I’m a little late posting. I took it upon myself to write this process into C++ code. My findings were interesting and showed several vulnerabilities still present in this encryption method.   
  
First, this approach doesn’t account for linguistic analysis, such as the most common letters. This approach is used in the TV show Wheel of Fortune, where players guess the most common letters used in the English language to reduce the number of guesses needed to solve the puzzle. This implies that there needs to be salting or an additional layer of obscurity.   
  
Secondly, the letter “a” has the value 1, which has a unique property. It raised to any power still yields 1. This means this character is exposed reaffirming the need for additional augmentation to the pre-encrypted text.

Lastly, some characters won't be changed with low-value prime numbers like 3 and 11. there is an issue that is overlapping for small values of “n”. This is because “n” represents the range of values we can make our subgroups of values to map to symbols. These groups must result in an injective and subjective function. Small values of n limit the potential outcomes of the mod operations which breaks the function requirement because values can be greater than “n”. However, this is where I throw my hands up in defeat. I have run out of time and energy (about 8 hours) to figure out how to reconcile the incongruent results on paper compared to my code without compromising reliability and security.