**Homework 3**

1. Read 117-128
2. (1 pts) Explain the purpose of the salt in the Unix password hash

3. (2 pts) Explain the threats in the following types of authentication:

   Alice sends password, Bob compares it against database of passwords

   Alice sends password, Bob hashes it and compares it against database of hashed passwords

   Alice computes hash of password, uses it as secret key in challenge/response protocol

   Alice computes hash of password, sends it to Bob, who hashes it and compares against a database of doubly-hashed passwords.

   Lamport’s hash

4. (3 pts) How about using more chars of the Unix pwd hash by encrypting 0 with the 1st 8 chars, then encrypting the output of that with the next 8 characters. Would this be a good scheme?
5. (2 pts) In problem 4., would the weakness enable you to find the user’s password?
6. (1 pt), chapter 5, problem 1. Would signing a message mod n rather than taking a hash of it and then signing the hash be secure?
7. (2 pts) problem 4: Suppose we create a 128-bit hash out of a secret key 64-bit block cipher by taking the blocks of the message (m1, m2, ... mi) and, starting with a constant, say 0, encrypting first with m1, then with m2, ... then with mi to get a 64 bit value, and then taking the blocks in the opposite order (encrypting 0 with mi, then mi-1, ... then m1) to get the other 64 bits of the hash. How can you, in $2^{32}$ operations, find two messages that hash to the same value?
8. (4 pts) Chapter 5, problem 14 find minimal sufficient conditions for x, y, and z that would make the following functions random:

   -x
   x XOR y
   x or y
   x and y
   (x and y) or (-x and z)
   (x and y) or (x and z) or (y and z)
   XOR (x, y, z)
   XOR (y, (x or -z))

9. (2 pts) Why do you need more bits of message digest if you want to make sure it’s infeasible to come up with two messages with the same digest, rather than if the property you want is to make sure you can’t come up with a message (or another message) with a particular digest?
10. (1 pt) Problem 5-18: Show how to do decryption with the scheme in 5.2.3.2
11. (3 pts) Problem 5-19. Change the pad in previous problem from MD(K,c_{i-1}) to MD(K,p_{i-1}). How would this be decrypted? Is it as secure?
12. (2) Show how to distribute a key via multicast to n parties with a message of length O(log n) when a member is removed from the group. Assume each party has an individual key it shares with the key distributor node (and other state, which you will describe). What does each member need to know? What would the key distributing message look like in the case where a member is removed from the group?
13. (2) How can one change the key efficiently when a new member joins?
14. (2) Describe how to digitally sign a very large message, such that any block can be modified and the entire file can be efficiently re-signed (without needing to digest the entire file).