# Assignment 1: Mixnets

Santiago Aragón

*s.e.aragonramirez@student.utwente.nl*

Owais Ahmed

*o.ahmed@student.utwente.nl*

*University of Twente*

---

---

**Assignment 1**

**Part A:** We developed an application mixnets.py [Appendix A] to send messages via the three mix nodes and the cache node that forwards the individual messages to the recipient.

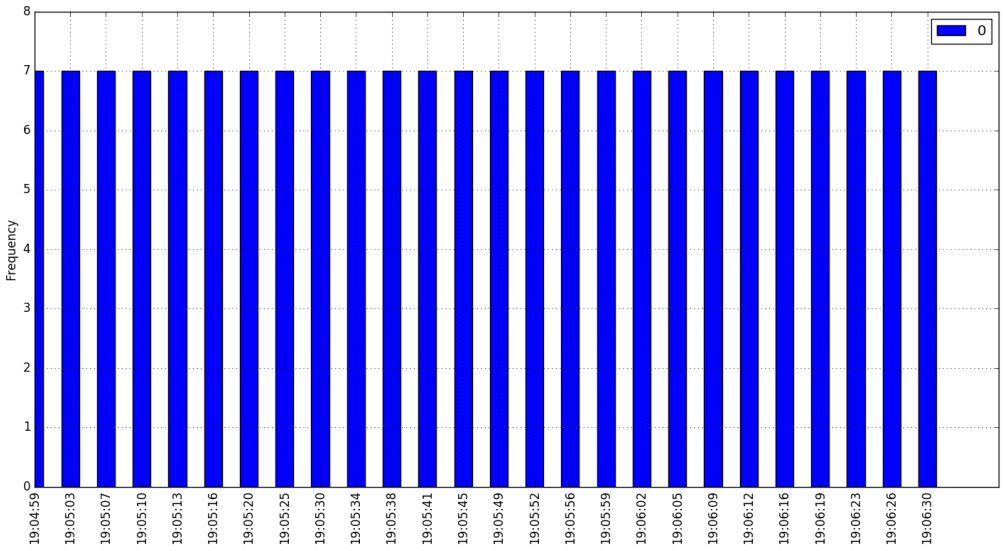
**Part B:** We sent a message to TIM from the send message method as shown below in mixnets.py application [Appendix A].

```
send_message('TIM     ','s1750542  and  s1736574')
```


**Part C:**

We parsed the cache log to analyse the individual messages and performed frequency analysis as shown in Figure: 1 by counting the number of messages received in a particular second and plotted the results in a bar chart graph. We observed that mostly seven messages were received in the cache log in a particular second, however in certain instances, the average of consecutive messages received in two seconds was seven. We therefore came to the conclusion that $n_C$ is 7, and since we know that the threshold of $n_A = n_B = n_C$, it implies that $n_A$ , $n_B$ and $n_C$ is 7.

Figure 1: Frequency of messages received against time in the same second.

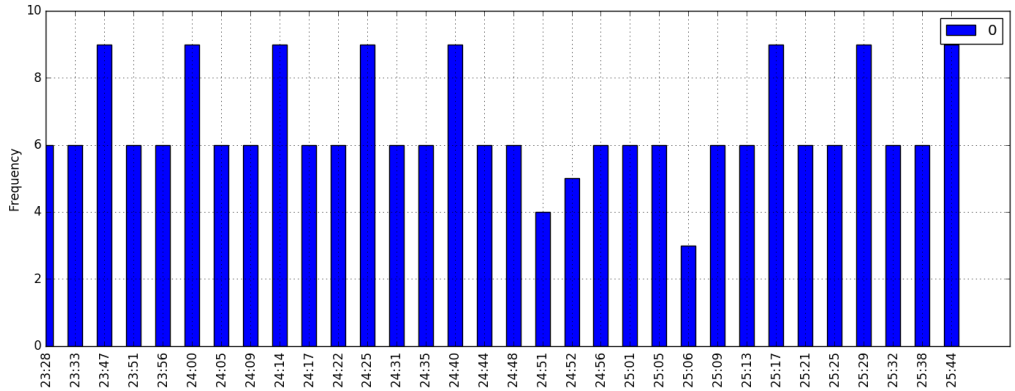



## Assignment 2

### Part A:

We sent individual messages one by one with a short time delay and observed the output via the cache log by parsing the message fields. We observed that the messages forwarded by MIX C to the CACHE NODE had a frequency of mostly 6 or 9. We learned that the sum of messages received in consecutive seconds was always a factor of 3, as shown in Figure: 2. We therefore, we came to the conclusion that the threshold of $n_C$ is 3.

Figure 2: Frequency of messages received against time in the same second.




We kept a count of the messages entering the second mixnet via MIX A and the messages

received in the cache log after reaching the threshold $n_C$. After the first 8 messages passed through MIX A, only 6 messages were displayed in the cache log. Since we know that the threshold of MIX B has to be at least more than $2n_C$ and less than $3n_C$. This implies that the threshold of MIX B is $6 < n_B < 9$. Furthermore, we have only inject 8 messages and we know that $n_A \geq 1$ , therefore we conclude that the threshold $n_B$ is 7.
We kept a count of the messages entering the second mixnet via MIX A and the messages received in the cache log after reaching the threshold $n_C$. After the first 8 messages passed through MIX A, only 6 messages were displayed in the cache log, that denotes that $1 \leq n_A, n_B \leq 8$. We sent more messages one by one until a second batch of 6 messages were received in the cache log. We noted that after sending 6 more messages, another batch of 6 messages was received in the cache log, this further helped us to analyse that MIX A always accepted even number of messages and the least common factor of the input messages to obtain an output was always 2. We therefore concluded that the threshold of $n_A$ is 2.

Threshold of $n_A$ is 2
Threshold of $n_B$ is 7
Threshold of $n_C$ is 3

**Part B:**

## Assignment 3

**Part A:**

To deanonymize the party that is communicating with TIM we launch a n-1 attack. We recall that we are a global active attacker with insert capabilities, namely, we have access to two logs one at the entrance (client log) of the mixnet and one at very end (cache log).

We have find the threshold of every MIX in section **??** and we know that the first 2 batches are triggered with 8 messages. Thus, after starting the mixnet we insert 7 messages and wait for a message coming from some mixnet user. The very first message that arrive will push 6 messages to the cache, to which we have access through the logs. By comparing the client log and the cache log we are able to deanonymize the first message that enters the mixnet after our first 7 messages. We define a function called *n_1_attack* where we start the mixnet, perform a n-1 attack and repeats until the deanonymized user is the one that communicates with Tim.

```
Python 2.7.10 (v2.7.10:15c95b7d81dc, May 23 2015, 09:33:12)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import mixnets as m
Mixer stoped
Mixer 3 started...
Mixer stoped
Alice is communicating with Tim
```

**Part B:** The first method is to prevent less than 3 users in one particular batch of messages to be flushed to the next mix node after reaching its threshold. The second way of preventing such type of attacks is by randomising the threshold of each node after every few random time delays. This will prevent the attacker to perform n-1 attack.

## Appendix A. Mixnet 1

## Appendix B. Mixnet 3

```
def n_1_a():
    start(3)
    sleep(.05)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    send_message('ME ', '-'*7)
    log = parseClientLog()
    if log == '':
        message_sent = 7
        while cache_num() < 6:
            pass
        stop()
        sleep(2)
        if not check_for_tim() and not_me() is not  None:
            rec = not_me()
            sen = first_client()
            print '%s is communicating with %s' %(sen, rec)
            n_1_a()
        elif not_me() is  None:
            n_1_a()
        else:
            rec = not_me()
            sen = first_client()
            print '%s is communicating with %s' %(sen,rec)
    else:
        n_1_a()
```

## References