Sep 18, 2016 at 6:33pm

[Quote](https://cicada3301.boards.net/post/54/quote/33)[like](https://cicada3301.boards.net/thread/33/modular-exponentiation-ciphers-runes)

Post Options

**Post by mortlach on Sep 18, 2016 at 6:33pm**

**Modular Exponentiation Ciphers With Runes**  
A method to encrypt runes using exponentiation ciphers is demonstrated. These types of ciphers have many deep connections with the mathematics and computer science behind modern cryptography. The number of connected topics is huge, and ranges from the theory of integers, primes and congruences to multiplicative functions (such as Euler’s Totient Function) to the number of bitwise operations required to perform calculations on a computer and digital security. I cannot rightfully claim that The Liber Primus is a puzzle based on such concepts: it has not been solved yet! All I can say is there are many clues to this area of investigation in previous puzzles, and all public attempts at decrypting have so far been unsuccessful.  
  
First a simple scheme that does not work properly for runes will be explained, then a fix that makes it workable for runes is added and developed.  
  
**Example 1:**  
Choose an odd prime, P, and an integer e such the greatest common divisor (gcd) of the two is 1:

gcd( P, e ) = 1

The gcd of two numbers is the largest positive integer that divides the numbers without a remainder. For example gcd(8,12) is 4. (If P and e are both primes then gcd(P,e)=1 is guaranteed.) We will chose P = 3301 and e = 29. Translate the runes in your message-text (MT) to numbers, (e.g. use their forward position in the Gematria):

MT = A,N,I,N,S,T,R,U,C,T,IO,N,C,O,M,M,A,N,D,Y,O,U,R,O,W,N,S,E,L,F,F,O,R,A,L,L,I,S,S,A,C,R,E,D

= 24,9,10,9,15,16,4,1,5,16,27,9,5,3,19,19,24,9,23,26,3,1,4,3,7,9,15,18,20,0,0,3,4,24,20,20,10,15,15,24,5,4,18,23

Element i of the cipher-text (CT) is calculated using the following formula:

CT(i) = ( MT (i)^ 29 ) % 3301

giving:  
  
CT = 2120, 1686, 1892, 1686, 1035, 1595, 774, 1, 2624, 1595, 1055, 1686,2624, 1471, 2192, 2192, 2120, 1686, 756, 93, 1471, 1, 774, 1471, 2943, 1686, 1035, 2997, 861, 0, 0, 1471, 774, 2120, 861, 861, 1892, 1035, 1035, 2120, 2624, 774, 2997, 756  
  
(It is interesting to note the 0,0 where message has F,F) To decipher CT we need to know the deciphering key which is an integer, d where:

d \* e = 1 % (P-1) d is an inverse of e % (P -1)

By ensuring gcd( P, e ) = 1 we ensure such a d exists. We can use d = 569 as 569 \* 29 = 16501 = 1 % 3300 Next we raise CT to the d’th power modulo P and recover the message:

( CT(i) ^ 569 ) % 3301 == MT(i) is true

This works well, however, in the Liber Primus we do not have CT numbers such as 2120, 1686, 1892, etc. we have runes, symbols in base 29. If we take CT modulo 29 and try the same procedure it does not work:

CT % 29 = 3, 4, 7, 4, 20, 0, 20, 1, 14, 0, 11, 4, 14, 21, 17, 17, 3, 4, 2, 6,21, 1, 20, 21, 14, 4, 20, 10, 20, 0, 0, 21, 20, 3, 20, 20, 7, 20, 20,3, 14, 20, 10, 2

(CT % 29) ^ 569 ) % 3301 = 1931, 1870, 1818, 1870, 2432, 0, 2432, 1, 2547, 0, 1278, 1870, 2547, 1595, 1363, 1363, 1931, 1870, 1848, 107, 1595, 1, 2432, 1595, 2547, 1870, 2432, 1316, 2432, 0, 0, 1595, 2432, 1931, 2432, 2432, 1818, 2432, 2432, 1931, 2547, 2432, 1316, 1848 (and blatantly != MT)

**Example 2:**  
One way to recover from the problems in Example 1 is to choose P = 29. Then everything works fine, let:

P = 29, e = 3301, d = 9 (as 9 \* 3301 = 29709 = 1 % 28 )

    giving:

CT(i) = ( MT (i)^ 3301 ) % 29 = 16, 22, 27, 22, 8, 25, 5, 1, 13, 25, 18, 22, 13, 14, 2, 2, 16, 22,

20, 15, 14, 1, 5, 14, 23, 22, 8, 10, 7, 0, 0, 14, 5, 16, 7, 7, 27, 8, 8, 16, 13, 5, 10, 20

CT(i) ^ 9 % 29 == MT(i) is true

**Adjustments to Match Frequency Distribution of the Liber Primus**  
There is one more adjustment that needs to be made. CT using the above method does not match the [frequency analysis of the Liber Primus](http://imgur.com/D7wpFZG) CT and [here](https://cicada3301.boards.net/thread/33/modular-exponentiation-ciphers-runes).   
  
There are two simple ways to fix this, the first is to find encryption keys, e, that when used to encrypt naturally give a low number of repeats and relatively uniform index of coincidence, (for example e could be a list of suitably chosen primes…). [This method has been shown to work for other encryption algorithms](https://cicada3301.boards.net/thread/33/modular-exponentiation-ciphers-runes)  
  
We will not show a suitable e here, but assume one can be found.  
  
Another method is to modify the algorithm as [discussed elsewhere](https://cicada3301.boards.net/thread/33/modular-exponentiation-ciphers-runes):  
  
  
First choose a shift of the Gematria such that a low frequency rune in the message has a value of 0, then when encrypting add the previous CT rune, giving the resultant CT a “memory” and naturally low numbers of repeated runes:  
  
    If i == 0  
        CT [ 0 ] = MT[0] ^ e[0] ) % 29  
    else  
        CT(i) = ( MT(i) ^ e(i) + CT( i-1 ) ) % 29  
 **Comments**  
I have tried to outline a simple implementation of a workable, rune-based, modular, exponentiation cipher. As with all methods there are many different and interesting variations. For example, using the rune-prime-equivalents for e gives a (mostly) workable solution, with just a few collisions. When a collision happens there is more than 1 rune that can encrypt to the same CT, these can generally be spotted by eye and are easy to correct as ~95 % of the message is generally readable.  
Types of e that could be used are the rune-prime-equivalents of the message, the cipher text, a keyphrase, etc. etc. etc.  
  
\*Comments, questions, suggestions, omissions etc ? please try [#cicadasolvers](http://webchat.freenode.net/?channels=cicadasolvers)  
  
MSGA