UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Improving anonymity in Tor through diversity

Dissertation presented by
Robin DESCAMPS

for obtaining the Master's degree in
Computer Science

Supervisor(s)
Olivier PEREIRA

Reader(s)
Olivier BONAVENTURE, Florentin ROCHET , Ramin SADRÉ

Academic year 2017-2018

If you do not know your enemies nor yourself,
you will be imperiled in every single battle.

SUN TZU (5th century BC)

Abstract

Tor is the most popular anonymous communication system nowadays, that set up a excellent compromise between anonymity and performance. However, Tor is by design vulnerable to end-to-end correlation attacks. Many techniques have been and are being developed in order to mitigate these attacks. This master thesis objective is to define the utility of a new relay, and to give a measure of its contribution to the diversity of the network. This is achieved with a modified version of the TorPS , which simulate the Tor network against both relay and network adversaries. More diversity means less end-to-end correlation attacks possible by the latter. At the end, we would like the volunteers to be aware of the utility they bring to the Tor network security with a certain relay configuration.

Acknowledgments

First and foremost I would like to offer my gratitude to Florentin Rochet, for his precious help for both conceptual and technical difficulties I encounter along this year, and for his continuous support to make this research possible.

I also give many thanks to my advisor Pr. Olivier Pereira, for his advices in methodology and his help in the different stages of this thesis.

Thanks also to the Tor community for the feedback and the support they express about this master thesis, especially to Iain Learnmonth, Tim Wilson-Brown (teor), grarpamp and beastr0 from the tor-dev team.

Finally, thanks to my family and friends for their support.

Contents

Introduction

Obtaining anonymity on Internet is a challenge that encompasses a lot of researches and solutions with all their compromises. Tor is the most popular system nowadays, because of its strengths, but is far to be perfect and has its share of drawbacks. Among them, the greater threat: the end-to-end correlation attack. The purpose of this research is to mitigate this weakness by increasing the network diversity under several assumptions and path selection algorithms. We will attempt to evaluate the diversity in Tor relays through well defined metrics regarding the most threatening adversaries nowadays.

We begin by reviewing all the solutions to get anonymity that exist today, to highlight the differences they have with Tor. We will see that the diversity definition we will give is specific to Tor, and cannot be applied to other anonymous communication systems.

Next, we detail the the way Tor works, in particular how paths are defined in the network. We will also present several solutions presented in the literature, aiming to improve the existing path selection algorithm.

Afterwards, we will explore previous work done regarding diversity in Tor by establishing a state of the art. We will see that in most previous research diversity was poorly defined, and that there is a lack of info about relays dispersion in the network.

Once all the important concepts highlighted, we start defining our methodology to measure diversity in the Tor network and the utility attached to (a) certain(s) node(s) configuration, and we conduct experiments on a certain period of time in Tor, to see which results we get from our metrics. We analyze them, and we give the best and the worst configuration for relay to deploy in the network in order to improve its diversity and thus its security.

Finally, we evoke the improvements and further work that can be done in order to complete and continue this research on diversity.

1 Anonymity on Internet

In order to achieve anonymous communication over the Internet, several strategies are possible, and various solutions have implemented them. We must know that these latter always define a trade-off between anonymity and latency. Therefore, we can classify anonymity systems into two categories: *high-latency* and *low-latency*.[17]

The high-latency systems have the objective of maximizing the anonymity, but that comes with a cost in terms of usability, the network being slow in terms of response. Consequently, interactive tasks such as SSH or web browsing experience too much lag to be reasonably usable by the users. Babel [22] and Mixminion [14] are examples of such systems, that are meant to exchange mails anonymously.

The low-latency systems are the ones that we are interested in. In contrast to the high-latency systems, they offer more usability but less anonymity. We begin by briefly describe the existing low-latency systems, to highlight the differences with Tor, that belongs to this category.

1.1 Low-latency systems

The simplest design to hide a user identity is to use an intermediate. Such intermediates are called anonymous proxies. Among all the benefits the users can obtain from such proxies is anonymity, destinations cannot know the identity of the clients. However, this solution is vulnerable if an adversary can observe the traffic in this single point, it therefore requires the so-called proxy trust.

The alternative is to build distributed-based design. First, we have the P2P design, where the nodes that generate traffic are the same as the ones that handle this traffic. These systems are mainly used to communicate inside the network, among the nodes belonging to the system. Examples are Tarzan [20], MorphMix [36], Crowds [35], Herbivore [21] and Freenet [13].

These systems also have a circuit-based design. Each time an anonymous communication is established, it is associated with a circuit built among the network's nodes. In such a design, the nodes only know their "neighbors" and are thus unable to match both ends of the communication locations. Tor use this design, particularly the *onion routing* design (that we detail later). Other approaches are possible, like the message-based design. Each communication is divided into messages, that are mixed and possibly delayed by the network. This approach is not vulnerable to end-to-end correlation attacks, unlike Tor (we will see why later), but comes with other disadvantages: a vulnerability to DoS attacks for example. I2P [48] is an application of this principle, which applies the *garlic routing*: instead of handling a stream with onion layers (like Tor), I2P handles its messages into cloves, which consist of an encrypted group of messages sent together.

A last design choice that must be taken into account is the protocol layer that we render anonymous. This consists of a trade-off between anonymity and flexibility. A system highly anonymous can use a specific protocol at its advantage and can, for example, hide identification information found in data units. On the other hand, a system highly flexible can handle several protocols, but will require extra mechanisms, not always portable to all operating systems, in order to render anonymity. Tor chose to support the TCP protocol, which is a half-way approach to this compromise.

2 The Tor network

We describe here the Tor network in more details, how it works, its purposes, assumptions and limitations. We present the ABWRS, the Tor selection circuit used nowadays, as well as other selection circuit algorithms, that aim to improve resilience against known attacks, and/or to provide more performance.

2.1 The network design

Tor is a popular low-latency anonymous system, based on volunteer relays across the world. In January 2018, it consists of about 6000 relays and 2 000 000 users that exchange 100Gbit/s [6]. It is an overlay TCP network that uses *onion routing* to create anonymity for its users [17]. A client that wishes to connect to a certain destination set up a circuit in the Tor network, consisting of three nodes chosen among the volunteer relays: a *guard* node (the entry), a *middle* node and an *exit* node. Each user circuit in Tor starts from the client to the guard, then passes by the middle node, then by the exit node to finally reach its destination. This circuit is the same in both the forward and reverse stream of the TCP connection. The client (or the destination when it responds) encrypts its data three times, resulting in three layers of encryption (what is called *onion encryption*). Each of the nodes (guard, middle and exit) in a circuit decrypts one layer, since each layer is encrypted with a different key, corresponding to the keys that each node knows. The last node sends the decrypted data to the destination (see Figure 1). Consequently, the client identity is only visible between the client and the guard node, and the destination identity is only visible between the exit node and the destination itself.

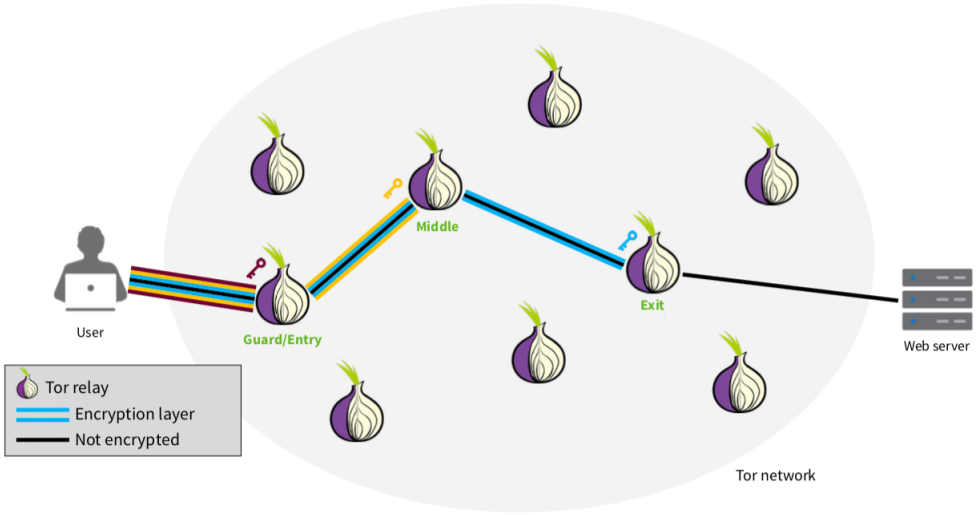

Figure 1: The Tor network

2.2 Assumptions, objectives and threat model

We now summarize the assumptions and goals of the Tor project [17]. Tor is meant to be simple in its design: there are not packet re-ordering, padding or delay, and no protocol normalization is provided (only TCP streams are considered, nothing more). Tor also needs to be usable and easy to deploy. Indeed, its strength relies first on its users: from the moment when a user hides among other users, the more they are, the better the anonymity is. The same goes for relays deployed by volunteers. The easier the relays are to set up, the more the volunteers are encouraged to

run a relay in the Tor network. More relays means more anonymity for the Tor users in general, although their contribution depends on their configuration. A last point to highlight is the possibility for the nodes to have a custom policy: each node can chose which hosts and ports it allows for its connections. This is a strong requirement in a volunteer-based system, since about half of the Tor traffic is considered illegal [10]. Exit nodes are thus sensitives, and may be exposed to complains or prosecutions.

In order to achieve the higher anonymity possible, we must define the adversary we consider in this undertaking. Often, anonymous systems define a global passive adversary, which is a kind of "omniscient" adversary, able to see all the exchanged traffic. Tor does not consider this adversary since it is unrealistic. Instead, we will basically consider a more practical adversary that can observe some part of the Tor traffic. We will detail later more precise adversary models that are considered in order to analyze diversity in the Tor network. Of course, other actions are considered feasible by adversaries that threat the network, such as denial of service attacks, performing antisocial activities from corrupted nodes to shut them down,... But many of these existing attacks are inefficient against the Tor network [17].

2.3 Main attacks

We present here the most threatening attacks against the Tor network in order to de-anonymize its users, by realistic adversaries described above.

2.3.1 End-to-end traffic correlation attack

This the most studied attack, because it will always be possible to apply end-to-end correlation to the Tor network, by design. If an adversary is able to observe the traffic between a client and its guard, as well as the traffic between its exit and its destination, he can then de-anonymizes the stream, by correlating the traffic volume and timing at both ends. There are thus several possibilities for an attacker. Either controlling the guard and the exit of a circuit or controlling an entity able to observe Tor traffic from/to client and from/to destination. Or, controlling a guard and an authority able to observe the traffic from/to the destination, or vice versa.

We see then that there are two ways to perform this attack: either controlling nodes (we can add ones or corrupt existing ones) or monitoring a part of the Internet (by controlling an authority such as an Autonomous System or a country). Research made about this attack thus only try to develop strategies to mitigate them, not to counter them, because it is impossible regarding the Tor design definition. This is the attack we will focus on here in order to define a diversity improvement in the Tor network.

2.3.2 RAPTOR

Routing Attacks on Privacy in Tor [43] is a recent research that present extensions to the end-to-end traffic correlation attack, by taking advantages of the BGP asymmetric routing between the Autonomous Systems (AS). In order to increase the Tor traffic quantity that an AS would normally, it has several ways to hijack traffic thanks to BGP mechanisms, and can manage to deflect Tor traffic, and thus manage to correlate more traffic and de-anonymize more users.

These attacks motivate the consideration of an AS adversary that consist a serious threat to the Tor network. When we will define the considered adversaries to improve diversity, We will see that the AS is the more important one.

2.3.3 Guard placement attack

This attacks consists for an adversary to deploy a guard relay in a specific location, regarding the Autonomous System and/or the geographic position. The location is chosen to target a set of clients (located in a defined area) that will choose this guard relay location more often than others, their anonymity being threatened by this adversary. The guard placement attack can become a problem with the DeNASA selection algorithm, that we will explain in the next sections.

2.4 Selection circuits

Choosing three nodes to build a circuit in Tor is not as simple as it appear to be. The path selection algorithm used nowadays is the Adjusted Bandwidth Weight Random Selection (ABWRS). Since the ABWRS is applied, several propositions have been made in the literature to improve users anonymity against different adversaries.

We will first explore the first selection processes used previously by Tor and explain the ABWRS algorithm. We will then describe other circuit selection designs, in particular the Waterfilling and DeNASA approaches.

2.4.1 First historical approaches

The first path selection algorithm used by Tor at its beginnings was the Simple Random Selection (SRS) [45]. The notion of *guard* nodes do not exit yet, we only have two sets of nodes (*middle* and *exit*). Nodes were simply selected randomly among all, in a uniform way. Although this strategy was considered secure regarding the onion routing model, it gives a poor performance in reality because of the different bandwidths offered by the nodes.

The logical consequence is then to weight all the nodes according to their bandwidth. This gives the second Tor approach: the Bandwidth Weight Random Selection (BWRS), a selection path algorithm in which the probability to select a node is directly proportional to its bandwidth. Some time later, the *guard* mechanism will be put in place. In this strategy, nodes with a great amount of bandwidth become a threat for the network. Furthermore, Tor has still performance issues because of the existing unbalance between the three node positions.

2.4.2 The Tor selection circuit: ABWRS

Tor uses nowadays the Adjusted Bandwidth Weight Random Selection (ABWRS) algorithm, which consists of "adjusting" the bandwidths announced by its nodes in order to maximize the network performance. Concretely, it is achieved through the resolution of the following system of equations [16]:

$$\begin{cases} W_{gg}.G + W_{gd}.D = M + W_{md}.D + W_{me}.E + W_{mg}.G & (guardbw = middlebw) \\ W_{gg}.G + W_{gd}.D = W_{ee}.E + W_{ed}.D & (guardbw = exitbw) \\ W_{ed}.D + W_{md}.D + W_{gd}.D = D & (consistency) \\ W_{mg}.G + W_{gg}.G = G & (consistency) \\ W_{me}.E + W_{ee}.E = E & (consistency) \end{cases}$$

Knowing that:

- W_{xy} is the weight for choosing a node with the flag y for the position x;
- A node flag can be Guard (g), Middle (m), Exit (e) or Guard and Exit (d);
- A position can be Guard (g), Middle (m) or Exit (e);
- Y is the total bandwidth for the nodes with flag y. Each Y is initialized to 1;
- Y can be Guards (G), Middles (M), Exits (E) or Guards and Exits (D).

The system ensures through the two first equations that the bandwidth allocated to the *guard* nodes is equal to the bandwidth allocated to the *middle* nodes and equal to the *exit* nodes. The other equations verify that all the bandwidth of the nodes is indeed allocated.

However, this system lacks two constraints which describe the network load. The Tor specification presents several possibilities where one or two position is (are) scarce compared to the other(s). In reality (so far), the Tor network is always in the case 3a(E=S), which describes a scarcity of the exit bandwidth compared to the other positions. This is due to their sensitivity regarding the law enforcement as we stated before. In this situation, we have the following additional constraints:

$$\begin{cases} W_{ee} = W_{ed} = 1 & \\ W_{md} = W_{gd} = W_{me} = 0 & \\ W_{mg} = 0 & if(G < M) \\ W_{mg} = \frac{G-M}{2.G} & else \\ Wgg = 1 - Wmg & \end{cases}$$

These equations maximize the bandwidth allocated to the exit position. This means that all the nodes with the flag guard and exit are devoted to the exit position. This is illustrated by the two first equalities, while the last equations indicate how to allocate the bandwidth for the guard and the middle position, in function of the guard bandwidth quantity (if it is scarce compared to the middle bandwidth or not).

All of these constraints are solved, and voted in a network status document. The directory authorities (special servers in the Tor networks) advertise these weights each hour. When a client builds a new circuits, he selects its three nodes according to the probabilities described in the last document published. Nevertheless, there remain a few constraints to satisfy:

- A node cannot be selected twice in the same circuit;
- The exit policy must accept the destination the client tries to connect to (IP and port);
- Each node address cannot belong to the same /16 subnet in the same circuit;
- Each node address cannot belong to the same family (same organization) in the same circuit.

A last point to mention is the *guard* set mechanism. When a client establishes a connection to the Tor network, a set of guards will be assigned to him. Nowadays, a client has a set of one guard (instead of three before 2014), that is changed after 90 days. The purpose is to limit points of observation for adversaries, to mitigate the end-to-end correlations, by avoiding a change of guard too often for clients [18]. However, the performance suffers from this strategy, since guards are not static, and can have failures. Discussion is still open today, a set of 1, 2 or 3 guards have each their advantages and their flaws [24].

2.4.3 Selection circuit algorithms proposals

We briefly introduce path selection methods from the literature, that we do not have the occasion to use in the context of the diversity we attempt to derive here, either because they are difficult to simulate or because their ineffectiveness has been demonstrated since.

Coordinate [39] is an attempt to increase Tor performance without harming the anonymity of its users. It is a link-based method that define virtual coordinates for each nodes. Each circuit defines its distance by summing the euclidean distances between each of the path nodes. The client is then able to choose the circuit with the shortest (virtual) distance. This method has been showed effective [46], but is difficult to simulate with our methodology (see next sections).

Snader/Borisov [42] is a method that aim to refine the Tor path selection with a tuneable parameter, proposing to the client to define its own compromise between anonymity and performance. Without giving the details, the SB strategy allows to balance between a uniform selection of relays and a high performance (select the nodes with the highest bandwidth possible). We do not take this method into account because of its definition: it increasing anonymity with a cost of performance (we would like the latter unchanged).

Congestion-aware selection [47] proposes to select circuits in Tor with the lowest congestion possible, based on round trip time measurements. In the details, there are short-term and long-term methods proposed, with records of the lowest round trip time measured for each circuits, compared to the average of the last (recent) measures when the circuit is selected. Basically we switch the circuit when the difference between this average and the lowest measure is above a certain threshold. This method has been demonstrated to be effective in a congestion context within Tor. It remains however difficult to measure when Tor is simulated.

LASTor [9] has two objectives: reduce the probability to have an AS on both sides of the established connections, and reduce latency by using geographic distance between nodes. Unlike Coordinate, LASTor uses the GeoIP service to map the relays with their locations. As for the AS avoidance, iPlane datasets are used. Recent evaluations [46] tend to demonstrate however its negative impact on Tor performance when realistic traffic of the network is taken into account.

2.4.4 Waterfilling selection circuit

The Waterfilling Adjusted Bandwidth Weight Random Selection (WABWRS) is a recent path selection algorithm proposition [37] which aims to reduce the threat of nodes with a guard flag that hold a great amount of bandwidth, by defining a threshold, a "water level" that limits the contribution of any node with the *guard* flag to the guard position. To compensate this limitation, the "little" guard-flagged nodes are fully devoted to the guard position. Of course, this algorithm proposal is only relevant when we are in the situation described earlier, namely with a scarcity of the exit-flagged nodes bandwidth. This principle can nevertheless be applied, in a symmetric way if, do we ever know, we have the opposite situation: a scarcity of the guard-flagged nodes. We will start from the constraints corresponding to the 3a(E=S) Tor specification case described in the previous section, and recompute weights according to the waterfilling principle we just described. We have the following system:

$$\begin{cases} W_{gg_i}.BW_i = W_{gg_{i+1}}.BW_{i+1} & \forall i \in (1, N) \\ W_{gg_i} = 1 & \forall i \in (N+1, K) \\ 0 \leq W_{gg_i} \leq 1 & \forall i \in (1, K) \\ \sum_{i=1}^{K} W_{gg_i}.BW_i = W_{gg}.G \end{cases}$$

We assume that we have K nodes with a guard flag in total. The first equation computes individual weights for guard nodes above the water level, knowing that N is the pivot node, i.e. the last node above that threshold. All nodes from $N+1$ to K are the "little" nodes that will provide all their bandwidth to the guard position, this is illustrated by the second equation. The two last equations stand for consistency, verifying that each weight is indeed between 0 and 1 (equation 3), and that all the bandwidth we devote the guard position with waterfilling is equal to the bandwidth we devote to the guard position when we use the ABWRS (equation 4). This is important to underline it, the waterfilling approach only distribute guards bandwidth differently, it does not modify its quantity. We can visualize this equality in Figure 2 where both gray areas have the same size.

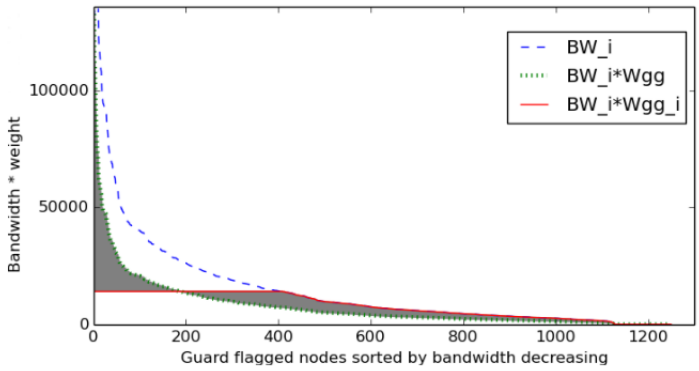

Figure 2: Waterfilling strategy applied to nodes bandwidths, with the consensus from 25th May 2015

It has been demonstrated [37] that the Waterfilling path selection increases the Tor security against a *relay adversary*, namely an adversary able to add or corrupt a set of nodes in the Tor network. Furthermore, compared to the original Tor path selection algorithm, Waterfilling shows similar performances. It is thus a better method to mitigate end-to-end correlation attacks against *relay adversaries* than the ABWRS original method.

2.4.5 DeNASA: Destination-Naive AS-Awareness

Destination-Naive AS-Awareness (DeNASA) [11] is another approach that complete the vanilla Tor path selection algorithm like the Waterfilling. However, the adversary we consider here is different: it is about *Autonomous System (AS)* adversaries. The goal is again to mitigate end-to-end correlation, but against top tier-1 AS able to observe traffic both between clients and guards, and between exits and destinations. Qiu and Gao inference tool is used here to derive the AS paths [30].

DeNASA is divided into two methods, namely *g-select* and *e-select*. G-select simply consists in banishing a set of ASes on the path between clients and guards. First, we need to compute the *Suspected AS list*, i.e. the ASes we encounter the most when we establish a connection to the Tor network (see Figure 3). The destinations were assumed to be random among the top 200

websites according to Alexa ranking. Then we select ASes from the suspect list to be avoided in the path between the client and the guard. This selection needs a careful decision, because removing too much tier-1 ASes can make the guard placement attack more threatening (see the attacks described above). It was demonstrated that the optimal trade-off between the two was to define a set of two top tier-1 ASes: 3356 (Level 3 Communications, Inc.) and 1299 (Telia Company AB).

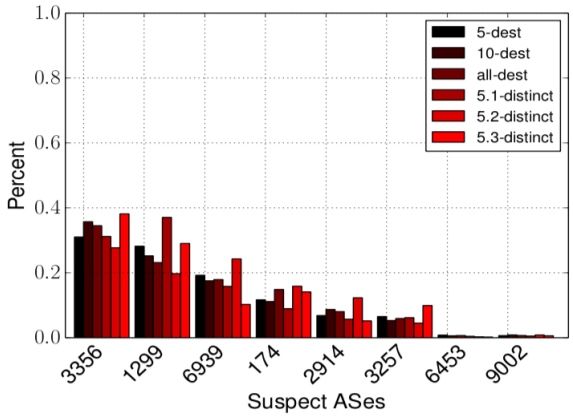

Figure 3: Probability of Suspect AS being able to observe both sides of a Tor stream, given that the Tor stream was vulnerable [11].

E-select on the other hand inspects the AS path between exits and destinations. We need to compute first T_{sus}, a table with all exits and their respective probabilities to encounter a suspected AS in the path towards their destination. The suspect ASes list is the same as we described above with g-select, the difference is that we take the eight ASes into account when we apply e-select. We also define τ, a tunable parameter set between 0 and 1, 0 representing maximum security and 1 representing maximum tolerance. The principle is, after selecting a guard and saved all the suspected ASes we found in the path from client to guard, select an exit only if the probability to retrieve on of these suspected ASes in the path from the exit to the destination is less than τ. See the Algorithm 1 for the pseudo-code corresponding to the e-select procedure.

It has been showed that the best configuration according to DeNASA is *g&e-select:0.1*, that is to say use g-select (with the set of two ASes defined above) and e-select with $\tau = 0.1$ from a security and performance point of view.

3 Diversity in Tor: a state of the art

We now explore previous work attempting to analyze and improve diversity in Tor network, according to different adversaries and using different metrics.

3.1 Existing measurements and statistics

First, the Tor metrics portal [6] does not provide metrics to measure diversity of the network. However, there are several graphs in which we can get a rough idea of the diversity: we can visualize for example the whole set of nodes represented proportionally to their allocated bandwidth, with a color indicating if they have the exit flag or not (see Figure 4). We can regroup the nodes by the sets of ASes or countries, to see which are the most represented by

Algorithm 1 e-select, AS-aware path selection.

Data : $P_{c \leftrightarrow g}$
Data : *Suspected AS list*
Data : T_{sus}
Data : τ
$flag \leftarrow 1$
select guard from Guard List
while $flag$ **do**
  select exit via Tor algorithm
  for all AS^{sus} *in* $P_{c \leftrightarrow g}$ **do**
    if $PR_{AS^{sus},AS^{ex}} < \tau$ **then**
      $flag \leftarrow 0$
    else
      $flag \leftarrow 1$
      break
    end if
  end for
end while

the nodes deployed in Tor in real time. To see a list of details about all the nodes, ranked from the highest bandwidth allocated to the lowest, we can use the relay search aggregation. Examples are `https://metrics.torproject.org/rs.html#toprelays` (all relays), `https://metrics.torproject.org/rs.html#aggregate/as` (relays regrouped by ASes) and `https://metrics.torproject.org/rs.html#aggregate/cc` (relays regrouped by countries).

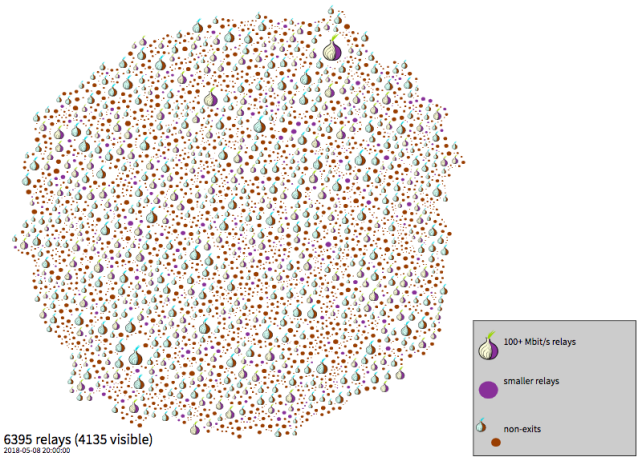

Figure 4: Relay diversity

OrNetStats [33] is a website that gives other useful statistics about the Tor network. We can see for example the top families in the Tor network (a family encompasses all relays belonging or administrated by a same or similar organization), or the Tor version distribution, along with all relays running an unsupported version nowadays, or other useful information such as the probability that an exit node support IPv6.

3.2 Holistic diversity metrics

What we see with existing statistics about Tor, is that we do not have any metrics able to give a measure of the diversity of the nodes, whether for geographic position, Autonomous System or

flags (*guard* and/or *exit*). In the literature, several attempts have been made to measure the uniformity of the network in a holistic approach.

The *degree of uniformity* based on Shannon entropy [15] measures the anonymity level of a system independently of its users. A degree maximum means that all associations between guard and exit nodes have the same probability. A degree to the minimum gives the opposite situation (a system fully biased).

The *Gini coefficient* is a similar measure, also commonly used as a security metric for Tor [41] that is a number defined between 0 and 1. 0 means perfect uniformity: selection probability is the same for all nodes, while 1 stand for perfect inequality between these probabilities.

The advantages of these two metrics is that they are easily computed. Nevertheless, the value we compute with an entropy measure has to be analyzed carefully. A lot of details are omitted, like the strategy performed to allocate the node selection probabilities, the adversary considered or other details handled in the Tor path selection (exit policies, subnet constraints,...) [44]. Thus, a perfect degree of uniformity or a perfect Gini coefficient could "hide" a Tor network with poor anonymity according to other criteria than the probabilities distribution among the nodes selection.

Another recent entropy measure proposed is the *guessing entropy* [37]: a measure of the number of nodes an adversary has to corrupt in order to de-anonymize a specific circuit. The adversary is considered strategic, meaning that he will take the control of the most powerful nodes first, i.e. the ones that maximize the chance to perform an end-to-end correlation attack, and will continue thereby, in function of the nodes he already controls. As a result, the *guessing entropy* is easier to interpret an can thus be more useful than other entropy measures.

To compute the guessing entropy, we need the set of probabilities $p_{i,j}$ describing the probability to select the guard node i along with the exit node j when building a circuit. We then define q_i by the marginal probability of a correlation attack that succeeded. The guessing entropy g is computed as follows (K being the guard nodes size, and N being the exit nodes size):

$$g = \sum_{i=1}^{N+K} i.q_i$$

To compute q_i, we proceed as follows: $q_1 = 0$ because we cannot perform a correlation attack with only one node. We then select the next node i with the objective to maximize the compromise, regarding the $i-1$ nodes we already control (the set denoted q):

$$\begin{cases} P_{G=x} = max(\sum_{y \in q} P(G = x, E = y) \forall x \notin q) \\ P_{E=y} = max(\sum_{x \in q} P(G = x, E = y) \forall y \notin q) \\ q_i = max(P_{G=x}, P_{E=y}) \end{cases}$$

Last metrics concern cumulative distributions of probability in a certain amount of time. It is about measuring the probability that a specific adversary will de-anonymize circuits over time. It can be the time until the first path compromise, or all the paths this adversary succeeded to de-anonymize over a certain period of time. Metrics can be derived from these computations.

Unlike the entropy metrics that perform measurements at a static point in time, this category describes an adversary's attack through a defined amount of time.

4 Methodology

Now that we defined the background about Tor and its diversity aspect, we present our methodology to give a measure of the network diversity under the following assumptions:

- We do not aim to modify the Tor protocol to change bandwidths amounts. We will be working with the nodes advertised bandwidth (though it does not correspond to real amount of bandwidths).
- We do not take /16 subnets diversity into account. Our methodology can however be extended to take these criteria into account.
- We do not take families diversity into account. This is because a node family is setup voluntary, so anyone can disturb this criteria, whether it is voluntary or not. However, like the /16 subnets constraint, our methodology can be extended to take families into account.
- We play our experiments on the entire month of **January 2018**, which will be considered to be representative of the Tor network state.
- The only attack considered is the end-to-end correlation attack.
- All the adversaries will be consider as equal threats, i.e. we make the hypothesis that among all adversaries attempting to attack Tor we consider all adversary models described in the next section in equal proportions.
- We do not take performance of Tor network into account
- The numbers and the locations of the Tor users are not taken into account.

The purpose of the following experiments is to add relay or groups of relays to the Tor network over a period of time, and simulate different models of adversaries, under different path selection algorithms, and compute a series of defined metrics in order to compare them between the original network and all the added relay configurations.

To realize this experimentation, we do not perform real simulations that aim to accurately mimic the Tor network, namely a client model that connects to specific destinations. This is because this way of doing has several drawbacks.

First, we must define clients and destinations models, that can be incoherent with the reality. The Tor client today will not be the same tomorrow. On the other side, estimate a model of a client that is representative of all the potentials clients across the world is difficult. The same goes for the destinations. We could use the *Alexa* websites ranking, but we still lack the clients habits, is the Tor network usage the same than the regular web browsing?

Second, these simulations are long to perform in the experiment context we just described. We will need to simulate a whole network over a long enough period of time for each relay(s) with specific configuration we would like to test. Furthermore, inferring the AS path for each of these connection is also too costly to perform. This is because the number of simulation samples need to be high enough in order to obtain a good estimation.

Instead, the choice was made to work directly with the probabilities to build a specific circuit. The probabilities are computed according to the path selection algorithm used. Another simplification was made to estimate the AS path as well. All the details of the process are explained in the following sections.

4.1 Adversaries models

We consider two categories of adversaries: *relay adversaries* and *network adversaries*.

The **relay adversary**: he is able to either add or corrupt nodes in the Tor network. He is considered strategic, i.e. he knows the path selection algorithm and chose the nodes to control carefully, to optimize his chances to perform end-to-end correlation attacks. Furthermore, he does not deploy nodes within the same /16 subnet nor in the same family to be sure that all its nodes can be combined to build a circuit.

The network adversary: An adversary that control a network and thus can observe all its traffic. A network can be either an **Autonomous System adversary** or a **country adversary**. As our experiments take place on January 2018, we will realistically consider the top AS and the top country in Tor at this moment.

The third network adversary we consider is the **tier-1 AS adversary**. It is different from the AS and country adversaries, because a tier-1 AS adversary does not possess directly a Tor node, but is yet able to observe Tor traffic. More particularly, we are interested in looking for ASes with the *customer cone* that has the most influence on the Tor network at the moment we study it. The reason we do not use the AS path inference method [30] is because it is too costly to compute in our experiment setup and remains subject to estimation errors. Instead we use the notion of customer cone, a worst-case AS threat measure.

A customer cone is a measure of an AS influence in terms of direct and indirect clients it possesses. To compute this cone, we look at the clients of an AS, and then we look recursively at the clients of the clients, until we cannot find clients anymore. This means we reach the bottom of the "cone" we drew (see Figure 5). In this philosophy, we consider a worst-case situation. Indeed, it is sufficient for a relay to be in the cone of a certain tier-1 AS to be threatened by the latter. In the worst case, the traffic will need to flow through the top tier-1 AS, but not always. The traffic could stay in the same AS, or take a path through a peer-to-peer link. On the other hand, an AS that does not stand in the cone of a certain tier-1 will never send its traffic to the latter through peer-to-peer link, according to the BGP protocol. Two assumptions are made here: we use a simplified model of the AS architecture which consists of only provider-to-customer and peer-to-peer links, and we do not take the direction into account, assuming that forward and reverse paths both flow through the tier-1 AS.

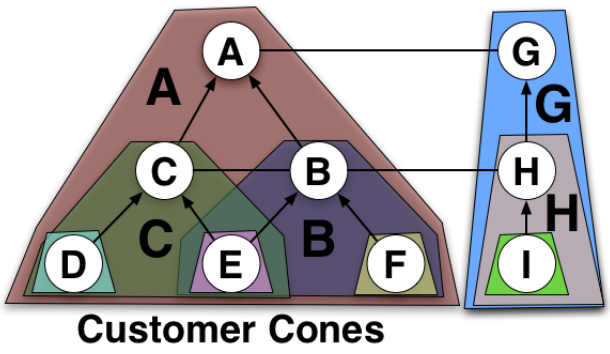

Figure 5: Customer cones of ASes example

Because we replaced path inference with a customer cone, the suspected ASes for the DeNASA method are more difficult to define. By a matter of simplicity, we will consider the two most threatening ASes in *g-select* method, and the next three in the *e-select*. Further research implicating AS path inference would be interesting to see if the most threatening tier-1 ASes distribution has the same trend than the one investigated back in 2015.

4.2 Proposed diversity metrics

4.2.1 Relay diversity

We chose to use the **guessing entropy** to measure the relay adversary threat on the Tor network. So far, this is the most accurate measure, as it pictures an intelligent adversary that knows the path selection process and that will try to control one relay after the other in an optimal way. The greater the value is, the more secure the network is against this model of adversary.

4.2.2 AS and Geographic diversity

We will be looking at two metrics concerning network adversaries (AS, country and tier-1 ASes).

First, the **number of paths compromised at the end of our period of time**. We consider one client connecting to the Tor network without interruption during one month. This means that this client builds a new circuit every 10 minutes [16]. In our case, a month of 31 days give $31 * 24 * 60/10 = 4464$ circuits in total. To represent a client that can come from anywhere, the connection is established each time on a random guard, according to the probabilities computed with the weights allocated by the circuit selection algorithm. The lesser the paths number is, the better the network security is.

Second, we look at the guessing entropy at an AS or country level. This means that each AS (resp. country) is considered as a node in our guessing entropy computation. The reason we select this metric here is motivated in the same way as for the relay adversary.

4.3 Circuit selection algorithms

We decided to compute our diversity metrics under different path selection algorithms: the **ABWRS** used by Tor nowadays, as well as the **Waterfilling** and the **DeNASA** strategies. Our choice is motivated by the positive results given by the analysis of this two recent propositions in a security point of view as well as a performance point of view. Unlike other proposals presented earlier, anonymity does not come at the price of performance, which is an important point to notice when we work with the Tor project. In addition to that, we have seen that the Waterfilling approach is made to mitigate attacks from *relay adversaries*, while the DeNASA goal is to mitigate attacks from *network adversaries*, in particular tier-1 AS adversaries. In this process, It would be interesting to see which one as the most positive impact on diversity, as much on their targeted adversary than on the other adversaries (that we described above).

4.4 Tools

Among all the tools that exist to simulate or emulate the Tor network, the most relevant one is TorPS [8]. For the other tools, some of them are no longer maintained (like ExperimenTor) or too costly in terms of resources (like SNEAC) [40]. Also, we did not use the Shadow tool [4] because we are not interested in observing networks effects. Furthermore, we consider not desirable to downscale the number of nodes in the context of the study of the network diversity.

We will then use TorPS, an open-source simulator for Tor circuits [8]. It was created to study the threat of AS and relay adversaries with typical users under defined path selection algorithms [27]. This tool works as follows: in the first step we need to collect the Tor networks states (published hourly), describing its nodes, their flags, addresses, assigned bandwidth as well as other many useful details. These state files are accessible from CollecTor [2]. Then, TorPS will process these

files to create its own states, keeping only the details it will use. At this step we can also modify the network by artificially adding adversary nodes with desired flag(s) and consensus bandwidth.

The second step is the simulation: we provide a client model, that states when and where it build a connection. This model usually describes a typical user with certain habits. We provide also the states we created in the previous steps, the period of time in which the simulation is executed in Tor, and precise the number of samples we want to produce (the number of clients establishing connections over the period of time defined). The output will be the list of all the circuits built in that period (with IP addresses of clients, guards, exits and destinations) if we choose a network adversary, or a list of compromise codes with corresponding time, indicating which and from when circuits are compromised if we choose a relay adversary. Analyzes can be conducted from theses simulation data.

We also use other tools in addition to TorPS: First, we use the IP2ASN website [3] that provide a service to download all the correspondence between all subnets and their respective AS and countries at once (this is downloaded in the form of a CSV file, that is updated hourly). Second, we also need to collect information about the ASes and countries that have the greater influence on the Tor network. To do so, we use the data rendered by the Tor metrics website [6], that indicate the top ASes and countries in Tor, in terms of controlled bandwidth by nodes they deployed on the network. As for the customer cone we already described, we use data provided by the CAIDA API [1].

4.5 TorPS modifications

We brought some modifications to the TorPS tool, so that we can use it to work directly with network states, instead of launching a series of simulations. We already explained the advantages of this method earlier in this section. We must first know that our implementation starts already from a modified version of TorPS that integrates the Waterfilling path selection algorithm as well as the guessing entropy metric computation, a version built by Florentin Rochet to test the Waterfilling algorithm [37].

The base principle is to look at the relay selection probabilities, weighted by their respective bandwidth. Each time a circuit is built, we have a certain probability to take each of the possible circuits. If a guard x has 0.5% of chance to be selected among all the guards each time, and if an exit y has 0.75% of chance to be selected among all the exits each time, the circuit x-y has $0.5 * 0.75 = 0.375\%$ to be selected each time.

In function of the adversary we look at, we compute how much guard bandwidth and exit bandwidth it controls in average in the month we run our experiments, since the states we observe represent snapshots of the Tor network, published hourly. For example, if an adversary controls half of the network, and if we build 4464 circuits, we consider that the adversary controls 2232 over the 4464 circuits (half of them). Also, we compute the exact moment in time when an adversary controls 1 circuit to determine the time to first path compromised. For example if the adversary controls 60% of the network, it will control 0.6 circuits after 10 minutes (1 circuit built), and 1.2 circuits after 20 minutes (2 circuits built). The time to first path compromised will then be $10 + 6.6 = 16.6min = 0.11h$. As for the guessing entropy metric, we simply need to build the probability matrix by ourselves, with the same probabilities we just described to define the chance to select a specific guard with a specific exit (see the computation details in the metrics section above). This is how the metrics are implemented.

The first prerequisite to run our diversity experimentation is to artificially add relays with specific

configuration. We define the relays we add by ourselves **diversity relays**. As for the guards and exits flags, no modification is necessary. In the case of an AS or a country attribute on the opposite, we need to add modifications. The principle is to generate a random IP possessed by a certain AS or country, this is achieved through correspondence between subnets and ASes or countries [3]. It is worth noticing that we differentiate the adversary and the diversity nodes adding in TorPS, for software evolution purpose, in case we would like to run specific users simulations like we described, and be able to differentiate adversary and diversity relays.

Another prerequisite to do so is to "spot" nodes that belong to the adversary. In the case of a relay adversary, no modification is required, since we can already observe bandwidth as well as the flags that define if they are guards or exits. In the case of AS or country adversary, we need to establish the link between the IP and their AS or country for each of them, to know if the network adversary controls the node or not. For tier-1 ASes adversaries, the process is similar (we look at subnets correspondence), but we need to compute the customer cones of the AS in question upstream. This is another (independent) adding to the program, that computes the customer cone of specific AS. List of subnets are then generated for each tier-1 AS adversaries to perform our matching.

Next, a feature that does not exist in TorPS is the DeNASA path selection algorithm (Tor and Waterfilling are already implemented). Basically, we add the g-select and the e-select methods to our relay probabilities observation. This filters first a part of the guards nodes, and then a part of the possible paths constructed. We use specifically e-select:0.0, because of the customer cone that represent a worst-case analysis. Instead of inferring AS paths like it is done in DeNASA, we simplify this aspect by considering that if a relay is in the customer cone of an AS, then it is compromised in 100% of paths it is implicated in, giving a worst-case e-select method with the parameter configured to 0.0.

The guessing entropy computation has also been extended to take into account other entities than IP addresses, such as set of ASes or countries, considered as (big) nodes in the computation. An optimization has also been added: a probability reduction. Indeed, computation time increases exponentially with the number of nodes in Tor. So, we use a reduction of the nodes in the following experiments when we compute the guessing entropy for all nodes (for ASes and countries set it not necessary since the sets are not too big). We work with groups of two nodes, considered as the nodes in the computation.

The last functionality added to TorPS is the automatic adversary determination. The principle is that our program computes the adversaries with the most influence on the network to apply correlation with guard and exit nodes they possess, i.e. $guard_probability * exit_probability$ for the network states we are working with. It derives thus the AS adversary, the county adversary and the DeNASA tier-1 AS set which we apply g-select and e-select on. To determine the set size of DeNASA tier-1 ASes, we take the five most influent tier-1 ASes from the customer cone point of view, given the probabilities we observed (see Test details below), but this number or selection criteria can be changed in function of the probabilities observed for a given network state. Of course, we can also define an adversary by ourselves. If we define several adversaries, the metrics are computed as an average for all the adversaries.

The diversity TorPS source code is available here:

`https://github.com/rodescamps/torps-diversity`.

4.6 Diversity score computation

The diversity score brought by a relay configuration is computed following the metrics and the adversaries we defined, under a selected path algorithm. Here are the adversaries and the metrics

we will use. The adversaries depend on the path selection method, the purpose is to take the most threatening one in each case. These adversaries are automated determined by our modifier TorPS tool.

- The relay adversary threat is measured by **the guessing entropy**
- The AS adversary is either *OVH* (AS16276) for Tor and DeNASA or *Online S.a.s.* (AS12876) for Waterfilling, and its threat is measured by **the number of paths compromised**. The **the guessing entropy** measures the global AS threat.
- The country adversary is *France* in all path selection methods and its threat is measured by **the number of paths compromised**. The **the guessing entropy** measures the global country threat.
- The tier-1 AS adversary is either *TELIANET* (AS1299) for Tor and Waterfilling, or [*TELIANET* (AS1299), *BTN-ASN* (AS3491), *Opentransit.* (AS5511), *UUNET* (AS701) and *XO-AS15* (AS2828)] for DeNASA, and its (their) threat is (are) measured by **the number of paths compromised**. The **the guessing entropy** measures the global tier-1 AS threat.

First, we need to compute all these metrics for the **vanilla** Tor network (network without modification). Then we compute the same metrics with added nodes following well-defined configurations.

The different configurations we are looking to observe for each adversary (relay, AS, country and tier-1 AS) are the initial network augmented in four different ways:

- 1 guard node with 100 MiB/s
- 5 guard nodes with 20 MiB/s each
- 1 exit node with 100 MiB/s
- 5 exit nodes with 20 MiB/s each

This gives four network configuration in addition to the vanilla network with which we observe the metrics changes. These four configurations are set up twice for each **network adversary**: first with nodes possessed by a threatening adversary we define. For example, for the country adversary, we add nodes that belongs to Germany or France. Second we add nodes possessed by any entity but the adversaries we observe. Again, for the country example, we could take Belgium. This gives **8 possible ways to add relays by network adversary** with which we observe metrics evolution. For the relay adversary, since the model is general and cannot change (there is no point in changing the AS or the country regarding the guessing entropy), we have here only **4 possible ways to add relays for the relay adversary**.

A first remark here concerns the bandwidth we define: it is about the consensus weight which is "a value assigned to a relay that is based on bandwidth observed by the relay and bandwidth measured by the directory authorities, included in the hourly published consensus, and used by clients to select relays for their circuits." [6]. The consequence is that our nodes will not have exactly 100 or 20 MiB/s, but a lesser value due to the measure performed by the Tor directory authorities. This measurement is not completely accurate from the reality point of view, but it prevents adversaries from cheating about any bandwidth they could announce to the network. This precision will affect our observed metrics (the absolute numbers), but will not influence our results comparison, since any node configuration is measured in the same way by the directory authorities.

Secondly, all the nodes we artificially add have the flags *Running*, *Valid*, *Fast* and *Stable*. This means that they respectively are online during our period (1 month) and thus never hibernate,

have a valid configuration (the node can be used by Tor), can handle a high-bandwidth circuit (minimum 10 Mbits/s), and can handle a long-lived circuit (hibernation is rare).

Finally, we repeat all the operations described three times, for each path selection algorithm. This gives in total $(1 + 4 + (8 * 3)) * 3 = 87$ tests to perform: 1 vanilla configuration, 4 relay adversary configurations, 8 network configurations for 3 network adversaries, all times the three selection path algorithms.

Once we get all the metrics from all our network modifications, we can compute increase or decrease in percentage for all the metrics. Since the numbers of paths compromised and the first time to compromise are derived from the same numbers, we keep only the numbers of paths in our computation. We will then determine which configuration is the best regarding a specific metric, under a specific path selection algorithm. We will also determine which adversary has the most impact on the diversity metrics, whether it is positive or negative.

We could go further if we take into account the assumption we made earlier: all the adversaries are considered equal threats. We could compute an average score that gives either an increase or a decrease of the Tor network diversity in terms of percentage, and see which configuration gives the best overall score. The possibilities are:

- ABWRS, WFABWRS or DeNASA?
- Guard(s) or Exit(s) nodes?
- 1 node or 5 equivalent nodes that split our bandwidth?

4.7 Identification of the adversaries

We would like to test in each case (adversary and metrics) the best and the worst case, to see the positive and negative impact by each adversary on each of their metrics. From TorPS we have the following probabilities: it is the influence of the adversaries, the part of the network they control ($P(guards) * P(exits)$) and the Tor network part in which they can perform correlation. We can then identify adversary for each category and for each path selection algorithm here. We also already discuss a little bit the distributions we obtain, nevertheless their analysis will be more detailed in next sections with our metrics applied.

DeNASA ASes and countries adversaries are not presented in this section, since, they all become negligible in the context of our methodology. This means that our metrics variations (number of paths compromised and guessing entropy) are not perceptible, compared to score variations that happen with the two other path selection algorithms. Even though the metrics changes are not significant enough also for the tier-1 AS adversary, we show here the shift of network adversary that happen when DeNASA is applied with our customer cones.

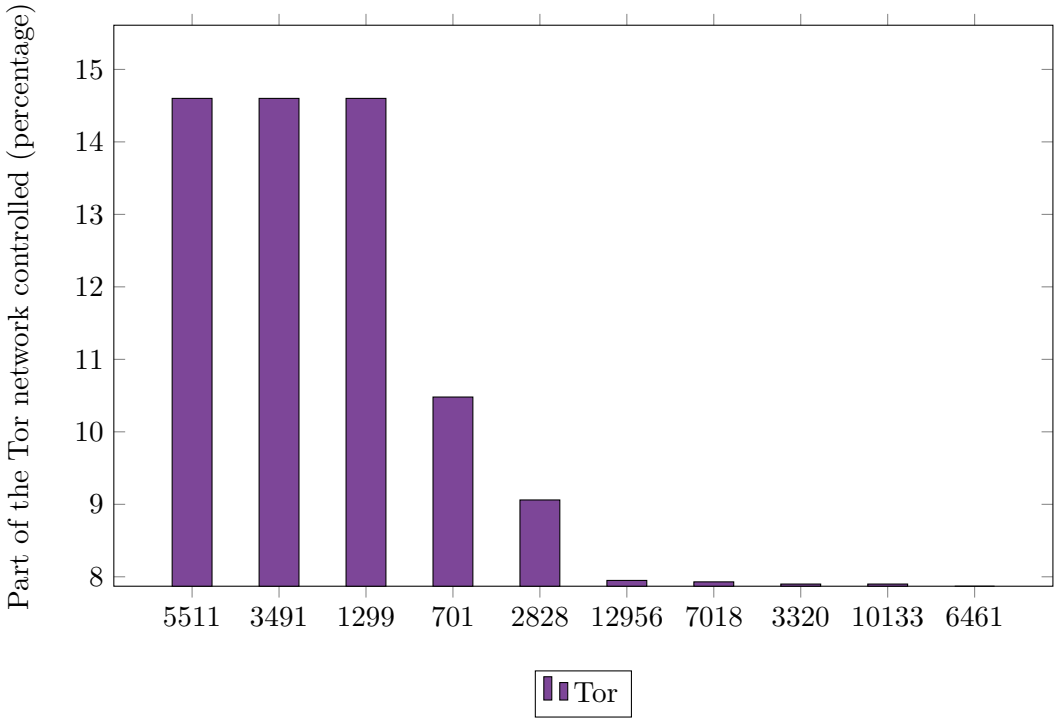

Figure 6: Part of the Tor network controlled **by tier-1 ASes** with Tor algorithm (ABWRS)

From the distribution of the tier-1 ASes customer cones influence with ABWRS (Figure 6), we can derive the adversaries used with DeNASA with *g-select* and *e-select*. We decided to take the two first for *g-select* and the next three for the *e-select*. This choice is not good nor bad in a general point of view; in fact, it set a trade off between (security from network adversary) and (security from relay adversary and Tor network performance). This compromise is discussed further in the analysis.

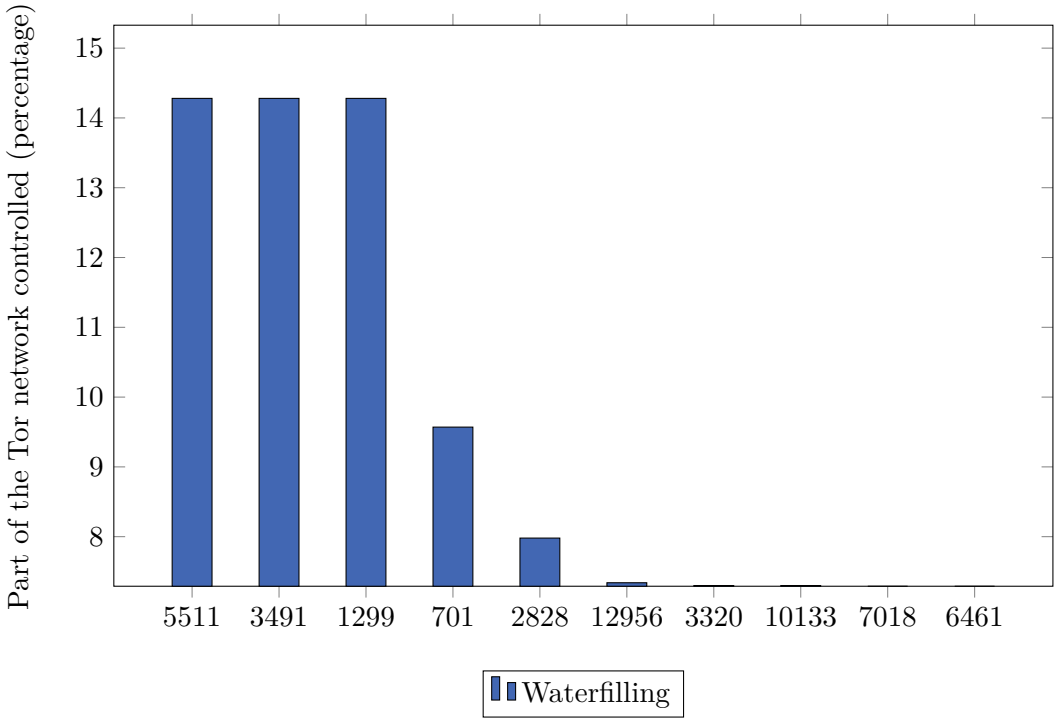

Figure 7: Part of the Tor network controlled **by tier-1 ASes** with Waterfilling (WFABWRS)

The tier-1 ASes customer cones influence with Waterfilling (Figure 7) follows the same distribution for the most threatening ASes as with the Tor algorithm. However, we can already observe that all the numbers (the influence percentage) are lower with Waterfilling, decreasing the tier-1 ASes influence in general.

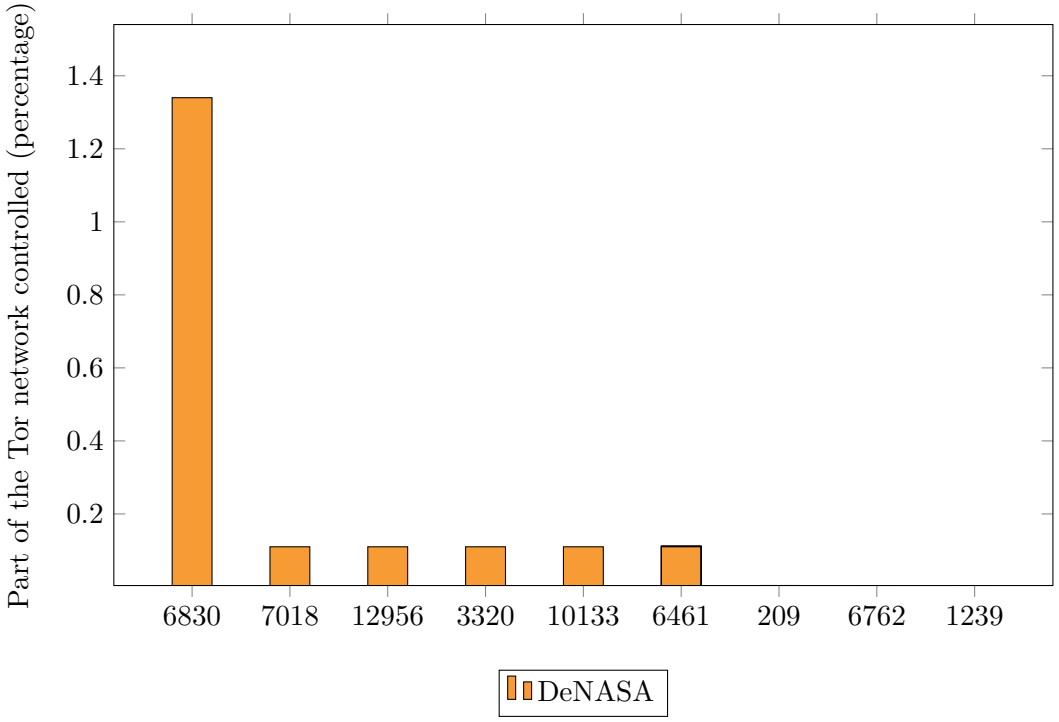

Figure 8: Part of the Tor network controlled **by tier-1 ASes** with DeNASA method

With DeNASA (Figure 8), a new tier-1 AS arises: AS6830, that luckily does not stand in any of the customer cones we wanted to restrict. Only AS6830 remains a serious threat, although percentage are way lower than with the two other path selection methods.

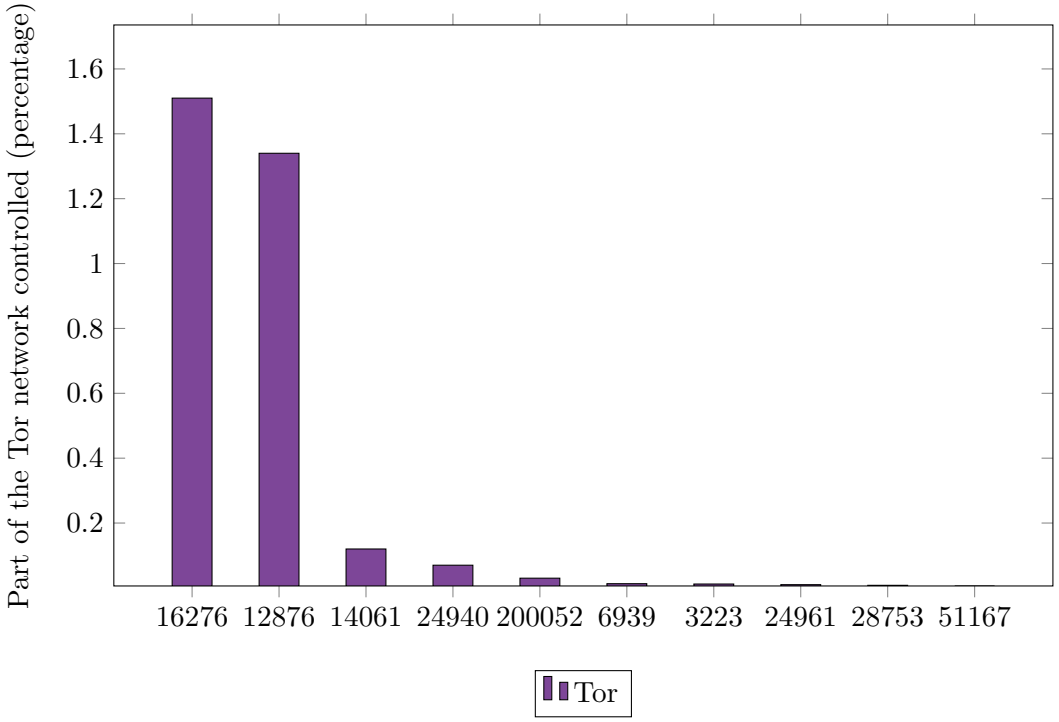

Figure 9: Part of the Tor network controlled **by ASes** with Tor algorithm (ABWRS)

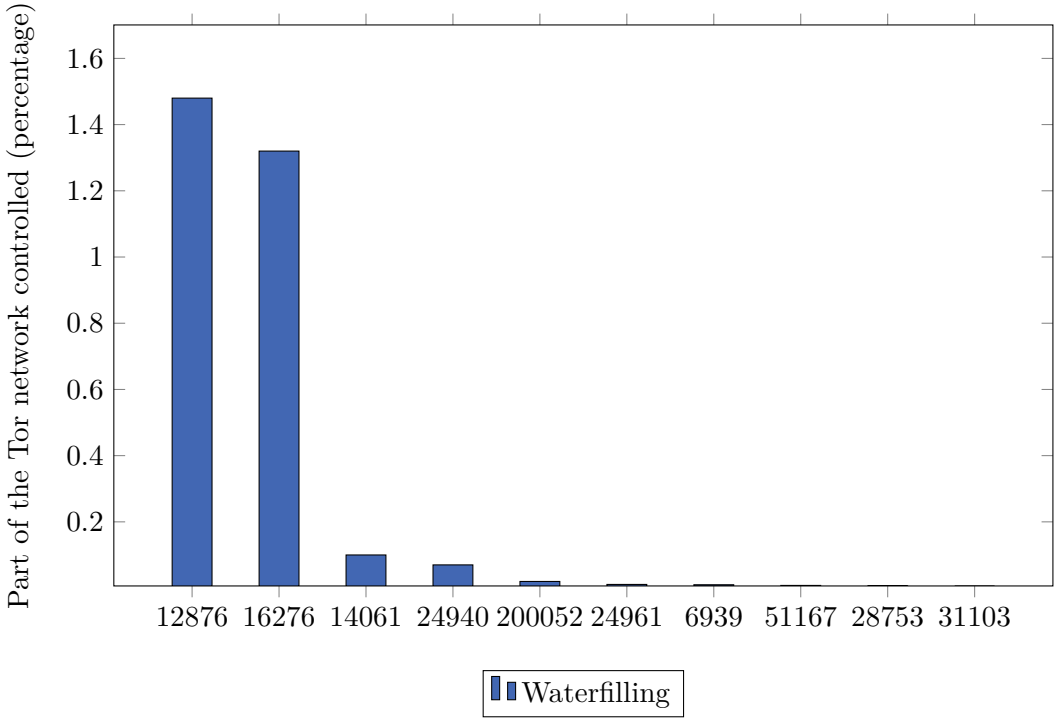

Figure 10: Part of the Tor network controlled **by ASes** with Waterfilling (WFABWRS)

We observe that the distribution follows the same trend for Tor and Waterfilling path selection methods (Figures 9 and 10). Two ASes stand out: AS16276 and AS12876, and they exchange their position in our ranking from Tor to Waterfilling. We can already deduce that AS16276 possesses less guards with more bandwidth than AS12876. Indeed, Waterfilling limits the influence of nodes with its water level, The AS12876 is then more adapted to this algorithm, and on the other hand possesses more guards with a bandwidth more divided.

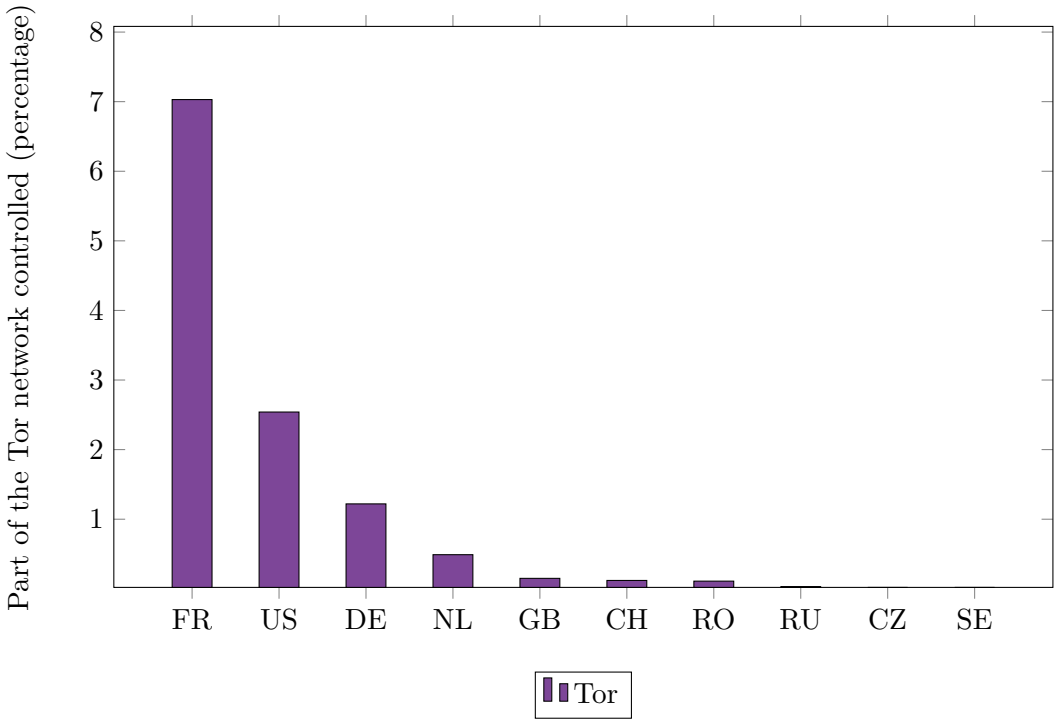

Figure 11: Part of the Tor network controlled **by countries** with Tor algorithm (ABWRS)

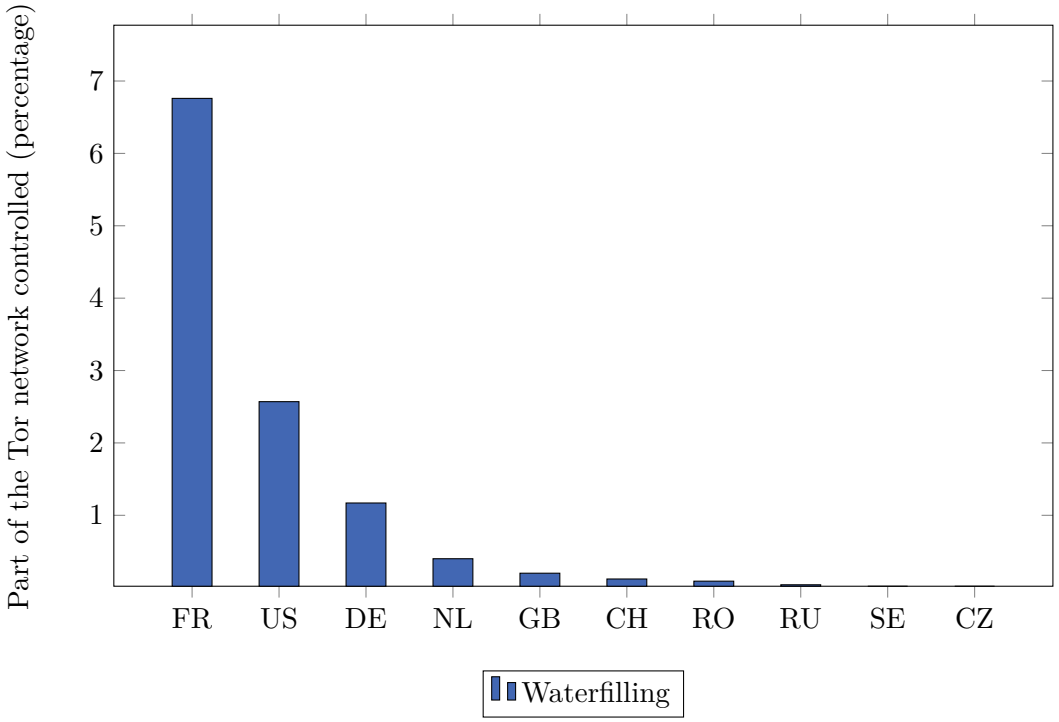

Figure 12: Part of the Tor network controlled **by countries** with Waterfilling (WFABWRS)

Concerning the countries adversaries, distributions are equivalent in Tor and Waterfilling (Figures 11 and 12). As for the DeNASA method (Figure **??**), the distribution is once again completely changed, because of the numerous ASes unable to perform correlation. Since almost all ASes belong to a country, this causes a lot of changes in the country distribution. Also, the influence of the new adversaries with DeNASA is less threatening than with Tor and Waterfilling: this is the same situation we observed with the ASes

4.8 Tests performed

Here are the configurations we use for our experiments, first for the AS adversary:

- Node(s) added in the AS16276 (*OVH*) with Tor
- Node(s) added in the AS12876 (*Online S.a.s*) with Waterfilling
- Node(s) added in the AS29518 (*BREDBAND2*)

For the country adversary:

- Node(s) added in France (FR)
- Node(s) added in Ireland (IE)

For the tier-1 AS adversary:

- Node(s) added in the AS1299 (*TELIANET*)
- Node(s) added in the AS7018 (*ATT-INTERNET4*)

Nodes added for the relay adversary have a random IP address, since their AS or country does not influence the guessing entropy. As for the AS and country adversaries, the objective was to have nodes deployed in AS/country adversary, and nodes deployed outside of the AS/country

adversary in order to compare the impact difference in the results. Finally, for the tier-1 adversary, we take a configuration that put node(s) in the most threatening one (AS1299) and outside of it (AS7018) for the Tor and Waterfilling method.

A last remark concerns the tests coherence: We do not intend to add relay in a certain country if we are observing an AS adversary, or vice versa. This would result in incoherent results since a country can encompass several ASes, and an AS can cover several countries.

4.9 Tests adaptation

What we just stated is the test initial plan, regarding the adversaries influence distribution from previous section. However, some results are not included in our analysis due to their lack of relevance.

For the tier-1 ASes adversaries, the guessing entropy will not be used for any path selection method, since the score difference between the configuration is too little (order of $10^{(} - 5)$) compared to other adversaries metrics variations.

The same phenomena goes for the DeNASA path algorithm with network adversaries. Since our customer cones adversaries remove a lot of nodes, the score differences are too little to be interpreted alongside with metrics score variations with the two other path selection methods.

5 Results and analysis

First, we analyze the metrics results by adversary again under all the circuit selection methods we tested. Then, we will compare metrics under the different path algorithms and summarize their strengths and weaknesses. Finally we attempt to give the best and the worst configuration for each path algorithm, by computing an average diversity score brought to the Tor network under all types of adversaries.

All the results that follow are presented as graphs to be easily interpreted. However, the exact numbers generated by our metrics for all adversaries models are detailed in Appendix A.

5.1 Relay configuration diversity

5.1.1 Relay adversary

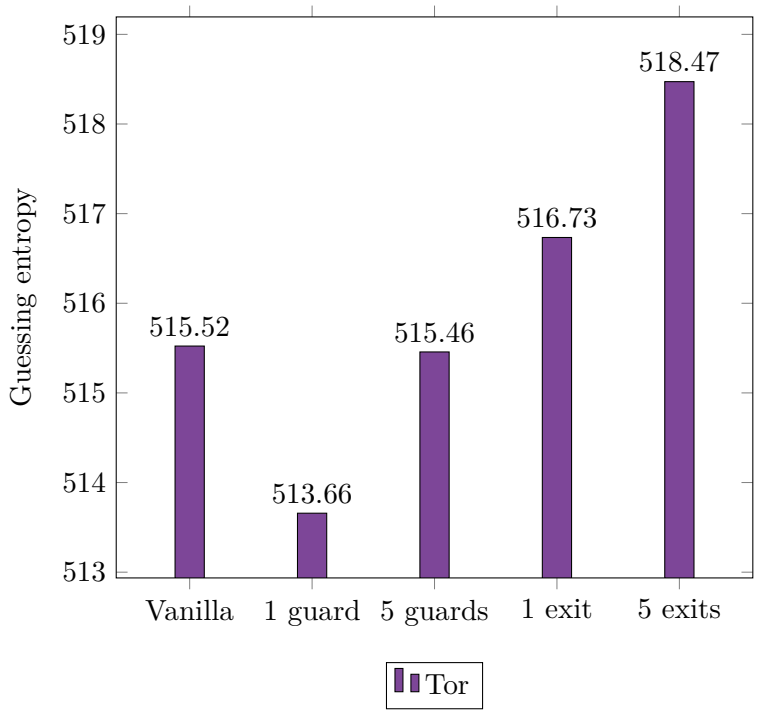

Figure 13: Guessing entropy under different network modifications with Tor algorithm (ABWRS)

The first observation here (Figure 13) is that adding guards node(s) decreases the guessing entropy, while adding exit node(s) increases it. We in fact empower a relay adversary influence by adding guard nodes to the existing network, which is, as we already stated earlier, scarce in exit nodes. Adding guard nodes will result in reinforcing this unbalance between guard and exit positions. More the network is unbalanced, the less nodes a relay adversary needs to control a specific circuit.

Secondly, adding five nodes instead of one node is better, since the guessing entropy only counts a number of nodes, not a bandwidth amount. It is obvious that put all your eggs in the same basket results in less security than splitting the bandwidth in different nodes. The relay adversary will have more difficulties if he must corrupt five nodes instead of one.

By combining these two observations, we see that the best score is obtained by the addition of five exit nodes. The exit position has the most impact on the guessing entropy here. Also the addition of five nodes instead of a single one is better both in guard or exit position.

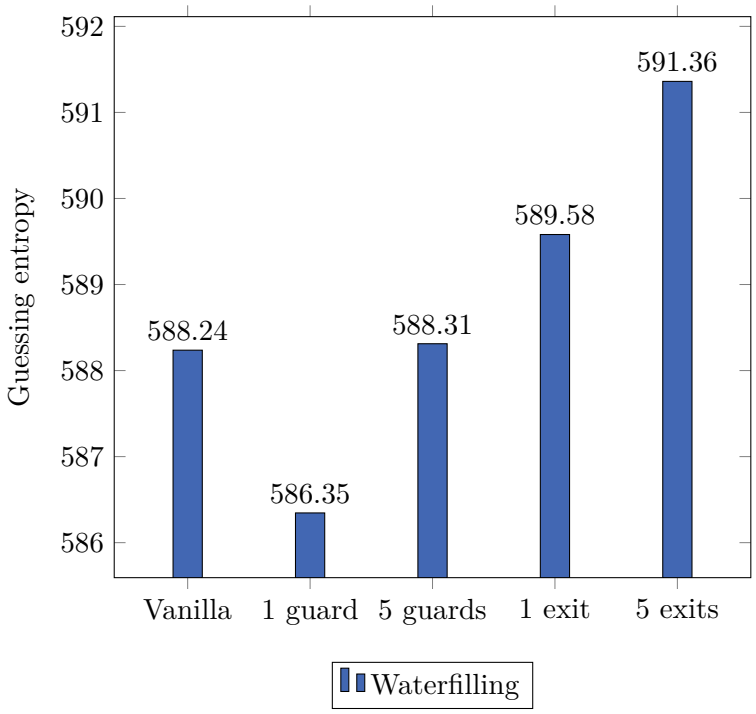

Figure 14: Guessing entropy under different network modifications with Waterfilling (WFABWRS)

With the Waterfilling strategy (Figure 14), we first observe similar guessing entropy scores in the big picture. But by looking further, we see that this time, the 5 guards added configuration gives a better result (even if it is slight), unlike the original Tor strategy. We might wonder why this little difference exits. It is due to the water level in the Waterfilling algorithm. To recall, its role is both to limit high-bandwidth guards and on the other side to increase bandwidth allocated to guard position for low-bandwidth guards. This is why five little guards have more power than a big guard with Waterfilling. As for the other configurations, the guessing entropy improvement (for the exit position) or deterioration (for the one guard added case) are in the same order of magnitude compared to the Tor path selection.

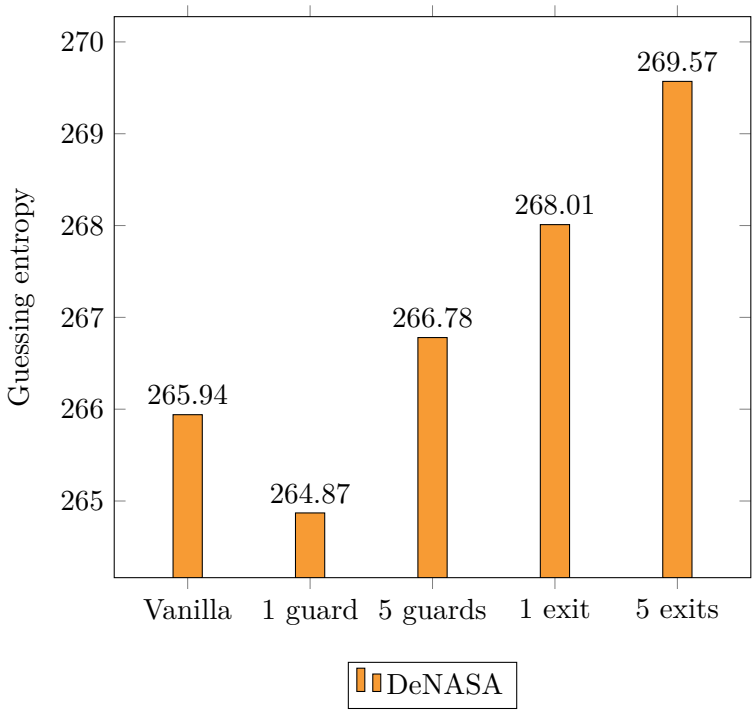

Figure 15: Guessing entropy under different network modifications with DeNASA method

Under the DeNASA method (Figure 15), the guessing entropy changes are greater between configurations. This is because we deleted a certain number of guards nodes, and thus balanced the network. The results are are difficult to interpret though, because the addresses of the nodes added are random. Some of them might be influenced by the *g-select* or the *e-select* method. We could conclude here for example that the one guard added was not excluded by *g-select* but forbids connections to a series of exit nodes, rendering the remaining nodes more powerful, and resulting in a weaker guessing entropy. For DeNASA, the trend remain the same, i.e. the best and worst scores are the same as the ones we observed for Tor and Waterfilling methods.

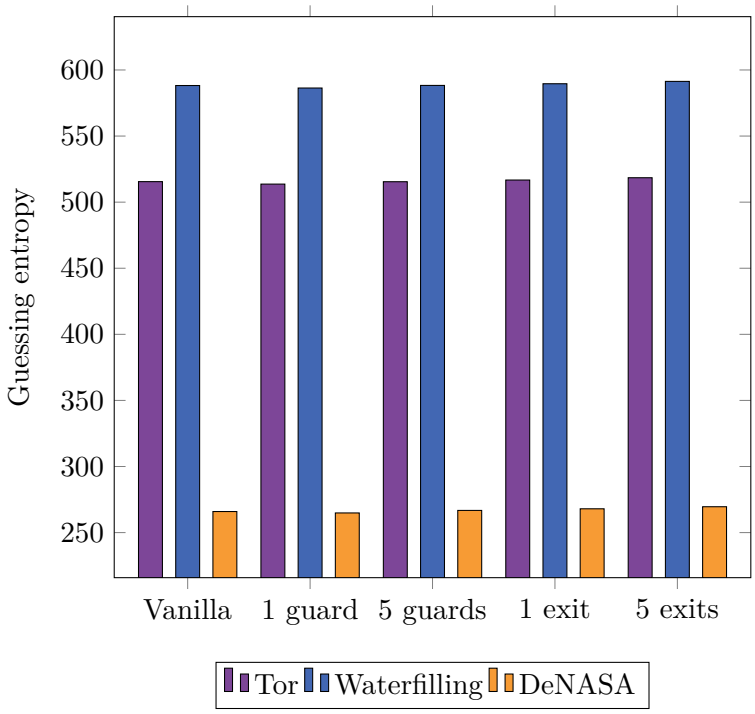

Figure 16: Guessing entropy under different network modifications for the three path algorithms

By comparing the guessing entropy resulting from our three path algorithms (Figure 16), we clearly see that the Waterfilling algorithm gives the best score, while the DeNASA method has the worst resilience to a relay adversary. These results are comforting, since Waterfilling is built to improve the network security against relay adversaries precisely, whereas DeNASA is a method designed to mitigate network adversaries, without any consideration for relay adversaries.

Now, as our purpose is to improve the Tor network against both relay and network adversaries, we may wonder which strategy is the best with our purpose in mind. We must analyze the score variation in each case to see which has the most impact. We observe here that DeNASA decreases the guessing entropy significantly more than Waterfilling is improving it. It remains to know the impact of these two strategies on the other metrics, to see if their impact on network adversaries compensate their influence against relay adversaries.

5.1.2 AS adversary

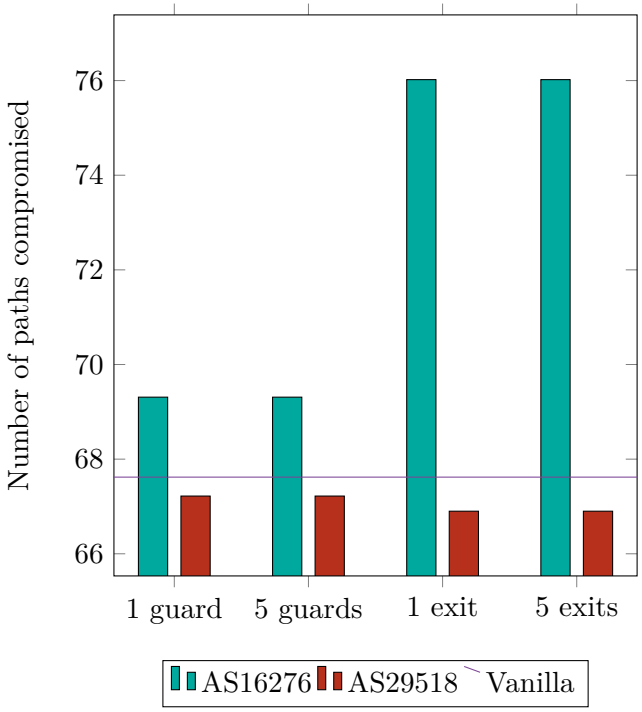

Figure 17: Number of path compromised by **AS16276** under different network modifications with Tor (ABWRS)

When we analyze the AS compromised by the most threatening adversary (AS16276) with the Tor algorithm (Figure 17), we first observe that adding 1 or 5 nodes do not change anything. It is normal, with ABWRS, either we divide our bandwidth or we concentrate all of it in one node, the probability we have to be selected in Tor circuits remains the same. Then, we see that when we strengthen the adversary, the exit position is the most dangerous, it is again what we expected, since there is a lack of exit nodes in general, and OVH (AS16276) is no exception, it has more guards than exits.

What is interesting to notice here, is that for the inverted case, when we try to minimize as much as possible the ASes threat, the impact in its absolute value is lower than the absolute impact done by the nodes added in the AS adversary. There is two reasons. First, as we have seen in the AS global probability distribution earlier, only a few ASes have a non negligible "grasp" on the Tor network. Therefore we need way more bandwidth to add in little ASes to hope balance the AS distribution. Of course, we also need to balance the guards and the exits bandwidth within an AS we wish to reinforce, and this the second reason the AS29518 has not a great impact. In our tests, we add nodes either in guard or exit position, but never in both.

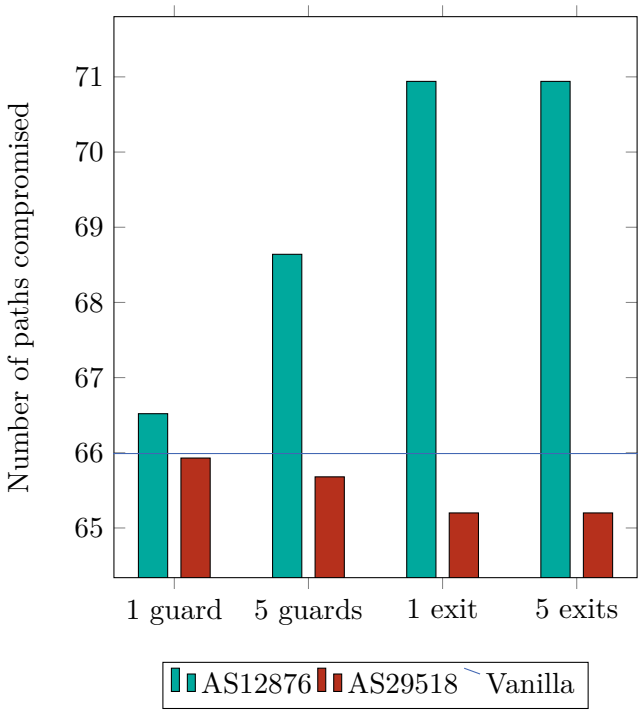

Figure 18: Number of path compromised by **AS12876** under different network modifications with Waterfilling (WFABWRS)

In Figure 18 we first we see that the negative impact (when we add nodes in the AS adversary) is greater than the positive impact when we try to minimize ASes influences, this is coherent with the Tor path selection. The difference here states in the guards nodes configuration. We now observe a difference between 1 and 5 guards, because Waterfilling set a limit to all guards nodes (the water level), that is below 100 MiB/s. So dividing our "power" such that the bandwidth we allocate to our guards never exceed the water level of the Waterfilling method maximizes our influence. Except that, all the other numbers are of the same order of magnitude. They are not equal because the adversary is not the same as when we use the original Tor algorithm, the AS12876 becoming the most threatening AS under the Waterfilling path selection.

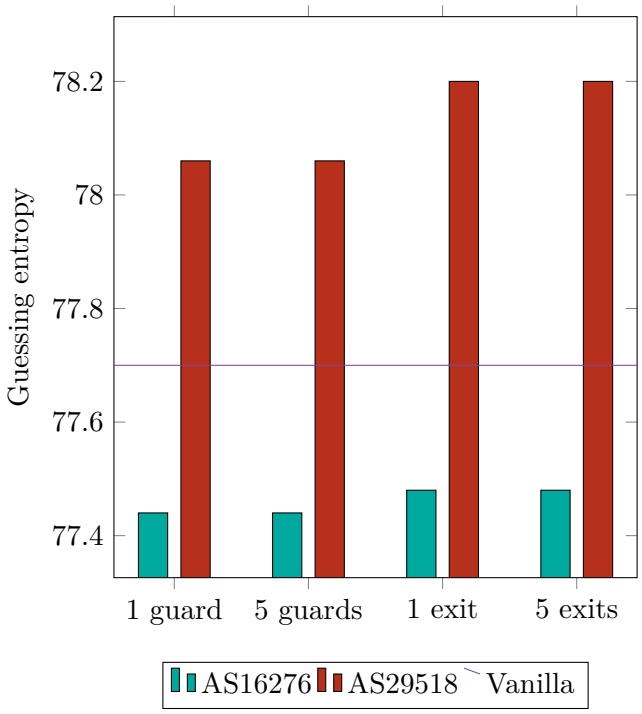

Figure 19: AS-level guessing entropy under different network modifications with Tor (ABWRS)

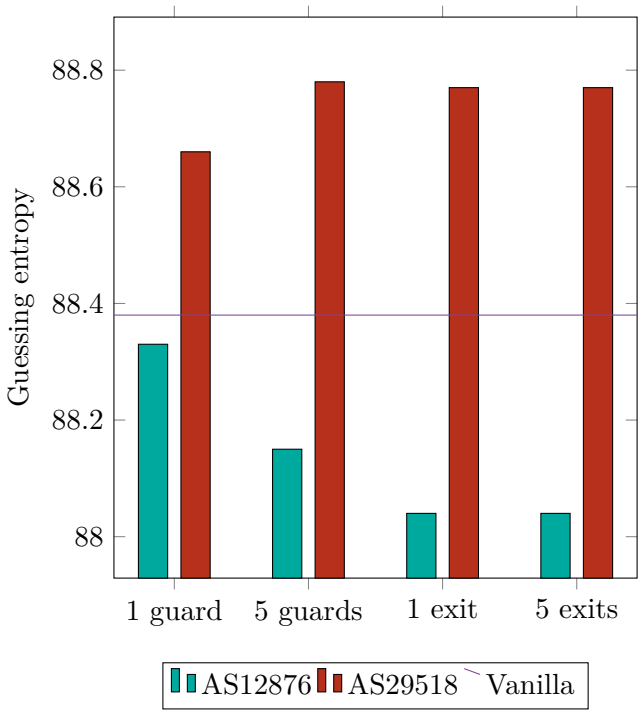

Figure 20: AS-level guessing entropy under different network modifications with Waterfilling (WFABWRS)

The results for the AS-level guessing entropy for Tor and Waterfilling (Figures 19 and 20) are coherent with the path compromised metrics. Nevertheless, the increasing of the metric when we add a good configuration (AS29518) almost equals the decreasing of the metric when we add a bad configuration (AS16276/AS12876). This is because of the metric nature: is is more close to

a variance than the numbers of paths compromised metric is, on the other hand, viewed with the eyes of high-threat adversary. What it means is that increasing the diversity like we do will have more impact with all nodes considered (guessing entropy) than it will have on a top adversary: more its threat is high, more it is difficult to decrease it.

5.1.3 Country adversary

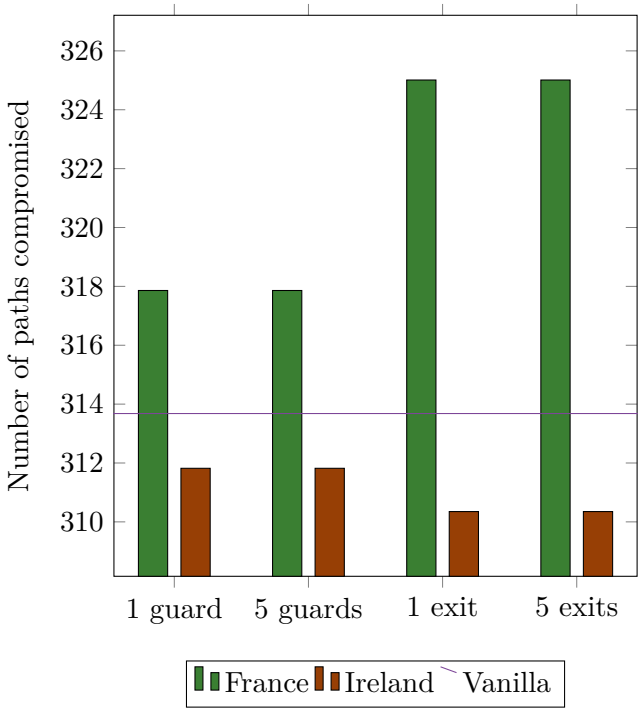

Figure 21: Number of path compromised by **France** under different network modifications with Tor (ABWRS)

The analysis we can give about the number of paths compromised by a country adversary (Figure 21) is similar to the analysis we did with an AS adversary. Again, since we use the Tor algorithm, there is no difference between adding one or five nodes with the same bandwidth in our possession. The positive impact induced by the nodes deployed in Ireland is certainly less in terms of absolute numbers than the negative impact caused by the nodes we added in the adversary, however, this difference is smaller here than when we try to increase diversity at the AS level. This is good news for country diversity, it means that the diversity here is easier to obtain than for ASes. This is caused by the entropy in the countries influences, which is greater (though it remains little) than the entropy we observed in the ASes distribution.

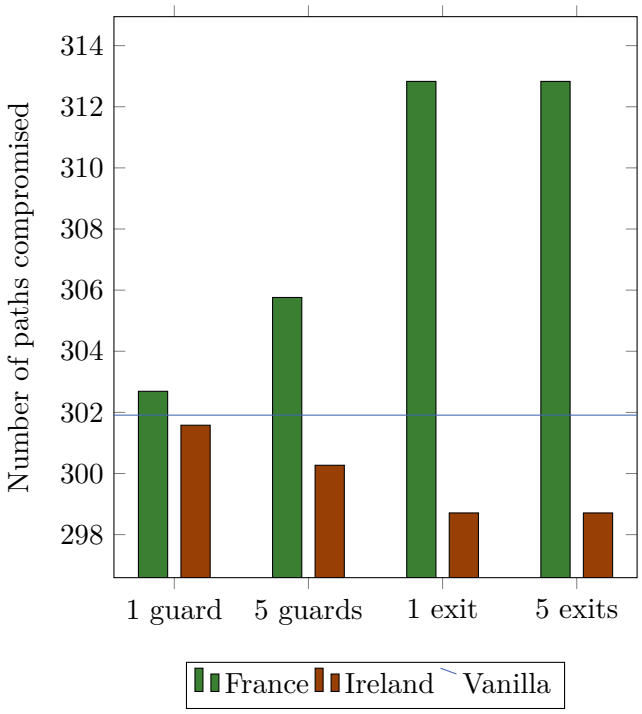

Figure 22: Number of path compromised by **France** under different network modifications with Waterfilling (WFABWRS)

Again, the analysis is here (Figure 18) similar to the AS diversity for the Waterfilling method. The only exception, that we already explained with the Tor algorithm, is that the negative impact inducted by the adding of nodes in the country adversary is not as great as the diversity decreasing in the AS level.

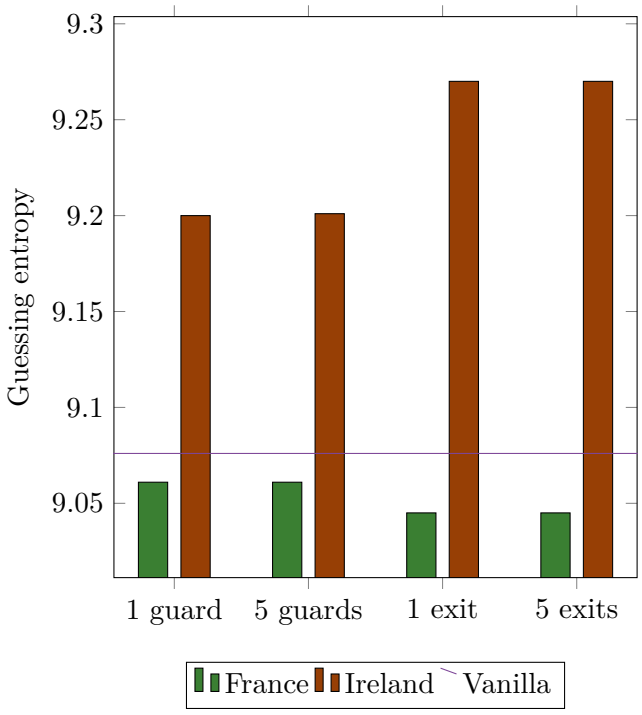

Figure 23: Country-level guessing entropy under different network modifications with Tor (ABWRS)

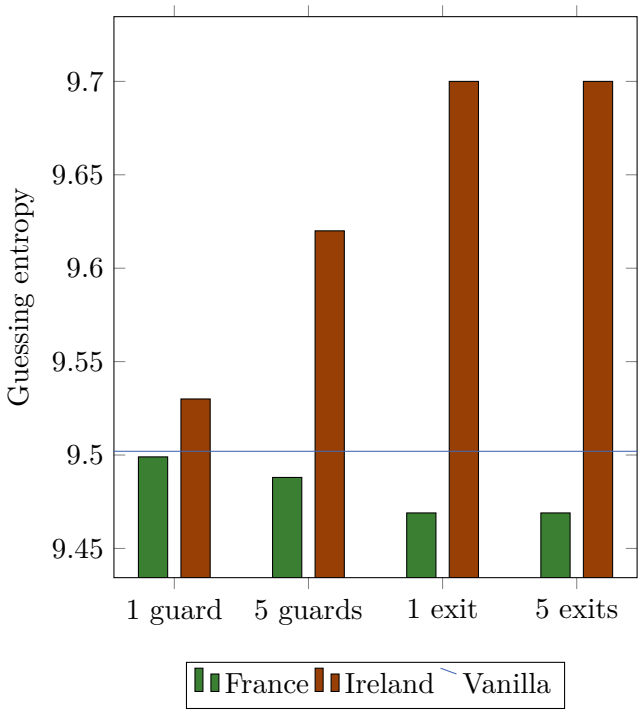

Figure 24: Country-level guessing entropy under different network modifications with Waterfilling (WFABWRS)

Results for guessing entropy at a AS-level (Figures 23 and 24) are coherent with the path compromised metric. Again, like the ASes measurements, the positive impact of a good configuration from a diversity point of view (here, node(s) deployed in Ireland) is greater when we use the guessing entropy metric instead of the number of path compromised by France. The explanation here is the same as the one we gave for the ASes guessing entropy.

Furthermore, we see a significant difference between positive and negative impact with the usage of Tor algorithm as well as the Waterfilling algorithm. This is because the distribution for countries influences is more uniform than the distribution we can observe for ASes. In our case, France has less power at the beginning, trying to reinforcing it will have an impact on diversity proportionally to the influence it possesses on the network. We already evoked this reason in the AS-level analysis, making results for countries metrics coherent.

5.1.4 Tier-1 AS adversary

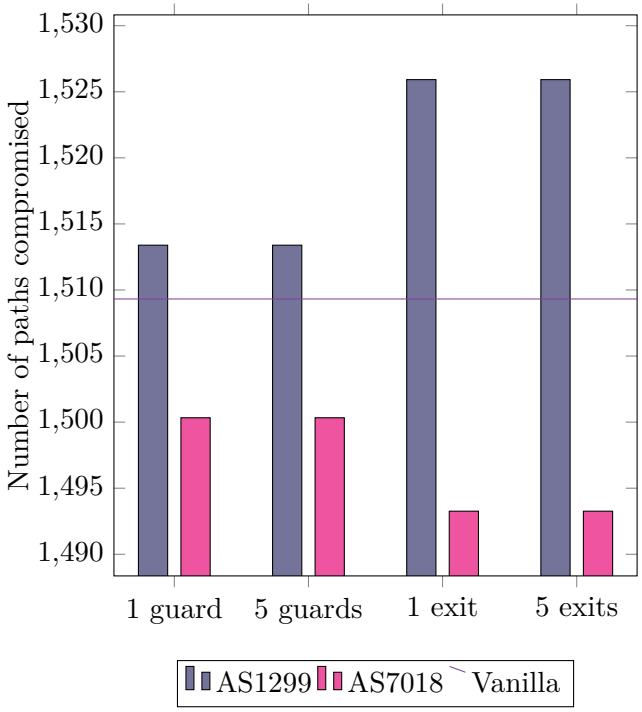

Figure 25: Number of path compromised by **Tier-1 AS1299** under different network modifications with Tor (ABWRS)

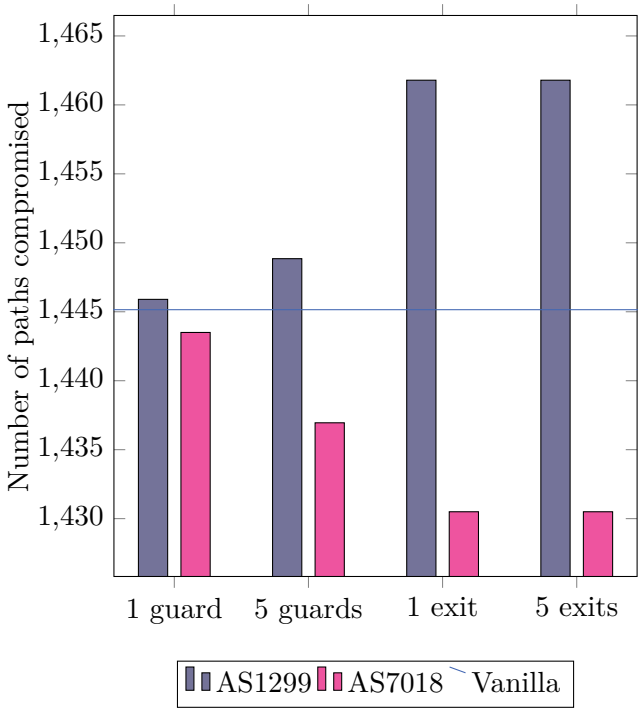

Figure 26: Number of path compromised by **Tier-1 AS1299** under different network modifications with Waterfilling (WFABWRS)

For the tier-1 AS adversaries (Figures 25 and 26), the negative and positive diversity caused by resp. the top adversary and (a) node(s) outside the top adversary are in the order of magnitude for exit nodes. For the guard nodes, we decrease the paths compromised more with the AS7018 than we increase them with the adversary AS1299. Of course, we see that this change is mitigated by the water level with Waterfilling in the case of a unique guard added, it is the behavior expected.

This is the opposite situation of the countries guessing entropy, in which we corrupt more paths with an adversary than we "save" them with an outsider country. Diversity is can thus be achieved easily from the tier-1 AS customer cones point of view. To finish the comparison, the AS-level entropy was more balanced, as it was as easy to add diversity than remove some of it.

5.1.5 Summary

In the the following tables, we gather all the data we collect with our metrics with all our test configuration to have a global overview of the diversity we achieved with the different adversaries:

Path algorithm	1 guard	5 guards	1 exit	5 exits
Tor (ABWRS)	-0.36%	-0.01%	+0.20%	+0.57%
Waterfilling	-0.32%	+0.01%	+0.23%	+0.53%
DeNASA	-0.40%	+0.32%	+0.78%	+1.36%

Table 1: Guessing entropy score variations under Tor, Waterfilling and DeNASA

The guessing entropy (at the node level) results show a clear trend in the Table 1, the worst configuration is in each path selection method the adding of a unique guard, and the best configuration is the adding of five exits. The second best choice is to add one exit, then comes the 5 guards configuration. This trend is simply explained by the Tor network state, which lacks of exit position nodes and thus has too much guard nodes, resulting in a total bandwidth poorly distributed.
The greater scores, both for negative and positive impact are achieved under DeNASA. The reason is that under DeNASA we have a Tor network deprived of a part of its guards nodes and a part of its possible circuits. Variations are thus greater when we have a smaller set of nodes.
The more balanced scores (the ones are closest to 0 for all configurations) are obtained with Waterfilling. It is logical, since Waterfilling improves the network balance the positions distribution by decreasing the guards influence, and thus bring it closer to the exits influence.

Added in	1 guard		5 guards		1 exit		5 exits	
	PC	GE	PC	GE	PC	GE	PC	GE
AS16276	+2.50%	-0.33%	+2.50%	-0.33%	+12.42%	-0.28%	+12.42%	-0.28%
AS29518	-0.59%	+0.46%	-0.59%	+0.46%	-1.06%	+0.64%	-1.06%	+0.64%
France	+1.33%	-0.16%	+1.33%	-0.16%	+3.61%	-0.34%	+3.61%	-0.34%
Ireland	-0.59%	+1.37%	-0.59%	+1.37%	-1.06%	+2.14%	-1.06%	+2.14%
AS1299	+0.27%		+0.27%		+1.10%		+1.10%	
AS7018	-0.60%		-0.60%		-1.06%		-1.06%	

Table 2: Diversity metrics score variations under Tor (ABWRS)

What we can observe in results from Table 2 is that the exit position is to take with caution: an exit added under the influence of an adversary is the worst case for diversity, while exits added

outside of adversaries's grasp form the best cases.
While looking at the numbers, we also observe that except for the tier-1 adversary, all negative variations for the number of paths compromised are greater in terms of absolute numbers than the positive variations. It is exactly the opposite for the AS and country levels guessing entropy: positive variations are greater than negative variations. As we have already explained above, this is due to the metric definition, the paths compromised being seen by a powerful adversary, while the guessing entropy is general.

Added in	1 guard		5 guards		1 exit		5 exits	
	PC	GE	PC	GE	PC	GE	PC	GE
AS12876	+0.80%	-0.06%	+4.02%	-0.26%	+7.50%	-0.38%	+7.50%	-0.38%
AS29518	-0.09%	+0.32%	-0.47%	+0.45%	-1.20%	+0.44%	-1.20%	+0.44%
France	+0.26%	-0.03%	+1.27%	-0.15%	+3.62%	-0.35%	+3.62%	-0.35%
Ireland	-0.11%	+0.29%	-0.54%	+1.24%	-1.06%	+2.08%	-1.06%	+2.08%
AS1299	+0.05%		+0.26%		+1.15%		+1.15%	
AS7018	-0.11%		-0.57%		-1.01%		-1.01%	

Table 3: Diversity metrics score variations under Waterfilling (WFABWRS)

The analysis we conducted on the Tor method Table remain valid for the Table of the Waterfilling score variations 3. The difference we can notice state in the variations for the 1 guard position. The variations are smaller (both positive and negative) because of the water level application. Obviously, the variations are different for the AS adversary, since it is not the same as with the Tor algorithm, while the countries and the tier-1 adversaries scores are almost equal.

5.2 Selection path algorithms diversity

The following array presents the different metric scores computed following the above definitions for the three selection path algorithms we study: **Tor (ABWRS)**, **Waterfilling (WFABWRS)** and **DeNASA**.

Relay adv.	AS adversary		Country adversary		Tier-1 AS adversary	
GE	PC	GE	PC	GE	PC	GE
515.52	67.62	77.70	313.68	9.08	1509.32	8.70
588.24	66.00	**88.38**	301.91	**9.50**	1445.15	9.67
265.94	**59.65**	49.43	**111.28**	7.91	**85.42**	**78.95**

Table 4: Vanilla network metrics under ABWRS, WFABWRS and DeNASA

From the Table 4 we can analyze the strengths and weaknesses of each circuit selection method. First, we see that Tor has no advantage compared to the two others, even though its scores are never far from Waterfilling. The latter is the best method to use against relay adversaries, and to maintain a better guessing entropy at the other levels as well. However, we see that DeNASA is significantly stronger than Waterfilling for all network adversaries. If we consider the four adversaries as equal threats, as we made on of our assumption in this research context, DeNASA is better for three of them, while Warterfilling is only better for relay adversaries. By comparing score numbers, we find that the great score Waterfilling obtain against relay adversary does not suffice to compensate the score increases DeNASA gains for other adversaries.

For all that, do we conclude that DeNASA should be used? Not necessarily, even if we forget the relay adversary. DeNASA has a several drawback. First, it renders powerful nodes useless in

the Tor network. This is nice for security, but a bad news from a performance point of view. DeNASA sacrifices performance in order to obtain security. Of course, an obvious solution is to use another estimation than the AS cutomer cones, that are too strict for performance. This leads to our second drawback: methods to estimate if a certain AS is able to observe traffic of a certain circuit are uncertain and not easy to deploy on the Tor infrastructure. This was the same reason the metrics we defined in this experiment are difficult to apply, AS paths are not easy to determine. Nevertheless, this is a whole subject in itself, trying to improve methods to define the AS paths are out of this research scope.

On the other hand, Waterfilling improves all score metrics computed with the Tor algorithm, and is strong against relay adversary, and is easy to deploy on the Tor network. What we were afraid of when setting up the experiment was that Waterfilling was a compromise that fight better relay adversaries while being weaker against network adversaries, but in all the results we obtained we see that it is not the case.

The ABWRS has also one advantage if we put in perspective all the arguments we just evoked: it is already implemented and used in the Tor network! Changes takes time, and a circuit selection method must be carefully analyzed before being launched at a large scale, our purpose remaining to maximize anonymity for Tor users; it is also about being careful and not to take unconsidered risks.

5.3 Configuration diversity ranking

We begin by stating the formula computing the general diversity brought by a certain node(s) adding configuration, regarding all adversaries presented and all metrics used in this research. Remind that this formula is relevant according to the hypothesis we made: all the adversaries defined are considered as equal threats, i.e. no one is more likely to exist than another. This is a simple attempt to measure diversity in general way in the Tor network, they can help to take a decision for the relay configuration to set. However, if we want to increase diversity regarding a well defined adversary, this formula is irrelevant.

$$Diversity\ Score = GE_r + PC + GE_n$$

Where

- GE_r is the guessing entropy variation for a relay adversary
- PC is the sum of variations of path compromised by the top AS adversary, the top country adversary and the top tier-1 AS adversary
- GE_n is the guessing entropy variation for a network adversary, being the sum of the variations of guessing entropy at the AS level, at the country level and at the tier-1 AS level

The variations are taken positively or negatively according resp. to their positive or negative impact on the diversity. Results presented are taken in their absolute value, with a negative sign if it decreases the diversity, with a positive sign if it increases the diversity following the metrics definitions.
Following our assumptions and our formula, let's compute it for each path selection method with a vanilla network, and with the best and the worst configuration, following the results we observed from our experiments.

5.3.1 ABWRS Diversity score

The worst case is the adding of one exit node to the AS16276 (OVH):

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = 0.2 - 17.13 - 0.62 = -17.55\% \\ GE_r = (0.2) = 0.2 \\ PC = (-12.42) + (-3.61) + (-1.10) = -17.13 \\ GE_n = (-0.28) + (-0.34) + (0) = -0.62 \end{cases}$$

The best case is the adding of five exit nodes in Ireland. We need an additional data, the AS scores: let's take AS15502 (Vodafone-Ireland-ASN). AS15502 is in the customer cone of AS1299.

- $PC_{AS} = -1.06\%$
- $GE_{AS} = +0.64\%\%$

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = 0.57 + 1.02 + 2.78 = +4.37\% \\ GE_r = (0.57) = 0.57 \\ PC = (1.06) + (1.06) + (-1.10) = 1.02 \\ GE_n = (0.64) + (2.14) + (0) = 2.78 \end{cases}$$

5.3.2 WFABWRS Diversity score

The worst case is the adding of one exit node to the AS12876 (Online S.a.s.) (AS12876 is in the customer cone of AS1299):

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = 0.2 - 12.27 - 0.73 = -12.80\% \\ GE_r = (0.2) = 0.2 \\ PC = (-7.5) + (-3.62) + (-1.15) = -12.27 \\ GE_n = (-0.38) + (-0.35) + (0) = -0.73 \end{cases}$$

The best case is again the adding of five exit nodes in Ireland. Let's take again AS15502 (Vodafone-Ireland-ASN). AS15502 is in the customer cone of AS1299.

- $PC_{AS} = -1.06\%$
- $GE_{AS} = +0.64\%\%$

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = 0.53 + 0.97 + 2.72 = +4.22\% \\ GE_r = (0.53) = 0.53 \\ PC = (1.06) + (1.06) + (-1.15) = 0.97 \\ GE_n = (0.64) + (2.08) + (0) = 2.72 \end{cases}$$

5.3.3 DeNASA Diversity score

The worst case is the adding of one guard node to the network that is outside of the customer cones of AS1299, AS3491, AS5511, AS701 and AS2828.

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = -0.40 + 0 + 0 = -0.40\% \\ GE_r = (-0.40) = -0.40 \\ PC = (0) + (0) + (0) = 0 \\ GE_n = (0) + (0) + (0) = 0 \end{cases}$$

The best case is the adding of five exit nodes to the network that are outside of the customer cones of AS1299, AS3491, AS5511, AS701 and AS2828.

$$\begin{cases} Diversity\ Score = GE_r + PC + GE_n = 1.36 + 0 + 0 = +1.36\% \\ GE_r = (1.36) = 1.36 \\ PC = (0) + (0) + (0) = 0 \\ GE_n = (0) + (0) + (0) = 0 \end{cases}$$

5.3.4 Diversity score implementation

From the diversity score we devised, we could easily imagine a service that automatically compute the diversity score in function of the relay(s) characteristics we would like to add to the existing Tor network. We could take also take recent network states to compute best and worst cases, and advice the volunteers for their relay configurations wished by Tor.

In this context, an important aspect to observe is the diversity score handling by the Tor authorities. Let's define a new adversary: the **diversity adversary**, that is supposed to control a part of the relays and/or an AS or some other entity. Its attack consists to run a maximum of node bandwidth in a configuration such that he twists the diversity score to encourage volunteers to put relays that will reinforce the control the adversary has on the network through its nodes and/or his As or other.

If we look for example in the network states from the last 24 hours, this attack will not be costly for the adversary, and the diversity score will quickly change to its advantage, and he just needs to convince as more volunteers as he can during this period to deploy nodes that will increases his influence on the Tor network, while the volunteers believe to increase the diversity! If the period is longer, the attack will be costly (the time to keep the nodes up and running) but will attract more volunteers relays. The question here is, in function of the Tor network state, the server costs, the diversity score formula and its result, is the **diversity adversary** a threat, i.e. is the attack we described can be deployed in a context such that it is worth the cost? We could also imagine, in addition to that, social engineering to trick volunteers, or other attacks combination. From now, it is hard to predict if such an adversary model is plausible.

6 Future work and improvements

Many improvements can be made to the work done in this research. First of all, it would be interesting to compare the results obtained with static observation, that is to say when we work with the probabilities derived from the Tor network states here, with user trace simulations. Of course, the results will be different, depending on the user trace behavior we define, but the difference gap is what we are interested in. Will a French user that uses Tor four times a day to connect to Google services give diversity results drastically different than a Japanese user that uses Tor once a day to connect to its local journal website? Or, are all different profiles we can think of each differ a little from our general diversity results?

A second critical point to investigate is the AS observing traffic. If we could know the exact AS paths between two nodes, we would be able to measure the impact of the Autonomous Systems in an exact way. Unfortunately, it is impossible unless we control all the ASes in the paths we would like to measure. The method that give the best estimations is the AS path inference [30]. We could use it to derive a more precise diversity score in our work, since we used the customer cone in place, to consider a worst-case scenario. The AS path inference takes also more time to compute, and does not always deliver a path estimation, depending on the researched paths.

We could also take the *Internet Exchange Points* into account in our adversary models. They are however as complex as the AS paths to compute. They do not have an impact as big as the ASes [11], but they are more likely to be controlled, since they are centralized unlike the bigger Autonomous Systems [27]. Also, more adversaries could be considered, regarding recent attacks discovered [43] [38], or future attacks that remain to be discovered. In the same principle other existing path algorithms could be used in the experiments we put in place, as well as future path algorithms yet to come. We hope that this research can provide an idea of the diversity a path algorithm can represent, compared to others.

Other evaluation side than diversity could be computed: if we use the Shadow tool, we could take the performance into account when we add a certain configuration of diversity nodes, to observe the trade off resulting from adding more diversity. By knowing the link between diversity and performance, we could derive a better score, adjusted with a balance between performance and diversity (thus security).

To help diversity, we could also encourage more people to deploy relays by making it easier. For example, nusenu developed an Ansible role for Tor relay operators [32]; we could go in that sense, less work to create and maintain a relay would definitely result in more people interested by running one. Support exist already for those who want to get help running Tor relays, with a growing community (see a recent example `https://blog.torproject.org/get-help-running-your-relay-our-new-advocate`). It sounds simple, but to encourage helping other relay operators is an important factor to develop diversity. Also, in a perspective that aims to automate and make a relay deployment easy, we could associate the diversity score in the same process. We could advise the best configurations to the Tor operators given the last 30 days network states, and conversely, computing the diversity score resulting from a proposed relay(s) configuration to add.

One last important point to mention that curbs the diversity development in the Tor network is the economic question. It is an important aspect, that we do not treat in this research, but that is essential if we really would like to get more diverse nodes in Tor. Indeed, the main reason the relays are numerous in certain providers or countries is because they are cheaper to deploy than in other providers. Since deploying a relay is a volunteer act, more it is costly, less the people are encouraged to take part to the Tor project. Furthermore, the Tor network is strongly influenced by countries politics. Some countries forbid the Tor usage in their territories, the relay deployment becoming impossible. It is however always possible to establish a connection to the Tor network, thanks to the *bridge* mechanism: it is an intermediate server that is the "bridge" between the user and a guard node. Promoting diversity is thus, by these aspects, difficult.

A solution lead could be to deploy an economic system in the Tor network. More the relay has a great diversity score, more it generates a certain currency with time. Such a currency could be a cryptocurrency specific to Tor, that could buy performance on the Tor network. Such performance bought should be proportional to the security and/or performance brought by our node(s). Let's suppose that such a cryptocurrency encounters success, because of a growing need to be anonymous on Internet, this will encourage more people to deploy more nodes with the greater diversity score possible, and then improve the network. This will also result in growing the Tor users, meaning inevitably increasing the network security. Research on economic development in Tor could be interesting for the future of the network.

Conclusion

With the help of the TorPS tool, recent research and available information about the Tor network, we were able to successfully attach a score, an utility to a node or a group of nodes addition to the existing Tor network. This utility has several components that analyze different adversaries models with several metrics. We were also able to analyze the score variation under recent path selection methods proposals: Waterfilling and DeNASA. We observed that Waterfilling was efficient against all our defined adversaries in addition to the relay adversary it was originally designed for. We encountered difficulties for the DeNASA method, that is complicated to evaluate for two reasons: (i) its simulation with nodes probabilities is difficult to implement and (ii) it gives results difficult to interpret without AS path inference.

We first established a state of the art of the diversity in Tor, which was brief, incomplete and misleading under some aspects. We adapted the TorPS tool in order to achieve simulations in very general and global way, in order not to have to make reduction or too much assumptions in a simplified Tor network. We perform simulations, and analyzed and highlighted the best and worst configuration in function of the metric, the adversary and the circuit selection algorithm. We also validated the result by comparing the scores variations differences, depending on the context and the deployment of node(s) configuration. We gave a formula starting point to effectively derive a score from a relay configuration, that need to be adapted depending on assumptions made about adversaries.

We have seen that this research could arises more work to be done in order to truly implementing utility scores, and that it can even arises more problems with potential new adversary.
I hope this master thesis will kick-off further research about relays diversity, with the purpose to improve the Tor network and reinforce its security or at least, help relay volunteers to be aware of the relays impact on network diversity and maybe helping them in their decisions.

References

[1] AS Rank: A ranking of the largest Autonomous Systems (AS) in the Internet. `http://as-rank.caida.org/`, 2018.

[2] CollecTor. `https://collector.torproject.org/`, 2018.

[3] Free IP address to ASN database . `https://iptoasn.com/`, 2018.

[4] The Shadow simulator. `https://shadow.github.io/`, 2018.

[5] The Tor Project. `https://torproject.org/`, 2018.

[6] Tor Metrics Portal. `https://metrics.torproject.org/`, 2018.

[7] Tor Relay Guide. `https://trac.torproject.org/projects/tor/wiki/TorRelayGuide/`, 2018.

[8] TorPS: The Tor path simulator. `https://torps.github.io/`, 2018.

[9] Masoud Akhoondi, Curtis Yu, and Harsha V Madhyastha. Lastor: A low-latency as-aware tor client. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 476–490. IEEE, 2012.

[10] Mhd Wesam Al Nabki, Eduardo Fidalgo, Enrique Alegre, and Ivan de Paz. Classifying illegal activities on tor network based on web textual contents. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 35–43, 2017.

[11] Armon Barton and Matthew Wright. Denasa: Destination-naive as-awareness in anonymous communications. *Proceedings on Privacy Enhancing Technologies*, 2016(4):356–372, 2016.

[12] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 92–102. ACM, 2007.

[13] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies*, pages 46–66. Springer, 2001.

[14] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[15] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 54–68. Springer, 2002.

[16] Roger Dingledine and Nick Mathewson. Tor protocol specifications. `https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt`, 2018.

[17] Roger Dingledine, Nick Mathewson, Steven Murdoch, and Paul Syverson. Tor: The second-generation onion router (2014 draft v1), 2014.

[18] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 43–54. ACM, 2012.

[19] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 66–76. ACM, 2004.

[20] Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM, 2002.

[21] Sharad Goel, Mark Robson, Milo Polte, and Emin Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[22] Ceki Gulcu and Gene Tsudik. Mixing e-mail with babel. In *Network and Distributed System Security, 1996., Proceedings of the Symposium on*, pages 2–16. IEEE, 1996.

[23] Angele Hamel, Jean-Charles Grégoire, and Ian Goldberg. The misentropists: New approaches to measures in tor. *Centre for Applied Cryptographic Research (CACR)*, 2011.

[24] Jamie Hayes and George Danezis. Guard sets for onion routing. *Proceedings on Privacy Enhancing Technologies*, 2015(2):65–80, 2015.

[25] Nicholas Hopper, Eugene Y Vasserman, and Eric Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security (TISSEC)*, 13(2):13, 2010.

[26] 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 20th ACM Conference on Computer and Communications Security (CCS 2013)*. ACM, 2013.

[27] Aaron M Johnson, Paul Syverson, Roger Dingledine, and Nick Mathewson. Trust-based anonymous communication: Adversary models and routing algorithms. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 175–186. ACM, 2011.

[28] Joshua Juen, Aaron Johnson, Anupam Das, Nikita Borisov, and Matthew Caesar. Defending tor from network adversaries: A case study of network path prediction. *Proceedings on Privacy Enhancing Technologies*, 2015(2):171–187, 2015.

[29] Brian Neil Levine and Clay Shields. Hordes: a multicast based protocol for anonymity 1. *Journal of Computer Security*, 10(3):213–240, 2002.

[30] Z Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On as-level path inference. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 339–349. ACM, 2005.

[31] Steven J Murdoch and Robert NM Watson. Metrics for security and performance in low-latency anonymity systems. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 115–132. Springer, 2008.

[32] nusenu. An Ansible Role for Tor Relay Operators. `https://github.com/nusenu/ansible-relayor`, 2018.

[33] nusenu. OrNetStats. `https://nusenu.github.io/OrNetStats/`, 2018.

[34] Lasse Overlier and Paul Syverson. Locating hidden servers. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

[35] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM transactions on information and system security (TISSEC)*, 1(1):66–92, 1998.

[36] Marc Rennhard and Bernhard Plattner. Practical anonymity for the masses with morphmix. In *International Conference on Financial Cryptography*, pages 233–250. Springer, 2004.

[37] Florentin Rochet and Olivier Pereira. Waterfilling: Balancing the tor network with maximum diversity. *Proceedings on Privacy Enhancing Technologies*, 2017(2):4–22, 2017.

[38] Florentin Rochet and Olivier Pereira. Dropping on the edge: Flexibility and traffic confirmation in onion routing protocols. *Proceedings on Privacy Enhancing Technologies*, 2018(2):27–46, 2018.

[39] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable link-based relay selection for anonymous routing. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 73–93. Springer, 2009.

[40] Fatemeh Shirazi, Matthias Goehring, and Claudia Diaz. Tor experimentation tools. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 206–213. IEEE, 2015.

[41] Robin Snader and Nikita Borisov. A tune-up for tor: Improving security and performance in the tor network. In *ndss*, volume 8, page 127, 2008.

[42] Robin Snader and Nikita Borisov. Improving security and performance in the tor network through tunable path selection. *IEEE Transactions on Dependable and Secure Computing*, 8(5):728–741, 2011.

[43] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. Raptor: Routing attacks on privacy in tor. In *USENIX Security Symposium*, pages 271–286, 2015.

[44] Paul Syverson. Why i'm not an entropist. In *International Workshop on Security Protocols*, pages 213–230. Springer, 2009.

[45] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

[46] Chris Wacek, Henry Tan, Kevin S Bauer, and Micah Sherr. An empirical evaluation of relay selection in tor. In *NDSS*, volume 13, pages 24–27, 2013.

[47] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. Congestion-aware path selection for tor. In *International Conference on Financial Cryptography and Data Security*, pages 98–113. Springer, 2012.

[48] Bassam Zantout and Ramzi Haraty. I2p data communication system. In *Proceedings of ICN*, pages 401–409. Citeseer, 2011.

A Relay configuration diversity numbers

Vanilla	1 guard	5 guards	1 exit	5 exits
515.52	513.66	515.46	516.73	518.47
	-0.36%	-0.01%	+0.2%	+0.57%

Table 5: Guessing entropy under different network modifications with Tor algorithm (ABWRS)

Vanilla	1 guard	5 guards	1 exit	5 exits
588.24	586.35	588.31	589.58	591.36
	-0.32%	+0.01%	+0.23%	+0.53%

Table 6: Guessing entropy under different network modifications with Waterfilling algorithm (WFABWRS)

Vanilla	1 guard	5 guards	1 exit	5 exits
265.94	264.87	266.78	268.01	269.57
	-0.40%	+0.32%	+0.78%	+1.36%

Table 7: Guessing entropy under different network modifications with DeNASA method

Vanilla: 67.62	1 guard	5 guards	1 exit	5 exits
AS16276	69.31	69.31	76.02	76.02
	+2.50%	+2.50%	+12.42%	+12.42%
AS29518	67.22	67.22	66.90	66.90
	-0.59%	-0.59%	-1.06%	-1.06%

Table 8: Number of path compromised by **AS16276** under different network modifications with Tor (ABWRS)

Vanilla: 65.99	1 guard	5 guards	1 exit	5 exits
AS12876	66.52	68.64	70.94	70.94
	+0.80%	+4.02%	+7.50%	+7.50%
AS29518	65.93	65.68	65.20	65.20
	-0.09%	-0.47%	-1.20%	-1.20%

Table 9: Number of path compromised by **AS12876** under different network modifications with Waterfilling (WFABWRS)

Vanilla: 77.70	1 guard	5 guards	1 exit	5 exits
AS16276	77.44	77.44	77.48	77.48
	-0.33%	-0.33%	-0.28%	-0.28%
AS29518	78.06	78.06	78.20	78.20
	+0.46%	+0.46%	+0.64%	+0.64%

Table 10: AS-level guessing entropy under different network modifications with Tor (ABWRS)

Vanilla: 88.38	1 guard	5 guards	1 exit	5 exits
AS12876	88.33	88.15	88.04	88.04
	-0.06%	-0.26%	-0.38%	-0.38%
AS29518	88.66	88.78	88.77	88.77
	+0.32%	+0.45%	+0.44%	+0.44%

Table 11: AS-level guessing entropy under different network modifications with Waterfilling (WFABWRS)

Vanilla: 313.68	1 guard	5 guards	1 exit	5 exits
France	317.86	317.86	325.01	325.01
	+1.33%	+1.33%	+3.61%	+3.61%
Ireland	311.82	311.82	310.35	310.35
	-0.59%	-0.59%	-1.06%	-1.06%

Table 12: Number of path compromised by **France** under different network modifications with Tor (ABWRS)

Vanilla: 301.91	1 guard	5 guards	1 exit	5 exits
France	302.69	305.76	312.83	312.83
	+0.26%	+1.27%	+3.62%	+3.62%
Ireland	301.58	300.27	298.71	298.71
	-0.11%	-0.54%	-1.06%	-1.06%

Table 13: Number of path compromised by **France** under different network modifications with Waterfilling (WFABWRS)

Vanilla: 9.076	1 guard	5 guards	1 exit	5 exits
France	9.061	9.061	9.045	9.045
	-0.16%	-0.16%	-0.34%	-0.34%
Ireland	9.20	9.20	9.27	9.27
	+1.37%	+1.37%	+2.14%	+2.14%

Table 14: Country-level guessing entropy under different network modifications with Tor (ABWRS)

Vanilla: 9.502	1 guard	5 guards	1 exit	5 exits
France	9.499	9.488	9.469	9.469
	-0.03%	-0.15%	-0.35%	-0.35%
Ireland	9.53	9.62	9.70	9.70
	+0.29%	+1.24%	+2.08%	+2.08%

Table 15: Country-level guessing entropy under different network modifications with Waterfilling (WFABWRS)

Vanilla: 1509.32	1 guard	5 guards	1 exit	5 exits
AS1299	1513.39	1513.39	1525.92	1525.92
	+0.27%	+0.27%	+1.10%	+1.10%
AS7018	1500.33	1500.33	1493.26	1493.26
	-0.60%	-0.60%	-1.06%	-1.06%

Table 16: Number of path compromised by **Tier-1 AS1299** under different network modifications with Tor (ABWRS)

Vanilla: 1445.15	1 guard	5 guards	1 exit	5 exits
AS1299	1445.90	1448.85	1461.79	1461.79
	+0.05%	+0.26%	+1.15%	+1.15%
AS7018	1443.50	1436.95	1430.50	1430.50
	-0.11%	-0.57%	-1.01%	-1.01%

Table 17: Number of path compromised by **Tier-1 AS1299** under different network modifications with Waterfilling (WFABWRS)

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**