Adversaries monitoring Tor traffic crossing their jurisdictional border and reconstructing Tor circuits

Herman Galteland* and Kristian Gjosteen

Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology, Trondheim
{herman.galteland, kristian.gjosteen}@ntnu.no

August 29, 2018

Abstract

We model and analyze passive adversaries that monitors Tor traffic crossing the border of a jurisdiction an adversary is controlling. We show that a single adversary is able to connect incoming and outgoing traffic of their border, tracking the traffic, and cooperating adversaries are able to reconstruct parts of the Tor network, revealing user-server relationships. In our analysis we created two algorithms to estimate the capabilities of the adversaries. The first generates Tor-like traffic and the second analyzes and reconstructs the simulated data.

Keywords. Onion routing, anonymity, simulations.

1 Introduction

The Onion Routine (Tor) protocol [8] is a well-established onion routing system that tries to provide a low-latency communications channel while also defending against network-level adversaries trying to reveal who is talking to whom. It is well understood how the Tor network behaves when an adversary compromises a fraction of the onion routers, and in particular that if the entire network is monitored little or no security is left.

In this paper we analyze the power of (coalitions of) less powerful adversaries who do not monitor onion router traffic directly, but instead partition the network into jurisdictions and monitor traffic crossing from one partition into another.

These kinds of adversaries are interesting because they are real, in particular of the form of programs to monitor traffic crossing borders. In 2008 the Swedish parliament passed a bill allowing the Swedish National Defence Radio Establishment (Försvarets radioanstalt) to monitor both wireless and cable signals passing the Swedish border [29].

In 2016, the Norwegian government appointed a group of experts to investigate whether or not the Norwegian Intelligence Service should be allowed access to communication crossing the Norwegian border, similar to the Swedish National Defence Radio Establishment. The investigating report concluded that the Norwegian Intelligence Service should be allowed to monitor the Norwegian border [10], however, this has not yet been put into effect. It seems likely that other countries have or plan to have similar programs.

*This work is funded by Nasjonal sikkerhetsmyndighet (NSM), www.nsm.stat.no.
1.1 Related work

Formal analysis of the Tor protocol comes in two variants. The first use an abstract model of the protocol and gives security bounds based on a worst case adversary [2, 12, 14, 15, 17, 18]. The second includes a detailed description of the protocols in their analysis when proving the security bound [3, 20, 35]. Backes et al. use their ANOA framework [2] in their formal analysis [3], where they focus on the Tor path selection algorithm and how it can be improved.

Adversaries that are able to observe both ends of a Tor circuit is able to correlate the traffic and connect a user with the server it is communicating with [24, 27]. The literature has considered a range of different adversarial power: from adversaries that only controls a portion of the onion routers, adversaries that controls an autonomous system (AS) [13, 33], or an Internet exchange point (IXP) [25], or several ASes and IXPs [20, 26]. It has been shown that ASes can observe both ends of Tor circuits [35]. Tor path selection algorithms has been proposed to avoid detection from ASes [1, 11].

Encrypting messages hides its content, but not necessarily the message’s length. An adversary with access to timing, packet size, and directionality of packets, sent over an encrypted HTTP tunnel, can use this information to launch traffic analysis attacks to reveal the identity of the server and user [4, 6, 19, 22, 32, 37]. Countermeasures to traffic analysis attacks in the literature includes padding messages [7] and morphing Tor traffic to mimic traffic associated to other servers [37]. However, hiding the packet length is insufficient [9].

Using a Tor network simulator makes it possible to analyze the effectiveness of adversaries versus the Tor protocol. Examples of such programs are Shadow with Tor plugin [30] and The Tor Path Simulator [34].

In the network intrusion detection literature a stepping stone is an intermediate node used by an attacker to conceal his identity. Algorithms used to detect stepping stones analyses streams of traffic to confirm or reject the existence of intermediate nodes between the analyzed traffic streams [5, 36].

1.2 Our contribution

In this paper we model and discuss a specific adversary versus the Tor protocol. The jurisdictional adversary is similar to an adversary controlling AS(es) or IXP(s), however, the ASes and IXPs are typically located inside a jurisdiction whereas we consider a passive adversary that only monitors traffic crossing the border of a jurisdiction. Further, an adversary controlling an AS or an IXP would see all traffic inside their network whereas an adversary monitoring jurisdictional borders would not.

We simulate a Tor-like network and the adversaries using two algorithms, a Tor network simulator and a reconstruction algorithm. The reconstruction algorithm is given traffic data generated in the Tor simulator and use this information to reconstruct the simulated network and connect users with the server they are communicating with. The reconstruction algorithm uses the timing, packet size, and directionality of packets in the reconstruction, that is, traffic analysis. We chose to write our own Tor network simulator since we wanted to partition the nodes into jurisdictions and have data we could use in the reconstruction algorithm.

In the simulations we look at both fixed (padded) and variable packet size to see if hiding the packet size is a possible countermeasures to the traffic analysis done by the adversaries. We do not morph the Tor traffic since the adversaries are only interested in the existence of traffic and not what it looks like.
Algorithms used to detect stepping stones analyzes streams of traffic to find intermediate nodes between the streams. Similarly, our reconstruction algorithm attempts to connect stream of traffic between known onion routers to recreate circuits. The techniques used to detect stepping stones could be used to detect onion routers and connect its incoming and outgoing traffic. The difference between the stepping stone literature and our work is the adversary we are modeling and analyzing, where we assume that the location of all onion routers is already known and we want to connect Tor traffic to reconstruct Tor circuits.

1.3 Overview

In Section 3 we model the jurisdictional adversaries monitoring Tor traffic crossing their borders. This model is employed in the simulation algorithm.

In Section 4.1 we describe the Tor network simulation algorithm that generates Tor-like traffic. This traffic is used by the Tor reconstruction algorithm described in Section 4.2.

In Section 5 we show the results from our simulation and reconstruction algorithms. We include a benchmark, showing the dependability of the reconstruction algorithm, two parameter experiments, which show how different parameter choices affects the reconstruction results, and an example using real world data.

We conclude with a short discussion of the jurisdictional adversaries in Section 6, where we look at a possible path countermeasure against this specific adversary and give a summary of their adversarial power.

2 Background

2.1 Anonymity

In an anonymous communication network there are different notions of anonymity that an adversary might want to break. If an adversary can reveal the identity of a user then the sender anonymity is broken. If an adversary is able to distinguish between the scenario where a single user sends two messages from the scenario where two users sends a single message each then the sender unlinkability is broken. If an adversary connects a user with the server it is communicating with then the relationship anonymity is broken [28].

Sender anonymity is the strongest notion, since it implies both sender unlinkability and relationship anonymity. Neither sender unlinkability nor relationship anonymity implies sender anonymity [2].

2.2 Tor

The Onion Router (Tor) protocol is an anonymous communication protocol [8] that allows Tor users to hide which server they are communicating with. The Tor protocol uses intermediate nodes called onion routers to achieve anonymity, where each onion router only knows the identity of its neighboring nodes. A user establishes a circuit in the Tor network to communicate with a server, see Figure 1. The last onion router in a circuit

\[ U \rightarrow OR_1 \rightarrow OR_2 \rightarrow OR_3 \rightarrow S \]

Figure 1: Example Tor circuit with the default setting of three onion routers. \( U \) denotes a user, \( OR \) an onion router, and \( S \) a server.
communicates with the server on behalf of the user. For simplicity we assume that all circuits consist of one user, three onion routers, and one server, which is the default Tor configuration. Restricting circuits to contain only three onion routers is not essential for our reconstruction algorithm.

A Tor circuit is constructed incrementally, when a user creates a circuit it picks three onion router and establishes a shared key with each router. The user first performs a key exchange with the first onion router in the circuit, then a second key exchange with the second onion router, via the first, and finally the last key exchange with the third onion router, via the first and second onion router. Only when the circuit is complete can the user communicate with the server.

2.3 Notation

We denote jurisdictions as \( J \) or \( J_i \), for some index \( i \). Observing traffic sent between the two nodes \( N_1 \) and \( N_2 \) is denoted as observing the connection \((N_1, N_2)\). Observing two connections \((N_1, N_2)\) and \((N_2, N_3)\) which is connected by the common node \( N_2 \) is denoted as reconstructing the partial circuit \((N_1, N_2, N_3)\). Similar, two connections \((N_1, N_2)\) and \((N_3, N_4)\) which is believed to be connected is denoted as reconstructing the partial circuit \((N_1, N_2, N_3, N_4)\).

3 Modeling jurisdictional adversaries

The physical network of the Internet consists of routers and computers that are connected by cables. We can describe this network as a graph with the routers as nodes, the computers as leaf nodes, and the cables as edges, see Figure 2a. The logical network is a graph describing the flow of data between the computers. The difference of the two representations is that the physical network shows the location of the various nodes and edges, while the logical is a complete graph that only shows the computers nodes and a direct edge connecting them, see Figure 2b.

We can describe the graph induced by the Tor protocol as a logical network, where the graph describes the flow of data sent by the protocol. The graph describing the Tor network is called an overlay network, where the users, onion routers, and server are leaf nodes.
3.1 Tor overlay network with jurisdictional adversaries

In our overlay model of the Tor protocol we include adversaries monitoring all Tor traffic crossing the border of a jurisdiction they control. That is, we have a set of jurisdictions and each jurisdiction has an adversary that monitors its border. The users, onion routers, and servers are represented as nodes. Each node in the overlay model will be located inside one of the jurisdictions, where the jurisdictions are connected in a complete graph, as in Figure 2c.

This model is a simplification of the real world situation. Traffic between two nodes inside a jurisdiction could very well cross the jurisdiction’s borders in the physical network. In fact, since routing is dynamic, it could cross borders one day and not cross borders the next. Hence, the adversaries probably get less information in our model than in the real world.

Modeling the jurisdictions as fully connected is probably also too generous. In the real world, depending on the jurisdictions, this might not be the case. Again, the adversaries probably get less information in our model than in the real world.

3.1.1 Observable and reconstructible traffic

When a Tor circuit crosses the borders of a jurisdiction the adversary observes one of the following three cases. Case 1, traffic sent from a user or a server to an onion router, see Figure 3a. Case 2, traffic sent from an onion router to a user or a server, see Figure 3b. Case 3, traffic sent from an onion router to another, see Figure 3c. That is, the adversaries can observe the following connections crossing their border

\[(U, OR), \ (S, OR), \ (OR, U), \ (OR, S), \text{ or } (OR, OR).\]

Note that for each connection two jurisdictions are able to observe it. In each case shown in Figure 3, the left jurisdiction observes the traffic as outgoing and the right jurisdiction observes the traffic as incoming. Since Tor traffic usually visit only one onion router inside a country (jurisdiction) it should be possible for a jurisdiction to connect its observed incoming Case 1 or Case 2 traffic with an outgoing Case 2 or Case 3 traffic. An incoming and outgoing connection can be combined if their timestamps and packet size difference is small, that is, if they are most likely part of the same circuit.

In Section 4 we show that the adversaries are able to connect their observed incoming and outgoing connections if they observe enough traffic. Therefore, they can reconstruct the following two cases. Case 4, the incoming and outgoing traffic of the jurisdiction.

Figure 3: The three types of connections that the adversaries can observe. In each figure, the left jurisdiction observes outgoing traffic and the right jurisdiction observes incoming traffic.
Figure 4: The adversaries are able to connect their observed incoming and outgoing traffic, see Figure 3, to create partial circuits.

connects in a common node inside the jurisdiction and their timing and packet size fits, see Figure 4a. Case 5, the timing and packet size of the incoming and outgoing fits, see Figure 4b. The Case 5 connections need more traffic than the Case 4 since it is not connected via a common node. Given enough traffic, the adversaries are able to construct the following partial circuits

$$\text{(U, OR}_1\text{, OR}_2\text{),}\ (\text{OR}_1\text{, OR}_2\text{, S) ,}\ (\text{OR}_1\text{, OR}_2\text{, OR}_3\text{),}\ (U, \text{OR}_1\text{, OR}_2\text{, OR}_3\text{, S) ,}\ (\text{OR}_1\text{, OR}_2\text{, OR}_3\text{, S) ,}\ or\ (U, \text{OR}_1\text{, OR}_2\text{, S).}$$

The middle node(s) are located inside the jurisdiction and the edge nodes are located outside, see Figure 4.

4 Simulation and reconstructing

In our algorithms we assume that the adversaries are able to recognize Tor traffic, since onion routers usually only send Tor traffic to other nodes and all onion routers are known. Hence, we only generate Tor traffic.

Further, we assume that the adversaries have analyzed the distribution of timing and packet size patterns of Tor traffic. They will use this knowledge to statistically connect traffic entering and exiting their jurisdiction. We base our timing on previous measurements [23], and we look at both fixed and variable packet size (to study the effect of padding countermeasures).

With these algorithms we can simulate and analyze different kinds of Tor traffic, measure how many of the reconstructed circuits reveals the relationship, and how much of the simulated Tor network we can reconstruct.

4.1 Tor simulator

The simulation algorithm employ the overlay network model described in Section 3.1. The traffic generated in the simulation does not include any cryptographic computations, we only include timing, packet size, and direction of traffic.

The simulation starts by creating a selected number of onion router and server nodes and partition them into jurisdictions, that is, we initialize a universe. The simulation runs for a fixed amount of seconds, it only needs a few seconds to simulate half an hour of traffic in the Tor network.

While the simulation is running it randomly executes one of the four actions below according to a specified distribution. When an action is completed a new is picked and executed. The distribution is set before the simulation starts and can be changed while running.
Add user: adds a user node to one of the jurisdictions and creates a circuit for the user.

Remove user: picks an active user with a circuit and destroys its circuit, this makes the user inactive.

Create new circuit: picks an active user with a circuit, destroys its old circuit, and creates a new circuit for the user.

Send traffic: picks an active user with a circuit and sends one packet from the user to the server. The server responds with a random number of packets to the user (between 1 and 5).

If there is no active users in the network then the add user action is initiated. Note that the number of responds sent from the server is a pessimistic number compared to real world traffic, more responds would improve the reconstruction results.

When we create a Tor circuit we make it incrementally, as specified in Section 2.2. The onion routers does not have any specific classification (for example entry or exit) and can be in any position of a circuit.

Destroying a Tor circuit is only initiated by the user. The onion routers receives a tear down circuit command, which they forward to the next onion router in the circuit.

The simulation stores the communication information sent between two nodes as a connection, 

Connection = (sender, receiver, (time stamp, packet size)).

The connection contains the identity of the sender node, receiver node, direction of traffic, and the time stamp and packet size for when the traffic crossed a border.

When the simulation is concluded the connections are sorted to the jurisdictions. A connection is given to a jurisdiction J if sender ∈ J and receiver /∈ J or if sender /∈ J and receiver ∈ J. That is, the traffic crosses the borders of J. Each jurisdiction classify its connection as either incoming or outgoing, as discussed in Section 3.1.1.

4.2 Tor reconstructor

In the reconstruction we only analyze the data collected by jurisdictions that are going to cooperate in breaking the relationship anonymity of the Tor users. Each participating jurisdiction starts by connecting their incoming and outgoing connection and stores the processed data as partial circuits,

Partial circuit = (sender, (intermediate), receiver, (time stamps, packets)_{sender}, (time stamps, packets)_{receiver}, score).

The sender and receiver are the end points of the partial circuit. There can be none or more than one intermediate nodes. The time stamps and packets are stored as lists, where the sender list is traffic sent to and from the sender and, similarly, the receiver list is traffic sent to and from the receiver. The score is used to check if the partial circuit is real, that is, a part of a real circuit used in the simulation.

To create partial circuits the reconstruction start by looking for outgoing Case 1 connections and incoming Case 2 connections, that is, circuit endpoints which cannot be reconstructed with other observed connections. These connection are stored as partial circuits of length two and removed. These partial circuits will be used to supplement partial circuits of length three, by adding any additional time stamps and packet sizes.
To reconstruct Case 4 and 5 partial circuits each jurisdiction connects their observed incoming and outgoing connections which has not been removed, that is, incoming Case 1, outgoing Case 2, and Case 3 connections. For every incoming connection the jurisdiction check all outgoing connections if they fits with the incoming based on the timing and packet size. If the time difference of the incoming and outgoing connection is within the expected value it is given a score based on the time difference. The closer the measured time difference is to the expected time difference the higher score it is given. An incoming connection is only matched with the outgoing connection that results in the highest score.

The scoring system is used to check if a partial circuit is real. Whenever a jurisdiction stores a partial circuit it first check if it has already made a similar partial circuit previously. If it has then the existing partial circuit receives the additional time stamp, packet size, and the new score is added to the existing score. If not then a new partial circuit is stored.

It is easier to reconstruct Case 4 partial circuits since they share a common node, whereas a Case 5 partial circuit is only connected because it fits based on the timing. Hence, a Case 5 partial circuit needs a higher score for it to be real. Any partial circuit with a low score will not be used in the rest of the reconstruction. A partial circuit has a low score if it is less than 95% probability to be real. The low score value is determined from the algorithms.

When all observed connections has been processed, the reconstruction algorithm starts to connect partial circuit into circuits. The jurisdictions look for partial circuits that overlap, that is, share two nodes, and connects them into longer partial circuits. The better two partial circuits overlap the higher score the connected partial circuit is given. Overlapping partial circuits should have the same time stamps and packet sizes if they are part of the same circuit. The more packet sizes and time stamps that is present in both partial circuits the higher score the new partial circuit is given.

When all partial circuits has been processed the algorithms outputs a list of circuits which is evaluated. Reconstructed circuits are either real circuits, circuits which was used in the simulated Tor network, or imagined circuits, circuits which was not used. The output circuits are classified as either assumed real circuits, which will be accepted, or assumed imagined circuits, which will be discarded. The classification is based on the circuit’s score.

We verify the result of the reconstruction algorithm by checking the assumed real and assumed imagined circuits. We compare the circuits in the output list with the circuits used in the simulated Tor network. Assumed real circuits shown to be real and assumed imagined circuits shown to be imagined shows the reconstruction algorithm is dependable. On the other hand, if an assumed real circuits was imagined then the reconstruction algorithm creates false circuits and accuse users to communicate with server they did not. Similarly, if an assumed imagined circuit was real then either the user of the circuit did not send enough traffic or the algorithm is unable to detect all circuits. Both of these results disproves the algorithm’s dependability.

The real circuits are classified as either relationship revealing circuit, showing both the user and server, or a partial circuit that either shows only the user, only the server, or only onion routers. A partial circuit does not show both the user and server, however it can show the user breaking the sender anonymity.

The runtime of our implementation of the reconstruction algorithm is, essentially, quadratic in the traffic volume per time interval.
Table 1: Benchmark results showing the dependability of the reconstruction algorithm. The algorithm outputs circuits which was either used in the Tor network simulator, a real circuit, or a circuit which was not, an imagined circuit.

(a) Percentages are of all Tor circuits created in the Tor network simulator algorithm.

|                  | Median and 95% confidence interval (in percentage) | Fixed packet size |
|------------------|--------------------------------------------------|-------------------|
|                  | 6 jurisdictions 10 jurisdictions 15 jurisdictions |                   |
|                  |                                                  |                   |
|                  | 47.34 [46.04, 48.36] 62.62 [61.54, 64.24] 71.41 [70.61, 73.09] % relationships revealed |                   |
|                  | 77.19 [76.46, 78.08] 85.88 [85.34, 86.59] 89.97 [89.6, 90.77] % reconstructed |                   |
| Variable packet size |                                                  |                   |
|                  | 52.76 [51.78, 54.74] 66.21 [64.87, 68.17] 73.47 [72.66, 74.89] % relationships revealed |                   |
|                  | 81.80 [81.22, 82.56] 88.64 [88.06, 89.44] 91.62 [91.35, 92.07] % reconstructed |                   |

(b) Percentages are of all circuits recreated by the reconstruction algorithm.

|                  | Median and 95% confidence interval (in percentage) | Fixed packet size |
|------------------|--------------------------------------------------|-------------------|
|                  | 6 jurisdictions 10 jurisdictions 15 jurisdictions |                   |
|                  |                                                  |                   |
|                  | 11.15 [7.66, 12.53] 6.92 [5.68, 8.46] 5.20 [4.30, 6.35] % imagined circuits |                   |
|                  | 0.04 [0.0, 0.11] 0.03 [0.0, 0.11] 0.03 [0.0, 0.14] % imagined circuits discarded |                   |
|                  | 0.79 [0.53, 1.14] 0.75 [0.47, 1.03] 0.67 [0.50, 0.96] % real circuits discarded |                   |
| Variable packet size |                                                  |                   |
|                  | 0.12 [0.0, 0.31] 0.04 [0.0, 0.17] 0.0 [0.0, 0.11] % imagined circuits |                   |
|                  | 0.0 [0.0, 0.07] 0.0 [0.0, 0.08] 0.0 [0.0, 0.04] % imagined circuits discarded |                   |
|                  | 0.0 [0.0, 0.04] 0.0 [0.0, 0.04] 0.0 [0.0, 0.04] % real circuits discarded |                   |

5 Results

We have two measurements of the real circuits output by the reconstruction algorithm, a relationship revealing percentage and a reconstruction percentage.

The relationship revealing percentage shows how many of the real circuits shows both the user and the server. That is, how many circuits the adversaries have broken the relationship anonymity.

The reconstruction percentage shows how much of the simulated Tor network has been reconstructed. All, real, partial circuits of length longer than three is included in this percentage and contains partial circuits that shows either a user, a server, only onion routers, or both user and server.

We include four experiments in our results. First, we show benchmarks that demonstrate the dependability of the reconstruction algorithm. Second, we look at how the number of jurisdictions alter the results. Third, we change the network pattern in the Tor simulation to see how different types of Tor traffic affects the adversary’s reconstruction. Fourth, we show a real world example to see how many nonuniform jurisdictions is needed to successfully reveal relationships. Note that the jurisdictions have uniform size in the three first experiments.
The benchmark checks the dependability of the reconstruction algorithm. In these simulations all jurisdictions cooperate in revealing the relationships. We run six benchmarks, two for 6 jurisdictions, two for 10 jurisdictions, and two for 15 jurisdictions. For each jurisdiction size we run one with fixed packet size and one with variable. Each iteration of the benchmark is initialized with a fixed number of jurisdictions, a fixed number of nodes, and the network pattern is fixed. See Table 2 for the initialization values.

Table 1a shows how much of the Tor circuits the adversaries are able to reveal the relationships of and how much of the network they are able to reconstruct. Table 1b shows the dependability of the reconstruction algorithm. It looks at how many reconstructed circuits are imagined circuits, if we correctly discard the imagined circuits, and if we incorrectly discard the real circuits.

The fixed packet size simulations only use the time stamps in the analysis and does not achieve as good results as the variable packet size simulations, which use both the time stamps and the packet size. We see that having fixed packet size increases the probability of reconstructing imagined circuit and it is higher for smaller cooperation sizes. In the fixed packet size simulations the jurisdictions recreate between 5.2% and 11.15% imagined circuits, compared to the total number of recreated circuits, where the variable packet size simulations has almost no imagined circuits. Hence, padding traffic has an effect on the reconstruction. A more thorough analysis and improved algorithm might reduce the number of imagined circuits.

For a circuit to be discarded it needs to have a low score. There are almost no reconstructed circuits with a low score because we discard all partial circuits with a low score before we start connecting them.

5.2 Changing the parameters

We can initialize the algorithm with different parameter settings to see when it is difficult for the adversaries to reveal the relationship of the users. In the first experiment we gradually change the number of jurisdictions and in the second we gradually change the

---

**Figure 5:** Increasing the number of jurisdictions. The number of nodes and the network pattern is fixed.
network pattern. In these simulations all jurisdictions cooperate.

For each experiment we run one with fixed packet size and one with variable. In the results, there seems to be a small difference between the fixed and variable packet size results, however, the variable packet size simulations create circuits that are more likely to be real as shown in the benchmark, see Section 5.1.

5.2.1 Changing the number of jurisdictions

We increase the number of jurisdictions while keeping the number of nodes fixed, making a finer grid around the nodes which makes the cooperating jurisdictions observe more traffic. See Table 3 for the initialization values, the nodes are uniformly distributed amongst the jurisdictions. The results is shown in Figure 5.

A cooperation size of 10 jurisdictions is able to reach around 50% relationship revealing percentage. To reach above 80% the cooperation size of should be at least 30. The percentage stabilizes above 40 jurisdictions.

5.2.2 Changing network pattern

We increase the frequency the users create new circuits. We start at having the users creating only one, never recreating any new circuits, and gradually change until the users creates a new circuit for every packet sent. The number of nodes and jurisdictions is fixed, see Table 4 for the initialization values, the nodes are uniformly distributed amongst the jurisdictions. The results is shown in Figure 6.

We observe that the less frequent a users creates new circuits gives better results for the adversary. Since the more traffic the users send over the same circuit the more data the jurisdictions have to determine if the observed traffic is part of the same circuit or not. Similarly, the more frequent users create new circuits the less traffic is sent over the same circuits.
5.3 Real world example

We have seen that different parameter choices change the results, hence, we want to know how many real world countries is needed to efficiently break the relationship anonymity Tor users. Although we say real world the simulation would not picture the real world exactly since our model does not.

The number of jurisdictions in the simulation is the number of cooperating adversaries plus one. The extra jurisdiction is the rest of the world and it does not participate in the reconstruction algorithm, only in the simulation. The number of onion routers per jurisdiction is sampled from a Tor network status web page [31]. The remaining nodes, the users and servers, are distributed according to the distribution of onion routers such that the size of the jurisdictions do not change. See Table 5 for the initialization values, the number of onion routers per jurisdiction is in Table 6. Note that the nodes are not uniformly distributed in these experiment.

We picked the following three sets of countries, because the smaller sets are included in any of the larger sets, that is, an increasing set of cooperating jurisdictions. We denote the sets by their cooperation size: Size 5, Size 9, and Size 14. They are also known as Five Eyes, Nine Eyes, and Fourteen Eyes [16].

**Size 5** Australia, Canada, New Zealand, the United Kingdom and the United States.

**Size 9** Previous five plus Denmark, France, Netherlands, and Norway.

**Size 14** Previous nine plus Germany, Belgium, Italy, Spain, and Sweden.

There is a total of six, ten, and fifteen jurisdictions in the Tor network simulation (the
cooperation size plus the rest of the world). We also change the network pattern, as in Figure 6, to get a range of different results. The results of the real world example is shown in Figure 7.

We observe that for a cooperation size of five jurisdictions only reveal between 0% and 1% of the relationships, nine jurisdictions between 3% and 8%, and fourteen jurisdictions between 12% and 25%. The less frequent the users creates new circuits the higher the percentage is.

To put these results in perspective we include the expected max relationship percentage, which shows the expected maximal obtainable relationship revealing percentage given an optimal reconstruction algorithm and enough traffic. The expected max relationship revealing percentage is 17.03% for the Size 5 cooperation, 45.83% for the Size 9, and 69.32% for the Size 14 cooperation. This implies that better reconstruction algorithm could exist, however, even a perfect algorithm cannot achieve maximum relationship revealing percentage if the users does not send enough traffic.

The expected max relationship percentage is estimated using the simulator. We simulated the same scenario and check whether or not it the recreating algorithm would be able to reconstruct the circuits created in the simulated network. This process is repeated to get a good estimate. A circuit is reconstructible if the first and the last connection crosses a border, that is, the adversaries is able to see both the user and server of the circuit.

For the Size 5 cooperation the reconstructed percentage is between 26% and 33%, for Size 9 it is between 44% and 55%, and for Size 14 it is between 57% and 71%. Even the Size 5 set is capable of reconstructing close to one third of the simulated Tor network.

Although the fixed and variable packet size results looks the same, the constructed circuits is more likely to be real in the variable packet size situation, as showed in Section 5.1.

6 Discussions

6.1 Path selection countermeasure

In Section 3.1.1 we discussed the three cases the adversaries can observe. The only connection which do not show the a user or the a server is traffic is sent between onion routers. If the Tor circuit is created such that the traffic that crosses the jurisdictional borders is sent between onion routers then the adversaries cannot see the user or the server and are never able to connect them.

The following path selection prevents the jurisdictional adversaries breaking the relationship anonymity. A user \( U \in J_U \), for a jurisdiction \( J_U \), wants to connect to a server \( S \in J_S \), for a jurisdiction \( J_S \). The user chooses the onion routers as follows: \( OR_1 \in J_U \), \( OR_2 \in J \), for any jurisdiction \( J \), and \( OR_3 \in J_S \). This means that the entry node is inside the jurisdiction of the user and the exit node is inside the jurisdiction of the server. The second onion router \( OR_2 \) can be chosen uniformly at random from any jurisdiction. The jurisdictions observing traffic sent on this circuit would only see \((OR, OR)\) connections and are unable to reveal the relationship, see Figure 8.

To prevent the adversaries breaking the sender anonymity the third node \( OR_3 \) can be chosen uniformly at random from any jurisdiction. That is, the user chooses the onion routers as follows: \( OR_1 \in J_U \), \( OR_2 \in J \) and \( OR_3 \in J' \), for any jurisdiction \( J \) and \( J' \).

Note that these path selections only avoids the jurisdictional adversaries, it is possible that other types adversaries could break the sender and/or relationship anonymity if the
Figure 8: Path selection where the jurisdictional adversaries are unable to connect the user $U$ with the server $S$ since they only observe $(OR, OR)$ connections.

users use this path selections.

6.2 Passive global adversaries

We claim that the best attack the jurisdictional adversary can do is to passively observe Tor traffic. Tor uses a TLS connection between Tor entities, which provides confidentiality and message integrity [7], which implies that Tor is IND–CCA [21]. Any active attack against messages sent between Tor entities will be detected and prevented, and the best attack the adversaries can do is, therefore, to passively observe Tor traffic (and possibly stop traffic).

An active attacker that corrupts onion router can potentially retrieve more information than the passive jurisdictional adversaries. As we mention in Section 6.1, if the observed traffic is sent between two onion routers then the passive adversary cannot reveal the user or the server. An active attacker corrupting onion routers inside the jurisdiction could. The jurisdictional adversary is blind for any traffic inside the jurisdiction, while an active attacker can have nodes inside the jurisdiction.

The jurisdictional adversaries indirectly monitor all onion routers inside its jurisdiction. If the jurisdictional adversaries cooperates in reconstructing Tor circuits they quickly become global, since they indirectly monitor a large portion of the Tor nodes. In addition, a large set of jurisdictions has the power to reveal the relationship of a circuit if they choose to do so.

References

[1] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. Lastor: A low-latency as-aware tor client. IEEE/ACM Trans. Netw., 22(6):1742–1755, December 2014.

[2] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. In Proceedings of the 2013 IEEE 26th Computer Security Foundations Symposium, CSF ’13, pages 163–178, Washington, DC, USA, 2013. IEEE Computer Society.

[3] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (nothing else) mator(s): Monitoring the anonymity of tor’s path selection. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS ’14, pages 513–524, New York, NY, USA, 2014. ACM.

[4] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In Proceedings of the 5th International Conference on Privacy Enhancing Technologies, PET’05, pages 1–11, Berlin, Heidelberg, 2006. Springer-Verlag.
[5] Avrim Blum, Dawn Song, and Shobha Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *Recent Advances in Intrusion Detection*, pages 258–277, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[6] Thomas Demuth. A passive attack on the privacy of web users using standard log information. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[7] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246, RFC Editor, August 2008. [http://www.rfc-editor.org/rfc/rfc5246.txt](http://www.rfc-editor.org/rfc/rfc5246.txt).

[8] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM’04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[9] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP ’12, pages 332–346, Washington, DC, USA, 2012. IEEE Computer Society.

[10] Digitalt grenseforsvar. [https://forsvaret.no/etjenesten/dgf](https://forsvaret.no/etjenesten/dgf). Accessed: 2017-10-04.

[11] Matthew Edman and Paul Syverson. As-awareness in tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS ’09, pages 380–389, New York, NY, USA, 2009. ACM.

[12] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM’09, pages 33–50, Berkeley, CA, USA, 2009. USENIX Association.

[13] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES ’04, pages 66–76, New York, NY, USA, 2004. ACM.

[14] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. A model of onion routing with provable anonymity. In *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security*, FC’07/USEC’07, pages 57–71, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Trans. Inf. Syst. Secur.*, 15(3):14:1–14:28, November 2012.

[16] Five eyes. [https://en.wikipedia.org/wiki/Five_Eyes](https://en.wikipedia.org/wiki/Five_Eyes). Accessed: 2018-06-07.

[17] Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES ’13, pages 225–236, New York, NY, USA, 2013. ACM.
[18] Alejandro Hevia and Daniele Micciancio. *An Indistinguishability-Based Characterization of Anonymous Channels*, pages 24–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[19] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET’02, pages 171–178, Berlin, Heidelberg, 2003. Springer-Verlag.

[20] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’13, pages 337–348, New York, NY, USA, 2013. ACM.

[21] Jonathan Katz and Moti Yung. *Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation*, pages 284–299. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[22] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS ’06, pages 255–263, New York, NY, USA, 2006. ACM.

[23] Sebastian Müller, Franziska Brecht, Benjamin Fabian, Steffen Kunz, and Dominik Kunze. Distributed performance measurement and usability assessment of the tor anonymization network. *Future Internet*, 4(2):488–513, 2012.

[24] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP ’05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[25] Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, PET’07, pages 167–183, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Rishab Nithyanand, Oleksii Starov, Phillipa Gill, Adva Zair, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. In *Proceedings of the Network and Distributed Security Symposium - NDSS ’16*. Internet Society, February 2016.

[27] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP ’06, pages 100–114, Washington, DC, USA, 2006. IEEE Computer Society.

[28] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2009.

[29] Förstärkt integritetsskydd vid signalspaning. regeringen.se/rattsdokument/proposition/2009/05/ Accessed: 2017-10-04.

[30] shadow-plugin-tor wiki. https://github.com/shadow/shadow-plugin-tor/wiki. Accessed: 2018-02-19.

[31] Tor Network Status. Tor network status. https://torstatus.blutmagie.de/index.php?SR=CountryCode. Accessed: 2017-10-27.
Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, SP ’02, Washington, DC, USA, 2002. IEEE Computer Society.

Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing attacks on privacy in Tor. In Proceedings of the 24th USENIX Security Symposium, August 2015.

The tor path simulator, torps. https://github.com/torps/torps. Accessed: 2018-06-07.

Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. An empirical evaluation of relay selection in tor. In NDSS, 2013.

Xinyuan Wang, Douglas S. Reeves, and S. Felix Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In Proceedings of ESORICS 2002, pages 244–263, October 2002.

C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In 2008 IEEE Symposium on Security and Privacy (sp 2008), pages 35–49, May 2008.

A Initialization values for simulations

Table 2: Initialization values for the benchmark simulations.

| Description              | Value(s)                      |
|-------------------------|-------------------------------|
| Iterations              | 50                            |
| Tor simulation runtime  | 5s                            |
| Setup nodes             | ORs: 6000                     |
|                         | Ss: 20000                     |
| Jurisdiction size       | 6                             |
|                         | 10                            |
|                         | 15                            |
| Network pattern         | New circuit: 10%              |
|                         | Add user: 10%                 |
|                         | Remove user: 10%              |
|                         | Send traffic: 70%             |
Table 3: Initialization values for the changing jurisdiction size simulations. The jurisdiction size starts at five and is increased by five in every fifth iteration.

| Description                  | Value(s)                  |
|------------------------------|---------------------------|
| Iterations                   | 50                        |
| Tor simulation runtime       | 5s                        |
| Setup nodes                  | ORs: 6000                 |
|                              | Ss: 20000                 |
| Jurisdiction size            | At start: 5               |
|                              | Size increase: 5          |
|                              | Step: 5                   |
| Network pattern              | New circuit: 10%          |
|                              | Add user: 10%             |
|                              | Remove user: 10%          |
|                              | Send traffic: 70%         |

Table 4: Initialization values for the changing the network pattern simulations. The New circuit increase value is 10 percentage points and the send traffic decrease value is 10 percentage points. The network pattern is changed in every fifth iteration.

| Description                  | Value(s)                  |
|------------------------------|---------------------------|
| Iterations                   | 45                        |
| Tor simulation runtime       | 5s                        |
| Setup nodes                  | ORs: 6000                 |
|                              | Ss: 20000                 |
| Jurisdiction size            | 20                        |
| Network pattern              | New circuit: 0%           |
|                              | Add user: 10%             |
|                              | Remove user: 10%          |
|                              | Send traffic: 80%         |
|                              | New circuit increase: 10 pp|
|                              | Send traffic decrease: 10 pp|
|                              | Step: 5                   |
Table 5: Initialization values for the real world simulations. The New circuit increase value is 20 percentage points and the send traffic decrease value is 20 percentage points. The network pattern is changed in every fifth iteration.

| Description            | Value(s)                        |
|------------------------|---------------------------------|
| Iterations             | 25                              |
| Tor simulation runtime | 5s                              |
| Setup nodes            | $ORS$: 6613, $SS$: 20000        |
| Jurisdiction size      | See Table 6                     |
| Network pattern        | New circuit: 0%                 |
|                        | Add user: 10%                   |
|                        | Remove user: 10%                |
|                        | Send traffic: 80%               |
|                        | New circuit increase: 20 pp     |
|                        | Send traffic decrease: 20 pp    |
|                        | Step: 5                         |

Table 6: List of selected jurisdictions with their number of nodes and cooperating set. Note that Size 5 is included in Size 9, and Size 5 and Size 9 is included in Size 14. The numbers was sampled at 2017-11-02 from [31].

| Jurisdiction     | Number of ORs | Set    |
|------------------|---------------|--------|
| Australia        | 50            | Size 5 |
| Canada           | 262           | Size 5 |
| New Zealand      | 14            | Size 5 |
| UK               | 258           | Size 5 |
| USA              | 1092          | Size 5 |
| Denmark          | 48            | Size 9 |
| France           | 910           | Size 9 |
| Netherlands      | 508           | Size 9 |
| Norway           | 52            | Size 9 |
| Belgium          | 24            | Size 14|
| Germany          | 1331          | Size 14|
| Italy            | 66            | Size 14|
| Spain            | 54            | Size 14|
| Sweden           | 190           | Size 14|
| **Total**        | **6613**      |        |