RSA Encryption

To start the project you should read Section 1.5 of the text book carefully. You are given an encrypted message below and your job is to decode it.

Before trying to decode the message, you are to state a theorem which generalizes Theorem 1.28 to include instead of just two primes p and q, any finite number of primes, say $p_1, p_2, \cdots, p_j$. You are then to prove the generalized theorem.

Now, after you understand the RSA encryption technique, write a few paragraphs explaning how it works. You should be sure to include what numbers are kept secret and what numbers are not.

Next, suppose that you intercept a message that Joe is sending to Mary. You know that to encode a message to Mary you use the numbers $k = 109283753$ and $n = 2357879497866699967$. Based on recent techniques to factor large numbers, you know that Mary's numbers are small enough that it is feasible to break her code. It is your job to do just that with the use of Mathematica or Maple. You also know that the way Mary converts a message into a number is to replace A with 01, B with 02, C with 03, and so on through Z with 26. Furthermore, a space is converted to 27, a period is converted to 28, a question mark is converted to 29, and an exclamation mark is converted to 30. So for example the message "HI." becomes 080930. If you write this number as 80930, you need to put a zero in front since the number of digits must be even. So, to send Mary a secret message, you first replace the message with a number and then use the scheme described in the book. When the message is long, it is just broken into smaller parts and each part is coded seperately. Using Mathematica, Maple, or your calculator, your job is to decode the following message consisting of the three numbers below. After decoding, remember to concatenate the three messages together to form one message. Each of the three decoded messages should have an even number of digits.

246919062243423957
1673829235453550548
1794946576363047813

If you are using Mathematica, then there are a few functions that help. Use the help menu to determine how to use the functions **FactorInteger** and **PowerMod**. Note that in PowerMod, you can use negative exponents.