**PART 1**

1. FBI is not able to get the information that they feel is relevant for the crime committed. FBI is asking Apple to create a backdoor for a particular phone (from the San Bernardino shooting) and ask Apple to open that phone, but Apple wants to keep its security. More specifically, FBI is asking Apple to create a special version of their operating system that eliminates the brute force attack and install it on the San Bernardino iPhone. Apple declined to create a software that would allow to bypass the phone’s security because it compromises their security features in their products. FBI responded to Apple’s denial by issuing a court order mandating that Apple create a new iOS software that could bypass the auto-erase function (where the phone deletes the information stored after 10 incorrect guesses). A few days later FBI dropped the hearing after announcing that they were able to unlock the phone in question
2. The FBI is working towards preventing crime. By unlocking this phone, it could potentially lead to more arrests and fewer terrorist crimes. The information held on the phone can potentially lead to more arrests and potentially tracking down other terrorists who would have been involved in this case, or any future attacks. The FBI could get information as to anyone else the shooter may have had contact with prior to the shooting, and this information can definitely help in preventing future terrorist attacks. Apple is trying to protect the information and privacy of its customers. By creating this backdoor that the FBI requests, it doesn’t stop in the San Bernardino case, and not necessarily at any other terrorist attack. You give them an inch and they take a mile, and they can use this new software to penetrate civilians phones and obtain information because it won’t just stop at this case. This software can be used again and again with no restraint from the law. Not only will the FBI have this software but it also makes it easier for hackers to gain access to people’s personal information on their phones and steal personal data.

**PART 2**

1. Oscardan = 01110 10010 00010 00000 10001 00011 00000 01101. C1 = **10100 00000 00110 01011 00101 10100 00101 11111**
2. C1⊕ K:

10100 00000 00110 01011 00101 10100 00101 11111

⊕ 11010 10010 00100 01011 10100 10111 00101 10010

01110 10010 00010 00000 10001 00011 00000 01101

1. C2 =

10100 00000 00110 01011 00101 10100 00101 10101

11010 10010 00100 01011 10100 10111 00101 10010

01111 10010 00010 00000 10001 00011 00000 00111

As expected, only the first 5 bits changed, but the only difference is the first 5 bits changed from 01101 compared to 00111 which according to MyCode it changes the N to a H.

1. This does not lead to an increase in security. A cipher with perfect secrecy doesn’t benefit from multiple enciphering’s the way this question is done. Unless we are assuming ideal conditions where the one-time pad is kept secrecy, used only once and is generated in a truly random way

**PART 3**

1. **K1 = 12, K2 = 13, X1= 56, X2 = 23**

Y2 = X1 XOR F(12,23)

F(12,23) = (12+23)2 mod 64 = 9

Y2 = 56 XOR 9 = 49

Y1 = 23, Y2 = 49

Y2 = Z2 = 49

Z1 = Y1 XOR F(13,49)

F(13,49) = (13+49)2  mod 64 = 4

Z1 = 23 XOR 4 = 19

Z1 = 19 Z2 = 49

1. Z1 = 19, Z2 = 49

Z2 = Y1 = 49

Y2 = Z1 XOR F(49,13)

F(57,13) = (57+13)2 mod 64 = 3844 mod 64 = 4

Y2 = 19 XOR 4 = 23

Y2 = X2 = 23

X1 = Y1 XOR F(23,12)

F(23,12) = (23+12)2 mod 64 = 1225 mod 64 = 1

X1 = 57 XOR 1 = 56

X1 = 56, X2 = 23 as the original plaintext

1. Z1 = 19 = 10011

Z’1 = 10010 = 18

Z2 = 49

Z2 = Y1 = 49

Y2 = Z’1 XOR F(K2, Z2)

F(K2, Z2) = (13+57)2 mod 64 = 4

Y2 = 18XOR 4 = 22

Y2 = X2 = 22

X1 = Y1 XOR F(K1, Y2)

F(K1, Y2) = (12+22)2 mod 64 = 4

X1 = 49XOR 4 = 53

X1 = 53 and X2= 22

1. Only 3 bits in the plain text remained the same as the original plaintext. Compared to question 2.3 where 7 bits remained the same. This is definitely better for encryption then the previous one since this changes the value of both plaintexts whereas the other only has the effect on 1 of the values
2. Let X1 = 1, X2 = 2, K1 = 12, K2 = 13. We can calculate the sequence:

1:

Y1 = 2

Y2 = 1 XOR F(2,12) = 4

Z2 = 5

Z1 = 2 XOR F(5,13) = 2

2:

Y1 = 5

Y2 = 2 XOR F(5,12) = 19

Z2 = 19

Z1 = 5 XOR F(19,13) = 1

3:

Y1 = 19

Y2 = 1 XOR F(19,12) =0

Z1 = 19 XOR F(0,13) = 58

Z2 = 0

Because we the numbers (2,5), (1,19), (58,0) we could pick either number at each iteration

**PART 4**

1. Round 1

Y1Y2 = IVX1

Z1 = (Y12 + Y22 + 11 ) mod 15 = 6 = 0110

Round 2:

Y1Y2 = Z1X2

Z2 = (Y12 + Y22 + 11 ) mod 15 = 8 = 1000

Round 3:

Y1Y2 = Z2X3

Z3 = (Y12 + Y22 + 11 ) mod 15 = 1 = 0001

Round 4:

Y1Y2 = Z2X4

Z3 = (Y12 + Y22 + 11 ) mod 15 = 7 = 0111

Therefore our last hash value is 0111

Part 1:

Y1Y2 = IV1

Z1 = (Y12 + Y22 + 11 ) mod 15 = 12 = 1100

Part 2:

Y1Y2 = Z1X2

Z1 = (Y12 + Y22 + 11 ) mod 15 = 11 = 1011

Part 3:

Y1Y2 = Z2X3

Z3 = (Y12 + Y22 + 11 ) mod 15 = 13 = 1101

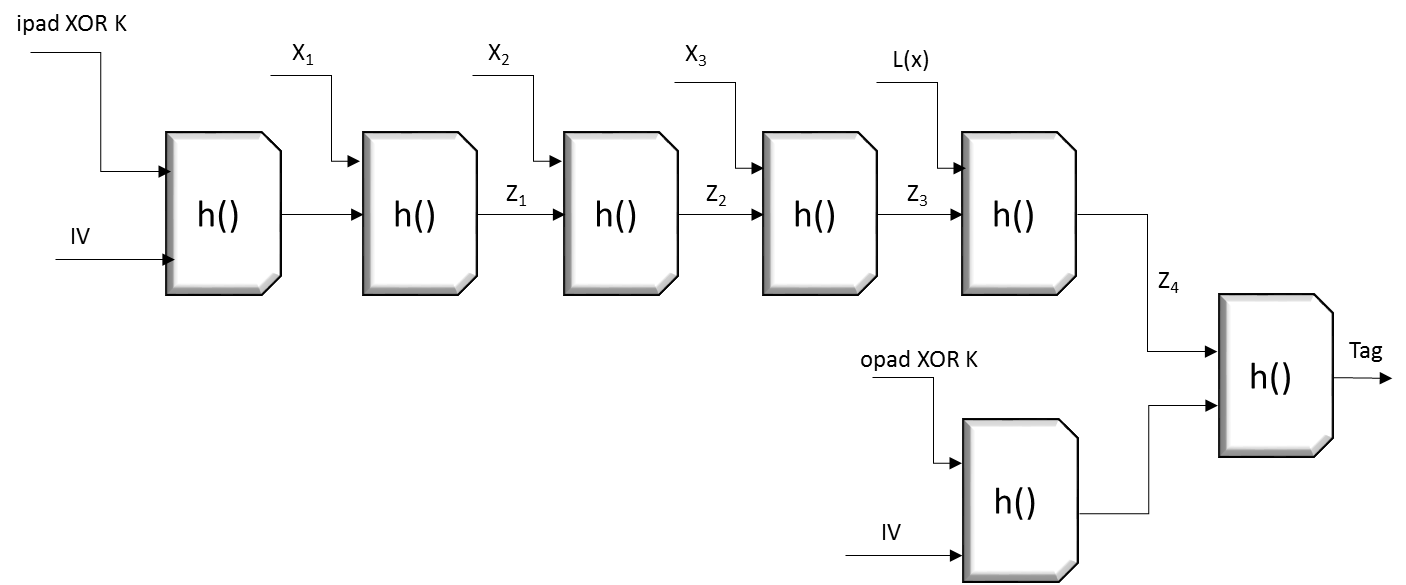
Part 4:

Round 4:

Y1Y2 = Z2X4

Z3 = (Y12 + Y22 + 11 ) mod 15 = 10 = 1010

Therefore our last hash value is 1010. This hash has a 3 bit difference from the hash of 1.



k XOR ipad = 0011

k XOR opad = 1100

Part 1:

Y1Y2 =IV(k XOR ipad)

Z1 =(Y12+Y22+11) mod15 =5 = 0101

Part 2:

Y1Y2 = Z1X1

Z2 =(Y12+Y22+11) mod15 = 1 = 0001

Part 3:

Y1Y2 = Z2X2

Z3 =(Y12+Y22+11) mod 15 = 3 = 0011

Part 4:

Y1Y2 = Z3X3

Z4 =(Y12+Y22+11) mod 15 = 6 = 0110

Part 5:

Y1Y2 = Z4X4

Z5 =(Y12+Y22+11) mod 15 = 12 = 1100

Part 6:

Y1Y2 = IV(k XOR opad)

Z6 =(Y12+Y22+11) mod 15 = 5 = 0101

Part 7:

Y1Y2 = Z6Z5

Z7 =(Y12+Y22+11) mod 15 = 0= 0000

We end up with tag = 0000

5 a.) Using an xor operation between tinyHMAC(k1,m) and tinyHMAC(k2,m) would be more efficient since it uses less numbers of extra bits to send each message. However with less bits the security would be less than the concatenation option. If we are using just the xor operation then the success chance would just be the total number of bits used for the mac, in this case it would be 1/16.

b.) using a concatenation instead of an xor operation would have better security of each message but this option sacrifices efficiency since more bits need to be sent. Therefore if large messages are sent, the messages could pile up. By concatenating it increases the number of bits and having a 1/256 probability of replicating the message

1. It would take at least 40 bits hash in order for it to be hash pre-image resistant to 40 bit security. This is because each bit has a possible of 2 elements, 0 or 1. Thus in order to have a success chance of 2^-40, it would be the same as saying having 40 bits each with 2 settings. Thus bringing the total of possibilities to 2^40. And since there would be only 1 correct hash, thus 1/ 2^40 is the success chance.