**Team 17 Results, Anonychat**

Ben Korza, Mark Fitzgibbon, Mitchell Caisse

# Results

The main goal of our project was to create an Anonymous chat program to allow users to communicate with each other anonymously. We have ran several tests to determine how practical our implementation is. These tests include anonymity tests, network utilization tests, and scalability tests. The result and anaylsis of the tests that we ran are discussed in detail below.

## Encryption and Decryption

To ensure the use of encryption wouldn’t add significant overhead to our protocol, we tested the time it takes for messages of various sizes to be encrypted and then decrypted. In order to do this we encrypted messages with a length with a range of 20 to 1000 increasing by 20 each time. As Figure 1 shows, the time required to encrypt a message is relatively minimum with the average time being around 0.15 milliseconds. The time required to decrypt a message is longer than encrypting, 8.13 milliseconds on average, but is still not that much of an overhead. These times are less than our metric of 250ms for encryption and decryption.



Figure Encryption and Decryption Times

Encryption time does vary greatly depending on the length of the message, as can be seen below in Figure 2. Messages with a longer length take considerably less time to encrypt than messages with a smaller length. However after the message length gets to around 150 characters encryption time levels out at around 0.15 milliseconds. There is a spike around 640 characters, which is most likely due to the encryption algorithm used. These results show that the encryption over head is dependent upon the message length, and there for the use of our chat program. If a user sends a lot of longer messages, encryption overhead will be lower than if they send longer messages versus smaller messages. As can be seen in Figure 3, the decryption time follows a similar trend to encryption.

Figure Encryption Time Trend

In addition to ensuring that encryption and decryption does not add too much over head to our protocol, we also wanted to ensure that a successful decryption will not expose a peer as a recipient. To do this we ran a similar test as above, but this time decrypted each message with the correct private key and an incorrect private key. In Figure 3, it can be seen that the time for an unsuccessful encryption does not differ by much of that of a successful decryption. An unsuccessful encryption is not always faster than a successful encryption. The average time difference between a successful and unsuccessful encryption is 0.0527 milliseconds. This difference is miniscule and will make it hard to determine the recipient of a message by analyzing decryption times.

Figure : Unsuccessful vs Successful Decryption Times

Another thing crucial to the success of our protocol was its bandwidth consumption. In order to make sure Anonychat was scalable we tested our network utilization. We did this by running the application on 25 clients running locally over the course of five minutes. During this time, messages were sent between clients at random intervals ranging from five to 35 seconds. As this happened, we measured the amount of packets that existed in the network. The results of these tests are reflected in Graph B.

Graph B

From these results we can gather some information.

Since we desired Anonychat to be a practical protocol, it was crucial we test the latency of sending packets across the network. To do this we created three nodes and had them communicate with each other. We recorded the RTTs of packets, and the results can be seen in the following three tables.

Node 1 RTTs

Sending RTT

Node (ms)

|  |  |
| --- | --- |
| 2 | 13.561035 |
| 2 | 12.787842 |
| 2 | 13.983887 |
| 3 | 7.923887 |
| 3 | 10.42920 |
| 2 | 10.164062 |
| 3 | 10.879150 |
| 2 | 13.978530 |
| 2 | 13.126953 |
| 2 | 13.251221 |
| 3 | 5.745850 |
| 2 | 11.463867 |
| 2 | 13.580078 |
| 3 | 8.060059 |
| 2 | 10.706299 |
| 3 | 3.977051 |

Node 2 RTTs

Sending RTT

Node (ms)

|  |  |
| --- | --- |
| 3 | 8.164062 |
| 3 | 7.083740 |
| 3 | 13.013672 |
| 1 | 13.936035 |
| 3 | 6.549805 |
| 1 | 12.625977 |
| 1 | 14.530029 |
| 3 | 7.388184 |
| 3 | 8.044922 |
| 1 | 12.937012 |
| 3 | 6.104004 |
| 3 | 6.746094 |
| 1 | 13.744141 |
| 1 | 13.084961 |

Node 3 RTTs

Sending RTT

Node (ms)

|  |  |
| --- | --- |
| 1 | 41.361084 |
| 2 | 31.485107 |
| 1 | 40.700928 |
| 2 | 31.006104 |
| 1 | 32.227051 |
| 2 | 28.937988 |
| 1 | 37.948975 |
| 2 | 27.944092 |
| 1 | 37.994873 |
| 2 | 111.469971 |
| 1 | 45.537109 |
| 2 | 33.472168 |

From this data we conclude something.

During this test the ANS reshuffled the peer list x times. These reshuffles took y ms on average.

The last thing we needed to check was encryption. Using wireshark, we confirmed the messages were indeed encrypted. No identical packets were found either.