Aggregating Local Metrics for Global Trust Calculations in the TOR Network

Nada M. Abdel Aziem
Computer Engineering Department
AASTMT, College of Engineering
Cairo, Egypt
eng_nadamostafa1988@yahoo.com

Sherif F. Fahmy
Computer Engineering Department
AASTMT, College of Engineering
Cairo, Egypt
fahmy@vt.edu

Attallah I. Hashad
Computer Engineering Department
AASTMT, College of Engineering
Cairo, Egypt
hashad@cairo.aast.edu

***Abstract*—This paper discusses the design and implementation of a distributed trust calculation mechanism that allows users in the peer-to-peer TOR anonymity network to determine whether or not the nodes that comprise the TOR network are malicious. This is done through a cooperative algorithm that allows individual nodes to measure the trust of the TOR nodes they communicate with. The paper measures the performance of this algorithm and shows that the algorithm can accurately identify several kinds of malicious nodes in the TOR network. This paper will focus on the following security issues: the problem of self reported bandwidth and uptime that can lead to low resource attack, geographical location and Denial of service (DoS) attack and how to aggregate all these criteria in one value that indicate whether or not the onion router is malicious. The aggregation is performed using two different techniques, each of which uses the exact same aggregation methods except for geographical location. The first method, which excludes entire geographical locations, produced slightly worse results than the second method, which assigned a scale between 0 and 0.99 for each geographical location to signify its trust**

Keywords—TOR, Anonymity, Trust.

I. INTRODUCTION

Anonymity is becoming increasingly important as people in many countries embrace the online age. The Internet is a very important tool in the dissemination of information, and its presence is a significant boost for freedom of expression. By allowing ordinary citizens to express themselves on the so-called Web 2.0, which broadly means social media and blogging platforms, the Internet has ushered in an unprecedented age of self-expression.

The Internet was not designed with anonymity in mind, and therefore contains no built-in capability for circumvention monitoring by public or private agents who are out to gather information about an individual's web use. In order to overcome this shortcoming of the Internet, several technologies have been developed to allow individuals to retain their anonymity while using the Internet.

Perhaps the most successful technology developed to do this so far is the TOR network [1] – The Onion Router – a network that routes a message through at least three different routers before it can reach its destination, with each link on the path being encrypted. While this provides a good solution to the anonymity problem, a major disadvantage appears because of the volunteer nature of the nodes making up this network.

There is no centralized vetting mechanism present in TOR networks, and thus there is no method for ascertaining whether a node in the TOR network is malicious or not. In order to address this issue, this paper presents a trust calculation algorithm that can be used to gauge the *"maliciousness"* of nodes based on their interaction with clients of the TOR system.

The paper shows how this local measurement of trust can then be aggregated to reach a system-wide view of trust. Aggregation means to combine many values to one value and this value indicates if the onion router is malicious or not. A slight modification of the TOR circuit establishment mechanism can then be used to make sure that only trusted nodes are present in any path used by clients of the network.

The remainder of this paper is organized as follows, Section II provides a brief overview of the TOR network, Section III is a brief literature review of related work, Section IV presents the proposed algorithm, section V presents the Trust Value calculation algorithm, section VI presents the results of the experiments performed to measure the performance of the trust value calculation algorithm, section VII presents Algorithm Complexity and overheads, and section VIII presents the Conclusion and Future Work.

II. THE TOR NETWORK

TOR [1], an acronym for The Onion Router, is free software for enabling online anonymity and circumventing censorship. TOR performs this function by routing Internet traffic through a volunteer network of more than five thousand nodes at latest count [2], to conceal the originating IP of the client accessing the Internet and to hide the identity of the destination that the client is trying to access from its Internet entry point – typically an ISP.

The term "Onion Router" refers to the fact that TOR uses layered encryption to protect traffic between the source and destination. This layered encryption is similar to the layers of an onion. TOR encrypts outgoing data, including the destination IP, several times and then sends it through a virtual circuit of its choosing from among the nodes in the volunteer network. At each node on the circuit, one, and only one, layer of the encryption is peeled off to reveal the identity of the next node on the path. Finally, when the data reaches the final node on the path, the innermost encryption layer is decrypted to

reveal the identity of the destination to which the data is to be delivered [3]. Figure 1 shows the typical structure of a TOR virtual circuit [4].

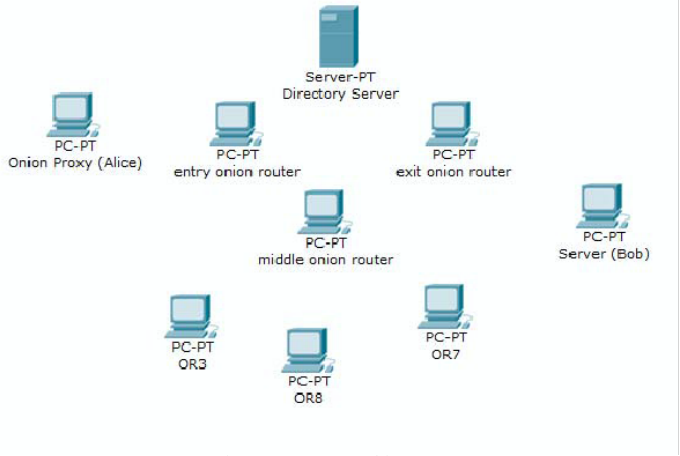

Figure 1: TOR Architecture

We now present a brief explanation of each of the components in the figure.

Onion proxy (OP): it runs on the client (Alice), to send the client data to the server through the TOR network.

Onion Routers (ORs): through which data are sent to the server. The circuit consists of three onion routers that propagate data through them until it is received by the server. The first onion router is called the entry router, the second one is called the middle router, and the third one is called the exit router. The three onion routers are connected with each other through secured links protected by a cryptographic protocol called TLS [5], to provide anonymous communication through the network.

Directory servers: they hold onion routers information, they contain a list of all the active TOR nodes along with their data (e.g. bandwidth, cryptographic keys).

TOR fails when the attacker sees the entry and exit onion routers. To solve this problem entry guards are used, each client selects a few relays at random, and chooses only from these relays to start the first hop "entry node"[6]. Entry guards are designed to protect against predecessor attack and denial of service as denial of anonymity attack [7]. If the entry guards are uncompromised, the user is safe [8].

Exit policies are implemented to prevent clients from abusing the system. Each TOR relay has an exit policy that assigns what connections are allowed or refused from that relay. The exit policies are propagated to TOR clients via the directory server, so the clients are able to avoid selecting exit relays that would reject to exit to their destination [9].

III. LITERATURE REVIEW

In this section of the paper, a brief review of the literature is presented. In [10], Kamvar et al. present a method for measuring the trust of peers in a peer-to-peer file distributing method. They present a method they refer to as EigenTrust. EigenTrust is an algorithm that allows nodes on peer-to-peer networks to measure their "trust" of other nodes based on whether or not they received authentic file chunks from them, the algorithm them aggregates this information using power iteration [11].

The technique in [10] is closely related to that presented in this paper. However, this paper modifies the algorithm in order to be able to identify untrustworthy nodes in a TOR network, something that cannot be done using a mere count of authentic file chunks received as can be done in file sharing peer-to-peer networks.

Vasilios A. Siris et al. 2006 [12], present algorithms that detect SYN flooding attack which is the most common type of Denial of Service (DOS) attack. These algorithms are: Adaptive threshold algorithm and a particular application of the cumulative sum (CUSUM) algorithm.

Adaptive Threshold Algorithm is based on detecting the attack by using a threshold that is adapted with recent traffic measurements. The alarm signals with the measurements exceeding the threshold $(\alpha + 1)\ \mu$ for a number of iterations where μ is the measured mean rate, and $\propto > 0$ is a parameter that indicates the percentage above the mean value that we consider to be an indication of anomalous behavior . However, this paper modifies the algorithm in order to be able to identify untrustworthy nodes in a TOR network.

Robin Snader et al. 2011[13], present a router selection algorithm that allows users to control the tradeoff between performance and anonymity. Also, it proposes a bandwidth measurement algorithm instead of self reported values of bandwidth.

The algorithm in [13] provides measurement of bandwidth that allows the users to choose the trusted onion routers according to these measurements instead of self reported values of bandwidth which allow each onion routers to assign their bandwidth alone. However, this paper modifies the algorithm in order to measure the trust values according to different criteria discuss later to be able to choose the non-malicious onion routers.

IV. PROPOSED ALGORITHM

As shown in figure 2, a proposed algorithm is presented so that it can be used to gauge the maliciousness of nodes based on their interaction with clients of the TOR system.

Individual nodes measure the trust of the TOR nodes they communicate with and report the local trust values of these nodes to the directory server.

After the nodes have calculated the local trust values of the TOR nodes according to the criteria that will be mentioned, they normalize the local trust values to prepare it for aggregation.

Finally, the directory server performs aggregation for these normalized local trust values to reach a system wide view of trust by knowing whether the node is malicious or not.

The directory server aggregates the normalized local trust values using EigenTrust Algorithm.

The purpose of a TOR directory server [14] is to advertise the list of available and trusted onion routers, that is, the nodes via which clients may build circuits for traffic.

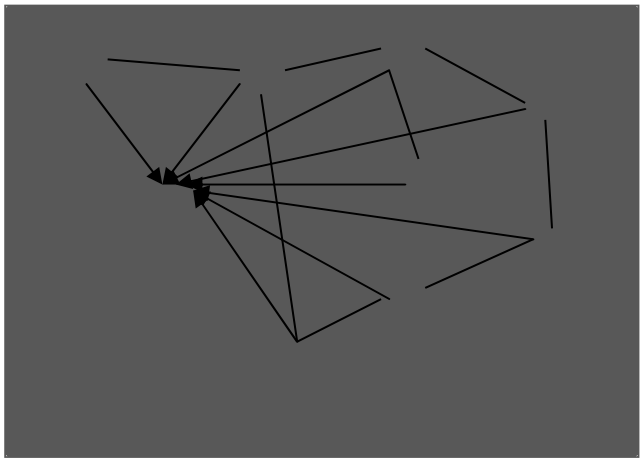

Figure 2: Block diagram for proposed algorithm

Normalization of Local Trust Values

The point of normalization is to make variables comparable to each other. To compare variables that may be measured using different scales, the values are normalized. Specifically, the parameters used to gauze the "maliciousness" of nodes in the proposed distributed trust calculation algorithm are normalized to ensure that no one value has more influence on the system-wide trust than the others. The onion routers normalize their trust values to be able to aggregate it later in the algorithm.

```
1   Constant integer size=764
2   Declare float sum=0.0
3   Declare float value [size]
4   Declare integer i

5   For i=0 to i=size do
6           Set Sum += value[i]
7   End do

8   For i=0 to i=size do
9           If value[i]!=0 then
10                  Set value[i]/=sum
11          Otherwise
12                  Set value[i]=0.0

13  End if
14  End do
```

Algorithm 1: Normalization Algorithm

Aggregation of the normalized local trust values

The directory server takes the normalized local trust values of the onion router to aggregate them. Aggregation means to combine several values into one value, and this value is a meaningful value as it indicates if the onion router is malicious or not. Aggregation is carried out by EigenTrust Algorithm [10].

```
1   t^(0) = p
2   Repeat
3           t^(k+1) = C^T t^(k);
4           t^(k+1) = (1-∝) t^(k+1) + α p;
5           δ = ||t^(k+1) − t^(k)||;
6   Until δ<ε;
```

Algorithm 2: Aggregation of normalized local trust values

Where, t is global trust vector, C is normalized local trust matrix, α is some constant less than 1, p is pre-trusted peers, ε is the stopping condition and δ is the difference between the time now and the time after.

Pre-trusted peers means selecting randomly some peers to be trusted peers as they guarantee convergence and break up malicious collectives. Therefore, the choice of pre-trusted peers is important. In particular, it is important that no pre-trusted peer be a member of a malicious collective. This would compromise the quality of the algorithm. To avoid this, the system may choose a very few number of pre-trusted peers.

Criteria to measure the trust of the onion routers

Calculate the local trust values of the TOR nodes according to the following criteria:

- Uptime.
- Bandwidth.
- Geographical Location.
- DOS (Denial of Service).

Uptime of the onion routers

This criterion measures the time during which the onion router has been working and available in the network. Measurement occurs through a number of phases:

First Phase involves: making query on each onion router if it is still alive in the system or not. This is done by sending many ICMP (Internet Control Message Protocol) echo requests (10 requests) for each onion router and waiting for its reply, when RTT (Round Trip Time) finishes it tells "request time out", this mean that the onion router is not found in the Network or making tasks and can't reply in that moment.

$$\text{Measured uptime} = \frac{number\ of\ replies}{total\ number\ of\ requests} \quad (1)$$

Second Phase involves: Calculating the proposed actual uptime of the router as shown in the proposed actual uptime Algorithm (Algorithm (3)).

Third Phase involves: subtracting the measured uptime from the proposed actual uptime, and setting the threshold value by 0.45.

If the result of the subtraction is greater than 0.45, then this onion router is untrusted, otherwise it is trusted.

Fourth Phase involves: Comparing the results according to the following features:

- False positive: percentage of good onion routers that are identified as malicious
- False negative: percentage of malicious onion routers that are identified as good
- Percentage of Malicious onion routers: percentage of malicious onion routers that has properly been identified as malicious
- Average: getting the mean of all values
- Standard deviation: showing if the values are close to the mean or not.

```
1   Choosing some onion routers to be malicious onion routers (210 onion routers), and choosing some onion routers to be user onion
    routers (554 onion routers).
2   Generate random numbers
3   If random number is even then
4           Factor=-1
5   Otherwise
6           Factor=1
7   End if
8   If nodes are malicious then
9           Actual=measured+ ((((float)Rand()/RAND_MAX)/2.0)+0.5)*Factor
10  Otherwise
11          Actual=measured+ (((float)Rand()/RAND_MAX)/2.0)*Factor.
12  End if
```

Algorithm (3): The proposed actual uptime Algorithm

Bandwidth of the onion router

It measures the performance of the onion router. Measurement occurs through a number of phases:

First phase involves: firstly, calculating the bit rate or number of bits received to an onion router in one second by using NS-2 (Network Simulator version 2). Secondly, selecting number of onion routers to send data (bits) to other onion routers in different time. Finally, creating links between onion routers to send data through it by making it duplex link (sending and receiving bits) and its bit rate 1.7Mbit and time 20 ms, setting queue size by 10 bytes (frames)

$$Bandwidth=\frac{number\ of\ bytes}{time}*8 \qquad (2)$$

Second phase involves: calculating the trust of the onion routers from the point of view to the other onion routers.

For example, calculation of the bandwidth of onion router 0 from the point of view of onion router 2:

$$Bandwidth=\frac{[BW(2->0)+\left[\frac{BW(1\to0)}{n}+\frac{BW(3\to0)}{n}+\cdots\cdots\cdots\cdots+\frac{BW(n\to0)}{n}\right]].}{2}. \qquad (3)$$

Third phase involves: calculating the average and the standard deviation of all onion routers.

Fourth phase involves: normalizing all values of bandwidth by the Normalization Algorithm discussed before(Algorithm (4)).

Fifth phase involves: aggregating all values of Bandwidth by the aggregation Eigntrust algorithm discussed before.

```
1   Constant integer size=764
2   Declare float sum
3   Declare float BW[size][size]
4   Declare integer i
5   Declare integer j
6   For i=0 to i=size do
7           Set Sum=0.0
8           For j=0 to j=size do
9                   Set Sum += BW[i][j]
10          End do
11          For j=0 to j=size do
12                  If BW[i][j]!=0 then
13                          Set BW[i][j]/=sum[i]
14                  Otherwise
15                          Set BW[i][j]=0.0
16                  End if
17          End do
18  End do
    Where, BW is bandwidth.
```

Algorithm (4): Bandwidth Normalization Algorithm

Sixth Phase involves: getting different values of alpha ranged from 0.55 to 0.99, also choosing some onion routers (33 onion routers) to be pretrust onion routers to make the convergence of the values.

Comparing the results from different alpha (α) values discussed in fifth phase according to the following features to get the best value alpha used:

- False positive
- False negative
- Percentage of Malicious onion routers
- F-measure [15]: measures the iteration's accuracy. It considers both the precision and the recall of the iteration to compute the accuracy .

$$Precision=\frac{tp}{tp+fp} \qquad (4)$$

$$Recall=\frac{tp}{tp+fn} \qquad (5)$$

$$F\text{-}measure=2*\frac{precision*Recall}{precision+Recall} \qquad (6)$$

Seventh phase involves: calculating the proposed actual bandwidth of the onion routers as shown below (Algorithm(5)).

Eighth phase involves: subtracting the measured bandwidth from the proposed actual bandwidth, and setting threshold value with double the standard deviation.

If the result of the subtraction is greater than the threshold, so this onion router is untrusted, otherwise it is trusted.

Ninth phase involves: Comparing the results according to the following features:

- False positive
- False negative
- Percentage of Malicious onion routers
- Average: getting the mean of all values
- Standard Deviation: showing if the values are close to the mean or not.

1	Choosing some onion routers to be malicious onion routers (210 onion routers), and choosing some onion routers to be user onion routers (554 onion routers).
2	Generate random numbers
3	If number is even then
4	Factor=-1
5	Otherwise
6	Factor=1
7	end if
8	If nodes are malicious then
9	Actual = measured+ ((rand () % 5 + 2) * standard deviation *factor)
10	Otherwise
11	Actual=measured+ ((rand () % 3) * standard deviation *factor)
12	end if

Algorithm (5): Proposed actual bandwidth Algorithm

Geographical Location of the onion routers

Knowing the location of each onion router in the network, and how these locations affecting the trust of the onion routers.

The location of onion routers can be determined by two methods that involve [16]:

First method involves: selecting some onion routers according to their locations to be malicious and the other onion routers is not malicious.

For example: selecting the onion routers in these locations: Egypt, Israel, Russian federation, and Denmark to be malicious onion routers and their values are equal to zero and the other onion routers are not malicious and their values are equal to one.

Second method involves: putting the trust value ranged between (0.0 to 1.0) according to the location of the onion routers in the countries. For example: the onion routers that are located in European countries have the trust values ranging between (0.67 to 1.00), the onion routers that are located in Asian countries have the trust values ranging between 0.34 to 0.66, and the onion routers that are located in African countries have the trust values ranging between 0.0 to 0.33.

Denial of Services (DoS)

It is popularly called SYN flooding. This attack is done by sending packets to all nodes in the system to make them busy all the time and to slow down the flow of data.

The attacker makes this attack to weaken the nodes in the network to destroy the network and make it unable to send or receive any true request.

DoS attack prevents the users from using any resource in the network.

It is detected through a number of phases:

First phase involves: simulating a DoS attack using NS-2 (Network simulator version 2). Firstly, selecting some routers acting as malicious onion routers (200 onion routers), and some onion routers acting as user onion routers (564 onion routers). Secondly, making the data rate of the malicious onion routers is 4Mb and data rate of user onion routers is 2 Mb.

Also making the size of the packets of the malicious onion routers is 200 Bytes, and size of the packets of the user onion routers is 1000 Bytes. Thirdly, running the simulator that made the onion routers (malicious or user onion routers) send packets to other onion routers in the network with different time, and calculating the number of packets received to all onion routers in the network. Finally, running this simulator 4 times to make sure if this onion router affected by DoS attack or not and to be sure if this onion router made DoS attack or not.

Second Phase involves: applying adaptive threshold algorithm that is one of the algorithms that can detect SYN flooding which is the most common type of DoS attack as shown in Algorithm (6).

1	It relies on testing whether the traffic measurement or the number of SYN packets over a given interval exceeds a certain threshold.
2	The value of the threshold is set based on the mean number of SYN packets which is computed from recent traffic measurements.
3	If the number of SYN packets in the n^{th} time interval (X_n), exceeds the mean rate estimated from measurements prior to n ($(\alpha+1)\mu_{n-1}$), then the alarm signals at time n.
4	When α value is more than zero this indicates that the percentage is greater than the mean value. α value is then considered to be an indication of anomalous behaviour.
5	The mean can be calculated from a past time window or by using an exponential weighted moving average (EWMA) of previous measurements as follows: $\mu_n = \beta\mu_{n-1} + (1-\beta)x_n$, Where β is the EWMA factor.
6	A simple modification that can improve its performance is to signal an alarm after a minimum number of consecutive violations of the threshold. In this case, the alarm condition is given by If $\sum_{i=n-k+1}^{n} 1_{\{x_i \geq (\alpha+1)\mu_{i-1}\}} \geq K$, ALARM signals at time n; Where the parameter k is more than one that indicates the number of consecutive intervals the threshold must be violated for an alarm to be raised.

Algorithm (6): Adaptive Threshold Algorithm

Third Phase involves: choosing different values of $\propto$ and β (about 18 trials), and set number of iteration in each trials (k) is four iteration that gets them from NS-2 simulator, and takes the majority of all iteration to identify which onion router is trusted and which is untrusted.

1	$\propto= 0.1, \beta = 0.7$
2	$\propto= 0.1, \beta = 0.98$
3	$\propto= 0.1, \beta = 0.998$
4	$\propto= 0.5, \beta = 0.7$
5	$\propto= 0.5, \beta = 0.98$
6	$\propto= 0.5, \beta = 0.998$
7	$\propto= 0.6, \beta = 0.7$
8	$\propto= 0.6, \beta = 0.98$
9	$\propto= 0.6, \beta = 0.998$
10	$\propto= 1, \beta = 0.7$
11	$\propto= 1, \beta = 0.98$
12	$\propto= 1, \beta = 0.998$
13	$\propto= 1.5, \beta = 0.7$
14	$\propto= 1.5, \beta = 0.98$
15	$\propto= 1.5, \beta = 0.998$
16	$\propto= 2, \beta = 0.7$
17	$\propto= 2, \beta = 0.98$
18	$\propto= 2, \beta = 0.998$

Fourth Phase involves: normalizing all values of DoS as shown in Algorithm (7).

Fifth Phase involves: aggregating all values of DoS by the aggregation Eigntrust algorithm discussed before.

Sixth phase involves: getting different values of alpha ranged from 0.55 to 0.99, also choose some onion routers be pretrust onion routers to make the convergence of the values.

```
1   Constant integer size=764
2   Declare float sum
3   Declare float DoS[size][size]
4   Declare integer i
5   Declare integer j
6   For i=0 to i=size do
7           Set Sum=0.0
8           For j=0 to j=size do
9                   Set Sum += DoS[i][j]
10          End do
11          For j=0 to j=size do
12                  If DoS[i][j]!=0 then
13                          Set DoS[i][j]/=sum[i]
14                  Otherwise
15                          Set DoS[i][j]=0.0
16                  End if
17          End do
18  End do
    Where, DoS is denial of service
```

Algorithm (7): DoS Normalization Algorithm

Seventh Phase involves: applying DoS Algorithm as shown below (Algorithm (8)).

```
1   Calculating the total of the values of the non-pretrust onion routers.
2   Calculating the average of the values of the non-pretrust onion routers.
    Average=(the total of the values of the non−pretrust onion routers) / (number of the non−pretrust onion routers)
3   If the value of router is greater than the average then
4           It is trusted onion router
5   Otherwise
6           It is untrusted onion router.
7   End if
```

Algorithm (8): DoS Algorithm

Comparing the results that come from different alpha (α) number of pretrust onion routers discussed in fifth and sixth phase, α and β discussed in third phase according to the following features:

- False positive
- False negative
- Percentage of Malicious onion routers
- F-measure [15]

V. Trust Value Calculation Algorithm

In this section we present how to calculate the trust values of onion routers using Trust Value Calculation Algorithm, which depending on the criteria's measurements that discussed in the previous section. These criteria are uptime, bandwidth, geographical location and DoS attack.

Measure the trust values of the onion routers

It will be measured by some phases:

First phase involves: normalizing all values that calculated from the criteria discussed before using normalization algorithm (algorithm 1).

There are two proposed methods to calculate the trust values of the onion routers:

First method involves: adding the trust values of uptime calculation with the aggregated values of both bandwidth and DOS attack then multiply them with the geographical location (first method "discussed in the previous section").

*Trust Value = (uptime + aggregated (bandwidth) + aggregated (DOS attack))*geographical location.* (7)

Then set the threshold with an equality trust values (1/764) of all values of the onion routers.

$$Threshold=\frac{1}{number\ of\ onion\ routers}=\frac{1}{764} \quad (8)$$

Finally, check the trust values of the onion routers with the threshold, if the trust value exceeded the threshold so it is malicious, otherwise it is not malicious.

Second method involves: adding the trust values of uptime calculation with the geographical location (second method "discussed in previous section"), with the aggregated values of both bandwidth and DOS attack.

Trust Value= uptime + geographical location + aggregated (bandwidth) + aggregated (DOS attack). (9)

Then set the threshold with an equality trust values of all values of the onion routers.

$$Threshold=\frac{1}{number\ of\ onion\ routers}=\frac{1}{764} \quad (10)$$

Finally, check the trust values of the onion routers with the threshold, if the trust value exceeded the threshold so it is malicious, otherwise it is not malicious.

Second phase involves: repeating the second phase 20 trials, as discussed before in the previous section that to get different values of uptime and bandwidth as their algorithm depend on random numbers.

Compare the results that come from this phase according to the following features:

- False positive
- False negative

- Percentage of Malicious nodes

Third Phase involves: calculating the average and the standard deviation of the trust values of the 20 iterations.

VI. Experimental Results

First method

As shown in table I., as discussed before in the last section that the algorithm will repeat 20 more iterations depending on different values of uptime and bandwidth. Then compare the results of the iterations using false positive, false negative and percentage of malicious onion routers.

As shown in this table, the false positive gets better values in all iteration. To identify if this method is better or not we are going to calculate the average and standard deviation of the three features.

TABLE I. TRUST VALUE OF THE ONION ROUTERS IN 20 ITERATIONS (FIRST METHOD)

	False positive	False negative	Percentage of malicious onion routers
1st iteration	38.80866426%	43.80952381%	56.19047619%
2nd iteration	41.15523466%	43.33333333%	56.66666667%
3rd iteration	39.53068592%	42.85714286%	57.14285714%
4th iteration	39.71119134%	44.28571429%	55.71428571%
5th iteration	38.98916968%	43.80952381%	56.19047619%
6th iteration	40.97472924%	43.80952381%	56.19047619%
7th iteration	39.71119134%	43.33333333%	56.66666667%
8th iteration	39.71119134%	43.33333333%	56.66666667%
9th iteration	41.87725632%	45.23809524%	54.76190476%
10th iteration	40.07220217%	44.28571429%	55.71428571%
11th iteration	38.80866426%	43.80952381%	56.19047619%
12th iteration	40.433213%	44.28571429%	55.71428571%
13th iteration	40.79422383%	44.28571429%	55.71428571%
14th iteration	40.433213%	43.33333333%	56.66666667%
15th iteration	39.53068592%	43.80952381%	56.19047619%
16th iteration	39.89169675%	43.80952381%	56.19047619%
17th iteration	39.71119134%	44.76190476%	55.23809524%
18th iteration	40.61371841%	43.33333333%	56.66666667%
19th iteration	38.98916968%	43.33333333%	56.66666667%
20th iteration	39.35018051%	44.28571429%	55.71428571%
Average (%)	39.95487365%	43.85714286%	56.14285714%
Standard deviation (%)	0.834969127%	0.576005066%	0.576005066%
95% confidence Intervals	From 39.5889323 to 40.320815	From 43.60730714 to 44.10958821	From 55.89041179 to 56.39530249

Second method

As shown in table II., as discussed before in the last section that the algorithm will repeat 20 more iterations depending on different values of uptime and bandwidth. Then compare the results of the iterations using false positive, false negative and percentage of malicious onion routers.

As shown in this table, the false negative and percentage of malicious onion routers get best results in all iteration. To identify if this method is better or not we are going to calculate the average and standard deviation of the three features.

TABLE II. TRUST VALUE OF THE ONION ROUTERS IN 20 ITERATIONS (SECOND METHOD)

	False positive	False negative	Percentage of malicious onion routers
1st iteration	52.52707581%	2.380952381%	97.61904762%
2nd iteration	54.33212996%	2.380952381%	97.61904762%
3rd iteration	52.52707581%	1.904761905%	98.0952381%
4th iteration	51.08303249%	2.380952381%	97.61904762%
5th iteration	51.44404332%	3.333333333%	96.66666667%
6th iteration	53.97111913%	1.904761905%	98.0952381%
7th iteration	52.52707581%	2.380952381%	97.61904762%
8th iteration	50.54151625%	1.904761905%	98.0952381%
9th iteration	53.6101083%	2.857142857%	97.14285714%
10th iteration	51.62454874%	1.904761905%	98.0952381%
11th iteration	50.36101083%	2.380952381%	97.61904762%
12th iteration	52.88808664%	1.904761905%	98.0952381%
13th iteration	53.79061372%	2.857142857%	97.14285714%
14th iteration	51.08303249%	2.380952381%	97.61904762%
15th iteration	52.88808664%	2.380952381%	97.61904762%
16th iteration	50.18050542%	2.380952381%	97.61904762%
17th iteration	53.06859206%	2.857142857%	97.14285714%
18th iteration	52.3465704%	1.904761905%	98.0952381%

19th iteration	50.18050542%	1.904761905%	98.0952381%
20th iteration	53.42960289%	2.380952381%	97.61904762%
Average (%)	52.22021661%	2.333333333%	97.66666667%
Standard deviation (%)	1.326567765%	0.405829346%	0.405829346%
95% Confidence Intervals	From 51.63882265 to 52.80161057	From 2.155470777 to 2.511195883	From 97.48880412 to 97.84452922

To know which method is the best there is two ways to do this:

The first way is to calculate F-measure [15] in both methods to know which method is the best as stated in section 4, equations (4), (5), and (6).

The F-measure of the first method is 0.572600783 and the F-measure of the second method is 0.781687021. So the best method according to F-measure is the second method.

The second way is according to the user, if the user wants to exclude the nodes that located in countries not trusted from the user so the first method is the best. If the user wants to put weights to all countries according to their truthfulness, if this country is untrusted put it low weight if it is trusted put it high weight so the second method is the best.

After that the directory server will include all active onion routers with their trust values, and the clients will choose the onion routers according to their trust values to build the circuit.

VII. Algorithm Complexities and Overhead

Uptime

According to the uptime the system ping on the nodes periodically to determine the uptime each time it pings the algorithm has O(1) asymptotic complexity. In addition, the algorithm sends 10 ping messages and receives the replies and therefore has 320 bits.

Bandwidth

According to bandwidth the nodes sending and receiving (n) packets so it has complexity of O (n).

Geographical Location

The complexity of this experiment is O(1) as the value of the geographical location depending on the location of the onion routers in the networks as discussed before the geographical location measured by two ways, the first way is to set zero for malicious nodes and 1 to non malicious nodes , the second way to put values ranged between 0 to 0.99 to the nodes.

DOS Attack

The complexity of this algorithm is O(1) as it didn't make any overhead on a network because only the behaviour of the malicious onion routers is affected the network.

Aggregation Algorithm

This algorithm quickly converges to a solution for example in our experiments the algorithm typically converged to a final solution in 7 iterations

Normalization Algorithm

The complexity of this algorithm is O(n^2)

VIII. Conclusion and Future Work

Conclusion

This paper presented a distributed algorithm for calculating trust of nodes in a TOR network. This was done by allowing nodes to compute local trust values of the nodes they deal with by recording measured uptime, bandwidth and whether or not a particular node participated in a DOS (or DDOS) attack on them. We show how the system is able to compute these values, and experimentally compute the best values for the adjustable parameters we use in computing these "trust metrics". We discuss the results of each of the experiments performed below. First, according to the uptime of the onion routers, the values of the false positive, false negative and percentage of malicious onion routers of the 20 iterations of the uptime are nearly close to each other, because the standard deviation is very small, when the standard deviation become smaller this mean that the values are close to the mean or average. Second, according to bandwidth of the onion routers, when the alpha of the EigenTrust aggregation algorithm increase the false positive, false negative and percentage of malicious onion routers get better results, so the best result of F-measure when alpha=0.99. Third, according to the geographical location of the onion routers, it had 2 methods so they can be chosen according to the users. If the users want to exclude the malicious onion routers from the network, they will use the first method and set the malicious onion routers to zero and the non-malicious onion routers to one. If the user want to set weight or trust value for each onion router according to their location so they will use the second method. Finally, according to denial of service attacks (flooding attacks) of the onion routers, when alpha (α) and number of pretrust onion routers of the EigenTrust aggregation algorithm and the alpha (α) and beta (β) of the DOS algorithm increase it get better results. So the best results of F-measure be when alpha of EigenTrust aggregation Algorithm α=0.97, the number of the pretrust onion routers=30 onion routers, alpha of the DOS algorithm α=2 and beta of the DOS algorithm β=0.998

Future Work

Our future work is to improve the performance of the onion routers by searching for more security issues to try to decrease the attacks on onion routers. In addition, we proposed to formally specify the fault tolerance of the proposed algorithm.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson. "TOR: the second-generation onion router," In *SSYM'04: Proceedingsof the 13th conference on USENIX Security Symposium*, Volume 13, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[2]TOR Status.http://TORstatus.blutmagie.de, 25-mar-2014.

[3] Juha Salo, "Recent Attacks On TOR," Aalto University, PhD thesis, 2012.

[4] Kevin Bauer1, Damon McCoy1, Dirk Grunwald1, Tadayoshi Kohno2, Douglas Sicker1, "Low-Resource Routing Attacks Against Anonymous Systems," Proceedings of the 2007 ACM workshop on Privacy in electronic society, Pages 11-20, 2007

[5] T. Dierks and C. Allen. The TLS Protocol — Version 1.0. IETF RFC 2246, January 1999.

[6]EntryGuards.https://www.TORproject.org/docs/faq.html.en, 19-mar-2014

[7] Entry Guards. https://blog.TORproject.org/category/tags/entry-guards, 19-mar-2014

[8] how-does-TOR-protect-against-an-attacker-just-running-thousands-of-nodes. http://security.stackexchange.com/questions/41542/how-does-TOR-protect-against-an-attacker-just-running-thousands-of-nodes, 19-mar-2014

[9]ExitPoLicies, https://www.TORproject.org/docs/faq.html.en#ExitPolicies, 23-mar-2014

[10] Sepandar D. Kamvar, Mario T. Schlosser, HecTOR GarciaMolina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," WWW '03 Proceedings of the 12th international conference on World Wide Web, ACM New York, Pages 640-651 ,2003.

[11]Power Iteration. http://mathfaculty.fullerton.edu/mathews//n2003/PowerMethodMod.html, 25-mar-2014.

[12] Vasilios A. Siris, Fotini Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE, Vol 4, Page 2050-2054, 2004.

[13] Robin Snader and Nikita Borisov, "Improving Security and Performance in the TOR Network through Tunable Path Selection," IEEE transactions on dependable and secure computing, vol. 8 no. 5, page 728-741,2011.

[14] L Overlier, P Syverson, "Locating hidden servers," SP '06 Proceedings of the 2006 IEEE Symposium on Security and Privacy,, Pages 100 – 114, 2006 IEEE Symposium on Security and Privacy, 2006.

[15] Yuanpeng J. Huang,Robert Powers, and Gaetano T. Montelione,"Protein NMR Recall, Precision, and F-measure Scores (RPF Scores): Structure Quality Assessment Measures Based on Information Retrieval Statistics," journal of the American chemical society, VOL 6, Page 1665-1674, 2005

[16] Location for IPs. www.ip2location.com, 21-june-2012