Up until now the methodology used has been to statistically analyse the plain and ciphertexts provided by Byrne and from that to draw conclusions on how the machine works. Over the course of many years, and with expenditure of much effort, our progress has been limited. Greg Mellen spotted what looks like enciphering blocks of 13. Jeff Hill noted that plain-cipher repeats occur very seldom at intervals less than 9. From that he inferred that the main cipher disk can only move short distances in order to avoid making a complete revolution in less than 9 letters. He postulated a random movement pattern of 1,2 or 4 places in an overall ratio of 2:1:1.

Moshe Rubin has pointed out that such movements do not completely preclude repeats at lower intervals than 9 and thus their absence is not fully explained by the Hill model.

So far nobody has been able to define, or tell us how to define, the key aspects of the machine: how many disks there are, what their movements are, what the mixed alphabets are nor what controls the movements of the disks. To get to this level of detail we must move on from the blunt instrument of statistical analysis to a finer method that is more precise. Also we must make fuller use of the extensive plain and ciphertexts that we have been given. In short we need a new analytical approach.

This paper outlines a possible way ahead that will use a computer program to develop the missing attributes mentioned above by a detailed and interactive analysis of the plain and cipher texts. This same program can also be used to test whether hypothetical machines would in fact produce Byrne's ciphertexts from their plaintexts.

Starting point.

I am going to start out with the initial letters of plain and ciphertext from Exhibit 1 to demonstrate manually the new analytical approach that will find the mixed alphabets and the movement key of the machine. Those who are interested can then program the process and make their own analyses.

I have to make three basic assumptions about the machine that may prove wrong and have to be revised. Such revision will not be a sign of failure but rather a homing-in on the correct assumptions. The process is Euclidian of putting up axioms and from them attempting to prove observations in the real world; if the proofs fail, the axioms must be changed.

My basic assumptions are:

1. the machine has two enciphering disks, each with a different mixed alphabet around its periphery. This assumption rests on Langen's reported comments.

2. disk 1 moves after each letter is enciphered. Disk 2 moves after 26 moves of disk 1. This assumption may well need to be fine tuned, especially the turnover.

3. the movements of disk 1 are either 1, 2 or 4 places. I borrow these movements from Jeff Hill, though not in the ratios he has mandated, purely as a starting point.

The analytical method.

I will first define an enciphering process for a 2-disk machine. Following Langen, each disk has a mixed alphabet around its periphery and is surrounded by a ring inscribed with the normal alphabet. I represent this as follows for two hypothetical mixed alphabets:

```
ring     abcdefghijklmnopqrstuvwxyz
disk 1   WHYJUMPFOLDINGBRACKETSQVXZ

disk 2   FROGSWALTZDUCKJIBVEXNYMPHQ
ring     ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

To encipher 'a': find 'W' below 'a' in disk 1 and then look for 'W' in disk 2. Below it find 'F' which is the encipherment. Plain 'a' = cipher 'F'. I call 'W' the **common letter** because it is the link between plain and cipher letters in the two mixed alphabets.

Having enciphered the first letter, disk 1 moves left by say 1 place to give:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     HYJUMPFOLDINGBRACKETSQVXZW
disk 2:     FROGSWALTZDUCKJIBVEXNYMPHQ
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Now 'l' enciphers to 'U'. And so the process continues, with disk 2 advancing after 26 letters have been enciphered.

With this enciphering process in mind, I can describe how the machine analysis is carried out.

1st cipher letter:
Take the first plain-cipher letters from Byrne's Exhibit 1, plain 'a' = cipher 'C'. We don't yet know the mixed alphabets so we don't know the common letter. We represent it with a symbol, for which I choose zero:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     0-------------------------
disk 2:     --0-----------------------
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Now disk 1 must move. We know it may move by 1, 2 or 4 places. The analytical process will start with a move of 1. If later this leads to conflict, it will return and try a move of 2. If that fails it will try 4. For the moment we try 1.

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     -------------------------0
disk 2:     --0-----------------------
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

2nd cipher letter.

The 2nd plain and cipher letters in Exhibit 1 are plain ‘l’ = cipher ‘L’. This time for the common letter we use the symbol 1 and enter it as follows:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     -----------1-------------0
disk 2:     --0--------1--------------
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Now we left shift disk 1 to get:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     ----------1-------------0-
disk 2:     --0--------1--------------
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

3rd cipher letter.
plain ‘l’ = cipher ‘Y’. We enter the symbol 2 appropriately as below:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     ----------12------------0-
disk 2:     --0--------1------------2-
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

And then left shift disk 1:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     ---------12------------0--
disk 2:     --0--------1------------2-
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

And so the process continues to encipher the next three letters ‘goo’ to 'TZP' in three more cycles to give:

```
ring        abcdefghijklmnopqrstuvwxyz
disk 1:     ---3--12----45------0-----
disk 2:     --0--------1---5---3----24
ring        ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The next plain letter is ‘d’ and it enciphers to ‘N’. We see there is now a conflict because in disk 1 there is already a symbol below plain ‘d’ in the ring. One of the earlier moves is wrong. Instead of moving 1 place, we should have moved 2 or 4 places. We don’t know where this mistake has occurred, but like Theseus in the maze of the Minotaur the computer program will go back to the last node, change the move and try again.

If you continue with this example you will find that the analysis indicates that one (or more) of the assumptions are wrong. Either the number of disks, or the disk moves allowed or the turnover period. The presence of a symbol above the 'Z' in ring 2 (see above) is a conflict for the 7th plain-cipher pair plain q-Z. The implication is that the turnover must occur before the 7th encipherment, requiring an adjustment to the initial assumption.

Using the analytical method in this way, I expect either to find the mixed alphabets and movement pattern used by Byrne, or to find (as above) that the process stops at an irreconcilable conflict. The latter would be a signal that one of the assumptions was wrong and it then would be a matter of changing assumptions and trying again.

I have summarized the algorithm below. It is in effect a Maze algorithm. The .cpp file of a prototype C++ implementation program is attached, which I have annotated freely and provided with multitudes of displays to show what the program is doing.

```
    MAIN SECTION
    make an array of disk 1 shifts allowed, index[3]= {1,2,4}
    get plaintext & ciphertext;
    fill the disk arrays with '.';
    set n=-1;
    set pointers for each letter =-1;

    START LOOP
    advance to next letter, n = n+1;
    find x= the position of the n'th plain letter in ring 1;
    find y= the position of the n'th cipher letter in ring 2;
    if position x is free in disk 1 & y in disk 2 there is a fit.
           Insert the symbol 'n' at x in disk 1 & at y in disk 2.
           increment pointer[n]
           if pointer[n] == 3, the number of moves available,
                 set flag=1 to go back 1 letter by implementing
                 procedure GO_BACK();
           else shift disk 1 by index[pointer[n]] & set flag=0.
    else
       there is no fit, so go back 1 letter by setting flag=1 to
       implement procedure GOBACK();

    while flag is equal to 1 implement the GO_BACK() procedure;

    return to start of loop.
-------------------------------------------------------------------------
    PROCEDURE GO_BACK()
         n=n-1;
         increment pointer[n];
         if pointer[n]== 3
               pointer[n]=0;
               flag=1;
               return;
         else
               move = index[this pointer]-index[last pointer]
               shift disk 1 by move;
               set flag=0;
```

Conclusion.

I hope this approach will enable a computer attack on the matched plain and cipher texts provided by Byrne, with a view first of finding the principles of operation of the machine and then recovering the keys - by which I mean the mixed alphabets and the disk movement key.

The task of explaining how the movement key is generated will still remain, but if the analysis can provide the first 50 integers then there is a good chance the generating system can be deduced.

As has been described, the variables to be evaluated are the movements that the disks can make, the turnover point, the number of disks and how they function to make the encipherment.