Privacy Enhancing Technology

Takang Kajikaw Etta Tabe (s2054752), Michiel Doesburg (s2180545, 4343875)

May 14, 2018

Question 1

1a

See the code included in the ZIP file.

1b

i. A MAC is used to guarantee integrity over communication. If no MAC is used, a third party can modify a sent message (even if it is encrypted) and thus sabotage communication. The fundamental idea behind mix-nets however, is to protect anonymity. The symmetric key is sent along with every message, encrypted by each mix-net's public key, there is no stateful shared secret between communicating parties, making a MAC meaningless. If you want to spoof another party, you can just send a message of your own.

ii. Since the IV is encrypted with the public key of the mix-net, the eavesdropper is not able to see the IV. Therefore using a constant IV does not make the eavesdropper able to link your messages together. If you are the only one using this constant IV, it does make the mix-net able to link your messages together, since it will see your IV.

1c

The shared threshold is eleven. This is easily deduced from the fact that messages show up in the exit log in batches of eleven, all with the same time-stamp. Meaning the last mix-net released eleven messages at a time. Since the threshold is shared, mix 1 and mix2 also have a threshold of eleven.

Question 2

2a

To get anything done we first needed to determine the threshold of mix 3 through trial and error. By sending a different number of messages at once we noticed that messages ended up in the exit log in batches of 12, 24 and 39. Since the only common factor of these three numbers is 3, the threshold of mix 3 must be 3.

After trial and error we found that 14 messages is the smallest number that produces any output. This means that both T1 and T2 must be 14 or less.

The first point at which all messages are released and no more messages remain in the mix net is at the least common multiple of T1, T2 and T3. We can find this multiple by incrementally sending messages and seeing what the first point is where no messages remain in the network. Using the code below, we found this point to be 78.

```
amount_of_lines = -1
amount_of_msgs = 5
```

```
while amount_of_lines != amount_of_msgs:
    send_msgs(1)
    amount_of_msgs = amount_of_msgs + 1

    time.sleep(.4)
    amount_of_lines = -1
    for line in urllib.urlopen(exitlog):
        amount_of_lines = amount_of_lines + 1

    if amount_of_msgs == amount_of_lines:
        print("found: " + str(amount_of_msgs))
```

Since T1 and T2 need to divide 78 and are both 14 or less, that leaves us with the following options: 1, 2, 3, 6, 13.

The first batch of messages only makes it through at the 14th message, meaning that one of T1 or T2 has to be 13 (since any combination without a threshold of 13 would release messages earlier).

$$C1 : T1 \vee T2 = 13 \tag{1}$$

If T1 were 13 then the thirteenth message would trigger a flush in mix 1. Consequently, a flush would also be triggered in mix 2 and mix 3 since they both have thresholds of 13 or lower. But since no output is presented before the fourteenth message, T1 cannot be 13.

$$C2 : \neg(T1 = 13) \tag{2}$$

From C1 and C2 we conclude that T2 is 13. At this point T1 must be 2, since no other threshold could let the fourteenth message produce the output. Thus we have

$$T1 = 2, T2 = 13, T3 = 3 \tag{3}$$

2b

By continuously sending messages to the mix-net at maximum capacity, the thresholds for mix 1 and mix 2 become irrelevant (since they process messages instantly), and the time after which mix 2 flushes can be easily deduced from subtracting the time-stamps of two batches of messages. After the flush timing delay of mix 2 has been found, you can start looking at how many messages it takes for no message to be left in the left, i.e. at the point where LCM(T1, T3) equals the number of messages sent. This is simply done by sending a number of messages within the flush delay of mix 2, and seeing if any messages remain in the mix-net. You can also figure out what the minimum amount of messages is that triggers any output by doing the same process. On top of that you can figure out the threshold of mix 3 by the same process, and seeing how large the batches are that are released. The GCD of all these batch sizes will be the threshold of mix 3.

Now all that's left is to figure out the threshold for mix 1. Knowing that T1 must divide LCM(T1, T3) and knowing an upper bound on what T1 is by finding the minimum amount of msgs to trigger output, there won't be many options left. You can try out these options and see when the corresponding output matches (considering T3's threshold), to figure out T1.

Question 3

3a

For the announcement (1, 1, 0, 1, 0, 1), the bit communicated is 0. This comes from the fact that XORing all the communicated bits from each participant will result in 0.

3b

i

No they can not tell. Since the colluding nodes A and E are on both sides of F, they know bits x_5 and x_6. The bit that F will send out is equal to

$$z = x_5 \oplus x_6 \oplus m \tag{4}$$

If x_5 xor $x_6 = 0$ then the bit F sends out will be 0, regardless of if it xor'ed with its message of 0 (since 0 xor 0 = 0). So A and E cannot tell if F was the sender. If x_5 xor $x_6 = 1$, then F will send out 1, regardless of if F xor'ed with his message 0 (since 0 xor 1 = 1). In both cases F will send out the same bit regardless of it is xor'ed with a message of 0.

ii

Not always. Although colluding nodes are connected to 2 out of the 3 edges of node D, Node C is also connected to one of the edges of Node D. The bit which is sent out by D is equal to:

$$y = x_7 \oplus x_3 \oplus x_4 \oplus m \tag{5}$$

Since A and E both know what x_7 XOR x_4 equals but do not have knowledge of x_3, they will not be able to tell if D was the sender of a message in the case where x_3 is zero. If x_3 is 1, then there might be a bit flip but this will be known since x_7 and x_4 are known by A and E.

If x_7 XOR $x_4 = 0$, then the bit D sends out will be dependent on the bit of x_3 which can also be of bit 0 or 1. If it is of bit 0, then the XOR of the 3 0's and that of the message is also 0 but if x_3 is 1, then the output of the XOR of x_3, x_4, x_7 and m will be 1, which can cause A and E to notice the bit flip.

3c

i

Yes A and E can tell, since this is the exact opposite of the scenario in question 3bi. Now the bit that F sends out will always be the opposite of what $t = x_5$ xor x_5 is: if $t = 0$ then F will send out 1, if $t = 1$, then F will send out 0. Thus A and E can tell F has sent a message of 1.

ii

Not always. The argument made in question 3bii still holds. Although the colluding nodes are connected to 2 of the 3 edges of D, x_3 also has a determining power. Since the message bit is 1, then having the XOR of x_7 and x_4 as 1 means an x_3 value of 1 leads to an output bit of 1 and having an x_3 value of 0 leads to an output bit of 0. Since the colluding nodes could observe changes in the output bit based on the value of x_3, then they can sometimes tell if it the message was from D and other times when x_3 does not lead to a bit change in the sent bit, then it becomes impossible to tell who the sender is.