PKC and RSA Exercises

Confidentiality *and* Integrity. Consider a generic public key system used by Alice and Bob. Alice has public and private keys PU_A and PR_A; Bob's public and private keys are PU_B and PR_B. Alice wants to send a message *M* to Bob such that it is confidential (only Bob can read it) and with integrity (only Alice could send it and the message hasn't been modified). Explain how Alice can send such a message using *only* the public key algorithm (e.g. no hash). How would bob decrypt the message and verify the integrity of the received message? Express your answer in terms of the message *M*, the encryption function E(), decryption function D(), and Alice and Bob's public and private keys.

Why use PKC. Consider an *n*-user network in which each user must be able to communicate confidentially with any other user. Assume the network does *not* use PKC. How many distinct symmetric keys will be required? If the network did use PKC, how many distinct public/private key pairs would be required?

Small RSA Computations. Perform encryption and decryption using the RSA algorithm for the following:

a. p = 5, q =11, e = 3, M = 9.
b. p = 7, q = 11, e = 17, M = 8.
c. p = 11, q = 13, e = 11, M = 7.
d. p = 17, q = 31, e = 7, M = 2.
e. You intercept the cipher text *C* = 10 sent to a user whose public key is e = 5, N = 35. Decrypt the cipher text.
f. The public key of a particular user is e = 31, N = 3599. What is the user's private key?

Medium RSA Computation. You intercept the encrypted message C = 4176229917282169 to a user with public key e = 65537, N = 19915121917840759. Decrypt the message (the answer will be readable text). The functions used to convert a string to an integer and vice versa are the same as those used in `rsa_example.py`. Hint: Wolfram Alpha can factor moderate-sized integers.

"Bad" plaintext blocks for RSA. You learn that the *plaintext* of an RSA-encrypted message has a factor in common with *N*. How can you use this information to break the system?