Lauren Watson*, Anupam Mediratta, Tariq Elahi, and Rik Sarkar

Privacy Preserving Detection of Path Bias Attacks in Tor

Abstract: Anonymous communication networks like Tor are vulnerable to attackers that control entry and exit nodes. Such attackers can compromise the essential anonymity and privacy properties of the network. In this paper, we consider the path bias attack– where the attacker induces a client to use compromised nodes and thus links the client to their destination. We describe an efficient scheme that detects such attacks in Tor by collecting routing telemetry data from nodes in the network. The data collection is differentially private and thus does not reveal behaviour of individual users even to nodes within the network. We show provable bounds for the sample complexity of the scheme and describe methods to make it resilient to introduction of false data by the attacker to subvert the detection process. Simulations based on real configurations of the Tor network show that the method works accurately in practice.

Keywords: differential privacy, outlier detection, privacy preserving analysis

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

Anonymity networks are designed to prevent a user's actions on the Internet from being visible to adversaries. Several networks for anonymity have been proposed such as Loopix, Babel and JAP [6, 15, 30] with Tor [11] being the most popular. These systems operate by creating an overlay network on the Internet where the route taken by a user's traffic is randomized to dissociate the origin of the traffic from its destination.

In systems such as Tor, certain nodes (relays) are marked (using flags) for traffic ingress to the overlay, and others for traffic egress from the overlay to the intended destination. In Tor, these are called *guards* and *exits* respectively.

In these systems, an adversary with control of one or more of these perimeter nodes is more likely to be successful in a de-anonymization attack [21]. A particular form of attack on Tor and similar networks is suggested by Borisov *et al.* [7] where the attacker induces user traffic to take certain routes. For example, an attacker controlling an entry node may be able to force a client's traffic to use an exit also controlled by the attacker, and thus discover their activity. We call this a *path bias attack*.

Monitoring the network to detect attacks is challenging due to the distributed nature of the Tor overlay network, and its probabilistic routing. Further, in practice, all network nodes or clients may not trust the monitoring authority to protect sensitive information. Therefore, network monitoring and attack detection has to be performed while preserving the privacy of individuals. Previous works on measurements on Tor [14, 16, 24] have focused on characterizing the behaviors of clients using the network. However, an attack or anomaly originating within the network infrastructure requires analysis of the behaviour of the network itself.

Our Contributions. In this paper, we propose a scheme to detect path bias (PB) attacks and similar anomalous behaviors in Tor. Our approach is to make use of the *middle nodes*—nodes without the guard or exit flags—to study the behavior of the network and see if the traffic pattern shows deviation suggestive of an attack. This deviation can be with respect to the configuration of the network (as recorded in the *consensus document* available in Tor), or with respect to expected behavior based on historical data.

In this method, an aggregator collects data from the distributed middle nodes to measure the behavior of traffic passing from each entry node to different exit nodes. We describe *PB attacks* in Section 4. In this attack, an attacker with control of one or more guard and exit nodes rejects circuits through its guard node that do not pass through a compromised exit, thus inducing a larger than expected volume of circuits to pass through compromised exits, making them sus-

***Corresponding Author: Lauren Watson:** U. of Edinburgh, E-mail: lauren.watson@ed.ac.uk
Anupam Mediratta: U. of Edinburgh, E-mail: a.a.mediratta@sms.ed.ac.uk
Tariq Elahi: U. of Edinburgh, E-mail: t.elahi@ed.ac.uk
Rik Sarkar: U. of Edinburgh, E-mail: rsarkar@inf.ed.ac.uk

ceptible to de-anonymization. Our main result shows that PB attacks diverting significant amounts of traffic can be detected—quickly and practically, with privacy guarantees—from small amounts of data. In Section 5 we describe the basic statistical test for such detection.

To protect individual users, our system uses differential privacy in reporting data. Middle nodes add noise to data before transmitting it to the aggregator, which ensures differential privacy irrespective of the subsequent use of the data or the security of communication channels. To prevent excessive noise in the system, a secret sharing mechanism is used between middle nodes, reducing the noise requirement, but also ensuring that unprotected data is never transmitted.

The variable bandwidths of the nodes in the network pose an additional challenge in privacy protection. The bandwidths determine the probability of selection of nodes in random routes. Therefore, very low bandwidth nodes handle few routes, which reduces reliability of statistical tests, particularly in the presence of differentially private reporting. To preserve test reliability, we introduce a binning technique, where nodes are processed in groups. This approach makes the algorithm resilient to low data volumes and noise due to differential privacy, but allows the attack to still be detected efficiently. In a real system, an adversary can also control some middle nodes and attempt to subvert the detection process by introducing false data. We describe sampling and voting mechanisms to mitigate such attempts. These enhancements are discussed in Section 6.

Our experiments (Section 7) rely on two forms of data collected from the Tor system. First, we used a Tor consensus document to generate random routes within the network which represents an *expected traffic profile*. Second, from Tor clients operated by us, we made a sequence of connection establishment requests to the Tor network and stored the successful routes – giving us an empirical traffic profile.

Experimental results on these datasets demonstrate the performance of the outlier detection algorithm for different types of attackers and show that outlier detection is accurate, obtaining high F1 scores in most scenarios, even with strong privacy requirements ($\epsilon = 0.1$) and using only half a day of circuit data.

Motivations. While we use Tor as our prototype system, the use of randomized paths, attacks on them, and statistical detection mechanisms are of wider interest. Mix networks operate using multi-hop randomized routes and are subject to similar path manipulation attacks. Random walk based anonymity measures have been proposed in sensor networks, and also have the risk of leaking information [32]. In IoT systems, compromised devices can send information to adversarial servers, which may be detected by statistical deviations in traffic distribution. Thus we expect our detection techniques to be generally useful against attacks on anonymity systems.

2 Background

This section presents a brief background on Tor, secret sharing and differential privacy, along with related works in Tor measurement studies.

2.1 Anonymous Communication Networks and Tor

Anonymous communication networks depend on a route-randomizing mechanism. Given a message or connection c between source s and destination t, a random route or circuit c is constructed so the pair (s, t) cannot be identified with high confidence by an adversary.

The Tor network routes a connection along a path consisting of three relays before reaching its destination. The route is selected at the source, that is, by the client. Along this route, nested encryption ensures that the first (entry) relay only knows the source of the connection and the next hop in the route. Relays with high uptime, stability, and bandwidth are assigned the *guard flag*. Only these guard relays may be used for entry into the Tor network.

Let us denote the set of guard, middle and exit relays as G, M and X respectively. The routes chosen by the Tor routing mechanism are of the form (g, m, x) where $g \in G, m \in M, x \in X$.

The client selects a relay to be on the route of c with probability proportional to its relative bandwidth within its type. These bandwidths and probabilities are public knowledge in the *Tor consensus document* published every hour. An analysis of consensus documents shows bandwidth distribution to be exponential, with few large relays and many relays that have very small bandwidth, and consequently a small probability of being chosen in a path. Indeed, hundreds of possible exit relays have probability 10^{-6} or smaller of being chosen.

2.2 Additive Secret Sharing

Additive secret sharing can be used by a group of entities to collaboratively and securely compute the sum of private inputs. We use this method for relays to share the counts of circuits they observe. .

Suppose each relay M maintains a count C of the number of circuits it sees. The relays wish to compute the sum of these counters without revealing individual values. For initialization, M randomly picks a secret value x_s modulo a large prime Q, and sets $C = x_s$. It then splits x_s in to N secret shares by creating $N-1$ random values and setting $x_N = x_s - (x_1 + x_2 + \cdots + x_{N-1}) \mod Q$. M then transmits the shares $x_1, \ldots, x_N$ to N designated *share keepers* for future use. Each share keeper adds the shares it receives from the relays together, modulo Q. During system operation, as each observation is recorded, M increments the count as $C = (C + 1) \mod Q$.

When computing the sum over all nodes, we proceed as follows: Each M and share keeper sends its value to the aggregation server. Once all values have arrived, the aggregator produces the aggregate as follows: It adds up all of the values it receives from the relays to get R, and separately adds up values it receives from share keepers to get S, it then computes $(R + S) \mod Q$ to find the sum of all observations across all M relays.

A compromise of up to $N - 1$ share keepers will not reveal any information about the secret x_s or any of the intermediate sums to a dishonest relay, share keeper, or aggregator. Like previous works, we assume that three share keepers is reasonable for the settings we consider.

2.3 Differential Privacy

To prevent the data or statistics published by the aggregator from revealing sensitive information, we use differential privacy, which is the current gold standard definition of statistical privacy.

Differential privacy operates by considering for each input dataset D, similar *neighboring* datasets:

Definition 1 (Neighbouring Databases). *Two databases D and D' are said to be* neighbouring *if they differ in exactly one row.*

Under this definition, differential privacy of a randomized algorithm requires that the probabilities of an output are similar for any two neighboring datasets:

Definition 2 (Differential Privacy). *A randomized algorithm K operating on the database satisfies ϵ-differential privacy if given any two neighbouring databases D and D' and a set of outputs $S \subseteq Range(K)$ then:*

$$P[K(D) \in S] \leq e^{\epsilon} P[K(D') \in S] \tag{1}$$

Smaller values of ϵ imply stronger privacy.

Usually, differential privacy is achieved by adding noise to variables in the database, or by adding noise to the final aggregated result.

2.4 Related Work

2.4.1 Privacy and Network Measurement

Monitoring anonymity networks while preserving privacy for users is challenging [25]. The statistics available at the Tor Metrics project [22] are indirect, limited in scope and many are based on assumptions about network usage. Early attempts at in-depth analysis [25], were later considered to be risky for privacy [35].

In response, Elahi *et al.* [14] proposed PrivEx, applying differential privacy to privately collect and release statistics for traffic between exit relays and several destination websites. They used a combination of secure multi-party computation—or distributed decryption as an alternative—and differential privacy.

PrivEx was later extended by Jansen *et al.* [16] into the PrivCount mechanism, allowing for more statistics to be privately collected (e.g. the number of unique clients). Mani *et al.* [23] and Wails *et al.* [37] extended the methods to be more robust to adversarial manipulation of the statistics gathered, which was further extended including a large measurement study in the follow-on work by Mani *et al.* [24]. These methods focus on collecting data on client usage of the system.

Our algorithm uses a similar setup, but for a different objective of analyzing statistics of traffic within the Tor network by collecting statistics from middle relays, and leads to different privacy and utility guarantees.

2.4.2 Attacks and Measurements on Tor

There is significant evidence that adversaries who can observe traffic at an entry and exit relay can re-identify clients in low-latency mix networks such as Tor [27, 29, 33, 38]. An attack was described by Borisov *et al.* [7] which effectively denies service for routes which cannot be compromised, therefore increasing the chance of a client using a route which is vulnerable to attack. Singh *et al.* [34] describe eclipse attacks for overlay networks. Sun *et al.* [36] studied a new form of routing attacks for anonymous communication networks based on Border Gateway Protocol (BGP) hijacking. Nithyanand *et al.* [28] provide estimates of the potential widespread nature of attacks compromising entry and exit relays, finding that up to 40% of circuits created by the existing Tor client were at risk.

2.4.3 Path Selection and Network Design

Several works [1, 3, 13, 17, 28] discuss AS-aware path selection methods, offering resistance against traffic analysis and correlation attacks. We propose a detection scheme which can be implemented in parallel to these methods. Recently, Leibowitz *et al.* [21] proposed a mix-network design that detects and mitigates the effect of malicious mixes in higher-latency networks.

3 Threat Model

The adversary's goal is to opportunistically compromise the privacy of as many clients as possible by compromising their circuits, and to simultaneously avoid detection.

Adversaries may be passive or active. A *passive* adversary controls one or many guard and exit relays and records the source and destination of circuits that pass through both compromised guards and exits, but does not attempt to manipulate circuits in any way. Such adversaries are hard to detect, but are also less of a threat.

Our main concern is an *active* adversary, who tries to modify the flow of circuits to capture more victims than is possible passively. In Tor, an adversary can take the following actions:

1. Drop and/or delay communication packets, including Tor circuit establishment requests.
2. Control some fixed fraction of the total Tor network bandwidth, by operating or controlling a fraction of the guard and/or exit relays.

Let BW_G and BW_X be the total guard and exit bandwidths in the Tor network, and let $BW_{\hat{g}}$ and $BW_{\hat{x}}$ be the adversarial guard and exit bandwidths, then the adversarial bandwidth fractions for guards and exits are $F_{\hat{g}} = \frac{BW_{\hat{g}}}{BW_G}$ and $F_{\hat{x}} = \frac{BW_{\hat{x}}}{BW_X}$ respectively.

Since clients choose relays on their circuit proportional to the relay bandwidths, the fraction of circuits controlled by the adversary is determined by their bandwidth. We assume that the cost of fielding this bandwidth is proportional to its size. Therefore, compromising or deploying nodes with high bandwidth is correspondingly more expensive.

The adversary is assumed to have full knowledge of our detection scheme and may strategically:

1. Redistribute their available bandwidth between relays, (Section 4.1).
2. Attack selectively to avoid detection by our scheme, at the cost of lower compromised circuit yield, (see Sections 5 and 6.3).
3. Control a limited fraction of data collection points (middle relays) and may manipulate the inputs they provide to thwart attack detection, (Section 6.3.1).

There is a trade-off between risk tolerance and success probability for the adversary. A larger attack incurs a greater risk of detection. We assume that the adversary does not alter these priorities during an attack, in this sense he is *non-mobile.*

There are two types of adversarial action in question. The *PB* adversary executes the PB attack and hopes to observe as much of client traffic as possible.

A *non-PB* adversary is an external observer that attempts to use the statistical reports generated by our scheme to infer client behaviour, possibly by cross analyzing it against other information.

4 The Path Bias Attack

Our scheme is designed to detect the PB attack, and is resistant to an active, strategic, non-mobile adversary who has knowledge of our algorithm and aims to avoid detection while carrying out the PB attack.

In a PB attack, the adversary's strategy is to subvert the randomized mechanism and force a circuit c to follow a route in the network with the adversary's compromised end-points. The active PB attack therefore allows the adversary to increase the number of compromised circuits in order to gain additional circuits in addition to those possible through the passive attack.

Suppose the adversary operates a specific guard relay $\hat{g}$ and an exit relay $\hat{x}$. While the adversary cannot actively dictate the route for a connection c to go through $\hat{x}$, they are free to selectively drop at $\hat{g}$ any connection that does not subsequently pass through $\hat{x}$ [7]. The adversary can use flow tagging, or watermarking, to mark a flow with a sequence of packets (that do not interfere with the client's traffic) that are detectable at their controlled relays [31].

The attacker thus proceeds to perform path bias as follows. For any connection c going out of $\hat{g}$, the attacker observes if the traffic watermark matches one of those arriving at $\hat{x}$. If not, $\hat{g}$ reports a failure of connection to s_c forcing the client to retry. The attacker can repeat this process until s selects $(\hat{g}, m, \hat{x})$ with an arbitrary middle node m as the route for c. Once a circuit is successfully diverted, the adversary may observe all the traffic that flows through that circuit, recording the domains, IP addresses, and any other traffic fingerprints.

The Tor network designers are aware of this attack and have implemented a mitigation: each client keeps a local circuit failure rate of its primary guard(s). If the rate is too high, the client marks the guard as potentially malicious. This mitigation is *not* turned on by default due to its unknown efficacy. Even if this mitigation was in use, its local nature allows a malicious guard to spread the attack across all clients that connect through it, albeit not as aggressively to avoid being flagged.

Our scheme addresses this by taking a global view from which to detect path biasing attempts by guards and attacks spread across a large population.

4.1 Effectiveness of PB attack

Let n be the total circuits created in the Tor network in an epoch of length t. An adversary can then passively (i.e. by obeying the protocol and not launching a PB attack) expect to observe $E_{\hat{g}\hat{x}} = n(F_{\hat{g}} \cdot F_{\hat{x}})$ circuits.

We now evaluate how effective the PB attack can be in the Tor network and how the adversary may optimize it. Using the PB attack the adversary can expect to compromise up to

$$E_{\hat{g}\hat{x},r} = (n \cdot F_{\hat{g}})(1 - (1 - F_{\hat{x}})^{r+1}) \qquad (2)$$

circuits, where r is the number of path bias attempts on any one circuit observed at the adversarial guard relay, $n \cdot F_{\hat{g}}$, in the epoch t. Note that $r = 0$ is the special case above of the passive attack (no PB attack).

We observe two facts: 1) the adversary may compromise at most $n \cdot F_{\hat{g}}$ circuits, and 2) as r grows the adversary's yield approaches $n \cdot F_{\hat{g}}$. Therefore, if r is allowed to be very large (infinite and ignoring t for the moment), the adversary may compromise up to $n \cdot (F_{\hat{g}} + F_{\hat{x}})$ by redistributing $F_{\hat{x}}$ into $F_{\hat{g}}$.[1]

Figure 1 shows that path biasing can allow an adversary with 0.2% and 0.02% adversarial bandwidth to compromise at most $4e5$ and $4e4$ circuits respectively (the green and blue horizontal lines). The orange and red curves show the compromised circuits as r is allowed to increase. Note the long tails indicating that large r is necessary to achieve those maximum compromise levels.

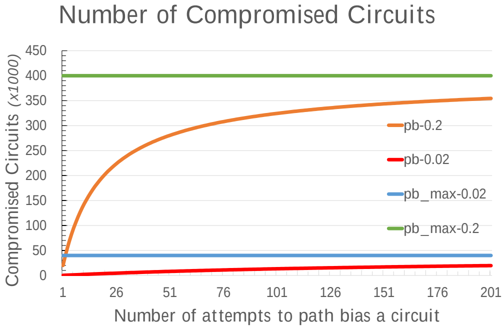


Fig. 1. The number of circuits an adversary may compromise when conducting a PB attack. The total number of circuits $n = 2e6$, with two adversaries, i) $F_{\hat{g}} + F_{\hat{x}} = 0.2$ and ii) $F_{\hat{g}} + F_{\hat{x}} = 0.02$ strategically distributed such that it maximizes the yield for a given number of PB attempts, r.

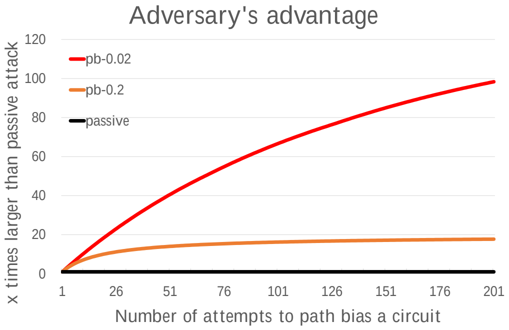


Fig. 2. The adversary's gain when conducting a PB attack as compared to the passive attack. The total number of circuits $n = 2e6$, with two adversaries, i) $F_{\hat{g}} + F_{\hat{x}} = 0.2$ and ii) $F_{\hat{g}} + F_{\hat{x}} = 0.02$ strategically distributed such that it maximizes the yield for a given number of PB attempts, r.

[1] For simplicity we assume here that the guard and exit pools are of similar size and that $F_{\hat{g}}$ and $F_{\hat{x}}$ can be exchanged at parity. However, we can generalize the result above with an appropriate exchange factor ($z = \frac{BW_X}{BW_G}$) giving the total compromisable circuits as $n \cdot F_{\hat{g}}(1 + z)$.

Figure 2, which captures the adversary's relative advantage, shows that a lower BW adversary gets relatively more (200-fold, top red line) reward for their bandwidth investment than the larger BW adversary (20-fold, middle orange line). However, note that this takes many more attempts (a larger r) than for the larger BW adversary. Specifically, we see from Fig. 2 that the larger adversary (middle orange line) may compromise up to 50%[2] of all the circuits that use its guard relay with just $r = 20$ PB attack attempts per circuit while the smaller adversary (top red line) requires more than $r = 201$ PB attack attempts per circuit to achieve the same compromise level. From the perspective of a PB attack victim, the larger adversary still poses a bigger threat than a relatively smaller adversary.

In practical terms, if we assume that the PB attack itself takes no time at all[3], the circuit construction phase itself still takes non-negligible time. Tor circuit build times vary across the network.[4] Assuming a typical figure of $250\,ms$ the smaller adversary would require up to $\approx 50\,s$ per circuit to compromise 50% of all circuits that use it, compared to the larger adversary who requires up to $\approx 5\,s$ per circuit to capture the same 50%. Therefore, in reality a large r may not be possible within the period t, either because 1) the adversary may not be able to wait due to operational costs or 2) the client may notice large delays in circuit construction and this could cause them to rotate away from the malicious guard, $\hat{g}$, resulting in loss of potential circuits to compromise. The strategic adversary will factor in the above and chose the appropriate r to calculate the optimal $F_{\hat{g}} : F_{\hat{x}}$ ratio before deploying their relays.

Practical limitations in attack strategies. In practice, in a Tor-like system, an adversary cannot gain significant real advantage by using very low bandwidth exits, which are selected with probability 10^{-6} or lower. For example, this requires thousands of rejections by the malicious guard before a client selects a low bandwidth exit. While Tor does not have a clear guideline on how many rejections marks a guard as defective, we can safely assume that a practical client is unlikely to accept this level of failure before changing guards.

On the other hand, the adversary may choose to run many exits of low bandwidth, hoping to increase the chances, but this will require deployment of hundreds or more of new low bandwidth exits to have an impact, which will be easily detected by Tor authorities as suspicious behaviour.

5 PB Detection Scheme

Overview. Our proposal allows the defender to detect when the observed number of circuits through a particular guard and exit pair exceeds the expected number of circuits.

Formally, we operate with a sample dataset S, and write $n = |S|$. The system is assumed to operate in epochs, where each epoch produces such a dataset used to detect presence of attacks. For any node a, we write S_a and n_a for the set and number of circuits passing through it. For any set of nodes $a, b, c, \ldots$ we correspondingly write $S_{a,b,c,\ldots}$ and $n_{a,b,c,\ldots}$ for set and number of circuits common to all of them. The variable $E_{a,b,c,\ldots}$ denotes the number of circuits through a, b, c we expect to see in sample S based on prior knowledge (such as probabilities for each node obtained from the Tor consensus document).

Quantitatively, we wish to ensure that the number of circuits through any pair gx of nodes does not significantly exceed E_{gx}—the expected number of paths through g and x given the sample size n. Or, in general, to ensure that the number of circuits n_x through exit x does not significantly exceed E_x for any guard g.

For this, we assume that the adversary, by employing the PB attack, intends to achieve $n_{\hat{g}\hat{x}} > E_{\hat{g}\hat{x}} + (\phi E_{\hat{g}\hat{x}} + \lambda)$. The $(\phi E_{\hat{g}\hat{x}} + \lambda)$ is the additional number of victims that the adversary may gain by increasing traffic through $\hat{g}\hat{x}$. The $\phi E_{\hat{g}\hat{x}}$ term implies at least a constant fraction increase over the base rate $E_{\hat{g}\hat{x}}$, and the additive constant λ is to ensure that if $E_{\hat{g}\hat{x}}$ is small, i.e. the number of circuits through $\hat{g}\hat{x}$ is very small, the adversary still gains some return for the attack.

On the other hand, the defender wishes to ensure that $E[n_{gx}] \leq E_{gx} + (\phi E_{gx} + \lambda)$ for every guard-exit pair, and flags an attack if this condition is suspected to be violated. The objective is to perform this test with a given confidence $1 - \beta$, based on a data sample which is subject to differential privacy.

[2] Similar results are obtained for other compromise proportions. See A.4 for a summary table of results.

[3] This is realistic since Rochet and Pereira's *dropmarking* confirmation attack can be performed in the interval after circuit construction and before any client traffic flows. [31]

[4] Circuit build time statistics are reported on the Tor Metrics portal here: https://metrics.torproject.org/onionperf-buildtimes.html.

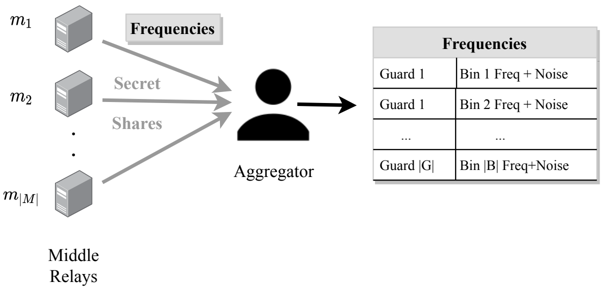
5.1 Basic Algorithm

In this section we will describe our basic approach. For simplicity of exposition, we assume that the the adversary controls a single guard and single exit node. More general strategic adversaries are discussed in Section 6.

Basic algorithm. The scheme works in epochs. Each epoch runs as follows:

1. At the start of the epoch, middle relays initialize differential privacy noise and secret shares.
2. The middle relays count the number of times a guard-exit pair (g, x) occurs among circuits passing through it.
3. They send this data to the aggregator using secret sharing to ensure security of the data.
4. The aggregator adds together the corresponding counts from all middle relays, to get the total number of times the pair $(g.x)$ occurred in the round.
5. The aggregator performs a statistical test to determine if the data shows evidence of attacks.

This process can be repeated for every exit for each guard; so that in the end, for each guard, the aggregator has a histogram with counts over all exits and can perform the test for each. This model is shown in Figure 3



Frequencies	
Guard 1	Bin 1 Freq + Noise
Guard 1	Bin 2 Freq + Noise
...	...
Guard |G|	Bin |B| Freq+Noise

Fig. 3. Output for the algorithm: a table of noisy frequency estimates for guard-exit pairs.

For ease of explanation, we first describe the statistical test, then we will describe the secret sharing and noise addition mechanisms.

5.1.1 Statistical Test at the Aggregator

Suppose that the whole dataset is a collection of n circuits, and n_x is the total number of circuits through an exit x that has been reported to the aggregator by middle relays.

Let us write E_x as the expected number of circuits through x given n circuits overall. Under a significant attack with parameters (ϕ, λ), suppose that the expected number of circuits becomes $\hat{E_x}$. Corresponding distributions are shown in Figure 4. When there is no attack, the observation is drawn from the green distribution centered at E_x, while during an active attack, the observation is drawn from the red distribution centered at $\hat{E_x}$. The statistical test checks n_x against the mid point $(E_x + \hat{E_x})/2$, and returns:

- **No Attack:** If $n \leq (E_x + \hat{E_x})/2$
- **Attack:** If $n > (E_x + \hat{E_x})/2$

In our setup, an adversary is looking for $\hat{E_x}$ to be at least $\phi E_x + \lambda$ larger than E_x, thus the threshold becomes $E_x + \frac{\phi E_x + \lambda}{2}$.

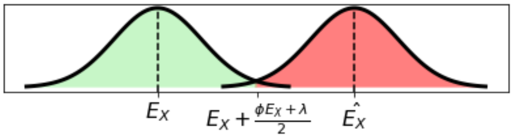


Fig. 4. Observed frequency distributions for exit x with no attack (centered around E_x) and with an attack defined by parameters ϕ and λ (centered around $\hat{E_x}$).

As more circuits are observed and the dataset size grows, the distributions become more concentrated around their respective means, thus reducing the chance of false positives or false negatives. We formalize this idea in the following lemma. For simplicity of notation, we use n to be the relevant data size, n_x to be the number of circuits through an exit x, and p to be the probability of any single circuit passing through x. Thus $E_x = np$ is the expected number of circuits through x.

Lemma 3. *A sample size of*

$$n \geq 12 \ln(\frac{1}{\beta}) \cdot \frac{1}{p} \cdot \frac{1}{(\min\{1, (\phi + \frac{\lambda}{E_x})\})^2}$$

suffices to ensure that $n_x \leq E_x + (\phi E_x + \lambda)/2$ *with probability at least* $(1 - \beta)$.

Proof. Here, n_x is a sum of random variables, each of which is true with a probability p. Thus, we can apply Chernoff bound [26], where $0 \leq \left(\phi + \frac{\lambda}{E_x}\right) \leq 1$, to obtain:

$$P[n_x > (1 + \phi/2 + \frac{\lambda}{2E_x})E_x] \leq e^{-\frac{E_x(\phi/2 + \lambda/2E_x)^2}{3}}.$$

Since this probability is to be bounded by β, we have $\frac{np}{4}(\phi + \lambda/E_x)^2/3 \geq \ln(\frac{1}{\beta})$.

Therefore, we need $n \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{p(\phi + \frac{\lambda}{E_x})^2}$ samples.

In the $\left(\phi + \frac{\lambda}{E_x}\right) \geq 1$ domain, a different version of Chernoff bound (Lemma 12) implies that $n \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{p(\phi+\frac{\lambda}{E_x})}$ circuits suffice. In that case, $\frac{1}{\min\{1,(\phi+\frac{\lambda}{E_x})\}} \geq \frac{1}{(\phi+\frac{\lambda}{E_x})}$, and thus

$$n \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{p} \cdot \frac{1}{(\min\{1, (\phi + \frac{\lambda}{E_x})\})^2}$$

samples suffice. □

False positive rate (bounded by β). The direct consequence of the lemma is to limit the false positive rate. It implies that a small number of samples suffices to ensure a false positive rate bounded by β. Sample size n increases only logarithmically with decreasing value of β, and it increases at most linearly as $\frac{1}{\phi^2}$. Observe that $1/p$ is the expected number of circuits needed to have any connection through x at all.

False negative rate (bounded by β). A false negative occurs when an attack is under way and the observation is drawn from the distribution on the right (Fig. 4), but the observation happens to be on the other side of the threshold. By a variant of Chernoff bound [26] the same sample complexity suffices as Lemma 6. Thus the false negative rate of the algorithm is also bounded by β.

5.1.2 Differential Privacy and Secret Sharing

To protect the privacy of circuits counted by the middle relays (e.g. from a non-PB adversary), each relay implements local differential privacy by adding noise to the count before reporting it to the aggregator. The quantity of noise required depends on a quantity called the sensitivity of the model:

Definition 4 (Sensitivity[12])**.** *The sensitivity of a function $f : D \to \mathbb{R}$ is defined as:*

$$\Delta f = \max_{D,D'} ||f(D) - f(D')|| \tag{3}$$

for all neighbouring databases D, D'.

Laplace mechanism of differential privacy. This mechanism requires drawing a random number γ from the Laplace distribution: $\gamma \sim Lap\left(\frac{\Delta f}{\epsilon}\right)$ with mean 0 and scale $\frac{\Delta f}{\epsilon}$. [5]
In the case of counting, the presence or absence of a single circuit can change the count by at most 1, and so $\Delta f = 1$. Thus, a middle node samples a noise γ from a Laplace distribution $\gamma \sim Lap\left(\frac{1}{\epsilon}\right)$. Instead of reporting a true count C, the middle node now reports $C + \gamma$. It repeats the process for every count reported.

The consequence of this method is that the probabilities of reporting the same count are similar (to within a factor of e^{ε}) whether a single particular circuit is counted or not. The presence of a constant number of circuits can be hidden by setting $\Delta f = k$, and thus drawing the noise from $Lap\left(\frac{k}{\epsilon}\right)$. The parameter k is a global constant for the system, set at initialization, and known to all nodes.

The method of each data source (middle relay) adding noise independently is called *local differential privacy.* This approach has the advantage that it is independent of the actions of the aggregator. By sanitising data locally, the middle nodes ensure that the shared information is permanently protected.

When added over a large number of source nodes, the noise accumulates. The following result from Chan *et al.* [9] gives a bound on the accumulated noise:

Lemma 5 (Sum of Independent Laplace Distributions)**.** *Given γ_i as independent random variables following Laplace distribution* $\text{Lap}(b_i)$ *with mean zero. Suppose $Y = \sum_i \gamma_i$ and $0 < \delta < 1$. Then $|Y|$ is at most $O(\sqrt{\sum_i b_i^2}\log(1/\delta))$ with probability at least $1 - \delta$.*

The bound, however, shows that we can only limit the noise to a limited degree. For example, in a system consisting of $|M|$ data sources, and each adding an independent noise from $\text{Lap}(1/\varepsilon)$, the noise is only bounded by $O(\frac{1}{\varepsilon}\sqrt{|M|}\log\frac{1}{\delta})$ with probability $1 - \delta$. Thus, as the number($|M|$) of reporting data sources grows, the noise grows rapidly as $\sqrt{|M|}$.

Secret sharing. This noise accumulation can be avoided by adding a single random noise $\gamma \sim Lap\left(\frac{k}{\epsilon}\right)$ to the total sum instead of one to each variable. We achieve this securely by employing additive secret sharing among the relays and the aggregator (See Section 2.2). In this design, the communication from each

[5] This Laplace distribution is described by the probability density function $P\left(x|0, \frac{\Delta f}{\epsilon}\right) = \frac{\epsilon}{2\Delta f}\exp\left(\frac{-\epsilon|x|}{\Delta f}\right)$.

relay is secure, and the aggregator only sees the total of the values from all the relays.

In a distributed setup, addition of a single random number is achieved by each middle adding a smaller noise to their value, which sums to sufficient noise.

When there are $|M|$ middle relays, each should add a noise equivalent to:

$$R \sim \Gamma\left(\frac{1}{|M|}, \frac{\epsilon}{k|M|}\right) - \Gamma\left(\frac{1}{|M|}, \frac{\epsilon}{k|M|}\right).$$

before transmitting to the aggregator. It can be shown that the sum of these $|M|$ random numbers follows the $Lap\left(\frac{k}{\varepsilon}\right)$ distribution [20].

5.2 System Design

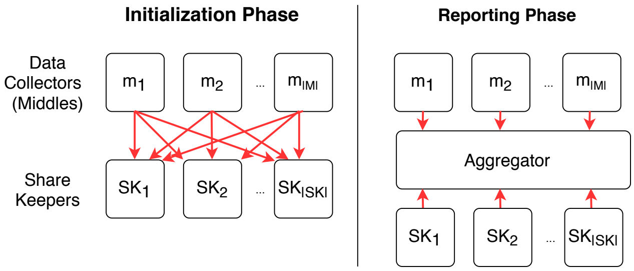


Fig. 5. Scheme design showing two of the phases where communication occurs. The red arrows denote shares.

Figure 5 shows the schematic view of the detection scheme, which has three phases of operation: initialization, data collection, and reporting.

In the *initialization phase*, the data collectors (middle relays) initialize frequency counters for all guard-exit pairs. These pairs are the same across all data collectors and the same binning parameters (γ, η – to be introduced in next section) are used for every middle relay. Secret sharing is initialized as discussed earlier. A single independent Laplace noise value, for differential privacy, is drawn by each middle relay for each counter it stores.

In the *data collection phase*, the data collectors increment the counter for each pair by 1 when a circuit with that guard-exit bin pair is observed. The counter is only incremented if some data travels via the circuit, therefore the results cannot be skewed by an adversary falsely creating many never-used circuits (see the following discussion for more details).

In the *reporting phase*, the data collectors send stored frequencies (plus the Laplace noise) to the aggregator. The share keepers combine the shares they received from the data collectors, and send them to the aggregator. The aggregator receives the shares from the share keepers and the noisy frequency estimates, and simply adds these to recover the original frequencies plus Laplace noise. The aggregator then releases the private frequency estimates.

After obtaining the noisy statistics, the end user of this system (for example, Tor Metrics) can use the results to identify presence of suspicious activity. The detection is statistical in nature and does not provide absolute guarantees. It is expected that the user will perform additional investigations before taking action against specific relays.

The enhancements described in Section 6.1 include a strategy of grouping similar exits together to reduce noise, which improves attack detection, but creates greater uncertainty about the compromised relay. Overall, our methods are thus not necessarily meant to pin point the adversaries, but designed to generate first alarms in case of attacks.

Discussions on Differential Privacy. If the number of circuits to be protected is k, then instead of $Lap(\frac{1}{\epsilon})$ noise, we add $Lap(\frac{k}{\epsilon})$ noise. In this scenario, the sample size bounds (and thus the utility guarantees) can be easily adjusted by simply increasing any terms in the sample size bound containing ϵ by a factor of k. Elahi *et al.* [14] suggest that for a collection period (epoch length) of 1 hour, $k = 6$ is appropriate. If $k + c$ circuits were observed, the resulting value of ϵ would be $\epsilon\frac{k+c}{k}$. For example, when $\epsilon = 0.1$, $c = 1$ and $k = 6$. Then, ϵ increases from 0.1 to 0.117.

We suggest counting circuits after they have been built and some data have been sent, for example, after the `RELAY_DATA` cell is seen[6]. A circuit rejected before this threshold does not affect the counts and therefore does not affect the differential privacy guarantee.

An adversary aware of the protocol may choose to wait until the threshold to discard the circuit, but this incurs extra cost for the adversary. It would also increase the overhead for the PB attack by both the size of a `RELAY_DATA` cell (586 Bytes), and the time it takes to transmit, per circuit path bias attempt. Using the examples of the large (20% TBW) and small (2% TBW) adversary from Section 4.1, the worst-case additional bandwidth overheads are 11.7 KB and 117.8 KB respectively per compromised circuit. The additional time overheads are $\approx 1\,s$ and $\approx 10\,s$ respectively per compro-

[6] Our scheme is not contingent on this condition, and can be adapted to other thresholds.

mised circuit. For a compromise rate of 50% of circuits going through their guards, the larger adversary incurs an additional BW cost of 2.34 GB, and the smaller adversary incurs an additional BW cost of 2.36 GB.

Further, a behaviour of rejecting established and counted circuits [7] in an attempt to carry out the PB attack will inflate the count for the guard itself, and can be detected by observing a larger $\hat{n}_g$, essentially with the same statistical test as Lemma 6:

Observation 6. *A sample size of*

$$n \geq 12 \ln(\frac{1}{\beta}) \cdot \frac{1}{p} \cdot \frac{1}{(\min\{1, (\phi + \frac{\lambda}{E_g})\})^2}$$

suffices to ensure that $n_g \leq E_g + (\phi E_g + \lambda)/2$ *with probability at least* $(1 - \beta)$.

A large adversary performing this attack on a ϕ fraction of its overall circuits, or a tiny adversary performing this attack on λ circuits, will be detected by this test.

Composition Across Epochs. The composition of T passes of an ϵ-differentially private algorithm over a given database D results in a final privacy guarantee of $T\epsilon$ [12]. Therefore, in the worst-case of a user's behaviour remaining identical across T epochs, the privacy guarantee for the user after T epochs degrades to $T\epsilon$. The epoch length can be increased to control this effect, with the downside of less frequent reporting. Given that there is a significant cost and delay involved in being approved as a guard, monitoring in epochs of days can be considered suitable for Tor.

Note that if users' behaviour is different across epochs, the databases used in each epoch would be disjoint and privacy degradation would be smaller.

5.3 Security

Our system design and communication protocols are based on the secret-sharing variant of PrivEx [14], therefore the security analysis from there also applies here. However, our algorithm uses different databases and noise mechanisms. We provide discussions of the privacy and utility guarantees provided by our algorithm.

The worst-case vulnerabilities posed by malicious relays, share keepers or aggregators are limited to subverting the attack detection or denial of service attacks on the algorithm operations. Both of these vulnerabilities risk the normal operation of the detection algorithm, but do not risk the safety of the network, or the privacy of individual users – our scheme *fails-safe.* We further address the risk of malicious middle relays misreporting frequencies in Section 6.3.1.

6 Enhanced Detection Algorithm

In the Tor network, the bandwidths of nodes and therefore their probabilities of being selected in a path vary. Some exit nodes have probability of 10^{-6} or smaller of being chosen on a particular circuit, which raises a challenge in performing detection when the epoch length, and therefore data volume, are bounded. In these realistic scenarios, low probability exits will have very small number of circuits to report, which can be dominated by the Laplace noise.

6.1 Binning Algorithm: Reporting Exits in Groups

To decrease noise in the system, we take the strategy of grouping multiple exit relays together into bins. A single unit of noise then suffices per bin, leading to less noise per relay.

We require the bins to have the following properties. For a bin B, and $x, y \in B$:

$$E_x \leq (1 + \gamma)E_y + \eta \tag{4}$$

$$|B| \leq m \tag{5}$$

For given parameters $\gamma, \eta \in \mathbb{R}^+, m \in \mathbb{Z}^+$.

The binning is achieved simply by sequentially processing exits in decreasing order of bandwidth and adding to a bin until the addition will violate one of the two conditions. See Algorithm 1 (Appendix A.3).

After running this algorithm, each middle has the same set of bins and allocations of exits to those bins (alternatively, the bins may be decided by the aggregator and reported to middle nodes). The middle nodes now report counts for bins, that is total for counts of all exits in a bins instead of counts for individual exits.

In the above strategy, m is a bound on the bin size. Parameters γ and η are set to describe the maximum possible difference in expected counts between exit relays in a single bin, where γ is the multiplicative factor and η is a small additive factor (e.g. 1e-5) allowing us

[7] In the event that the adversary is following an inefficient form of the PB attack as compared to the one shown by Rochet and Pereira.

to group together extremely small probability exits that would otherwise require a large value of γ.

The binning based method does not by itself pin point the compromised exit. Its purpose is to detect the presence of an attack quickly using a small amount of data, which can then be investigated further. As before, differential privacy can be achieved by adding $\text{Lap}\left(\frac{1}{\epsilon}\right)$ noise. However, in this case this unit of noise is added to the count in a bin instead of individual exits.

We now show that statistical testing based on binning is efficient.

6.1.1 Analysis of Binning Algorithm

For binning parameters γ, η, m as defined above, with E_B denoting the expected frequency of the bin B,

Theorem 7. *A sample size of:*

$$n > 12\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma)+\eta] + 6\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi p}$$

suffices to ensure that $n_B \leq E_B + \frac{1}{2}(\phi E_x + \lambda)$ with probability at least $1-\beta$.

See Appendix A.1 for the proof of Theorem 7. This result implies that the sample size increases only linearly with the size of the bin m.

6.1.2 Analysis of Differentially Private Detection with Binning

When we add Laplace noise to the counts in bins, the required sample complexity does not increase beyond small factors. As before, the results apply symmetrically to bounds for false positive and false negative rates.

Theorem 8. *With a noise $\zeta \sim \text{Lap}(\frac{1}{\epsilon})$ a sample size of:*

$$n \geq 4\left(12\ln(\frac{2}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma)+\eta] + 6\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi p}\right)$$
$$+ 4\left(\frac{1}{\epsilon} \cdot \ln(\frac{1}{\beta}) \cdot \frac{1}{\phi} \cdot \frac{1}{p}\right)$$

suffices to ensure that $n_B + \zeta \leq E_B + \phi E_x + \lambda$ with probability at least $1-\beta$ and ϵ-differential privacy.

See Appendix A.1 for the proof of Theorem 8. Note that this theorem is for the goal of obfuscating the presence of any single circuit. When the objective is to hide any k circuits to the same level of privacy, the sample complexity (or more precisely, the second term in expression of n) grows by a factor of k.

Sample Sizes in practice. The Tor network constructs more than 1.2 billion circuits per day [24]. Consider the case where $\beta = 0.05$ (a 95% confidence for identified outliers), $\phi = 1$, $p = 0.01$, $m = 10$, $\gamma = 5$, $\eta = 0.01$ and $\epsilon = 0.1$. In this case, a sample size of $n \geq 275,785$ is sufficient. If the attack size decreases to $\phi = 0.1$ a sample size of $n \geq 26,661,792$ is sufficient.

These theoretical estimates are loose, and are meant to provide a conceptual bound that the data requirement does not grow excessively fast. Experimental results in Section 7 show that in practice substantially smaller datasets produce reliable results.

An additional note here is that these theoretical results are with a view to reliably detecting a difference in the circuit distribution, and thus the sample size is required to be a multiple of $1/p$, to ensure that there are circuits through x for detection of attack. In practice, however, absence of circuits implies that there is no significant attack action to detect.

Attack Parameters in practice. The values of ϕ and λ are selected based on the characteristics of the attacker and the desired trade-off between false positives and false negatives.

Suppose that an adversary controls a 0.1 proportion of both the total guard relay bandwidth and the total exit relay bandwidth. As demonstrated by Figure 2, this translates to an increase of between 1x and 20x times more circuits observed through those compromised relays, implying a level of ϕ between 1 and 20. The possible value of ϕ increases as the proportion of bandwidth controlled decreases (Figure 2).

If $Lap(\frac{k}{\epsilon})$ noise is added, then $\lambda = -\ln(0.05)\frac{k}{\epsilon}$ is the 95% confidence level for the increase in observations due to this noise. Setting λ to this value or larger will reduce the number of false positives due to the noise, particularly for pairs where $\phi E_{\hat{g}B}$ is small.

6.2 Overhead Costs

Running the detection algorithm introduces some cost for the relays, share keepers and aggregator. Suppose the system comprises of $|G|$ guard relays, $|M|$ middle relays (data collectors), $|B|$ exit bins, $|SK|$ share keepers and a single aggregator. Let relay identifiers have at most I characters. Let us consider a scenario in which

$|G| = 3000, |M| = 3000, |B| = 100, |SK| = 10$ and $I = 16$. These values approximate typical numbers for Tor.

Database size. The size of the database determines the basic communication and storage costs. The database stores the count for $|G| \times |B|$ guard-bin pairs. Assuming that each count is stored as a double precision floating point number, each taking 8 Bytes to of space, which for our scenario, amounts to a database size of 2.4 MB at each node.

Initialization Phase. During the intialization phase, data collectors (middles) initialize storage for the $|G| \times |B|$ counters, and an database of size 2.4 MB.

Each data collector sends the key information for all the pairs (2.4 MB) to the share keepers.

Data Collection Phase. During the data collection phase, data collectors (middles) increment the appropriate counter by 1 each time a new circuit uses the pair and has suitable traffic.

Reporting Phase. During the reporting phase, data collectors (middles) report all frequencies, sending the database of size 2.4 MB to the aggregator. The aggregator receives a 2.4 MB database from each middle relay, as well as the shares from the share keepers.

Discussion of Overhead Costs: The communication costs are constant, and small for any reasonable length of epochs. In each epoch, a middle relay communicates 2.4 MB for the detection algorithm, resulting in ≈ 7.2GB for the whole network. Tor transports 517 TiB of data per day [24] (21.54 TiB/hr). Thus the cost of the detection algorithm is negligible compared to the volume of Tor client traffic, even for short epochs of one hour.

From the perspective of a single node, 2.4MB is also a small cost. For example, in the consensus labelled 2019-09-04-18-00-00, approximately 95% of non-exit-flagged relays (possible middle relays) have bandwidth of above 0.01 MB/s (40 MB/hr), while 84% and 64% have bandwidths of above 0.1 MB/s (360 MB/hr) and 1 MB/s respectively.

The costs of initialising and incrementing counters are also small. In 1000 trial runs on a 2.9 GHz Intel Core i7 processor, binning the exits (once per epoch) takes 13 ms on average. Initializing the frequency counters and random noise, takes 536 ms. Incrementing a counter when a new circuit is observed takes on average 0.45 μs. For example, if 1 million circuits were observed, the total time cost would be 450 ms over the epoch.

6.3 Strategic Attacker and Practical Considerations in Tor

It is natural to ask if instead of allocating its entire bandwidth to single guard and exit nodes, the adversary can gain any substantial benefit by dividing their available bandwidth and running multiple relays.

To understand the effect of this strategy, let us consider the attack parameters ϕ and λ separately. When $\lambda = 0$ and $\phi > 0$, the adversary can gain no benefit by changing the distribution of bandwidths; this follows simply by linearity. For example, if the adversary runs a single exit $\hat{x}$ with expected $E_{\hat{x}}$ circuits, then they gain at most $\phi E_{\hat{x}}$ additional circuits by the active attack without being detected. If the adversary were to split the the bandwidth into multiple exits, the total expected circuits remains the same and thus the additional circuits remains the same at a factor of ϕ.

When $\lambda > 0$, the adversary can gain some advantage. In this case, the adversary can direct λ additional circuits without detection to each bin containing a compromised exit. For nodes with large bandwidth, $\phi E_x >> \lambda$, the approach does not gain significant benefit over a passive attack. The approach becomes meaningful only when λ is comparable to ϕE_x. Since λ is a small constant, the strategic adversary is restricted to considering multiple low bandwidth exits.

In practice, low bandwidth exits are not particularly strong attack tools since they are unlikely to be chosen by the Tor path selection algorithm. An adversary may insist on rejecting circuits many times to compromise them, but simple calculations show that this will mean rejecting a circuit hundreds of times before the compromised exit is picked. Clients are likely to assume that the guard is defective, and change to a different guard.

An adversary may choose to deploy multiple exits in the hope of increasing their chances, but it is not possible to stretch this far. For example, the current Tor system has ≈ 1000 exits, and deploying hundreds of new exits will be easily detected as suspicious behavior.

6.3.1 Sampling to Avoid Adversary Interference

Suppose an adversary controls a proportion $\frac{q}{|M|}$ of the middle relays $\hat{m} \in M$ and attempts to subvert detection by misreporting frequencies.

As a strategy to avoid this interference, the aggregator can sample a random subset of K middle relays from M and then aggregate the observed frequencies from only these relays. The probability of the result being affected is reduced to $1 - \left(1 - \frac{q}{|M|}\right)^K$, which decreases with decreasing q and K. For example, if $|M| = 1000$, $q = 10$ and $K = 100$, the probability of result being affected by the adversary is reduced to 63%. If $q = 5$, the probability reduces to 40%.

Lemma 9. *A sample size of:*

$$n \geq \frac{4}{\sum_{i \in \{1,\ldots,K\}} p_{m_i}} \left(12 \ln(\frac{2}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma)+\eta]\right)$$
$$+ \frac{4}{\sum_{i \in \{1,\ldots,K\}} p_{m_i}} \left(6 \ln(\frac{1}{\beta}) \cdot \frac{1}{\phi p}\right)$$
$$+ \frac{4}{\sum_{i \in \{1,\ldots,K\}} p_{m_i}} \left(\frac{1}{\epsilon} \cdot \ln(\frac{1}{\beta}) \cdot \frac{1}{\phi} \cdot \frac{1}{p}\right)$$

circuits overall suffices to ensure that $n_B \leq E_B + \phi E_x + \lambda$ with probability at least $1 - \beta$ given that the aggregator randomly selects K middle relays without replacement from the set of $|M|$ middle relays.

Lemma 9 describes the cost of this strategy, which is an increase in the number of observations required (sample size) by a factor of the reciprocal of the proportion of network traffic observed by the sampled middle relays.

For example, if the sample of middle relays used to report frequencies represents 50% of the total probability mass over selecting middle relays, then the sample size increases from n to $2n$. See Appendix A.1 for the proof of Lemma 9.

Choosing a large value of K which is likely to cover a large fraction of the network reduces the basic sample size, but the probability of the adversary affecting the result increases as K becomes large. Therefore, there is a natural trade-off here between protection against misreporting and the increase in sample size required. The exponential dependence on K suggests that sampling alone results in a trade-off which could be further improved, underlined by the relatively high revised probabilities of the adversary affecting the result.

Despite this, sampling the middle nodes also has the benefit of increasing the privacy of the algorithm, as there is now a lower chance of any one group of circuits being included in the result:

Observation 10. *The subroutine of sampling has been shown to increase the level of privacy obtained by an ϵ-differentially private algorithm [2, 4, 5, 8, 10, 18]. As outlined by Balle et al. [2], uniformly sampling K relays from $|M|$ without replacement would amplify the ϵ-differentially private algorithm described, resulting in a new value of $\epsilon' = \log\left(1 + \frac{K}{|M|}(e^\epsilon - 1)\right)$. For example, if $\epsilon = 0.1, K = 100, |M| = 1000$ the multiplicative guarantee e^ϵ improves from 1.105 to 1.005.*

6.3.2 Voting to Prevent Adversary Interference

An alternative approach to reduce the effect of the adversary compromising up to q middle relays would be for each middle relay to run the statistical test on its own and submit the decision as a vote. The aggregator may collect votes from K of the $|M|$ middle relays, and check the total against a threshold T. The aggregator in this scheme works as follows:

1. Randomly select K from $|M|$ middle relays.
2. Receive a binary vote (outlier or not) for each guard-exit pair from each middle relay.
3. For each guard-exit pair, if the total number of positive votes exceeds a threshold T, identify that pair as an outlier.

The defense provided by this method is to reduce the influence of misreporting relays by collecting a vote, instead of a potentially unbounded count, from each relay. This strategy prevents a small number of compromised relays from flipping the decision for any pair by reporting a large value. It ensures a high probability of correctly identifying an outlier pair despite the compromised votes (Lemma 11).

Lemma 11. *For up to q compromised relays, the probability of correctly identifying if a guard-exit bin pair is an outlier is at least:*

$$1 - e^{-\frac{\left(1 - \frac{T+q}{(1-\beta')K}\right)^2 (1-\beta')K}{2 + \left(\frac{T+q}{(1-\beta')K} - 1\right)}}$$

given that each individual relay has a probability $(1 - \beta')$ of reporting the correct result for that pair.

See Appendix A.1 for the proof of Lemma 11. Note that in order for an individual relay to have at least a $(1 - \beta')$ probability of reporting the correct result for a given pair, they must individually satisfy the sample size requirement. Therefore, similarly to sampling, the sample size bound must be scaled up by the reciprocal of the individual's proportion of network traffic, with β in Lemma 9 replaced by β'.

Discussion of parameters. Increasing β' increases the probability of non-compromised relays voting incorrectly. Decreasing β' increases the probability of correct non-compromised votes, but also increases the size of the dataset required. Increasing the value of K increases the probability that compromised relays will be included by the aggregator. Decreasing K increases the relative effect of including compromised votes. The severity of this effect depends on the threshold T. Increasing T implies that more positive votes are needed to identify a pair as an outlier. T should be low enough to allow for some errors by non-compromised relays, reducing the required dataset size by allowing for a higher value of β'. However, T should also be high enough to avoid compromised relays from easily overwhelming the votes of non-compromised relays e.g. when T is close to K.

Using our previous example, if $q = 10$, $|M| = 1000, K = 100, \beta' = 0.1$ and $T = 50$, the probability of correctly identifying an outlier is 0.99. Doubling the probability of incorrect non-compromised votes to $\beta' = 0.2$ changes the probability of correct identification in this scenario to 0.95. Doubling the number of compromised relays to $q = 20$ results in a probability of 0.92. Allowing T to be close to K with $T = 95$ (as advised against), reduces the probability of correct detection to 0.68. The probability of correct detection approaches 1 as K increases.

7 Simulations

In this section we present experimental evidence demonstrating the performance of the above algorithms for detecting attacks on datasets of routes generated according to the Tor process. The accuracy of the algorithm is defined as its F1-score, which is a combined measure of the false positive and false negative rates of detection. The main observations are:

1. Attacks using high bandwidth relays can be detected using a low amount of data (10 million circuits $\approx$ 10 minutes of traffic on Tor) when the privacy guarantee is strong ($\epsilon = 0.1$). This is reflected by an F1 score of over 0.99. (Section 7.4).
2. If all other parameters are held constant, increasing the privacy guarantee from $\epsilon = 1$ to $\epsilon = 0.1$ causes a decrease in the F1 score from 0.99 to 0.84. (Section 7.3).
3. This decrease in accuracy due to a higher level of noise can be offset by increasing λ as ϵ decreases. In this case, even for $\epsilon = 0.1$ an F1 score of 0.98 is attained. (Section 7.3).
4. Attacks which are spread out between many low bandwidth exit relays are harder to detect. However, increasing the dataset size recovers the lost accuracy, resulting in F1 scores of over 0.99, as long as the guard relay is either high bandwidth or medium bandwidth. (Section 7.4).
5. Attacks using many low bandwidth guards are the most difficult to detect, with F1 scores of between 0.79 and 0.88. (Section 7.5).

See Section 7.2 for a full explanation of the terms used.

7.1 Data Sets

Two datasets were used in the completion of these experiments.The first dataset (RW-Tor) captures a total of $353,331$ successful circuits generated via Tor clients from four different regions: Canada(Central, $86,740$ circuits), Frankfurt(Europe, $96,072$ circuits), Oregon(US-West, $91,397$ circuits), and Tokyo(Asia, $79,122$ circuits). Our clients selected routes using the latest version of the Tor routing mechanism. See Appendix A.2 for details of the collection procedure. Due to the large cost of generating the RW-Tor data set, a second data set (SYN-Tor) was *synthetically* generated containing 1 billion circuits (around a day of traffic across the Tor network [24]) by outputting routes with a distribution matching the relay selection probabilities described by the Tor consensus document labeled 2019-09-04-18-00-00.

Comparing Synthetic to Real World Tor Circuit Data. Our expected number of circuits observed at a node comes from the consensus documents generated by the Tor network every hour. However, the actual circuits that clients build depend on 1) the Tor path selection algorithm that introduces additional rules for security and performance reasons, 2) transient network effects, and 3) path skews due to client location on the network [19]. We use our RW-Tor dataset for this analysis.

Due to our limited sample size, we analyze the top 10 guard-exit pairs since we can be more confident ($\beta = 0.05$) at this end of the range. Our results show that these pairs are more frequently selected than is expected, on average 1.96 times more often ($\sigma^2 = 0.753$). This may be for the reasons we have already mentioned above. To accommodate these phenomena, our scheme can be tuned with this in mind by setting $\phi = 0.96$, in this case. With a larger sample size we could better tune the scheme by analysing the network deviation across all guard-exit pairs and updating the expected probability distributions and scheme parameters accordingly.

7.2 Simulation Set-Up (SYN-Tor)

The ability of the algorithm to detect an attack depends on factors including the amount of data (n), the expected probability of the compromised relay (p), the attack size (ϕ, λ) and the privacy level (ϵ). In order to explore the relationships between these interacting factors, we defined several dataset sizes of between 2 million and 1 billion circuits and examined the accuracy of the algorithm for various attack scenarios. Following [14], we let $k = 6$, our results then obscure changes of up to 6 circuits per epoch.

The attack scenarios are defined by the 'size' of the relays being compromised (p) and the magnitude of the attack (i.e. the probability p_A that a route through a compromised guard will be rejected).

We consider two types of simulated adversary:

1. A high resource adversary who has the resources available to run both a single compromised high bandwidth guard and high bandwidth exit for the desired amount of time. These relays have a $\approx 3 \times 10^{-3}$ proportion of the total available guard/exit bandwidth.
2. A medium resource adversary who has the resources available to run both a single compromised medium bandwidth guard and medium bandwidth exit for the desired amount of time. These relays have a $\approx 1 \times 10^{-4}$ to 1×10^{-5} proportion of the total bandwidth

A low bandwidth relay has $p < 1 \times 10^{-6}$. Note that the adversary can divide resources into multiple lower bandwidth relays in each of the two cases above. For example, a high resource adversary could compromise approximately 10 medium bandwidth guard/exit relays.

In each scenario, the attacks were simulated by selecting compromised relays of the appropriate expected frequencies and then, with probability $p_A = 0.1$, removing routes generated which contained a compromised guard relay but did not pass through a compromised exit relay. These routes were replaced with routes passing through the compromised exit relay. The appropriate expected frequencies were those described by the Consensus labeled 2019-09-04-18-00-00.

7.3 The Effect of ϵ

We first examine the effect which the level of privacy guarantee has on accuracy. As shown in Figure 6, when all other parameters are held constant, increasing the strength of the privacy guarantee reduces accuracy due to the addition of a larger amount of random noise. For example, using a dataset of 50 million circuits letting $\epsilon = 0.1$ results in an F1 score of 0.84 whereas $\epsilon = 1$ produces an F1 score of 0.99. Figure 6 demonstrates that for low values of ϵ such as 0.1, the F1 score can be improved by increasing the dataset size. In this case, for $\epsilon = 0.1$, the F1 score increases to 0.87 if the dataset size increases to 1 billion circuits (∼1 day of traffic).

Larger improvements can be made by allowing λ to increase as ϵ decreases to reflect the larger magnitude of noise added. Figure 6 demonstrates that the loss in accuracy is significantly reduced, with all F1 scores between 0.977 and 0.99, if we allow $\lambda = -\ln(0.05)\frac{6}{\epsilon}$ in order to reflect the 95% confidence interval for the Laplace distribution with scale $\frac{6}{\epsilon}$.

7.4 Detecting Different Attack Scenarios

In this section, scenarios including those which spread the attacks in this way are considered.

High Resource Adversary. Suppose we define high, medium and low bandwidth relays as before. A high resource adversary could then use any combination of a single high bandwidth guard, some medium bandwidth guards (i.e. 10) or many (i.e. 100) low bandwidth guards with a single high bandwidth exit, some medium bandwidth exits or many low bandwidth exits.

As shown in Table 1, detection quickly reaches an F1 score of 1 using any combination of high and medium bandwidth relays.

	Datasize (mil.)					
	10	50	100	200	500	1000
High BW Guard						
High BW Exit	**0.99**	**1**	**1**	**1**	**1**	**1**
Medium BW Exits	**0.99**	**0.99**	**1**	**1**	**1**	**1**
Low BW Exits	**.76**	**.98**	**.99**	**.99**	**1**	**1**
Medium BW Guards						
High BW Exit	**.99**	**.99**	**1**	**1**	**1**	**1**
Medium BW Exits	**.92**	**.99**	**1**	**1**	**1**	**1**
Low BW Exits	**.67**	**.69**	**0.79**	**0.92**	**0.99**	**1**
Low BW Guards						
High BW Exit	**0.65**	**0.70**	**0.74**	**0.76**	**0.79**	**0.82**
Medium BW Exits	**0.65**	**0.66**	**0.66**	**0.67**	**0.72**	**0.74**
Low BW Exits	**0.65**	**0.66**	**0.66**	**0.66**	**0.66**	**0.66**

Table 1. F1 Score vs datasize with $\epsilon = 0.1, \phi = 10, \lambda = 150$.

Attacks using low bandwidth exit relays with high or medium bandwidth guard relays also reach an F1 score of 1 when the dataset size is increased to ap-

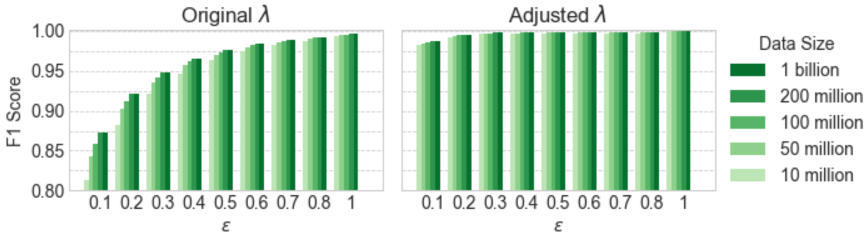


Fig. 6. High bandwidth guard and exit pairs, using $\phi = 10, \lambda = 20$ and $\phi = 10,\ \lambda = -\ln(0.05)\frac{6}{\epsilon}$ respectively.

proximately 1 hour of data for a single high bandwidth guard, and 1 day of traffic using some medium bandwidth guards.

The same is not true of attacks using many low bandwidth guards. However, as discussed further in Section 7.5, calibrating binning parameters of the algorithm more carefully leads to significant improvements in the performance of the algorithm in this scenario.

Medium Resource Adversary. According to our definition of a medium resource adversary, this type of adversary could use any combination of a single medium bandwidth guard relay or some (i.e. 10) low bandwidth guard relays with a single medium bandwidth exit relay or some low bandwidth exit relays.

Table 3 (Appendix A.5) demonstrates that the conclusions in the medium resource adversary scenario are similar to those of the high resource adversary scenario.

7.5 Binning Strategy for Low Bandwidth Guards

The algorithm's performance when the adversary uses low bandwidth guards can be improved by altering the generic parameter settings used above to increase the number of exit relays per bin. This is done by increasing the maximum number of relays per bin from $m = 20$ and $\eta = 1 \times 10^{-6}$, to $m = 100$ and $\eta = 1 \times 10^{-5}$.

The maximum F1 scores attained for the high resource adversary case with multiple low bandwidth guards were originally 0.82, 0.74 and 0.66 (Table 1). The revised binning strategy improved these to 0.86, 0.84 and 0.80 respectively (Table 4, Appendix A.5). In the medium adversary case, the maximum attained F1 scores improved from 0.86 and 0.61 (Table 3) to 0.88 and 0.79 (Table 4).

8 Conclusion

Path manipulation attacks can compromise various anonymity systems. We presented a statistical framework for detecting such attacks with rigorous guarantees for Tor. We have shown that the accuracy of the detection improves rapidly with available data as well as with scale of attacks. Thus, as the popularity of such systems grows, or adversaries gain more powerful nodes and attacks become more dangerous, the detection mechanism becomes more effective. We also expect our approach to be useful in other anonymous communication systems such as mix networks and for securing IoT networks.

Acknowledgements

This work was funded in part by the UK National Centre for Cyber Security. We thank Katharina Kohls and the anonymous reviewers for many helpful suggestions that improved the paper.

References

[1] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. 2014. LASTor: A Low-Latency AS-Aware Tor Client. *IEEE/ACM Trans. Netw.* 22, 6 (Dec. 2014), 1742–1755. https://doi.org/10.1109/TNET.2013.2291242

[2] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*. 6277–6287.

[3] Armon Barton and Matthew Wright. 2016. DeNASA: Destination-Naive AS-Awareness in Anonymous Communications. *Proceedings on Privacy Enhancing Technologies* 2016 (02 2016). https://doi.org/10.1515/popets-2016-0044

[4] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. 2010. Bounds on the sample complexity for private learning and private data release. In *Theory of Cryptography Conference*. Springer, 437–454.

[5] Amos Beimel, Kobbi Nissim, and Uri Stemmer. 2013. Characterizing the sample complexity of private learners. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. ACM, 97–110.

[6] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. 2001. Web MIXes: A system for anonymous and unobservable Internet access. In *Designing privacy enhancing technologies*. Springer, 115–129.

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

[8] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. 2015. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 634–649.

[9] T-H Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)* 14, 3 (2011), 26.

[10] Kamalika Chaudhuri and Nina Mishra. 2006. When random sampling preserves privacy. In *Annual International Cryptology Conference*. Springer, 198–213.

[11] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router*. Technical Report. Naval Research Lab Washington DC.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[13] Matthew Edman and Paul Syverson. 2009. AS-awareness in Tor path selection. In *Proceedings of the 16th ACM conference on Computer and communications security*. 380–389.

[14] Tariq Elahi, George Danezis, and Ian Goldberg. 2014. Privex: Private collection of traffic statistics for anonymous communication networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1068–1079.

[15] Ceki Gulcu and Gene Tsudik. 1996. Mixing E-mail with Babel. In *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*. IEEE, 2–16.

[16] Rob Jansen and Aaron Johnson. 2016. Safely measuring tor. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1553–1567.

[17] Aaron Johnson, Rob Jansen, Aaron D Jaggard, Joan Feigenbaum, and Paul Syverson. 2015. Avoiding the man on the wire: Improving Tor's security with trust-aware path selection. *arXiv preprint arXiv:1511.05453* (2015).

[18] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[19] Katharina Kohls, Kai Jansen, David Rupprecht, Thorsten Holz, and Christina Pöpper. 2019. On the Challenges of Geographical Avoidance for Tor.. In *NDSS*.

[20] Samuel Kotz, Tomasz Kozubowski, and Krzystof Podgorski. 2012. *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Springer Science & Business Media.

[21] Hemi Leibowitz, Ania M Piotrowska, George Danezis, and Amir Herzberg. 2019. No right to remain silent: isolating malicious mixes. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1841–1858.

[22] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. 2010. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010) (LNCS)*. Springer.

[23] Akshaya Mani and Micah Sherr. 2017. HisTorε: Differentially Private and Robust Statistics Collection for Tor.. In *NDSS*.

[24] Akshaya Mani, T Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. 2018. Understanding tor usage with privacy-preserving measurement. In *Proceedings of the Internet Measurement Conference 2018*. ACM, 175–187.

[25] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. 2008. Shining light in dark places: Understanding the Tor network. In *International symposium on privacy enhancing technologies symposium*. Springer, 63–76.

[26] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press.

[27] Steven J Murdoch and George Danezis. 2005. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 183–195.

[28] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. 2015. Measuring and mitigating AS-level adversaries against Tor. *CoRR* abs/1505.05173 (2015). arXiv:1505.05173 http://arxiv.org/abs/1505.05173

[29] Lasse Overlier and Paul Syverson. 2006. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*. IEEE, 15–pp.

[30] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The loopix anonymity system. In *26th USENIX Security Symposium, USENIX Security*. 16–18.

[31] Florentin Rochet and Olivier Pereira. 2018. Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols. *Proceedings on Privacy Enhancing Technologies* 2018, 2 (April 2018).

[32] Rui Shi, Mayank Goswami, Jie Gao, and Xianfeng Gu. 2013. Is random walk truly memoryless—Traffic analysis and source location privacy under random walks. In *2013 Proceedings IEEE INFOCOM*. IEEE, 3021–3029.

[33] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*. Springer, 18–33.

[34] Atul Singh, Tsuen wan "johnny Ngan, Peter Druschel, and Dan S. Wallach. 2006. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*.

[35] Christopher Soghoian. 2011. Enforced community standards for research on users of the Tor anonymity network. In *International Conference on Financial Cryptography and Data Security*. Springer, 146–153.

[36] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2015. {RAPTOR}: Routing Attacks on Privacy in Tor. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 271–286.

[37] Ryan Wails, Aaron Johnson, Daniel Starin, Arkady Yerukhimovich, and S Dov Gordon. 2019. Stormy: Statistics in Tor by Measuring Securely. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 615–632.

[38] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. 2004. On flow correlation attacks and countermeasures in mix networks. In *International Workshop on Privacy Enhancing Technologies*. Springer, 207–225.

A Appendix

A.1 Proofs of theoretical results

Lemma 12. *Let X denote the sum of n independent random variables, $X_1, ...X_n$, where $\mu = \mathbb{E}[X]$. Then, for $\delta \geq 1$,*

$$P[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta\mu}{3}} \tag{6}$$

Proof. By [26], $P[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$.

Given that $\ln(1+\delta) > \frac{2\delta}{2+\delta}$ for all $\delta > 0$, we have

$$\begin{aligned}\ln\left[\left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}\right] &= \mu\left(\delta - (1+\delta)\ln(1+\delta)\right) \\ &\leq \mu\left(\delta\left[1 - \frac{2(1+\delta)}{2+\delta}\right]\right) \\ &= \mu\left(\frac{-\delta^2}{2+\delta}\right).\end{aligned}$$

When $\delta \geq 1$, we have $e^{\frac{-\delta^2\mu}{2+\delta}} \leq e^{-\frac{\delta\mu}{3}}$. Thus :

$$P[X \geq (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2+\delta}} \leq e^{-\frac{\delta\mu}{3}}.$$

□

Proof of Theorem 7:

Proof. We can split n into $n = n_1 + n_2$ where:

$$n_1 \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma) + \eta]$$

and

$$n_2 \geq 6\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi p}$$

Then,

$$\begin{aligned} & n_1 \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma) + \eta] \\ \implies & n_1 \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m\left[(1+\gamma) + \frac{\eta}{np}\right]\end{aligned}$$

assuming that $n_1 > \frac{1}{p}$. This implies:

$$n_1^2 p^2 \geq 12\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi^2} \cdot m[(1+\gamma)n_1 p + \eta]$$

$$\begin{aligned} \implies & \frac{n_1^2 p^2 \phi^2}{m[(1+\gamma)np + \eta]} \geq 12\ln(\frac{1}{\beta}) \\ \implies & (\phi\frac{E_x}{E_B})^2 E_B \geq 12\ln(\frac{1}{\beta}) \\ \implies & (\phi\frac{E_x}{E_B} + \frac{\lambda}{E_B})^2 E_B \geq 12\ln(\frac{1}{\beta}) \\ \implies & e^{-\frac{\frac{1}{4}(\phi\frac{E_x}{E_B} + \frac{\lambda}{E_B})^2 E_B}{3}} \leq \beta.\end{aligned}$$

By Chernoff bound, with $\frac{1}{2}\frac{\phi E_x + \lambda}{E_B} \leq 1$:

$$P[n_B \geq E_B + \frac{1}{2}(\phi E_x + \lambda)] \leq e^{-\frac{\left(\frac{1}{2}(\phi\frac{E_x}{E_B} + \frac{\lambda}{E_B})\right)^2 E_B}{3}} \leq \beta.$$

Now observe that:

$$\begin{aligned} & \phi n_2 p > 6\ln\frac{1}{\beta} \\ \implies & \phi E_x + \lambda > 6\ln\frac{1}{\beta} \\ \implies & \left(\frac{\phi E_x + \lambda}{E_B}\right) E_B \geq 6\ln\frac{1}{\beta} \\ \implies & e^{-\left(\frac{\phi E_x + \lambda}{6E_B}\right)E_B} \leq \beta\end{aligned}$$

When $\frac{1}{2}\frac{\phi E_x + \lambda}{E_B} > 1$ this implies by Lemma 12, that

$$P[n_B \geq E_B + \frac{1}{2}(\phi E_x + \lambda)] \leq e^{-\left(\frac{\phi E_x + \lambda}{6E_B}\right)E_B} \leq \beta.$$

□

Proof of Theorem 8:

Proof. We can split n into $n = n_1 + n_2$ where:

$$n_1 \geq 4\left(12\ln(\frac{2}{\beta}) \cdot \frac{1}{\phi^2} \cdot \frac{1}{p} \cdot m[(1+\gamma) + \eta] + 6\ln(\frac{1}{\beta}) \cdot \frac{1}{\phi p}\right)$$

$$n_2 \geq 4\left(\frac{1}{\epsilon} \cdot \ln(\frac{1}{\beta}) \cdot \frac{1}{\phi} \cdot \frac{1}{p}\right).$$

The following result then follows in the same way as Theorem 7, substituting ϕ with $\frac{\phi}{2}$ and β with $\frac{\beta}{2}$. A sample size of n_1 ensures that

$$P[n_B \leq E_B + \frac{1}{4}(\phi E_x + \lambda)] \leq e^{-\frac{\left(\frac{1}{4}(\phi\frac{E_x}{E_B} + \frac{\lambda}{E_B})\right)^2 E_B}{3}} \leq \frac{\beta}{2}.$$

.

For n_2, we have:

$$\begin{aligned} \phi n_2 p &\geq 4\frac{1}{\epsilon}\ln(\frac{1}{\beta}) \\ \implies \phi E_x &\geq 4\frac{1}{\epsilon}\ln(\frac{1}{\beta}) \\ \implies \phi E_x + \lambda &\geq 4\frac{1}{\epsilon}\ln(\frac{1}{\beta}) \\ \implies \frac{1}{2}e^{-\frac{1}{4}(\phi E_x+\lambda)\epsilon} &\leq \frac{\beta}{2}. \end{aligned}$$

Since $\zeta \sim \mathrm{L}ap(0, \frac{1}{\epsilon})$ this implies:

$$P[\zeta \geq \frac{1}{4}(\phi E_x + \lambda)] \leq \frac{\beta}{2}.$$

Therefore, as $n \geq n_1$ and $n \geq n_2$:

$$P[n_B + \zeta \geq E_B + \frac{1}{2}(\phi E_x + \lambda)] \leq \frac{\beta}{2} + \frac{\beta}{2} = \beta.$$

□

Proof of Lemma 9:

Proof. Middle relays $m_1, ...m_K$ receive a proportion $\alpha = \sum_{i\in\{1,...,K\}} p_{m_i}$ of the total network traffic.

The result describing the sample size in Theorem 8 still holds in this case, however now if n_o circuits are observed by the network, only αn_o are expected to be recorded by the sampled middle relays.

Therefore the result found in Theorem 8 increases by a factor of $\frac{1}{\sum_{i\in\{1,...,K\}} p_{m_i}}$.

□

Proof of Lemma 11:

Proof. Say $V = \sum_{i=1,...,K} v_i$ where v_i is a binary vote from middle relay i indicating if a given pair is an outlier or not an outlier.

If the pair is an outlier, the expected value of K reported votes is $(1-\beta')K$. This is correctly identified when q votes are misreported if $V \geq T + q$. By Chernoff Bound:

$$P(V \leq (1+\alpha)(1-\beta')K) \leq e^{-\frac{\alpha^2(1-\beta')K}{2+\alpha}}$$

Setting $(1+\alpha)(1-\beta')K = T + q$ we obtain the required bound:

$$P(V > T+q) \geq 1 - e^{-\frac{\left(\frac{T+q}{(1-\beta')K}-1\right)^2(1-\beta')K}{2+\left(\frac{T+q}{(1-\beta')K}-1\right)}}$$

□

A.2 Further Experimental Details for RW-Tor

Our approach was to run 120 Tor clients (30 in each of the four regions) in parallel, one per virtual machine deployed on Amazon Web Services, and construct and record successful circuits over the course of one hour. This process was approved by the Tor research safety board and our University ethics board.

We utilized Tor version 4.0.5 (May 2, 2019) which we built from source. Our circuit collection script interacted with the running Tor client through the Tor controller port using the STEM library. We constructed circuits as follows:

1. We allow Tor to start up
2. We download the consensus through STEM and save it for future reference.
3. We drop the guards the client knows about so far, thus forcing the Tor client to re-pick a guard(s) for future circuit creation requests.
4. We create 1 circuit, logging each successful circuit (i.e. three hops) built. The next circuit is requested after the we log the previous one.
5. We repeat the last two steps 3600 times.
6. Finally we download the consensus again and store it for future reference before ending the experiment.

Our 120 Tor clients attempted to construct 378943 circuits. However, in reality circuits do not always build successfully (i.e. circuits of length less than three) within the 1 second between the circuit creation request and the logging. Removing those we captured 353331 circuits (1-2pm on 20/05/2020).

Circuits were created using Stem; specifically, the via the module Controller. Note that our *torrc* configuration uses the default settings, except that we open a control port (un-commenting the line *ControlPort 9051*). We observed that the percentage of failed circuits was ≈ 6.7% on average across all the regions.

A.3 Binning Algorithm PseudoCode

```
Data: Data size $n$, probabilities of exits.
      Parameters $\gamma$, $\eta$, $m$
Result: Result a histogram of counts in bins.
S=sorted($x \in X$) (exits descending order) ;
k=0 (bin number);
for $s \in S$ in descending order do
    if ($\exists x \in B_k : E_x \geq (1+\gamma)E_s + \eta$) or
      $|B_k| \geq m$ then
        $k = k + 1$ (move to next bin);
        $B_k = B_k \cup \{s\}$ (initialize with the
          current element);
    else
        $B_k = B_k \cup \{s\}$ (add to current bin)
    end
end
```

Algorithm 1: Binning algorithm

A.4 PBA under various compromise rates

	Retries per circuit, r		
	10%	20%	50%
pb-0.02	29	56	201
pb-0.2	2	4	20

Table 2. Number of retries per circuit, r, required to compromise 10%, 20%, 50% of total circuits seen by the large and small adversary.

A.5 Simulation Results

	Datasize (mil.)					
	10	**50**	**100**	**200**	**500**	**1000**
Medium BW Guard						
Medium BW Exit	**0.97**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**
Low BW Exits	**0.65**	**0.65**	**0.66**	**0.68**	**0.98**	**0.99**
Low BW Guards						
Medium BW Exit	**0.67**	**0.73**	**0.77**	**0.81**	**0.84**	**0.86**
Low BW Exits	**0.61**	**0.61**	**0.61**	**0.61**	**0.61**	**0.61**

Table 3. F1 Score vs datasize with $\epsilon = 0.1$, $\phi = 10, \lambda = 150$.

	Datasize (mil.)					
	10	**50**	**100**	**200**	**500**	**1000**
High Resource Adversary - Low BW Guards						
High BW Exit	**0.66**	**0.76**	**0.80**	**0.83**	**0.84**	**0.86**
Medium BW Exits	**0.66**	**0.73**	**0.77**	**0.80**	**0.78**	**0.84**
Low BW Exits	**0.65**	**0.70**	**0.72**	**0.72**	**0.77**	**0.80**
Medium Resource Adversary - Low BW Guards						
Medium BW Exit	**0.66**	**0.78**	**0.83**	**0.85**	**0.85**	**0.88**
Low BW Exits	**0.63**	**0.68**	**0.69**	**0.69**	**0.76**	**0.79**

Table 4. F1 Score vs datasize with $\epsilon = 0.1$, $\phi = 10, \lambda = 150$.