Privacy Enhancing Technologies, assignment 1

Fedor Beets, s1227874
Robin Pronk, s0138746

May 11, 2016

1 Question 1

1.1 1a

See Assignment1.java. To make this work add the bouncycastle (bcprov...) as library, which is include in our handed in file.

1.2 1b

We used function runAssignment1B for this.

1.3 1c

The shared thresshold for this mixnet is 7.
Run the mixnet for some time without sending own messages, download the client and cache node files. Look at the timestamps of the messages. The clients continuously send messages to the mixnet. However batches of 7 messages are received by the cache node within 0.15 seconds of each other, after this there is a waiting period of several seconds. Furthermore 7 is a prime number so it cannot be created by consecutive bursts of messages other than if the threshold was 1. The threshold is not 1 as otherwise messages would be sent more frequently, this can be verified by looking at the client logs. The threshold of all 3 mixers is 7.

2 Question 2

2.1 2a

We used function runAssignment2A and runAssignment1A for increments for this.

Start a fresh mixnet and then send one message at a time waiting for something to appear on the cache node. That amount of messages will be the maximum combined threshold value for all three nodes, but does not have to be a threshold value. The amount of messages appearing at the cache node will be the threshold value of the last mix node. This gives a threshold for the last mixer of 6 and the combined maximum value is 8.

Next we will increase the amount of message sent, all containing a number so we can identify the message. After 36 messages we got 30 messages on the cache node, which is five times its threshold. This output contains message number 35 but never 36. At this stage the last mixer has just been flushed but can still hold new messages. The third mixer can only get incoming messages, causing it to flush, when the second mixer flushes its bucket into to third. The same holds for second mixer with the first mixer flushing. However the first mixer only receives one message at a time from us so after it flushes it is empty as long as we don't send anything new.

So at this moment the first mixer is empty and the other two may contain more messages. This means the second and third mixer together contain 6 (36-6) messages. Earlier runs show that the first mixer also flushes on 28, 22, 14 and 8. The threshold of the first mixer thus has to be a common divisor for 36, 28, 22, 14 and 8, which is 2.

As message 35 is the highest one out (many runs used to beat the odds of mixing), number 36 is held by mixer two and not by one, which is empty, or three, otherwise message 36 should be mixed in by mixer two sometimes. We find an equal case on 28 in, 24 out with 23 being the highest number in the output. So mixer two flushes at 35 and 28, so its threshold is a common divisor of 35 and 28, namely 7.

To conclude:

- nA = 2
- nB = 7
- nC = 6

2.2 2b

This answer is under the limitation that we can only provide input to 1 mix node, and that there is always output from only the last node. Furthermore a side-channel attack on the timing, to see the difference between messages being buffered up in the last node or messages being flushed by the first or second node and then directly between the third node is also not considered feasible.

To distinguish the thresholds under these conditions the set of input messages combined with the set of output messages for each input must be unique for every possible threshold. Otherwise two sets of thresholds may be identical under any inputs and outputs.

Under these limitations the thresholds cannot always be determined. A mixnet with thresholds of 2 for mix A, 1 for mix B and 1 for mix C (2-1-1) cannot be distinguished from a mixnet with threshold 2 for mix A, 2 for mix B and 1 for mix C (2-2-1). There is no set of input messages for which the set of outputs is different between these two thresholds combinations.

3 Question 3

3.1 3a

We used the function runAssignment2A.

It automatically performs an n-1 attack; injecting messages and reading the logs. See source file for comments on this. We ran it multiple times to see if it always gives the same answer and it does.

Alice is the bad user!

```
Found bad users: Alice | Alice | Alice | Alice | Alice |
```

3.2 3b

- Instead of using a fixed threshold use a dynamic pool mix. Pass on a randomly selected fraction of the messages in the mixer bucket after time interval and keep the rest.
- Use a binomial pool mix. At an interval flip a virtual coin to determine if a certain message in the mixers bucket is forwarded or retained on a time based interval.