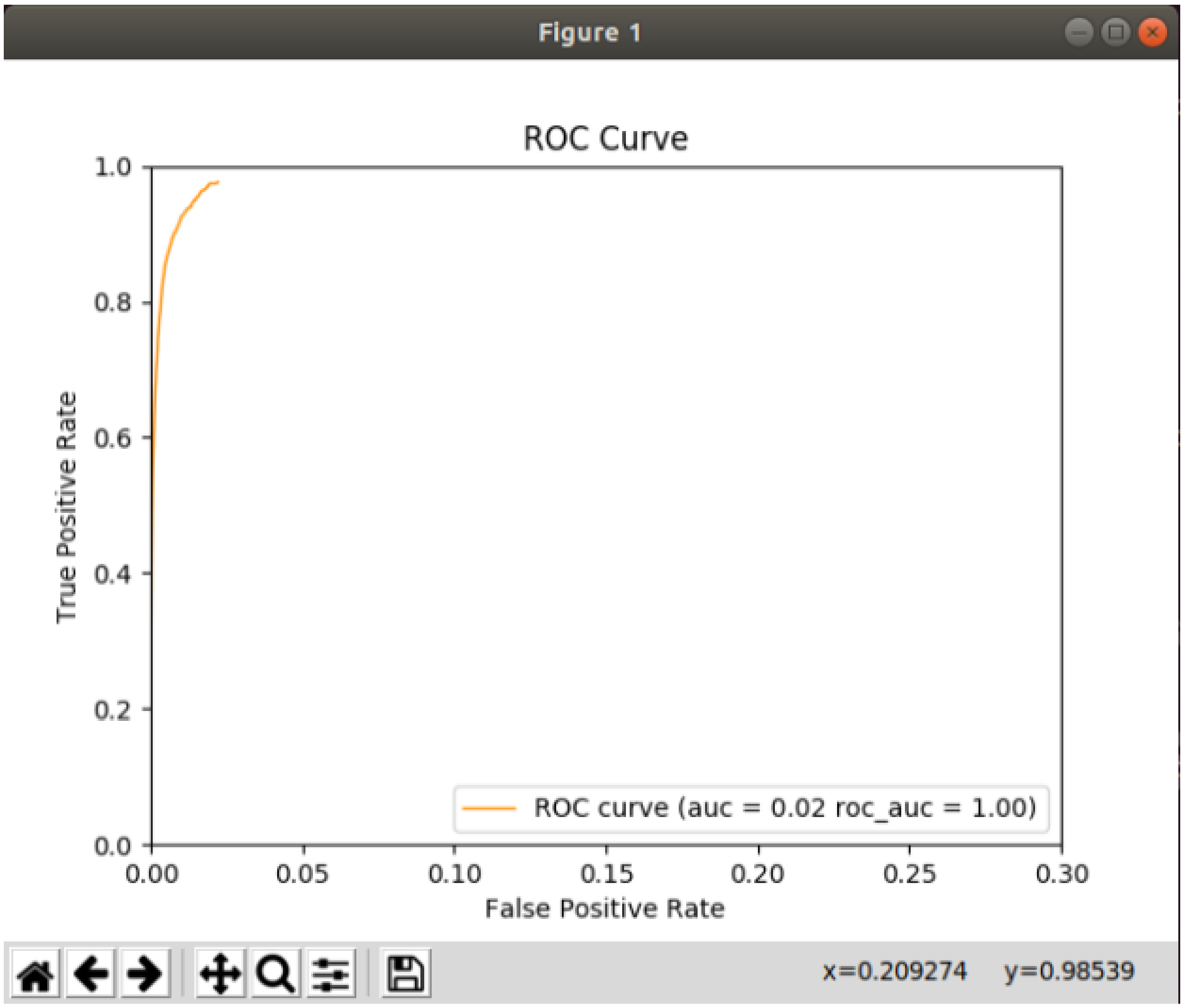
Cameron Clark
1/26/19
Anonymity & Tor

Timing Analysis Report

Introduction

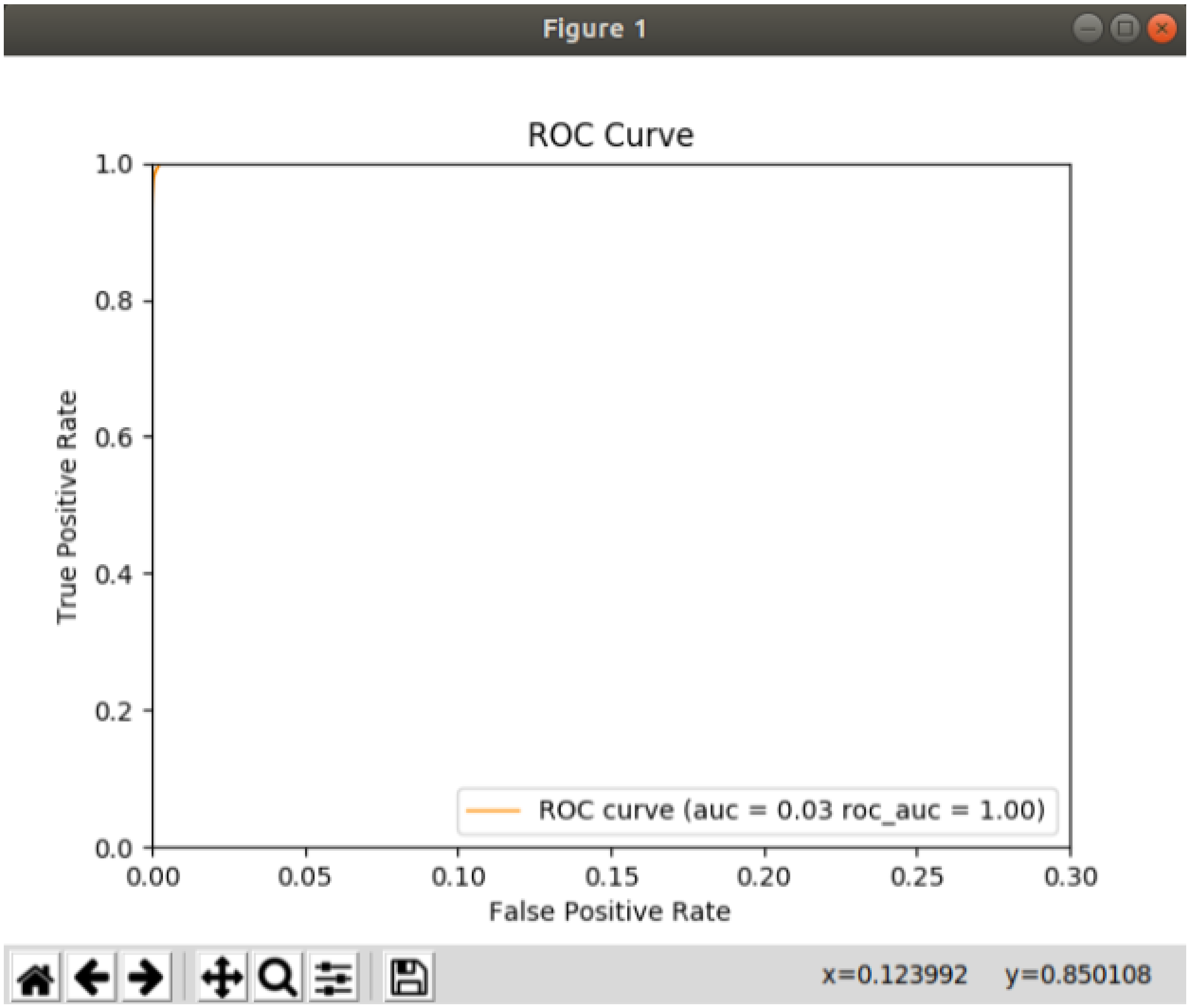
In this experiment, we are discovering the effectiveness of end to end correlation attacks for Tor traffic. We are given starting times of packets and we then generate the ending times of packets for that sites instance. Then going between all the sites and each of there instances. Using the Pearson correlation algorithm, we are measuring the relationship between the first and last hops. If the correlation coefficient is above a certain threshold we say that those two sets of traffic are pairs. From this, we can determine the number of true positives, in which we were able to correctly identify the first hop and last hop packet times from the same instance. We also determine the number of false positives. This occurs when we identify that a set of first hop times and last hop times as pairs, but in fact, they were either from the same site but different instances, or different sites entirely.

Experiments

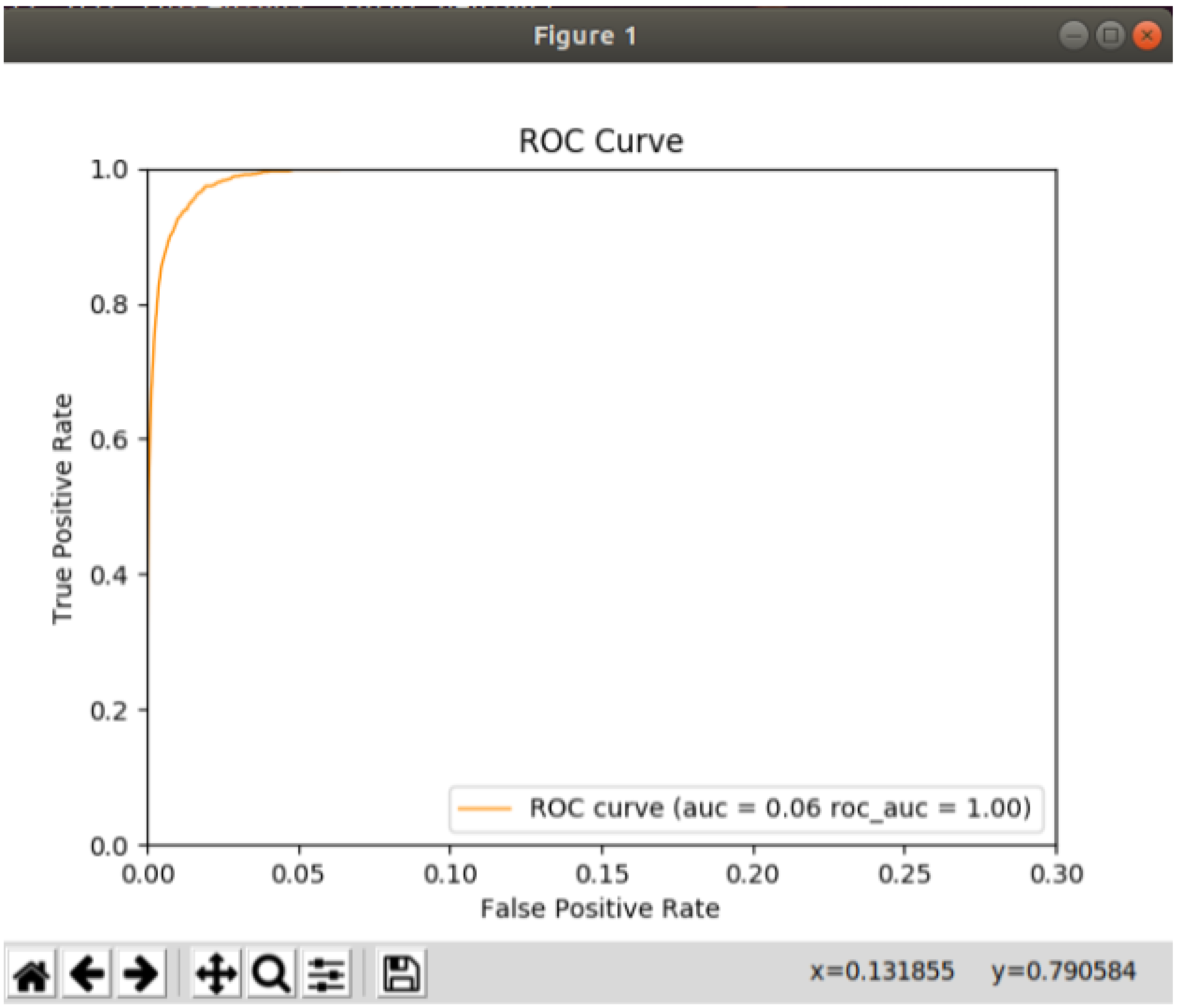


ROC curve (auc = 0.02 roc_auc = 1.00)

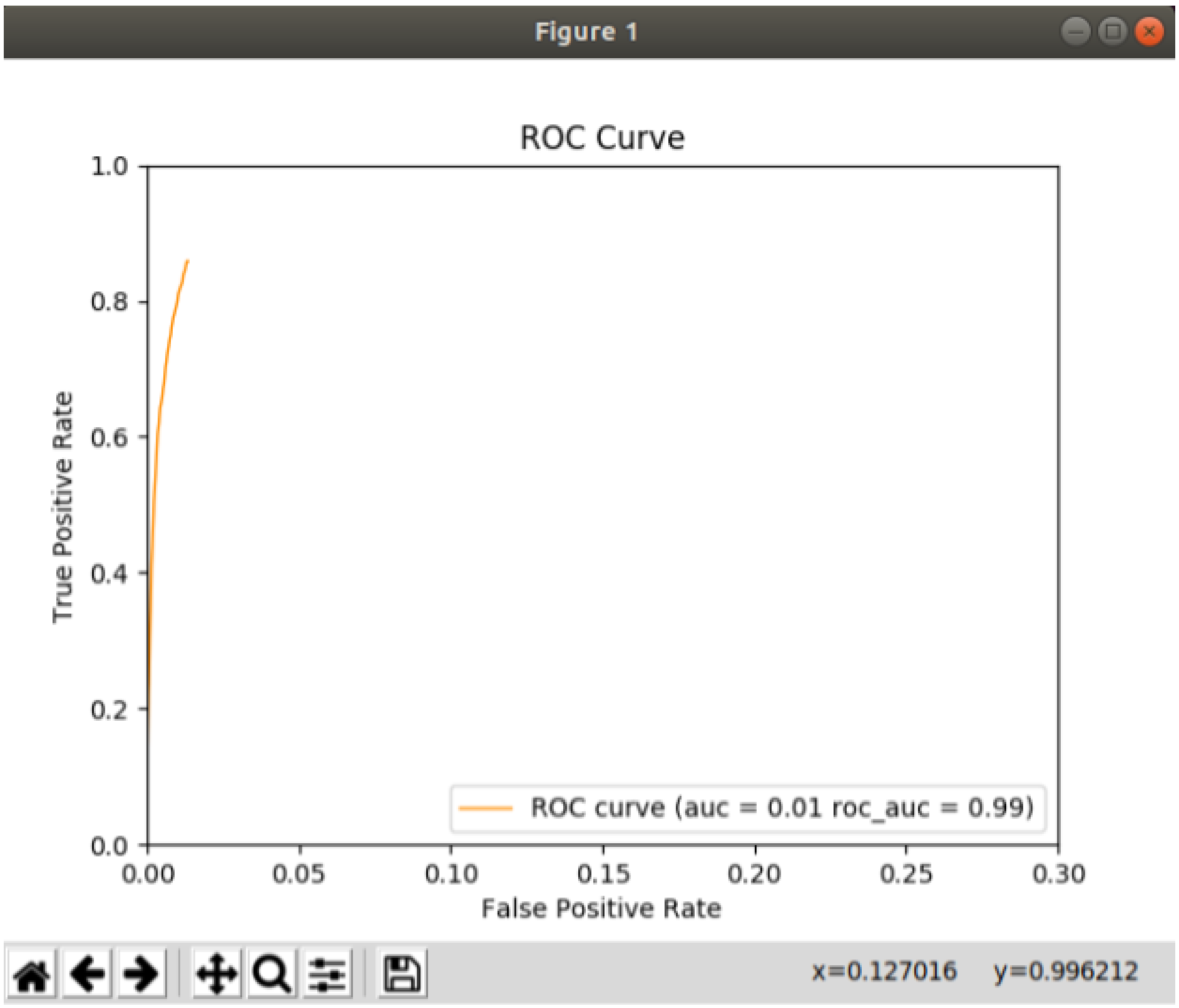
A = 10 MIN_THRESHOLD = 0.9 MAX_THRESHOLD = 1.0



A = 2 MIN_THRESHOLD = 0.9 MAX_THRESHOLD = 1.0



A = 10 MIN_THRESHOLD = 0.8 MAX_THRESHOLD = 1.0



A = 20 MIN_THRESHOLD = 0.9 MAX_THRESHOLD = 1.0

Random parameter	Area under curve (auc function)	Area under curve (roc_auc_score function)
10	0.02	1.00
2	0.03	1.00
20	0.01	0.99

All thresholds are between 0.9 and 1.0

From these graphs, we can see with our correlation function we were able to achieve a very high true positive rate versus a very low false positive rate. The threshold started off decently high at 0.9 for it to be considered a match. The correlation function was able to perform well up until reaching the 0.95 thresholds and above which is when we began to see the performance fall off. A majority of these tests were run with the random delay value being 10 or higher. Which is a extremely high latency even for Tor network standards. Looking at the graph where we had a random parameter of just 2. The performance is nearly perfect up until the 0.97 thresholds. The AUC metric specifies how accurately the correlation algorithm can detect when a match is actually a match and when it is not. When performing the experiments I was using the *auc()* function from the *sklearn* python library, the value I was getting didn't seem to be indicative of the ROC curve I was seeing, so I switched to using the *roc_auc_score()* function from the same package and the results seen still seem to be a bit off. As they are higher which makes sense because the correlation algorithm had a very high success rate. But getting values of 1.0 for the AUC would mean the correlation algorithm performed perfectly, which it did not.

The key takeaways here are that the lower the threshold we specified we were able to see high true positives, but also saw higher false positive rates which are not good for an attacker. By keeping the threshold reasonably high, we get high true positive rates with little to no false positives. In addition to this, the larger the random parameter *'a'* we gave, the less true positives we were able to see. With that much variation, the correlation algorithm performs poorly.

Responses

Question #1:

With 500 active users, we would have 250000 pairs of first and last.
Threshold (0.960): TP = 500, FP = 1750, TN = 247750, FN = 0
% of matches that are FP = 77%
Threshold (0.994): TP = 458, FP = 0 , TN = 249500, FN = 42
% of matches that are FP = 0%

The higher threshold 0.994 is much better for the attacker as they may not get every match possible, but there are no false positives so they can be completely sure that the matches they are getting are correct. Whereas with the lower threshold, they do get all the correct matches, but they get additional matches that are incorrect, being over two-thirds of the total matches. Which gives a lot of misleading data.

Question #2:

False positive results for the same site, can still be considered a good metric and used for the likes of website fingerprinting because it was able to compare two different instances of the same site and conclude that they are the same. The different site false positives are the false positives that we want to avoid completely as the two data sets have no relation to each other at all, so a false positive in this case means the correlation algorithm failed completely.

Question #3:

The delay model just adds a simple random delay to each packet time. It could be more useful and realistic to have a base latency to add that represents what a normal connection would look like. And from that add some delays throughout the connection to that base latency that can simulate some smaller blips in the connection. This will make the connections seem more like actual traces from the websites.

To improve the attack we could look at more than just the packet times coming in as that is a very small amount of information to make decisions on. In the data, we are given a number that represents if the packet is from the client or the server. We can use that data in addition to the packet times to fingerprint the website the traffic is going to.

An improvement to the experiment would be to utilize other correlation algorithms and compare how they perform under the various randomness and thresholds. This could give a better overall look at what thresholds are the best in the attacker's case, and in turn, the defender's case.

Conclusion

In conclusion, we were able to see the effects that increasing and decreasing the threshold has on the true positive versus false positive ratio. And the effect that the given random parameter has on the ratio as well. The more variance we add to the packet times, the harder it is for the correlation algorithm to produce true positive results. In addition to this, we compared the BRF between two thresholds, a discovered that while a higher threshold does give us some false negatives, it greatly reduces the number of false positives seen, giving us more reliable results overall.