1. Diffie-Hellman questions
   1. YA = αXa = 75
   2. YB = αXb = 712
   3. The shared secret = αXaXb = 7(5)(12)
   4. α is specifically chosen for having the property of being a primitive root of modulus q. Without this property, there is no guarantee that the logarithm based x and mod q with will exist. If this doesn’t exist then the algorithm is moot.
2. Hashing Questions
   1. We assume that the attacker can induce senders to sign messages. The attacker would then generate a set of 2m/2 valid messages with similar enough meanings. The attacker would then generate a set of 2m/2 fraudulent messages that mean what the attacker wants to trick the receiver into believing. Now, because of the nature of the birthday paradox, there is a greater than 50% chance that there is a valid pair of generated message/frauds. The attacker than has the sender sign the semantically similar half of the message/fraud pair and does whatever he pleases.
   2. To process both the 2m/2 semantically similar message and the 2m/2 fraudulent messages should require processing (2m/2)(2m/2) = 2(m/2)+(m/2) = 2m messages/fraud pairs.
   3. Considering that an attack would be generating 2m message/fraud pairs and m=64 in this system. At that speed, an attack would take 264/220 = 244 seconds.
   4. In a 128 bit hash. The memory usage should be 2128 message/fraud pairs that would take 2128/220 = 2108 seconds to hash.
3. Working under the assumption that we’re supposed to use Merkle-Hellman Knapsack
   1. Generate public key {5, 9, 21, 45, 103, 215, 450, 946} \* 1019 mod 1999 = {1097, 1175, 1409, 1877, 1009, 1194, 779, 456}
   2. Encrypting ’0101 0111’ gives C = 1175 + 1877 + 1194 + 779 + 456 = 5481
   3. To continue we need a-1 mod 1999. We must solve a = 1 mod 1999. Below we find that 1 = [1999] 209 - (1019) 410. Which means 1 mod 1999 = (1999) 209 - (1019) 410 mod 1999. 1 mod 1999 = (1019) -410 mod 1999. Thus a-1 = -410 mod 1999 = 1589 mod 1999.
      1. 1999 = (1019) 1 + 980
      2. 1019 = (980) 1 + 39
      3. 980 = (39) 25 + 5
      4. 39 = (5) 7 + 4
      5. 5 = (4) 1 + 1
      6. 1 = 5 - (4) 1
      7. 1 = 5 - (39 - (5) 7) 1
         1. 1 = 5 - (39 - (5) 7)
         2. 1 = 5 - 39 + (5) 7
         3. 1 = 5 + (5) 7 - 39
      8. 1 = (5) 8 - 39
      9. 1 = (980 - (39) 25 ) 8 - 39
         1. 1 = ([980]8 - (39) 200 ) - 39
         2. 1 = [980]8 - (39) 201
      10. 1 = [980]8 - (39) 201
      11. 1 = [980]8 - (1019 - (980) 1 ) 201
          1. 1 = [980]8 - (1019 - (980) 1 ) 201
          2. 1 = [980]8 - ([1019] 201 - (980) 201)
          3. 1 = [980]8 - [1019] 201 + (980) 201
          4. 1 = [980]8 + (980) 201 - [1019] 201
          5. 1 = (980) 209 - [1019] 201
      12. 1 = (1999 - (1019) 1) 209 - [1019] 201
          1. 1 = [1999] 209 - (1019) 209 - [1019] 201
          2. 1 = [1999] 209 - (1019) 410
      13. 1 = [1999] 209 - (1019) 410
   4. To decrypt we multiply the ciphertext with the inverse of a. C \* a-1 = 5481 \* 1589 mod 1999 = 1665
   5. Decompose the value into elements of the private key set. 1665 = 946 + 450 + 215 + 45 + 9
   6. Match the decomposition with elements in the private key set, 1 if the number is present; 0 otherwise and we get P = 0101 0111