Towards Predicting Efficient and Anonymous Tor Circuits

Armon Barton, Mohsen Imani, and Jiang Ming, ***University of Texas at Arlington;***
Matthew Wright, ***Rochester Institute of Technology***

https://www.usenix.org/conference/usenixsecurity18/presentation/barton

This paper is included in the Proceedings of the 27th USENIX Security Symposium.

August 15–17, 2018 • Baltimore, MD, USA

ISBN 978-1-931971-46-1

Open access to the Proceedings of the 27th USENIX Security Symposium is sponsored by USENIX.

Towards Predicting Efficient and Anonymous Tor Circuits

Armon Barton
University of Texas at Arlington
armon.barton@mavs.uta.edu

Mohsen Imani
University of Texas at Arlington
mohsen.imani@mavs.uta.edu

Jiang Ming
University of Texas at Arlington
jiang.ming@uta.edu

Matthew Wright
Rochester Institute of Technology
matthew.wright@rit.edu

Abstract

The Tor anonymity system provides online privacy for millions of users, but it is slower than typical web browsing. To improve Tor performance, we propose *PredicTor*, a path selection technique that uses a Random Forest classifier trained on recent measurements of Tor to predict the performance of a proposed path. If the path is predicted to be fast, the client then builds a circuit using those relays. We implemented PredicTor in the Tor source code and show through live Tor experiments and Shadow simulations that PredicTor improves Tor network performance by 11% to 23% compared to Vanilla Tor and by 7% to 13% compared to the previous state-of-the-art scheme. Our experiments show that PredicTor is the first path selection algorithm to dynamically avoid highly congested nodes during times of high congestion and avoid long-distance paths during times of low congestion. We evaluate the anonymity of PredicTor using standard entropy-based and time-to-first-compromise metrics, but these cannot capture the possibility of leakage due to the use of location in path selection. To better address this, we propose a new anonymity metric called *CLASI*: Client Autonomous System Inference. CLASI is the first anonymity metric in Tor that measures an adversary's ability to infer client Autonomous Systems (ASes) by fingerprinting circuits at the network, country, and relay level. We find that CLASI shows anonymity loss for location-aware path selection algorithms, where entropy-based metrics show little to no loss of anonymity. Additionally, CLASI indicates that PredicTor has similar sender AS leakage compared to the current Tor path selection algorithm due to PredicTor building circuits that are independent of client location.

1 Introduction

Privacy threats on today's Internet include targeted advertising, large-scale user profiling, and dragnet surveillance by government agencies. These threats, along with the desire to protect freedom of speech and overcome censorship on the Internet, have resulted in an increase in public interest for anonymity systems. The Tor Network [13] in particular has received enormous attention and currently serves millions of users from all over the world. Tor users can connect to the Internet through an encrypted tunnel by first building a path through three Tor routers (called a *circuit*) chosen from a set of approximately 7,000 volunteer routers. Part of Tor's anonymity is attributed to the size of the user base, called the *anonymity set*, and attracting a large anonymity set is thus important for privacy of Tor users.

Performance. Unfortunately, Tor is slower than typical web browsing. Several groups have proposed new circuit-building approaches that aim to improve performance by optimizing properties such as bandwidth or latency. Wacek et al. [41] examined these approaches and determined that *Congestion-Aware Routing (CAR)* [42] offered the best performance-anonymity trade-off. CAR is a decentralized approach, where clients opportunistically measure circuit congestion during circuit creation and select the best one for use. This decentralized approach is limited because clients only have a small subset of relevant congestion information. Global knowledge of congestion in Tor and performance of circuits more generally would enable better choices for all clients.

Building on this insight, we propose *PredicTor*, a path selection technique that leverages performance measurements of many circuits to select less congested nodes and geographically shorter paths with greater probability. PredicTor uses a Random Forest classifier trained on recent measurements of Tor circuits to predict the performance of a proposed path. If the path is predicted to be fast, then a circuit is built using those relays. We implemented PredicTor in the Tor source code and show through simulations in Shadow that PredicTor improves Tor network performance by 23% compared to Vanilla

Tor and by 13% compared to CAR. This resulted in a speed up over Vanilla of over 500ms in the median case, and over 1.5s in the 90th percentile. We show that PredicTor utilized approximately 30% more Tor relays compared to Vanilla resulting in greater load distribution and allowing PredicTor to make better use of limited network resources.

Moreover, we performed live Tor experiments and show that PredicTor improves network performance partly due to avoiding highly congested nodes, and partly due to building lower latency circuits. In fact, PredicTor is the first path selection algorithm that dynamically considers both congestion and latency according to the state of the live Tor Network. In the live Tor experiments, during times of high congestion, we show an improvement of 7%-13% in the median case for PredicTor compared to Vanilla Tor.

Measuring Anonymity. Any proposal for building efficient Tor circuits must thoroughly evaluate anonymity. For example, a method that focuses on using high-bandwidth relays could concentrate traffic into fewer nodes, making it easier for a few attackers to compromise more circuits. Unfortunately, existing metrics do not address all aspects of Tor that need to be considered in evaluating new path selection proposals.

Current anonymity metrics fall into two complimentary categories: methods that aim to quantify anonymity and metrics that empirically measure all-or-nothing compromises. Most metrics that quantify anonymity are based on entropy [31, 11, 30]. Syverson [38] points out that while entropy-based metrics represent the average case well, they do not represent worst-case scenarios well. Other quantification methods [5, 4] perform information theoretic inferences about Tor clients with probability equal to $1/|N| + \delta$ where N is the number of clients, and δ is the degree to which the inference is successful beyond a best guess. Due to the large user base in Tor, these inference probabilities can be minuscule. Therefore, it is difficult to justify how these inference probabilities may indicate an advantage for an adversary to fully compromise anonymity.

The latter category, metrics that empirically measure all-or-nothing compromises, includes time-to-first compromise, a measure of how long it takes until a client uses a compromised circuit [24]. Though such metrics give us a good understanding of properties that lead to full deanonymization, they offer less insight into the state of anonymity for users that have not been fully deanonymized. As such, we need a metric that offers some insight into the state of anonymity before full deanonymization and one that shows an adversary's ability to infer key attributes about the user.

In this work, we present an anonymity metric called *CLASI* (Client AS Inference). CLASI measures an all-knowing adversary's ability to infer clients' Autonomous Systems (ASes) by fingerprinting their circuits at the network and country level, along with other auxiliary information such as relay bandwidth. We give our adversary full knowledge of all the connections in the Tor Network, and our results thus represent an upper bound. Information revealed about the clients' AS, rather than the client directly, could potentially be more useful for adversaries for several reasons:

– The number of popular Tor client ASes is far lower than the number of Tor clients. Thus, inferring a client's AS is more achievable and may be a first step in reconnaissance for an adversary;

– High-resource adversaries such as nation-states are known to target ASes for infiltration in efforts to passively observe network traffic;

– Making inferences at the client level may yield negligible results due to $Pr[1/|N| + \delta]$ being small in most cases, especially when N is large.

We evaluate this method empirically by testing a recently proposed *location-aware* algorithm called DeNASA [6]. Comparing DeNASA to Vanilla Tor, we find anonymity loss using CLASI that is not apparent when using entropy-based metrics. We note that DeNASA is not a performance-based path selection algorithm, but rather that it seeks to improve security by routing around network-level adversaries to avoid traffic analysis attacks. Thus, CLASI can be useful for evaluating other such algorithms [14, 35, 6, 23, 17] and for schemes that seek to avoid active BGP hijacking attacks [37, 36].

Finally, we evaluate the anonymity of *PredicTor* using both CLASI and entropy-based metrics. We find that AS leakage for PredicTor is similar to Vanilla and slightly better than CAR due to PredicTor clients building paths independently of their own network location.

Contributions In summary, we make the following contributions:

1. We show circuit classification accuracy for machine learning algorithms that are trained using data currently available from the Tor consensus files.
2. We present PredicTor, implement it in the Tor source code and show significant performance benefits in Shadow simulations.
3. We perform live Tor experiments and show that PredicTor is the first path selection algorithm to dynamically optimize for congestion and path length depending on path conditions.
4. We present the CLASI anonymity metric. Our evaluation shows that CLASI indicates anonymity loss for location-aware path selection algorithms where entropy-based metrics show little to no loss of anonymity.

5. We evaluate PredicTor with CLASI and other metrics and find that PredicTor's path selection maintains high anonymity.

2 Background and Related Work

Tor is a low-latency anonymity system for TCP-based applications [13]. The Tor network comprises approximately 7000 volunteer-operated relays [26] that are deployed throughout the world. It was recently shown by Jansen et al. [21] that Tor has approximately 550,000 active users at any given time. Each client selects a three-hop path of relays and builds a multi-hop encrypted tunnel, called a *circuit*, through this path. The first, middle, and last relays on the circuit are called the *guard*, *middle*, and *exit* relays, respectively.

A client uses a single guard node as the first hop for all of its circuits to help prevent attacks such as the *predecessor attack* [43, 44, 29], the selective denial of service attack [7], and statistical profiling. A new guard is chosen only if the presently selected guard becomes unavailable, or if a period of 60 days to 9 months is reached [12].

To provide fast connections for web browsing, relays are selected for circuits such that traffic is evenly distributed over the available bandwidth in the Tor Network. A set of *directory servers* are responsible for securely maintaining the list of relays, along with their bandwidths and other information. Once per hour, each client receives a *consensus document* from the directory servers, and this document contains weights assigned to each relay based on the relay's position in the circuit and its bandwidth. Then, load balancing is achieved by selecting each relay in proportion to its consensus weights.

2.1 Improving Network Performance

Tor is slower than typical web browsing, and a number of research groups have attempted to address this [34, 33, 1]. Wacek et al. [41] examined these approaches and determined that *Congestion-Aware Routing* (CAR) [42] offered the best performance-anonymity trade-off. In this paper, we thus use CAR as a benchmark for comparison.

CAR aims to intelligently select Tor circuits with the lowest levels of congestion. Congestion measurements for circuits are performed by the clients by sampling round-trip times (RTTs) of both circuit-building and application connections. A circuit is selected for use only if it's measured *congestion time* (the current RTT minus the shortest RTT) is the lowest out of three randomly selected circuits. If at any point during the life of that circuit, the mean of the last five congestion times is greater than 0.5 seconds, the client will switch to another circuit.

2.2 Measuring Anonymity

The existing literature provides substantial contributions in measuring Tor's anonymity [31, 11, 34, 30, 5]. Our approach, CLASI, builds on the AnoA framework proposed by Backes et al. [4] for computing quantitative bounds on the anonymity in Tor. The AnoA framework is modeled as a challenge-response game between an adversary and a challenger. The adversary possesses two tables (D_0 *and* D_1) in which each line is populated with a sender, a receiver, and auxiliary information. The two tables differ in exactly one row in the sender field. For this special row, the sender field for D_0 contains sender S_0, and the sender field for D_1 contains sender S_1. The adversary A sends tables (D_0 *and* D_1) to a challenger CH. The challenger chooses D_b according to its input b where $b \subseteq \{0,1\}$, and successively feeds each row to an idealized Tor protocol. At any point, the adversary outputs their decision b. Sender anonymity for this protocol is then measured in terms of δ where:

$$Pr[b = 0 : 0 \leftarrow A, CH(0)] \leq Pr[b = 0 : 0 \leftarrow A, CH(1)] + \delta.$$

As anonymity of the protocol decreases, δ increases due to the fact that adversary A guesses b correctly with greater probability.

We use a framework similar to AnoA as the foundation for designing our CLASI metric. The most important and distinguishing characteristics of the CLASI metric are:

- We equip the adversary with a probabilistic classification model trained on realistic Tor simulated data.
- Our adversary is all-knowing, and thus our metric provides an upper-bound.
- Our adversary classification model is configured to infer the Autonomous System of the client.

In the CLASI classification model, we use three features for each relay in a circuit: the bandwidth of the relay from the consensus file (BW), and the network (AS) and country (CC) that the relay is located in. We decided to use AS, CC, and BW features because the proposed path selection algorithms in Tor are generally designed to optimize performance or security based on relay bandwidth [34], network location of relays [1, 33], or by routing around relays that located in certain ASes [14, 35, 6, 23]. We measure an all-knowing adversary's ability to infer clients' ASes because knowledge of the clients' AS is a probable first step for adversary reconnaissance. The CLASI design and evaluation are described in Sections 6 and 7, respectively.

2.3 Related Work

Routing Protocols. Snader and Borisov [34] proposed a change to Tor's path selection algorithm that allows

the client to tune the degree to which relay selection is weighted in proportion to bandwidth. The tunable parameter can be increased to bias relay selection in favor of high bandwidth relays or decreased to reduce that bias and induce more uniform relay selection. A limitation of this approach is that selecting relays weighted too heavily towards bandwidth can cause high-bandwidth relays to become overloaded and low-bandwidth relays to become starved, resulting in poor performance.

Sherr et al. [33] proposed a latency-aware relay selection strategy in which relays participate in a virtual coordinate embedding system. Clients then estimate the latencies of anonymous circuits by summing the virtual distances between relays' advertised coordinates. Akhoondi et al. [1] proposed an approach that aims to reduce latency of paths by accounting for inferred locations of relays while choosing paths. Some limitations to these approaches were pointed out by Wacek et al. [41], who performed an empirical study in which they compared the routing protocols mentioned above. Their results indicate that relay selection algorithms perform best when bandwidth is considered as a factor. Moreover, CAR was shown to perform close to the best in throughput and time-to-first-byte, in addition to significantly outperforming other algorithms in anonymity.

One important disadvantage of CAR is that circuit-RTTs can be manipulated during circuit creation by malicious exit nodes. This disadvantage is compounded in another similar approach called Navigator [3], in which active RTT measurements and a-priori information from the distribution of globally measured RTT values are used to select circuits. Additionally, Geddes et. al [15] suggested that the use of RTT measurements for latency improvements also results in an increase in the effectiveness of latency-based attacks.

More recently, Geddes et al. [16] proposed ABRA (the avoiding bottlenecks relay algorithm). Their approach aims to increase network utilization by having relays estimate the extent to which they are a bottleneck on each circuit and spread this information to clients. They showed that ABRA results had better network utilization compared to CAR. However, they did not show results for time-to-first-byte or time-to-last-byte measurements, so there is no evidence that ABRA offers any improvement in these measures of end-user performance.

Anonymity Metrics. Existing anonymity metrics for Tor can be categorized into works that use information theoretic or rigorous methods to quantify anonymity of Tor users and works that aim to empirically measure all-or-nothing compromises of Tor users. Our proposed anonymity metric lies within the former category.

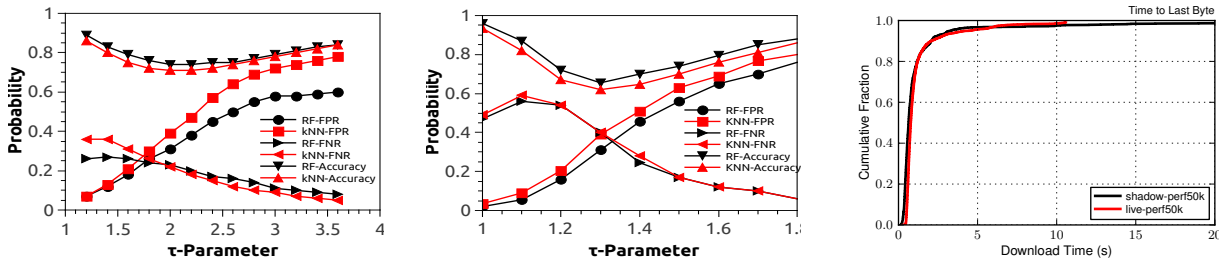
In the area of quantifying anonymity, Serjantov and Danezis [31] and Diaz et al. [11] propose using Shannon entropy [32] to measure the uncertainty of the distribution of guard/exit pairs selected by senders. Rochet et al. [30] proposed a metric based on *guessing entropy* that indicates the expected number of nodes that must be compromised in order to mount a successful correlation attack. Snader and Borisov [34] apply the Gini coefficient to measure the equality of selection probability for Tor relays. A Gini coefficient of 0 means all relays were chosen with equal frequency (maximal anonymity), and a coefficient of 1 means the same relay was always chosen (minimal anonymity).

One limitation for entropy based metrics – pointed out by Syverson [38] – is that the results can be misleading because the worst case is not always represented. Additionally, entropy does not indicate a loss in anonymity if clients select relays differently, as long as the distribution of selected relays is near uniform. To consider an extreme example, suppose client A always selects relay X and client B always selects relay Z; the entropy would be 1. This is a misleading result in terms of anonymity because both clients are fully identifiable with knowledge of their selected relay.

To establish tight upper bounds on anonymity, Meiser et al. [5] presented a rigorous methodology for quantifying anonymity of Tor with respect to budget adversaries. In their analysis, they show anonymity impact for a system with two senders connecting to two receivers using several proposed path selection algorithms over an idealized Tor network. Their analysis, however, does not show anonymity impact for users who are masked within large anonymity sets or for varying user destinations. In our proposed metric, these parameters are tunable, allowing researchers to understand anonymity impact for different client models and different user models.

In the area of empirical measurement – being complimentary to anonymity quantification metrics such as our proposal – Johnson et al. [24] measured time to first compromise by *relay-level* and *AS-level* adversaries by modeling the Tor network and taking empirical measurements. Murdoch and Watson [28] presented an analysis of proposed path selection algorithms against adversaries that deploy malicious Tor nodes. Sun et al. [36] proposed a metric that measures the resilience of the Tor network to active attacks on BGP routing called *RAPTOR* attacks.

These empirical measurement approaches are complimentary to our proposed metric because they measure all-or-nothing compromises, while our metric quantifies the ability of an all-knowing adversary to infer clients' ASes – a property that could lead to a compromise and thus indicates a loss of anonymity for path selection algorithms under study.



(a) Shadow Accuracy (b) Live Tor Accuracy (c) Live Tor Circuit Length

Figure 1: Effect of varying τ on accuracy, false positive rate, and false negative rate for k-NN and Random Forest in (a) Shadow and (b) Live Tor. (c) TTLB for Shadow *perf clients* compared to live Tor perf [26] for 50KiB downloads.

3 Path Classification

In this section, we motivate the design of PredicTor by first showing machine learning classification results for k-NN and Random Forest models trained on Tor descriptor data. Our goal is to classify Tor circuits into two classes: fast and slow. We used two distinct methods for acquiring the training data and show results for both.

Shadow Data. In the first method, we ran a Tor network simulation with 1000 clients using Shadow [20], a discrete-time event simulator. More details about the simulation are discussed at the end of this section. We generated a training set of 120,000 streams from one simulation run and a testing set of 25,000 streams from another simulation run. Each stream consisted of a *Vanilla client* downloading a file from a server through a circuit. For each stream, we recorded the time-to-last-byte (TTLB) download time that was measured from the client during the simulation. We then set a threshold τ and labeled each data point as *True* if the TTLB was less than τ, i.e. the stream was fast, and *False* if the TTLB was greater than τ, i.e. the stream was slow.

Live Tor Data. In the second method, we gathered training data from the live Tor Network by deploying a server that hosted 20 VMs, each running Tor version 0.3.0.9. From each VM, circuits were built over the live Tor network and requests were made to download an 80 KiB file from a US destination server. For each file download, we measured the time-to-last-byte download time. The labels were then set to *True* if the TTLB was less than τ, and *False* if the TTLB was greater than τ. Using this technique, we collected approximately 50,000 *training samples* on Dec. 5, 2017 from approximately 17:00 to 18:00 GMT. Then, during the subsequent hour (18:00 to 19:00 GMT), we collected approximately 20,000 *testing samples* .

Feature Set. In Tor circuits, there is a relationship between download times and the consensus bandwidth of each relay, as well as between download times and the network location of each relay. Due to this relationship, we believe that a recognizable pattern exists such that download times can be predicted (to some degree) by inspecting bandwidth and network location of each relay in a circuit. As such, we resolve each relay into three features: 1) Autonomous System (AS), 2) Country Code (CC), and 3) Consensus Bandwidth (BW). This yields nine features for the circuit in total. For the *AS features*, we use the AS number directly as an integer. For the *CC features*, we use the decimal representation in ASCII of the two-character country code. For the *BW features*, the consensus bandwidth is used and represented as an integer. We used distance-weighted k-NN with $k = 9$ [27], and Random Forests [8] to classify each circuit into class True (Fast) or False (Slow). We tested the k-NN model with $k = 3, 5, 7, 9$, and 11, and found that $k = 9$ produced the highest accuracy compared to other values of k.

Classification Accuracy. Figures 1a and 1b show the accuracy, false positive rate, and false negative rate of the k-NN and Random Forest models in predicting circuit performance with respect to varying τ. For both models, as τ increases, we observed an increase in false positive rate and a decrease in false negative rate. The false positive rate and false negative rate converge near the median download time for the training data. The median download time for Shadow and Live Tor was approximately 1.8s and 1.4s, respectively. The Shadow results in Figure 1a show the accuracy at the median to be 76% and 70% for Random Forest and k-NN, respectively. The false positive rate and false negative rate at the median was approximately 25% for both k-NN and Random Forest. The Live Tor results in Figure 1b show the accuracy at the median to be 70% and 64% for Random Forest and k-NN, respectively. The false positive rate and false negative rate at the median was approximately 45% and 28%, respectively, for both k-NN and Random Forest.

For both Shadow and Live Tor models, accuracy is minimal at the median download time. For greater values of τ, both accuracy and false positive rate increase. Likewise, for lower values of τ, both accuracy and false negative rate increase. In the context of predicting fast circuits for Tor clients, high values of τ allow clients to accept a large percentage of slow circuits due to the high false positive rate. Low values of τ cause clients to become more selective in general and lead to dramatically higher circuit build times. Based on these results, we use Random Forest for PredicTor's classification model with the τ parameter always set to the median download time with respect to the training data.

Shadow Simulation Details. Our Shadow configuration consisted of 1000 clients from the top 10 countries by directly connecting users [26], 400 relays from a live Tor Consensus, and 70 destination servers from the Alexa list of top websites [2]—forming a client to relay ratio of 2.5 : 1. All clients and relays were assigned to an enhanced network topology of 17,250 vertices and 150 Million edges based on their AS [19]. In this simulation, there were two classes of clients, *web clients*, and *perf clients*. *Web clients* randomly selected servers from which they performed HTTP GET requests to download 320 KiB files over the modeled Tor network [18], and *perf clients* downloaded 50 KiB files over the Tor network. Each client measured the time from when the first request was made to when the last byte was received (*TTLB*). We validated our Tor model against live Tor by comparing the results of *perf clients* to historical Tor data from Tor Metrics [26]. Figure 1c shows the live Tor performance for fixed file size downloads of 50 KiB from historical Tor network data [26] compared to Shadow *perf clients*. The results show that live Tor performance was not significantly different than Shadow *perf client* performance for our simulation, indicating that our Tor model performs statistically similar to live Tor.

4 Speeding up Tor with PredicTor

We now describe PredicTor, our proposed approach for improving Tor path selection. In PredicTor, the guard selection policy is identical to Vanilla Tor, and a client will use a single guard as long as it is available for up to nine months. To complete a path, middle and exit relays are selected according to consensus bandwidth weights as per standard Tor protocol. The resulting proposed circuit is then classified by a classification model as described in Section 3. If the proposed circuit is predicted to be fast, the circuit is built; otherwise, new relays are selected.

Let us define function $M_\tau(C)$ that, for a given threshold τ, returns *True* when a proposed circuit C is predicted to be faster than τ and *False* when it is predicted to be slower than τ. In the PredicTor path selection method, Tor proposes C as per standard bandwidth weighted selection. Then, if $M_\tau(C) == True$, the circuit is built. Otherwise, the loop runs until the condition is met. Note that when τ is set to be the median download time, then the loop runs two times on average, though this can vary between clients and depends on the guard selected.

Experimental Setup. We implemented PredicTor in the Tor source code and tested its performance compared to Vanilla, Congestion-aware routing (CAR), and Snader and Borisov (SB) path selection using both *Shadow* and *live Tor*. Prior work shows that *SB* has competitive performance under medium congestion with the parameter s set to 9 [41, 34]. We tested *SB* with two settings: *SB-9*, with $s = 9$ for *partial* bias to high bandwidth and *SB-15*, with $s = 15$ for *heavy* bias to high bandwidth.

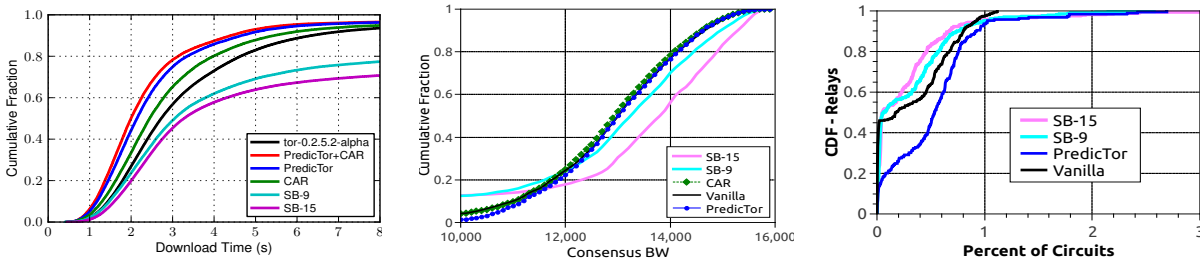
In the Shadow simulation, we used the same configuration as described in section 3. For all path selection techniques, the respective clients requested a 320 KiB file download from a server selected uniformly at random from a set of 70 destination servers.

In the live Tor experiments, for all path selection techniques, the respective clients requested the home page of websites selected uniformly at random from a set of 1000 sites from the Alexa list of top sites [2].

For the PredicTor experiment in both Shadow and live Tor, τ was set to the median download time with respect to the training set, and Random Forest was used for the classification model. Note that in a Shadow simulation, we can observe how performance is affected when *all clients* use a given path selection technique. This is not possible in live Tor because we can only deploy an insignificant fraction of clients compared to the full user base. However, two of Shadow's limitations are: 1) the network size is significantly smaller than the real-world Tor Network, and 2) the simulation does not fully model real-world network dynamics. In live Tor, we can observe how path selection techniques respond under dynamic real-world network conditions. Therefore, for measuring performance of path selection techniques, it is useful to test in both Shadow and live Tor.

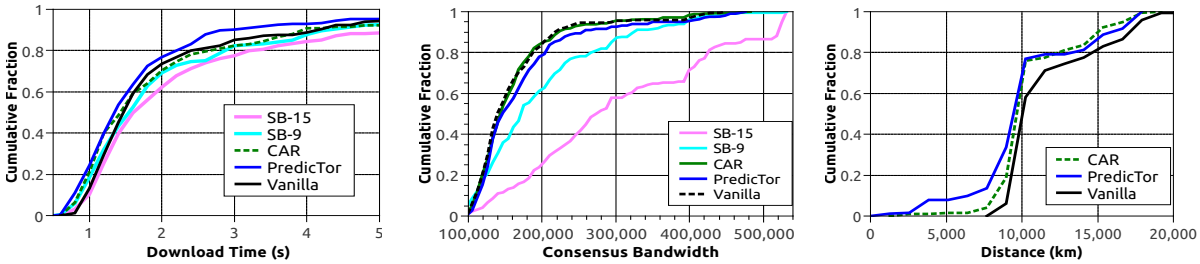
Performance Results. Figures 2a and 3a show Shadow and live Tor download times for Vanilla, PredicTor, CAR, SB-9, and SB-15. In both Shadow and live Tor, PredicTor was the fastest. In the Shadow simulation, PredicTor had a 23% and 13% median improvement compared to Vanilla and CAR, respectively, and a 28% improvement in the 90th percentile compared to Vanilla. This resulted in a speed up over Vanilla of over 500ms in the median case, and over 1.5s in the 90th percentile. In the live Tor experiments, PredicTor had 11% and 6% median improvements compared to Vanilla and CAR, respectively, and a 28% improvement in the 90th percentile compared to Vanilla. This resulted in a speed up of over 1.0 second in the 90th percentile compared to Vanilla.

Circuit Bandwidth. SB-9 and SB-15 performed the slowest in both Shadow and live Tor. Figures 2b and 3b show the Shadow and live Tor circuit consensus bandwidths for Vanilla, PredicTor, SB-9, and SB-15. As expected, SB-9 and SB-15 build circuits with significantly higher bandwidth compared to other techniques, particularly in live Tor, where SB-9 and SB-15 circuits used 22% and 97% more bandwidth in the median than Vanilla. The Shadow results suggest that selecting relays weighted heavily towards bandwidth causes high bandwidth relays to become overloaded, resulting in poor per-



(a) Shadow Download Times (b) Shadow Circuit Bandwidth (c) Shadow Relay Utilization

Figure 2: **Shadow Experiments:** a) TTLB. b) Circuit consensus bandwidth. c) Relay utilization.



(a) Live Tor Download Times (b) Live Tor Circuit Bandwidth (c) Live Tor Circuit Length

Figure 3: **Live Tor Experiments:** a) TTLB. b) Circuit consensus bandwidth. c) Circuit length.

formance. Moreover, the live Tor performance and consensus bandwidth results suggest that high-bandwidth relays experience some persistent congestion due to a large user base using Vanilla's bandwidth-weighted selection policy. The persistent congestion causes poor performance even if one client weights their selection heavily toward these high bandwidth relays.

Performance gains in PredicTor, on the other hand, cannot be attributed exclusively to selecting high-bandwidth relays. In Shadow, PredicTor did not build higher bandwidth circuits compared to Vanilla. In live Tor, Predictor uses approximately the same median consensus bandwidth as Vanilla, though it uses 25% more bandwidth at the 90th percentile.

Node Congestion. Wang et al. [42] concluded that congestion is a property of the Tor router itself. Though congestion comes in bursts in the short term, each node's congestion characteristics do persist over time, and thus some nodes are consistently more congested than others.

Figure 2c shows the empirical distribution (ECDF) of relays with respect to the percent of circuits that they were used on for the Shadow simulation. Vanilla completely avoided selecting approximately 45% of relays in the network, and SB-9 and SB-15 completely avoided selecting approximately 50% of relays in the network. Under-utilizing the network in this way caused more persistent congestion on the 65% and 50% of relays that Vanilla and SB did utilize, respectively. On the other hand, PredicTor utilized approximately 85% of the network. These results suggest that, when all clients use PredicTor, more relays are utilized, resulting in greater load distribution and lower persistent congestion for high-bandwidth relays.

PredicTor+CAR. One advantage for PredicTor compared to CAR is that clients have more global knowledge of persistent network congestion for all nodes during circuit creation. This helps PredicTor clients avoid consistently congested nodes and select consistently non-congested nodes with greater probability. In CAR, on the other hand, clients only have knowledge of congestion characteristics for a small subset of nodes that are opportunistically measured during circuit creation.

We combined PredicTor with CAR because we suspected that CAR should have better performance if nodes are selected using the PredicTor scheme first, then opportunistically measured. Figure 2a shows a 28% improvement in the median case for PredicTor+CAR compared to Vanilla. These results indicate that a hybridized scheme combining centralized and decentralized congestion measurements for relay selection results in better performance compared to either scheme alone.

Circuit Length. Although geographic distance is not a good measure for Internet latency, it can provide a point of reference for a system like Tor, where a circuit might traverse multiple intercontinental hops. Figure 3c shows live Tor circuit lengths for Vanilla, CAR, and PredicTor. To measure circuit lengths, we first resolved each relay into coordinates. Then, we calculated the distance between relays using Vincenty's Formula [40]. The circuit length was taken as the sum of the distances between the guard and middle and between the middle and exit. In the median case, PredicTor built circuits that were approximately 680 km shorter compared to Vanilla. In the 90th percentile, PredicTor and CAR built circuits that were approximately 592 km and 2,043 km shorter compared to Vanilla, respectively. These results suggest that the performance gains for PredicTor and CAR are partially due

to building shorter circuits, and thus, circuits of lower latency.

4.1 Discussion

We conclude that performance gains for PredicTor are achieved by considering three key factors: 1) congestion, 2) bandwidth, and 3) latency. From the Shadow simulation, we observed that PredicTor utilizes the network in a way that leads to more efficient load distribution and lower congestion for high-bandwidth nodes. Additionally, the live Tor results suggest that PredicTor avoids highly congested nodes while building circuits of slightly higher bandwidth and lower latency compared to Vanilla.

Quantifying improvement. Our experiments provide strong evidence that PredicTor should result in an overall improvement for all clients in Tor, with 23% improvement in Shadow with all clients using PredicTor and 11% improvement in the live Tor experiments with one client using PredicTor. We do not claim, however, that our experiments can show the exact quantitative gains that PredicTor would provide when deployed in Tor. One way to more fully quantify the improvement for a live deployment of PredicTor would be to test in a wide-area testbed where all clients use PredicTor. Since no such testbed was available for this study, we leave this for future work.

Malicious Relays. We also highlight some important mitigation steps against relays that attempt to manipulate their bandwidth contribution during live PredicTor measurements to win more traffic. First, PredicTor selects guards exactly the same way as Vanilla, by consensus weight. Thus, malicious relays cannot win more guard traffic because they do not have the ability to change their consensus weight by gaming PredicTor measurements. A malicious exit relay may attempt to win more traffic by prioritizing measurement circuits and throttling all other connections, thereby appearing fast during measurement. This can be mitigated by selecting probe destinations from the distribution of most popular destination websites as observed from (honest) exit nodes, reported safely using a system like PrivCount [22]. Since popularity of websites is heavily concentrated in relatively few sites [10], a moderate-sized list of probe destinations should suffice to make the attacker unable to distinguish quickly between a measurement circuit and the majority of non-measurement user activity.

In contrast, a major disadvantage for methods that use RTT measurements such as CAR and Navigator is that malicious exit nodes can easily manipulate RTT measurements. Geddes et. al [15] show how the use of RTT measurements for latency improvements results in an increase in the effectiveness of latency-based attacks.

5 Client Location and Guard Diversity

We performed additional live Tor experiments from three additional client locations: 1) United States (US), 2) Germany (DE), and 3) Japan (JP). For each client location, the experiment was performed during prime Internet surfing hours for both the US and Europe (approximately 14:00 GMT) and during a time that is evening in the US and middle of the night in Europe (approximately 00:00 GMT). We call the experiments run at 14:00 GMT as the *high-congestion* condition and the experiments run at 00:00 GMT as the *low-congestion* condition.

Due to the single-guard selection strategy in Tor, clients may be connected to a slow or fast guard for long periods of time. We desire to understand PredicTor performance when connected to guard nodes of various consensus weights. Thus, for each client location, the experiment was performed using a *slow guard* (consensus weight 1770) and a *fast guard* (consensus weight 35600).

In Appendix A, Table 1, we show the median and 90th percentile improvement for PredicTor compared to Vanilla. We observed that the best performance improvement from PredicTor was realized during times of high congestion while connected to a fast guard. From the US location, there was a 9.7% and 17.7% improvement in the median and 90th percentile, respectively. From the DE location, there was a 12.8% and 25.3% improvement in the median and 90th percentile, respectively. From the JP location, there was a 6.3% and 10.8% improvement in the median and 90th percentile, respectively.

During times of low congestion while connected to a fast guard, PredicTor performance did not improve compared to Vanilla as much as in the high-congestion experiment. The median improvements from the US and DE were 4.2% and 7.3%, respectively. Wang et al. [42] also state that CAR should get better performance during high congestion times compared to low congestion.

We observed a slight improvement for PredicTor while connected to a slow guard during both high and low congestion for the median time (3.3% to 7.3% faster). For slow guards with high congestion, PredicTor showed larger improvements for the 90th percentile, between 14.0% and 19.1% improvement. Slow guards typically act as a bottleneck in most circuits, but when congestion is high, PredicTor can find and select faster circuits.

Circuit Distance. In Appendix A, Table 1, we show the median and 90th percentile circuit distance improvements for PredicTor compared to Vanilla. We observed the best improvement in circuit distance for PredicTor during times of low congestion while connected to a fast guard. From the US location, there was a 52% improvement in the median, with circuits that were approximately 1,600 km shorter. Similar results were observed for the DE and JP locations. During times of high

congestion while connected to a fast guard, PredicTor showed more modest improvements in circuit distance of between 11% and 26% in the medians. We observed little to no improvement in circuit distance for the slow guard experiments. We believe this is due to the slow guard acting as a bottleneck in most connections.

We conclude that PredicTor intelligently picks relays in a way that has never been done by any other algorithm. During times of high congestion, PredicTor correctly avoids highly congested nodes. During times of low congestion, when there are fewer congested nodes to avoid, PredicTor correctly builds lower-latency circuits of shorter geographic distance.

6 CLASI: Client AS Inference

Since PredicTor and other Tor path selection algorithms such as TAPS [23], DeNASA [6], and LASTor [1] use network location information to select paths, it is important to understand the extent to which these choices lead to predictability and loss of anonymity. In particular, we seek to understand whether an attacker can infer something about the location of the client from the choices of paths that she makes. To this end, we now describe CLASI, a metric for measuring the ability of the attacker to infer the client AS. If the client AS is known, then the adversary may be able to efficiently target clients with a BGP hijacking attack [37]. Additionally, many state-level adversaries are known to collude with ISPs [24]. As such, an adversary may target ISPs that are suspected to serve clients.

CLASI is a challenge-response game between an adversary and a challenger. The adversary possesses a path simulator *PS* that is an idealized Tor network with a given path selection algorithm to generate paths over the sender space S, the relay space R, and the destination space D. We denote one path P being generated from *PS* as $P \leftarrow PS$, and a set of paths $\mathbf{P}$ being generated from *PS* as $\mathbf{P} \leftarrow PS$. Each path P is a set of nodes where $P = \{p_1, p_2, p_3, p_4, p_5\}$, such that: $p_1 = clientIP, p_2 = guardIP, p_3 = middleIP$, $p_4 = exitIP, p_5 = destinationIP$.

Adversary A sends path simulator *PS* to the challenger *CH*. *CH* generates a path P' from *PS* and removes the sender p'_1 such that $P' = \{p'_2, p'_3, p'_4, p'_5\}$. *CH* then sends P' to A. A attempts to predict the network location L of sender p'_1. More precisely, L is the sender's AS, and we let S_L be the set of all possible sender ASes. Then let L' be A's prediction for the location of the sender. Sender location information leakage for the idealized Tor network is then represented by ε_s, where:

$$Pr[L = L'] = \frac{1}{|S_L|} + \varepsilon_s.$$

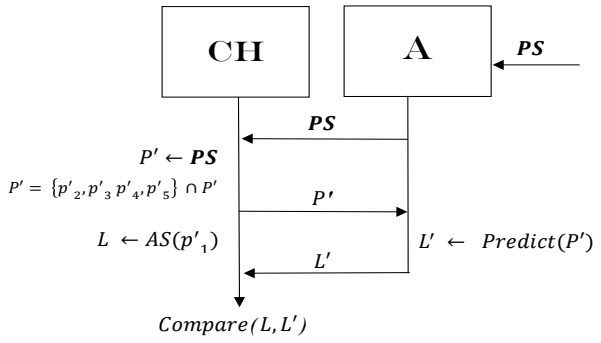


Figure 4: CLASI challenge-response game between adversary and challenger.

There is no leakage ($\varepsilon_s = 0$) if the attacker can do no better than guessing L' uniformly at random from among all possible sender ASes in S_L. Otherwise, the attacker has some advantage in guessing the sender's AS ($\varepsilon_s > 0$).

The CLASI challenge-response game sequence is shown in Figure 4. Adversary A uses the function $Predict(P)$ that extracts the features from the path P and uses them to classify the AS of the sender. Each Tor relay has the features AS, BW, and CC, represented as described in Section 3, while the destination has the features AS and CC. *Predict* uses a probabilistic classification model that is trained on the feature set of paths generated from *PS* and labels that represent the sender's AS. In our evaluation, we used k-NN with $k = 9$ for the adversary classification model. It is possible that different classifier models each tuned to the system in use could improve the adversary's performance, but using one high-quality model allows us to quantitatively compare client AS leakage between different path selection techniques.

7 CLASI Evaluation

We now evaluate CLASI's ability to measure sender location information by running a simulation of the Tor network using TorPS [39] and testing a location-aware Tor path selection protocol called DeNASA [6].

7.1 Tor Model

TorPS [39] is a Tor path selection simulator that uses historical data to recreate network conditions experienced by Tor users in the real world [24]. Circuits are created according to past network state, and streams are attached to those circuits according to simulated user behavior.

For the set of destinations used in our simulation, we tested the 200 top Alexa [2] websites. We modeled clients connecting from the top 10 countries by directly connecting users according to Tor Metrics [26]. The sim-

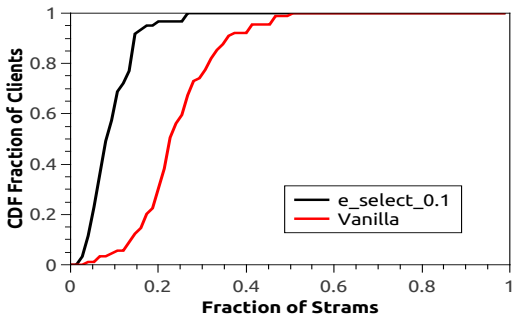


Figure 5: Fraction of vulnerable streams for DeNASA e-select compared to Vanilla with respect to AS adversaries.

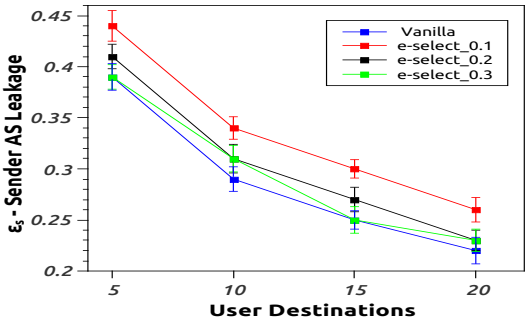


Figure 6: **CLASI:** Sender AS leakage for DeNASA exit selection variations compared to Vanilla Tor as a function of user model.

ulated clients connected from distinct ASes chosen partly from the list proposed by Edmond and Syverson [14], partly from the list proposed by Juen [25], and partly from CAIDA Top Ranking ASes [9].

We tested four distinct users models: 1) 5-destination, 2) 10-destination, 3) 15-destination, and 4) 20-destination. According to the number of destinations specified for each user model, clients selected their destinations uniformly at random from the set of 200 sites at the start of the simulation. During the simulation, clients connected to destinations selected uniformly at random from this pre-selected set.

7.2 DeNASA Protocol

Location-aware protocols are designed to increase security against AS-level threats [24], or the threat of BGP hijacking attacks [37]. For our evaluation, we chose to test a location-aware protocol called DeNASA (destination-naive AS-awareness) because DeNASA's tunable parameters allow users to increase or decrease location awareness in exchange for more or less security against AS-level adversaries respectively. We would expect, however, that increasing location awareness would cause an increase in sender information leakage.

DeNASA increases security against AS-level adversaries by creating circuits that have higher probability of avoiding some Tier 1 ASes from the client to guard, and simultaneously from exit to destination. Barton et al. identify eight Tier 1 ASes, called *suspect ASes* that are the most likely to appear on both sides of Tor circuits.

The two methods used in DeNASA are: 1) *e-select*, and 2) *g-select*. *E-select* determines how clients select exit relays based on a tunable parameter τ ranging from 0 to 1. When $\tau = 0.1$ clients are restricted to selecting from a smaller set of exits that have lower probability of traversing the suspect ASes. Additionally, the set of exits available for each client is dependent on the client's location. As τ increases, the restriction is relaxed.

Using the described TorPS configuration, we ran a nine month simulation for e-select $\tau = 0.1$ and Vanilla. We denoted streams as being *vulnerable* if AS3356 or AS1299 appeared on both sides of the stream. As shown in Fig. 5, the median vulnerable stream rate for e-select and Vanilla was 8% and 22% respectively – indicating that e-select builds 63% fewer vulnerable streams compared to Vanilla.

The *g-select* method ensures that clients only select guards for which there are no suspect ASes in the AS-level path from client to guard. The suspect AS list is tunable such that the client can avoid from one to eight suspect ASes. By avoiding more suspect ASes from the client side, clients maintain a more restrictive set of possible guards to choose. Additionally, the set of guards available for each client is dependent on the client's location.

7.3 Experiments

For each experiment, we generated 1.8 million Tor paths to train the CLASI adversary classification model. We then ran the CLASI challenge-response game for 3,000 new paths on which the adversary made prediction attempts. For each data point, this process was repeated 30 times and we plot the mean along with a 95% confidence interval. In Figure 6, we plot sender AS leakage for different variants of *e-select* compared to vanilla Tor. The Figure indicates that sender AS leakage is higher for *e-select* compared to vanilla Tor. Moreover, sender AS leakage increases for *e-select* as the threshold τ is decreased. For example, for the *5-*, *10-*, *15-*, and *20-destination* user models, sender AS leakage increased by 7%, 10%, 11%, and 13%, respectively, for $\tau = 0.1$ compared to $\tau = 0.2$. We found similar results when τ was increased from 0.2 to 0.3. This was expected due to the set of exits being more restrictive for clients when using lower values for τ.

Vanilla path selection. Sender leakage should be equivalent to random guessing ($\varepsilon_s = 0$) for uniform relay and uniform destination selection. For Vanilla path selection, we observed that sender leakage was signifi-

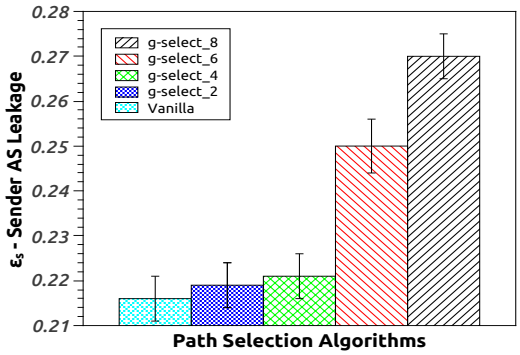


Figure 7: **CLASI:** Sender AS leakage for DeNASA guard selection variations compared to Vanilla Tor.

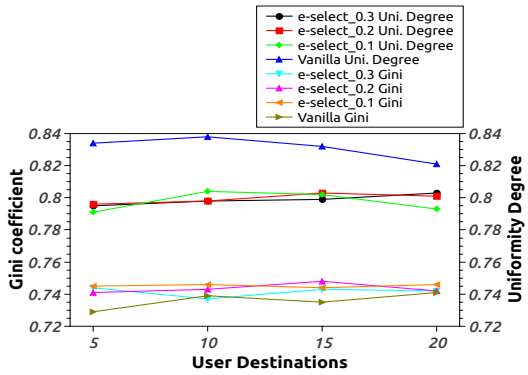


Figure 8: Gini Coefficient and Uniformity degree for DeNASA exit selection variations compared to Vanilla Tor.

cantly higher than random guessing. This is due to the fact that: 1) clients are partitioned into subsets with respect to their selected guards, and 2) clients are partitioned with respect to the set of destinations they connect to. We observed that sender leakage decreased for *all path selection* algorithms as *user destinations increased* due to a greater overlap in the destination space across clients as the destination sets increased. More specifically, for Vanilla Tor, sender AS leakage decreased by 26% for the *10-destination* user model compared to the *5-destination* user model, by 14% for the *15-destination* user model compared to the *10-destination* user model, and by 12% for the *20-destination* user model compared to the *15-destination* user model. This highlights the impact that the user model may have on security results. Moreover, we note that CLASI is sensitive to changing user models and should be a useful tool for researchers seeking to gain more understanding of security implications for their proposed path selection algorithms under different user models.

Figure 7 shows sender AS leakage for DeNASA guard selection variations compared to Vanilla Tor. Sender AS leakage was not significantly different for Vanilla Tor compared to *g-select* when two to four suspect ASes were avoided. On the other hand, there was a 14% increase in leakage for when six suspect ASes were avoided compared to when four suspect ASes were avoided. Similarly, there was an 8% increase in leakage for when eight suspect ASes were avoided compared to when six suspect ASes were avoided.

7.4 Entropy Based Metrics

To understand the value of CLASI as an anonymity metric, it is necessary to see the results of other anonymity metrics. In this section, we show results for DeNASA using two anonymity metrics, Gini coefficient [34] and Uniformity degree [11].

In Figure 8 we plot Gini coefficient and uniformity degree for DeNASA exit selection variants compared to Vanilla Tor. The x-axis shows the number of user destinations. The two measures have an inverse relationship. As Gini coefficient grows, anonymity goes down, while as uniformity degree grows, anonymity goes up. We see that both measures show little difference for different values of the threshold τ or the number of user destinations. In contrast, CLASI does show a significant differences in sender location leakage as these parameters vary. This highlights an advantage for CLASI, in that it can be used by researchers to understand the anonymity impact of path selection algorithms under various user models in Tor.

Additionally, there was not a significant change in Gini coefficient for Vanilla compared to DeNASA's exit selection variants. This anomaly highlights a significant disadvantage for Gini coefficient in measuring anonymity for path selection algorithms in Tor. The result is due to the fact that Gini coefficient is a measure of equality of relay selection for all clients in the anonymous communication system taken together. As an extreme example, suppose that all Tor users are split evenly into users from country A and those from country B. Also suppose that all Tor relays are also split into two groups with equal bandwidths. If all users from country A select only relays from the first group and country B users from the second group, then the Gini coefficient will be the same as Vanilla Tor, even though the choice of relays clearly indicates which country the user is in. Thus, Gini coefficient is not suitable for understanding anonymity loss when clients use some bias relevant to their location to select paths.

There was a significant decrease in uniformity degree for DeNASA's exit selection variants compared to Vanilla Tor. However, there was no significant change in Uniformity Degree with respect to changing values of τ for the three DeNASA exit selection variants themselves. On the other hand, CLASI did show a significant differ-

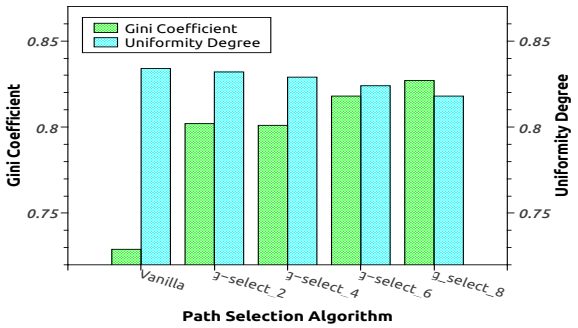


Figure 9: Gini Coefficient and Uniformity Degree for DeNASA guard selection variants compared to Vanilla Tor.

ence in anonymity among the three DeNASA exit selection variants. This shows a disadvantage for uniformity degree in measuring anonymity for path selection algorithms in Tor.

In Figure 9 we plot Gini Coefficient and Uniformity Degree for DeNASA's guard selection variants compared to Vanilla Tor for the *5-destination* user model. The results indicate a loss in anonymity, as Gini Coefficient significantly increased for all three g-select variants compared to Vanilla Tor. However, there was no significant change in Gini coefficient among the three variants of g-select.

There was no significant change in uniformity degree for Vanilla compared to DeNASA's guard selection variants. This shows that uniformity degree did not indicate that there was a *guard placement attack* vulnerability even if clients were configured to avoid up to eight suspect ASes while selecting their guard nodes.

We conclude that gini coefficient and uniformity degree are not sufficient replacements for the CLASI metric when measuring anonymity of path selection algorithms in Tor.

7.5 Time To First Compromise

Time to first compromise is an all-or-nothing measure of how long it takes until a client uses a compromised circuit [24]. Using the TorPS configuration described in Section 7.1, we ran a nine-month simulation for e-select $\tau = 0.1$ and Vanilla. We denoted streams as being *vulnerable* if AS3356 or AS1299 appeared on both sides of the stream. As shown in Figure 10, approximately 60% of Vanilla and e-select clients built at least one vulnerable stream within the first two weeks. After the nine month period, approximately 80% of Vanilla and e-select clients built at least one vulnerable stream. On the other hand, according to Figure 5, DeNASA builds 63% fewer vulnerable streams with respect to AS adversaries compared to Vanilla. Therefore, some DeNASA clients should realize some security improvement compared to Vanilla clients because they build less vulnerable streams, even

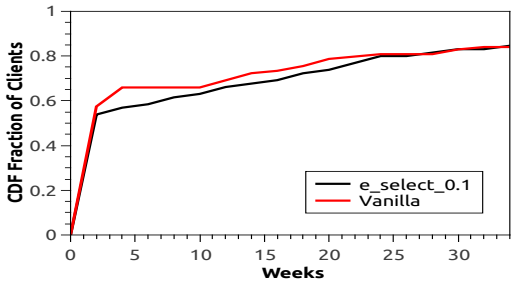


Figure 10: Time to first compromise for DeNASA e-select compared to Vanilla Tor

though the time taken for DeNASA clients to build their first vulnerable stream is similar to Vanilla Tor.

The results indicate that DeNASA e-select $\tau = 0.1$ provides approximately the *same* security against AS-level adversaries with respect to *time to first compromise*, and better security against AS-level adversaries with respect to *vulnerable stream rate*. Conversely, in Figure 6, CLASI shows a security *reduction* in client AS leakage of 11% for e-select $\tau = 0.1$ compared to Vanilla. These contrasting results support our assertion that *time to first compromise* alone is not sufficient in fully understanding the security implications of path selection algorithms. Though *time to first compromise* is an important metric, we point out that other metrics including CLASI should be used when measuring anonymity for path selection algorithms in Tor.

8 PredicTor Security Evaluation

To understand the anonymity level of PredicTor and PredicTor+CAR compared to CAR and Vanilla, we first generated 500,000 paths for each algorithm from the Shadow experiment described in Section 3. Then, we measured anonymity of each path selection algorithm using Gini coefficent, Uniformity degree, and CLASI.

Figure 11 shows Gini coefficient and Uniformity degree for all tested algorithms. The Gini coefficient was 0.21 higher for PredicTor over Vanilla and 0.25 higher for PredicTor+CAR over Vanilla. According to the Gini coefficient metric, Vanilla and CAR had similar anonymity while PredicTor and PredicTor+CAR had significantly worse anonymity. These results suggest that PredicTor clients select some relays with higher probability and avoid other relays, causing an inequality in relay selection compared to Vanilla. In contrast, there was a slight decrease in Uniformity degree for PredicTor and PredicTor+CAR compared to CAR and Vanilla.

In Figure 12, we plot sender AS leakage using CLASI for PredicTor and PredicTor+CAR compared to CAR and Vanilla. We found that PredicTor clients had similar AS leakage compared to Vanilla, likely due to clients choosing paths independently of their location in PredicTor. On the other hand, CAR clients build paths based

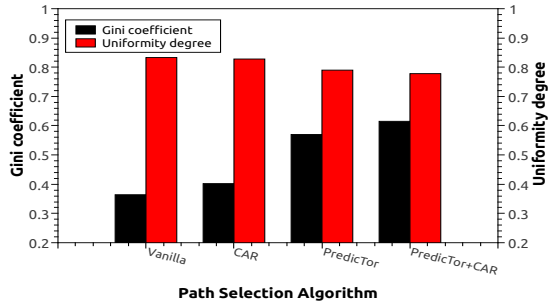


Figure 11: Gini coefficient and Uniformity degree.

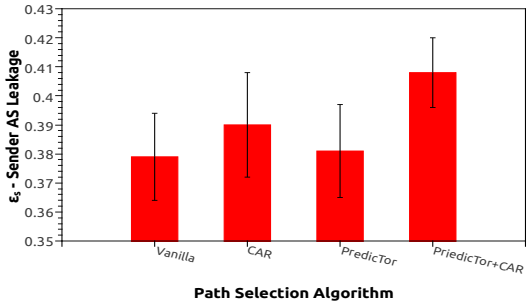


Figure 12: Sender AS leakage

on opportunistic measurement from the client, and thus, their paths should have some relationship to their location. Accordingly, we found a slight increase in AS leakage for CAR and a significant increase of about 9% for PredicTor+CAR compared to Vanilla and PredicTor alone. We note that both Gini coefficient and Uniformity degree did not indicate a significant difference in anonymity for PredicTor+CAR compared to PredicTor.

From our findings, we conclude that AS leakage for PredicTor is similar to Vanilla and slightly better than CAR due to PredicTor clients building paths independently of network location. On the other hand, PredicTor does select certain relays with higher probability causing an inequality in relay selection compared to Vanilla and CAR. However, the entropy loss is minimal, indicating that the distribution of selected relays for all PredicTor clients is similar to Vanilla.

Time to First Compromise Johnson et al. [24] showd that time to first compromise is strongly related to guard selection policy. As PredicTor chooses guards exactly the same as Vanilla, we believe time to first compromise for PredicTor due to relay-level and AS-level adversaries should be similar to Vanilla Tor.

9 Discussion and Future Work

In this section, we discuss deployment ideas for PredicTor and future work possibilities for CLASI.

9.1 PredicTor Deployment

There are two main challenges that would need to be addressed for the successful deployment of PredicTor: 1) clients should routinely receive comprehensive training data, and 2) the training data should be gathered securely, such that an adversary has little chance of directing traffic to malicious relays.

In the Live Tor experiments, we built a training set by measuring download times for approximately 50,000 streams from a centralized authority over the course of one hour. This training set was given to a PredicTor client and used to build circuits during the subsequent hour. Gathering the training data added an additional load on the Network of approximately 4.0 GiB/hr, or just .07 GiB/s. The current bandwidth of the Tor Network is approximately 60 GiB/s. Thus, we believe that the measurement load should not be a problem for deployment.

Similar to the Tor consensus, the training set should be sent to each client once per hour. We believe this should not be problematic because our training set was approximately 233 KB. Adding this information to the consensus would result in a file size increase of only 10%. It may be possible to reduce this further with optimizations.

In a live deployment of PredicTor, the training data can be gathered by one single authority or by multiple authorities. If the data is gathered by one authority, then that authority should be trusted. If the training data is gathered by multiple authorities, then there should be a voting process that is used to resist manipulation from a subset of malicious authorities.

Additionally, PredicTor measurement circuits should be made indistinguishable from regular circuits by randomizing the destination domain and payload size of the measured stream. We note that using RTT measurements for performance is fundamentally insecure because they can easily be manipulated by malicious exit relays.

Model Features. The AS and CC features used within the model are categorical, and thus have no particular order. Neighboring ASes and countries may have quite distinct codes. It may be that reordering the AS and CC numbers to provide some correspondence to network location or geographic location, or just directly using geographic location, could improve the performance of PredicTor. In our system, we assume that there are enough training data points to provide multiple values with exact matches in each given categories most of the time, i.e. some measurements using the same AS or CC. Given that, PredicTor can usually classify a circuit correctly without resorting to data about other ASes and countries which might have questionable relevance to the circuit being considered.

9.2 CLASI

Adversary Model. The adversary model within the CLASI challenge-response game is an all-knowing adversary. Therefore, our results yield an upper bound. Meiser et al. [5] showed that a budget adversary model results in a tight upper bound. The CLASI adversary model can be modified such that the adversary's knowledge is bounded by their budget. The budget can be defined in terms of cost or bandwidth, for example.

Sender/Receiver Anonymity. CLASI is designed to measure AS leakage from the sender. However, the classification model can be modified to also measure AS leakage from the receiver. This could give researchers even more insight into anonymity implications, especially for *destination-aware* path selection algorithms that use destination information to build circuits [35, 14].

10 Conclusion

To address Tor performance, we presented *PredicTor*, a path selection technique that uses a Random Forest classifier trained on a set of recent Tor paths to predict the performance of a proposed path. We implemented PredicTor in the Tor source code and showed through simulations in Shadow that PredicTor improved Tor network performance by 23% compared to Vanilla Tor and by 13% compared to Congestion-Aware Routing. In our live Tor experiments, during times of high congestion, PredicTor had an improvement of 7% to 13% in the median case compared to Vanilla Tor. We evaluated the anonymity of PredicTor using standard entropy-based metrics, and we proposed a new anonymity metric called *CLASI*: Client Autonomous System Inference. Our results indicated that CLASI showed anonymity loss for location-aware path selection algorithms where other entropy based metrics showed little to no loss of anonymity. Additionally, CLASI indicated that PredicTor had similar client AS leakage compared to Vanilla due to PredicTor building circuits that are independent of client location.

11 Acknowledgements

We would like to thank Roger Dingledine and Rob Jansen for insightful discussions about testing Tor performance. Additionally, we thank Sebastian Meiser for insightful discussions about anonymity metrics for Tor. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1423163 as well as Rochester Institute of Technology under a Signature Interdisciplinary Research Areas grant.

References

[1] AKHOONDI, M., YU, C., AND MADHYASTHA, H. V. LASTor: A low-latency AS-aware Tor client. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P'12)* (2012).

[2] ALEXA.COM. Alexa top sites., 2017. http://www.alexa.com/topsites.

[3] ANNESSI, R., AND SCHMIEDECKER, M. Navigator: Finding Faster Paths to Anonymity. In *IEEE European Symposium on Security and Privacy (EuroS&P' 16)* (2016).

[4] BACKES, M., KATE, A., MANOHARAN, P., MEISER, S., AND MOHAMMADI, E. AnoA: A Framework For Fnalyzing Anonymous Communication Protocols. In *Proceedings of the 26th IEEE Computer Security Foundations Symposium Computer (CSF'13)* (2013).

[5] BACKES, M., MEISER, S., AND SLOWIK, M. Your Choice MATor (s). In *Proceedings on Privacy Enhancing Technologies (PETS'16)*.

[6] BARTON, A., AND WRIGHT, M. Denasa: Destination-naive AS-Awareness in Anonymous Communications. In *Proceedings on Privacy Enhancing Technologies (PETS'16)*.

[7] BORISOV, N., DANEZIS, G., MITTAL, P., AND TABRIZ, P. Denial of Service or Denial of Security? In *Proceedings of the 14th ACM conference on Computer and communications security (CCS'07)* (2007).

[8] BREIMAN, L. Random Forests. vol. Vol 45, Springer.

[9] CAIDA. Caida as ranking. http://as-rank.caida.org/.

[10] CROVELLA, M. E., TAQQU, M. S., AND BESTAVROS, A. Heavy-tailed probability distributions in the world wide web. *A practical guide to heavy tails 1* (1998), 3–26.

[11] DIAZ, C., SEYS, S., CLAESSENS, J., AND PRENEEL, B. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies (PET'02)* (2002).

[12] DINGLEDINE, R., HOPPER, N., KADIANAKIS, G., AND MATHEWSON, N. One Fast Guard For Life (or 9 Months). In *Proceedings of 7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs'14)* (2014).

[13] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium (USENIX Security'04)* (2004).

[14] EDMAN, M., AND SYVERSON, P. AS-Awareness in Tor Path Selection. In *Proceedings of the 16th ACM conference on Computer and communications security (CCS'09)* (2009).

[15] GEDDES, J., JANSEN, R., AND HOPPER, N. How Low Can You Go: Balancing Performance with Anonymity in Tor. In *International Symposium on Privacy Enhancing Technologies Symposium (PETS' 13)* (2013).

[16] GEDDES, J., SCHLIEP, M., AND HOPPER, N. ABRA CADABRA: Magically Increasing Network Utilization in Tor by Avoiding Bottlenecks. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society (WPES'16)* (2016).

[17] IMANI, M., BARTON, A., AND WRIGHT, M. Guard Sets in Tor using AS Relationships. In *International Symposium on Privacy Enhancing Technologies Symposium (PETS' 18)* (2018).

[18] JANSEN, R., BAUER, K. S., HOPPER, N., AND DINGLEDINE, R. Methodically Modeling the Tor Network. In *Proceedings of the 5th USENIX Conference on Cyber Security Experimentation and Test (CSET'12)* (2012).

[19] JANSEN, R., GEDDES, J., WACEK, C., SHERR, M., AND SYVERSON, P. Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security'14)*.

[20] JANSEN, R., AND HOPPER, N. Shadow: Running Tor in a Box For Accurate And Efficient Experimentation. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'12)*.

[21] JANSEN, R., AND JOHNSON, A. Safely Measuring Tor. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS'16)* (2016).

[22] JANSEN, R., AND JOHNSON, A. Safely measuring tor. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM, pp. 1553–1567.

[23] JOHNSON, A., JANSEN, R., JAGGARD, A. D., FEIGENBAUM, J., AND SYVERSON, P. Avoiding The Man on the Wire: Improving Tor's Security with Trust-Aware Path Selection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'17)* (2017).

[24] JOHNSON, A., WACEK, C., JANSEN, R., SHERR, M., AND SYVERSON, P. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the 20th ACM SIGSAC conference on Computer & communications security (CCS'13)* (2013).

[25] JUEN, J. Protecting Anonymity in the Presence of Autonomous System and Internet Exchange Level Adversaries.

[26] METRICS, T. Tor metrics, June 2015. https://metrics.torproject.org.

[27] MITCHELL, T. *Machine Learning*. McGraw-Hill, 1997.

[28] MURDOCH, S. J., AND WATSON, R. N. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS'08)*.

[29] OVERLIER, L., AND SYVERSON, P. Locating Hidden Servers. In *Proceedings of the 27th IEEE Symposium on Security and Privacy (S&P'06)*.

[30] ROCHET, F., AND PEREIRA, O. Waterfiling: Balancing the Tor Network with Maximum Diversity. In *Proceedings on Privacy Enhancing Technologies (PETS'17)*.

[31] SERJANTOV, A., AND DANEZIS, G. Towards an Information Theoretic Metric for Anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (2002)*.

[32] SHANNON, C. E. A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review 5*, 1 (2001), 3–55.

[33] SHERR, M., BLAZE, M., AND LOO, B. T. Scalable Link-Based Relay Selection for Anonymous Routing. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies (PETS'09)* (2009).

[34] SNADER, R., AND BORISOV, N. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of the 16th Annual Network & Distributed System Security Symposium (NDSS'08)* (2008).

[35] STAROV, O., NITHYANAND, R., ZAIR, A., GILL, P., AND SCHAPIRA, M. Measuring and mitigating AS-level adversaries against Tor. In *Proceedings of the 24th Annual Network & Distributed System Security Symposium (NDSS'16)*.

[36] SUN, Y., EDMUNDSON, A., FEAMSTER, N., CHIANG, M., AND MITTAL, P. Counter-RAPTOR: Safeguarding Tor Against Active Routing Attacks. In *Proceedings of the 38th IEEE Symposium on Security and Privacy (S&P'17)* (2017).

[37] SUN, Y., EDMUNDSON, A., VANBEVER, L., LI, O., REXFORD, J., CHIANG, M., AND MITTAL, P. RAPTOR: Routing Attacks on Privacy in Tor. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security'15)* (2015).

[38] SYVERSON, P. Why I'm not an entropist. In *International Workshop on Security Protocols* (2009), Springer, pp. 213–230.

[39] TORPS. TorPS: The Tor path simulator., 2013. http://torps.github.io.

[40] VINCENTY, T. Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. In *Survey Review* (1975).

[41] WACEK, C., TAN, H., BAUER, K. S., AND SHERR, M. An Empirical Evaluation of Relay Selection in Tor. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS'13)* (2013).

[42] WANG, T., BAUER, K., FORERO, C., AND GOLDBERG, I. Congestion-Aware Path Selection for Tor. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security (FC'12)* (2012).

[43] WRIGHT, M., ADLER, M., LEVINE, B. N., AND SHIELDS, C. Defending Anonymous Communications Against Passive Logging Attacks. In *Proceedings of the 24th IEEE Symposium on Security and Privacy (S&P'03)* (2003).

[44] WRIGHT, M. K., ADLER, M., LEVINE, B. N., AND SHIELDS, C. Passive-Logging Attacks Against Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC) 11*, 2 (2008), 3.

Appendices

A Client Location and Guard Diversity

In Table 1, we show the median and 90th percentile performance and circuit distance improvements for PredicTor compared to Vanilla.

Date & Time	CC	Guard	Cong.	Time		Distance	
				Median	90th per.	Median	90th per.
2017-07-18 00:00	US	Fast	Low	4.2%	15.6%	52.0%	16.5%
2017-07-19 14:00	US	Fast	High	9.7%	17.7%	23.0%	10.8%
2017-09-01 00:00	US	Slow	Low	4.4%	6.3%	2.2%	−1.2%
2017-08-31 14:00	US	Slow	High	6.7%	19.1%	−1.0%	−5.2%
2017-08-15 00:00	DE	Fast	Low	7.3%	23.1%	29.4%	16.2%
2017-08-03 14:00	DE	Fast	High	12.8%	25.3%	26.2%	21.8%
2017-08-22 00:00	DE	Slow	Low	3.3%	2.6%	2.0%	5.9%
2017-08-17 14:00	DE	Slow	High	6.9%	16.9%	0.0%	0.0%
2017-08-30 00:00	JP	Fast	Low	7.4%	11.2%	28.8%	18.0%
2017-08-29 14:00	JP	Fast	High	6.3%	10.8%	11.3%	10.9%
2017-08-24 00:00	JP	Slow	Low	7.2%	4.5%	1.8%	0.0%
2017-08-25 14:00	JP	Slow	High	7.3%	14.0%	2.1%	4.1%

Table 1: Improvements in download time and circuit distance for PredicTor compared to Vanilla.