# $Re^3$: $Re$lay $Re$liability $Re$putation for Anonymity Systems

Anupam Das, Nikita Borisov, Matthew Caesar
University of Illinois at Urbana-Champaign
{das17,nikita,caesar}@illinois.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

## ABSTRACT

To conceal user identities, Tor, a popular anonymity system, forwards traffic through multiple relays. These relays, however, are often unreliable, leading to a degraded user experience. Worse yet, malicious relays may strategically introduce deliberate failures to increase their chance of compromising anonymity. In this paper we propose a reputation system that profiles the reliability of relays in an anonymous communication system based on users' past experience. A particular challenge is that an observed failure in an anonymous communication cannot be uniquely attributed to a single relay. This enables an attack where malicious relays can target a set of honest relays in order to drive down their reputation. Our system defends against this attack in two ways: first, we use an adaptive exponentially weighted moving average (EWMA) that ensures malicious relays adopting time-varying strategic behavior obtain low reputation scores over time, and second, we propose a filtering scheme based on the reputation score that can effectively discard relays involved in such attacks.

We use probabilistic analysis, simulations, and real-world experiments to validate our reputation system. We show that the dominant strategy for an attacker is to not perform deliberate failures, but rather maintain a high quality of service. Our reputation system also significantly improves the reliability of path construction even in the absence of attacks. Finally, we show that the benefits of our reputation system are realized after only a moderate number of observations, making it possible for individual clients to perform their own profiling, rather than relying on an external entity.

## Keywords

Anonymity, Reputation Model, Tor Network, DoS attack.

## 1. INTRODUCTION

Anonymous communication systems play a vital role in protecting users against network surveillance and traffic analysis. The widely-used Tor network [27] has approximately 4 500 relays and serves an estimated 400 000 unique users in a day, as of November, 2013[1]. The effectiveness of Tor depends on the reliability of these relays. Unreliable relays can both degrade the user experience and impair the anonymity guarantees, as certain users will decide to abandon the system, decreasing the anonymity set, and the remaining users will end up retransmitting messages, presenting further opportunities for observation. This latter problem can be exploited by malicious relays by strategically affecting the reliability of communications paths to increase their odds of compromising user anonymity [19, 20]. Detecting such selective denial-of-service (DoS) attacks is challenging, as an observed failure in an anonymous path cannot be uniquely attributed to a single relay. Given previous instances of active attacks on Tor [3, 9, 13], as well as recent governmental endeavors [1, 4] to deanonymize Tor users, make identification of active attackers in anonymity systems is an important problem to address.

We propose a new reputation model, $\boldsymbol{Re^3}$, that can be used to detect and penalize relays involved in selective DoS. The main challenge in building our model is that it is hard to pinpoint the single relay responsible for an observed failure. This enables an attack where malicious relays can target a set of honest relays in order to drive down their reputation. Moreover, malicious relays can oscillate between good and bad behavior in order to evade detection. To address these challenges we propose using an exponentially weighted moving average (EWMA) that can dynamically adjust its weighting co-efficient to capture the dynamic behavioral trend of a given Tor relay. $\boldsymbol{Re^3}$ ensures that a malicious relay who oscillates between reliable and unreliable states obtains low reputation score overtime. We then propose a filtering protocol based on relays' reputation score that can effectively discard relays mounting active attacks.

We analyze the security of our filtering protocol both probabilistically and through a prototype deployment in the live Tor network. We find that attackers gain no advantage through active attacks like selective DoS with $\boldsymbol{Re^3}$ deployed. We also show that our filtering scheme is not vulnerable to strategic attacks like the *targeted attack* and a particularly serious form of targeted attack known as the "*creeping death attack*" [28]. Furthermore, we study adaptive attackers who tailor their strategy specifically for our detection scheme, performing active dropping only if their reputation is above a chosen threshold. We conclude that with $\boldsymbol{Re^3}$ deployed the dominant strategy for such attackers is to not perform any circuit dropping. Finally, we show that our reputation model provides benefits even outside the context of active attacks, and is able to substantially increase the reliability of circuit construction in Tor.

**Contributions.** We offer the following contributions:

- We present a reputation system that assigns quantitative scores to relays based on the reliability that they provide during anonymous communications. Our reputation system captures dynamic behavioral change by relays and penalizes relays exhibiting be-

[1]https://metrics.torproject.org

havioral oscillation.

- We probabilistically analyze the security of our filtering protocol against the selective DoS attack, including its randomized variants. We also study strategic attacks against our reputation model such as the targeted attack and creeping-death attack.
- We perform simulation and experiments on the live Tor network to demonstrate that our reputation model can effectively filter out compromised relays.
- We demonstrate the benefits of our approach even outside the context of active attacks. Using real world experiments on the Tor network, we find that our filtering protocol is able to significantly improve the reliability of circuit construction.
- We present two strategies to incorporate our reputation model into Tor. One way is to run it locally at individual clients and the other is to run it at shared directory authority (DA) servers.

# 2. BACKGROUND

In this paper we take Tor, one of the most widely used low-latency anonymity system, as a case study to profile its participating relays. So, we present a brief overview of the Tor network, and then discuss how active attacks can lower anonymity in Tor. We also briefly discuss different types of reputation systems.

## 2.1 Tor: A Low-latency Anonymity Network

To anonymize TCP connections, a Tor user constructs a *circuit* comprised of several Tor *relays* (also known as routers). The relays form a pipeline through which traffic is forwarded back and forth between the user and destination. Circuits typically involve three relays: the *entry*, *middle*, and *exit*. Tor protects the contents of the traffic by using a layered encryption scheme [38], where each relay decrypts a layer while forwarding. As a result, any individual router cannot reconstruct the whole circuit and link the source to the destination. The relays in a circuit are chosen using specific constraints [25]. Each user selects the *entry* relay from a small, fixed number of relays that are flagged as "fast" and "stable". These relays are called *guard relays* [44]; their use is designed to defend from the predecessor attack [45]. To choose the exit relay, the user picks from among those relays that have an exit policy compatible with the desired destination. After these constraints, the relays for each position are chosen randomly, weighted by their bandwidth.[2]

Tor aims to provide low-latency traffic forwarding for its users. As a result, as traffic is forwarded along the path of a circuit, timing patterns remain observable, and an attacker who observes two different relays can use timing analysis to determine whether they are participating in the same circuit [33, 41, 43, 47]. Thus to compromise anonymity it suffices to observe the entry and the exit relays for a circuit. Standard security analysis of Tor [27,43] shows that if $c$ is the fraction of relays that are observed, an adversary can violate anonymity on $c^2$ of all of the circuits. Due to bandwidth-weighted path selection in Tor, $c$ is best thought of as the fraction of total Tor *bandwidth* that belongs to relays under observation[3]. The security of Tor, therefore, relies on the assumption that a typical adversary will not be able to observe a significant fraction of Tor relays. For most adversaries, the easiest way to observe relay traffic is to run their own relays. It should be noted that other forms of adversaries do exist, such as ISP-level adversaries, and Internet exchange-level adversaries [14,29,36], but these adversaries are typically assumed to be passive and are thus not the focus of this paper.

[2]This is a simplified description of the path selection; a detailed specification can be found at [26]. The omitted details do not significantly impact our analysis, and we use the full specification in our experiments.

[3]To be more precise, the correct fraction would be $c_g \cdot c_e$, where $c_g$ and $c_e$ are the fractions of the guard and exit bandwidth under observation, respectively. For simplicity of presentation, we will assume $c_g = c_e = c_m = c$ in the rest of the paper.

## 2.2 Active Attack: Selective DoS in Tor

An adversary who controls a Tor relay can perform a number of active attacks to increase the odds of compromise [19,20]. One approach is selective denial-of-service (DoS) [20]. A compromised relay that participates in a circuit can easily check whether both the entry and exit relays are under observation. If this is not the case, the relay can "break" the circuit by refusing to forward any traffic. This will cause a user to reformulate a circuit for the connection, giving the adversary another chance to compromise the circuit. A simple analysis shows that this increases the overall fraction of compromised circuits to: $\frac{c^2}{c^2+(1-c)^3} > c^2$, because only circuits with compromised entry and exit relays ($c^2$) or circuits with no compromised relays ($(1-c)^3$) will be functional, and out of those $c^2$ will be compromised. For example, if 20% of the bandwidth is controlled by an adversary (i.e., $c = 0.2$) then the selective DoS attack nearly doubles the overall fraction of compromised circuits from 4% to 7.2%.

The use of guard relays changes the analysis somewhat. If none of a user's guards are compromised, then the user is effectively immune from the selective DoS attack, since the user will never use a compromised entry regardless of the attack. If, on the other hand, one or more of the guards are malicious then the user is significantly impacted, as the dishonest guard(s) chosen for a significant fraction of all circuits will break any circuit that does not use a compromised exit. For $c = 0.2$, if one of the guards is compromised then the selective DoS attack increases the overall fraction of compromised circuits from 6.7% to 13.5% and for two compromised guards this value increases from 13.3% to 38.5%. Therefore, guard relays mitigate the selective DoS attack in that it will affect fewer users if they choose honest guards, but can adversly affect users who are unlucky enough to choose dishonest guards.

## 2.3 Reputation Models

A reputation model [39] collects, aggregates, and distributes feedback about participants' past behavior. Reputation models help users decide whom to trust, encourage trustworthy behavior, and discourage participation by users who are dishonest. Reputation models can be classified as either *local* or *global*, based on the way information is aggregated [35]. In a local reputation model, feedback is derived only from direct encounters (first-hand experience) whereas in a global reputation model feedback is also derived indirectly (second-hand evidence) from other users. Hence, in the case of a global reputation model [31,40,46], a user aggregates feedback from all users who have ever interacted with a given participant, thus enabling it to quickly converge to a better decision. However, global reputation models are much more complex to manage than local approaches as malicious users have the opportunity to provide false feedback. Our focus is on building a local reputation model that accumulates only first-hand experiences about Tor relays.

# 3. $Re^3$: OUR REPUTATION MODEL

Our goal is to construct a local reputation model that can be used by a Tor user to filter out less reliable Tor relays. This section discusses the different components of our model.

## 3.1 Reputation Score

To evaluate the reputation of a given Tor relay, a user keeps track of its own local experience with the relay through a modified form of exponentially weighted moving average (EWMA). EWMA

combines both recent and historical evaluations of a chosen feature which enables it to compute a representative evaluation of the feature under question. However, a conventional EWMA assigns fixed weights to recent and historical evaluations and as a result it fails to capture any oscillating behavior. In our system we are considering strategic relays, capable of altering their behavior in any way that may benefit them. Aringhieri et al. [17] proposed dynamically modifying the weighting co-efficient of EWMA to obtain a better evaluation, but they do not consider strategic oscillations where malicious users oscillate between building and milking their reputation. Srivatsa et al. [42] proposed a PID-controller based reputation framework which can handle strategic malicious behavior. We adopt a similar approach where we use PID-controller based techniques [16] to dynamically change the weighting coefficient of our adaptive EWMA in such a way that it penalizes any relays exhibiting such oscillations. Figure 1 shows an overview of how we compute a relay's reputation value.

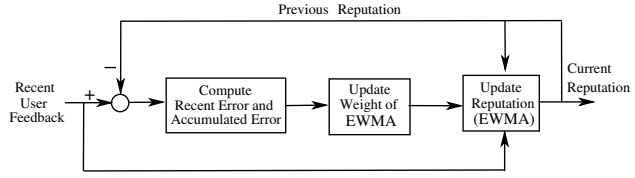



**Figure 1:** Block diagram of our reputation model. A feedback loop is used to model the dynamic behavior of a Tor relay. We dynamically adjust the weighting coefficient of our EWMA based on a relay's behavior.

Let $\boldsymbol{R_n(x)}$ represent the local reputation of relay $\boldsymbol{x}$ after $\boldsymbol{n}$ interactions (where $\boldsymbol{R_n \in [-1, 1]}$). The local reputation update function is defined as follows:

$$\boldsymbol{R_n(x) = \alpha_n(x) \times R_c + [1 - \alpha_n(x)] \times R_{n-1}(x)} \quad (1)$$

with $\boldsymbol{R_0(x) = 1}$; i.e., all relays are initially assumed to be good because we want all relays to be initially usable. Here, $\boldsymbol{R_c}$ represents the rating of the most recent experience. For simplicity we have used a binary rating system where a user rates a relay based on whether the circuit built through that relay was usable or not.

$$\boldsymbol{R_c = \begin{cases} -1, & \text{if circuit failed} \\ 1, & \text{otherwise} \end{cases}} \quad (2)$$

The weighting coefficient $\boldsymbol{\alpha_n(x)}$ determines the degree to which we want to discount older observations. To penalize relays oscillating between good and bad behavior, we keep track of accumulated deviation $\boldsymbol{\xi_n(x)}$ from recent error $\boldsymbol{\delta_n(x)}$, and lower reputation proportionally (with proportional constant, $\boldsymbol{K_p}$). The update function of $\boldsymbol{\alpha_n(x)}$ is given as follows:

$$\boldsymbol{\alpha_n(x)} = \boldsymbol{K_p \times \frac{\delta_n(x)}{1 + \xi_n(x)}} \quad (3)$$

$$\boldsymbol{\delta_n(x)} = \boldsymbol{\begin{cases} \frac{R_c(x) - R_{n-1}(x)}{\mu}, & \text{if } R_c(x) \geq R_{n-1}(x) \\ \frac{R_{n-1}(x) - R_c(x)}{\nu}, & \text{if } R_c(x) < R_{n-1}(x) \end{cases}} \quad (4)$$

$$\boldsymbol{\xi_n(x)} = \boldsymbol{\xi_{n-1}(x) + \delta_n(x)} \quad (5)$$

Here proportional constant, $\boldsymbol{K_p (0 \leq K_p \leq 1)}$ controls to what extent we want to react to recent error ($\boldsymbol{\delta_n(x)}$) compared to accumulated deviation ($\boldsymbol{\xi_n(x)}$). We set $\boldsymbol{K_p = 0.5}$ after performing a sensitivity analysis (please see Appendix-B for more details).

As stated earlier we want our reputation function to penalize oscillating behavior and to achieve that we update the error function $\boldsymbol{\delta_n(x)}$ using a reward and punishment strategy where we reward relays for successful communication and punish them for unsuccessful communication. Incorporating such a strategy enforces relays to behave faithfully. In equation (4), $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ represent the reward and punishment factor respectively. Both $\boldsymbol{\mu, \nu \in \Re}$, but we must ensure that $\boldsymbol{\mu > \nu}$ because the impact of punishment should be greater than that of reward. In other words, $\boldsymbol{\delta_n(x)}$ should increase more for unsuccessful communication than successful communication, because that would in turn increase $\boldsymbol{\alpha_n(x)}$, giving higher significance to the recent bad evaluation.

To get an understanding of how $\boldsymbol{Re^3}$ reacts to different scenarios like random network failures or strategic oscillating behavior, we evaluate the reputation score of a relay dropping circuits at different rates. Figure 2 shows the evaluated reputation score for different characteristics. We see that any form of circuit dropping results in lowering reputation. Even strategic oscillating behavior (i.e., strategically building and milking reputation) is punished *severely* and this is evident from the lower reputation score for the same drop rate. For example, oscillating at 50% drop rate results in a lower reputation score compared to 50% random drop rate. Hence for a relay to achieve good reputation, it will have to provide good service consistently (more results available in Section 6.1.3). Appendix-B describes how different parameter choices affect our reputation model. We also analyze the stability of our reputation model in Appendix-A.

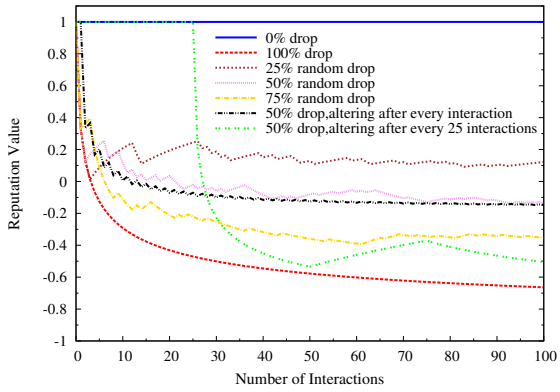



**Figure 2:** Dynamic behavior of our reputation metric under different combinations of cooperative and malicious behavior. We can see that different dropping attacks result in lowering reputation.

## 3.2 Confidence Factor

"The more you interact with a relay the more confident you become about that relay's behavior". Following this principle we associate a confidence factor with our local reputation score. In other words, if a user $\boldsymbol{A}$ performs $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ (where $\boldsymbol{n_1 > n_2}$) interactions with relay $\boldsymbol{B}$ and $\boldsymbol{C}$ respectively, then the reputation score reported for relay $\boldsymbol{B}$ is more close to its stable score than that reported for $\boldsymbol{C}$. We, therefore, formulate the confidence factor as a monotonically increasing function of the number of interactions with a given relay. We use the following confidence metric:

$$\boldsymbol{C_n(x) = \beta^{\frac{1}{n}}} \quad (6)$$

where $\boldsymbol{n}$ represents the number of times a user has communicated through relay $\boldsymbol{x}$. Here, $\boldsymbol{\beta}$ $\boldsymbol{(0 < \beta < 1)}$ is a user defined variable which determines how quickly the user becomes confident about his/her rating (sensitivity analysis is provided in Appendix-B).

## 3.3 Ranking Score

The ranking score reflects how well a given relay has been performing so far. To obtain a high ranking score a relay will have to persistently exhibit good behavior for a long period of time. We formulate the final ranking score of a relay by multiplying the rep-

utation score with the confidence metric.

$$\boldsymbol{Rank_n(x) = R_n(x) \times C_n(x)} \quad (7)$$

This ensures the ranking score of a relay is high only when both its reputation score and confidence value is high. A relay with high reputation score but low confidence value will result in an overall low ranking score. This hinders whitewashing attack.

# 4. FILTERING PARTICIPANTS IN TOR

In this section we describe how $\boldsymbol{Re^3}$ can be used to filter out potentially compromised relays. As mentioned above, $\boldsymbol{Re^3}$ matures as the user gathers more experience and by assuming that only a small fraction of all relays are compromised, we can probabilistically show that the average reputation of honest relays will be higher than that of compromised ones. So to filter out potentially compromised relays, we only need to find outliers in terms of ranking score. Our filtering protocol assumes $\gamma$ fraction of the relays are potentially outliers (ideally $\gamma$ is equivalent to the fraction of compromised relays in the system). Now, to filter outliers a client takes the following steps:

- First, the client computes the average ($\boldsymbol{\mu}$) and standard deviation ($\boldsymbol{\sigma}$) of the top (in terms of ranking score) $\boldsymbol{1-\gamma}$ fraction of the relays he/she has interacted with.
- Filters out any relay $\boldsymbol{x}$ with $\boldsymbol{|Rank(x) - \mu| > k \times \sigma}$ as outlier. Here, $\boldsymbol{k}$ represents to what degree of deviation we are willing to tolerate from the expected ranking score. We filter outliers in both directions because when large fraction of the guards are compromised, compromised exits tend to obtain a higher ranking score (as majority of the circuits have a compromised guard in such scenario) compared to the other honest relays.

From a security perspective, we are interested in cases when clients have some compromised and some honest guards. In such cases, we can adopt the following two strategies with respect to selecting guards:

- **Strategy 1**: Consider all guards that are not outliers.
- **Strategy 2**: Consider only the highest ranked guard.

The reason behind using strategy 2 is that if 1 or 2 of the guards are compromised then their reputation score should be lower than that of the honest ones, so strategy 2 helps to filter out potentially compromised guard relays. We evaluate both strategies later in Section 6. We want to point out that clients consider the filtered list of Tor relays (after profiling a large set of Tor relays) for future circuit construction following Tor's conventional bandwidth-proportional path construction protocol (potentially the reputation score of relays could be considered as one of the decision parameters). Another point, in Tor, Sybil attack is frustrated by bandwidth. If an adversary tries to use different IP addresses from the same 'subnet' we could potentially flag the whole subnet as suspicious. Furthermore, our confidence metric thwarts an adversary from changing its pseudonym because in that case the adversary has to *again* participate in multiple circuits to obtain any stable reputation.

# 5. PROBABILISTIC ANALYSIS

In this section we probabilistically analyze the effectiveness of $\boldsymbol{Re^3}$ in filtering out compromised relays under four adversarial strategies. Filtering is done based on the ranking score of relays, which in turn is computed based on the fraction of times a relay receives positive (+ve) and negative (-ve) feedback from a user (see equation (2)). Hence, we conduct our analysis by computing the probability of receiving positive and negative feedback for different types of relays. First, we analyze the selective DoS attack scenario. Next, we analyze the impact of targeted attack against $\boldsymbol{Re^3}$; followed by an analysis of creeping-death attack. Finally, we analyze random drop strategy where the adversary randomly drops communication with the aim to remain undetectable. To carry out our probabilistic analysis we consider the following parameters:

**Table 1:** Probabilities under selective DoS attack

| Relay | Probability of Positive (+ve) Feedback | Probability of Negative (-ve) Feedback |
|---|---|---|
| Honest | $\frac{1}{2}gc + (1-g)(1-c)$ | $1 - \frac{1}{2}gc - (1-g)(1-c)$ |
| Compromised | $\frac{1}{2}(gc + g)$ | $1 - \frac{1}{2}(gc + g)$ |

- $\boldsymbol{g}$: fraction of guard relays per user that are compromised (by default each user has 3 guard relays).
- $\boldsymbol{g'}$: fraction of guard relays per user that are targeted (for targeted attack analysis only).
- $\boldsymbol{c}$: fraction of other relays in the network that are compromised.

Our probabilistic computation assumes that different types of relays can appear only in the middle and exit position of a Tor circuit, as guard relays are preselected by users and they are changed only after a period of 30 to 60 days (uniformly chosen). For the following analyses we give more emphasis to scenarios where $\boldsymbol{g = 1/3, 2/3}$, as $\boldsymbol{g = 0, 1}$ are trivial scenarios.

## 5.1 Analysis of Selective DoS Attack

We start by computing the fraction of positive and negative feedback that a given Tor relay will obtain when compromised relays are carrying out selective DoS attack. For example, a compromised exit relay will only allow a given circuit to continue if the guard relay is also compromised. So, the probability of receiving positive feedback is at most $\boldsymbol{g}$ for a compromised exit relay. We can similarly calculate the other probabilities. Table 1 summarizes the different probabilities.

Now our filtering protocol can successfully filter compromised relays only if they are outliers in the reputation spectrum. We can approximate reputation scores as the difference between the fraction of positive and negative feedback. For $\boldsymbol{g = 1/3}$, we see that the reputation of compromised relays is less than that of honest relays for any value of $\boldsymbol{c < 0.75}$. But for $\boldsymbol{g = 2/3}$ we see that the reputation score of compromised relays is higher than that of honest relays, however, if we assume compromised relays as a minority group then they become outliers compared to honest relays making it possible for $\boldsymbol{Re^3}$ to filter them. More analysis is available in Section 6.1.2.

## 5.2 Analysis of Targeted Attack

In this section we analyze the impact of targeted attack against $\boldsymbol{Re^3}$. Here we assume that the set of compromised relays target a set of honest relays (preferably high bandwidth relays) and try to frame them as compromised. The attack strategy is defined as follows: *"Whenever a compromised relay discovers that a targeted relay lies on the circuit, it kills the circuit forcing the user to lower the reputation score of the targeted relay"*. Now lets assume $\boldsymbol{c}$ fraction of the compromised Tor relays target $\boldsymbol{t}$ fraction of the honest relays. Intuitively this scheme will work only if $\boldsymbol{c > t}$, otherwise both the compromised and targeted set of relays will have similar low reputation scores causing them to be filtered out. Table 2 summarizes the probabilities of receiving positive and negative feedback for the different types of relays (e.g., if a compromised relay is in the middle position of a Tor circuit then it will let the circuit

continue only if the adjacent relays are not in the targeted set, this corresponds to $\frac{1}{2}(1-g')(1-t)$, one of the terms in the table)[4].

**Table 2:** Probabilities under targeted attack strategy

| Relay | Positive Feedback(+ve) | Negative Feedback(-ve) |
|---|---|---|
| Target | $\frac{1}{2}[(1-c)(2-g)-gt]$ | $1-\frac{1}{2}[(1-c)(2-g)-gt]$ |
| Compromised | $\frac{1}{2}[(1-t)(2-g')-g'c]$ | $1-\frac{1}{2}[(1-t)(2-g')-g'c]$ |
| Other | $\frac{1}{2}[2-g'c-gt]$ | $\frac{1}{2}[g'c+gt]$ |

Now we want to check if our filtering protocol can identify relays mounting targeted attacks. To do so we need to ensure that the reputation score of compromised relays is outside the acceptable region, i.e., outside $(\mu - k \times \sigma, \mu + k \times \sigma)$. First, we approximate the reputation score of the different types of relays. Next we compute the average ($\mu$) and standard deviation ($\sigma$) of the top $1-\gamma$ fraction of the relays. Finally, we determine the value of $k$ for which the reputation score of compromised relays is outside $(\mu - k \times \sigma, \mu + k \times \sigma)$.

Lets first approximate the average reputation score of different types of relays. Since we use a binary rating system (see equation (2)), we can approximate the average reputation score of different types of relays using the following equations:

$$R_T = \Pr(+ve_{fb})_{target} - \Pr(-ve_{fd})_{target}$$
$$R_C = \Pr(+ve_{fb})_{compromised} - \Pr(-ve_{fd})_{compromised}$$
$$R_O = \Pr(+ve_{fb})_{other} - \Pr(-ve_{fd})_{other}$$

Here the probabilities represent the fraction of the feedbacks that are positive and negative. $R_T$, $R_C$ and $R_O$ refers to the reputation score of relays belonging to the targeted, compromised and non-targeted honest set respectively. From Table 2, we see that for $c > t$, the reputation score of a compromised relay is greater than that of a targeted relay. But both of their reputation score are much lower than that of a non-targeted honest relay, i.e., $R_O > R_C > R_T$. If we assume that $c$ is still a minority group, we can use this discrepancy to filter out the malicious relays from the selection.

A client starts by ignoring a fraction $\gamma$ of the relays with the lowest reputation and then computes the average and standard deviation of the remaining reputation scores. If $c < \gamma < c+t$, then this computation will exclude all of the targeted relays but include a fraction of the malicious relays:

$$\mu = \frac{(1-t-c)\cdot R_O + (c+t-\gamma)\cdot R_C}{1-\gamma}$$
$$\sigma = \sqrt{\frac{(1-t-c)\cdot(\mu - R_O)^2 + (c+t-\gamma)\cdot(\mu - R_C)^2}{1-\gamma}}$$

The client only keeps the relays whose reputation score lies within $k$ standard deviations from the mean, i.e., in the interval $(\mu - k\sigma, \mu + k\sigma)$. Some calculations show that if we set $\gamma = 0.2$ and $k < \sqrt{3}$, then $R_C$ will fall outside this interval. We tested for both $g$ and $g'$, taking their values from the set $\{1/3, 2/3\}$.

Figure 3(a) shows the probabilities of receiving positive and negative feedback for different types of relays when $c = 0.2$. As we can see, the targeted relays suffer the most under this attack scenario. This trend becomes more clearer from Figure 3(b), which highlights the average reputation score of the different relays along with the acceptable upper and lower bound. Now, let us assume that the compromised relays perform targeted attacks first, and once the targeted relays are blacklisted they start performing selective DoS attack to increase their chance of being selected. Figure 3(c) shows the probability of selecting compromised circuits for different values of $k$ with different fractions of honest relays being targeted. From Figure 3(c), we can see that with the proper setting of $k < \sqrt{3}$ we can effectively filter compromised relays even when they launch a targeted attack on a small set of honest relays.

[4]Our analyses are approximations with negligible error where we are assuming replacement of relays after selection, however, with a large number of relays this introduces negligible error.

**Table 3:** Probabilities of under creeping-death attack strategy

| Relay | Positive Feedback(+ve) | Negative Feedback(-ve) |
|---|---|---|
| Honest | $(1-g)(1-c)+gc$ | $1-(1-g)(1-c)-gc$ |
| Compromised | $g+c-gc$ | $1-g-c+gc$ |

Thus, we can see that compromised relays do lower the reputation score of the targeted set (as evident from Figure 3(b)). However, this in turn increases the reputation score of the remaining relays (i.e., relays that are neither in the compromised or targeted set), because under this attack strategy the targeted set is much smaller compared to the vanilla selective DoS scenario where all honest relays are included in the targeted set. And since we use the average and standard deviation of the top $1-\gamma$ fraction of the relays, this raises the bar of acceptance, making it harder for compromised relays to cross the bar. If adversaries adopt the strategy to change the target set then relays in the previous target set will eventually be reconsidered for usage due to their honest nature. Thus shifting the target set does not affect the system as compromised relays will also have to wait until they buildup a good reputation.

## 5.3 Analysis of Creeping-Death Attack

Dingledine et al. defined the "*creeping death attack*" [28] as follows: "*Whenever a compromised relay discovers that majority of the relays on the circuit are honest, it kills the circuit forcing the user to lower the reputation score of the honest relays*". In Tor, circuits contain three relays, so a honest relay will succeed only if the other two participating relays are honest or compromised (with probability $(1-g)(1-c)+gc$). Table 3 summarizes the probabilities for different types of relays. Again we see that for $g = 1/3$ the the reputation score of honest relays is higher than that of compromised relays, but for $g = 2/3$ it is reverse. But assuming compromised relays are minority we can successfully filter them by setting $k < \sqrt{3}$.

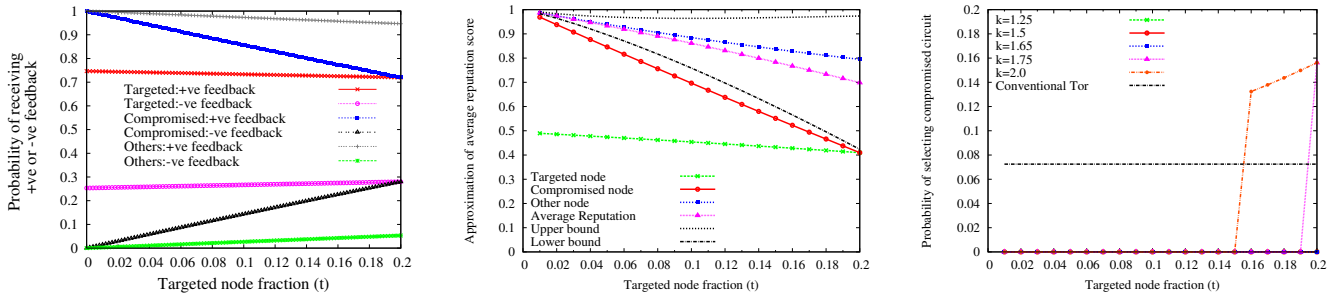
## 5.4 Analysis of Random Dropping Attack

Next we analyze the impact of random dropping strategy, a variant of selective DoS. In this scenario, compromised relays randomly drop non-compromised circuits. The benefit of random dropping can be realized through the following probability of selecting compromised circuit:

$$\Pr(CXC) = \frac{c^2}{c^2 + (1-c)^3 + [1-c^2-(1-c)^3](1-d)} \quad (8)$$

where $CXC$ represents a compromised circuit where $C$ refers to a compromised relay and $X$ refers to any type of middle relay. From equation (8), we see that as the drop rate increases the probability of selecting compromised relays also increases. *Hence, any form of dropping seems to be a beneficial strategy in the absence of* $Re^3$.

Now, let us compute the probabilities of receiving positive and negative feedback for different types of relays with $Re^3$ deployed. Table 4 summarizes the different probabilities. For $c = 0.2$, we see verify that any form of dropping results in reducing reputation, but compared to honest relays compromised relays suffer greatly for $g = 1/3$. For $g = 2/3$, the reputation score of compromised relays is greater than that of honest relays, but that makes them outliers which in turn helps our filtering scheme to easily filter them. Thus random circuit dropping ($d > 0$) becomes a losing proposal




(a) (b) (c)

**Figure 3:** For the targeted attack scenario: (a) Probabilities of receiving positive and negative feedback for different types of relays. (b) Average ranking score of different types of relays along with the acceptable upper and lower bounds. We see that the reputation score for both the targeted and compromised relays lie outside the acceptable region. (c) Probability of selecting a compromised circuit once filtering has been done. As we increase the acceptable range of reputation (by increasing $k$) the probability of selecting a compromised circuit increases.

**Table 4:** Probabilities under random drop attack strategy

| Relay | Positive Feedback (+ve) | Negative Feedback (-ve) |
|---|---|---|
| Honest | $(1-g-c+\frac{3}{2}gc)+[g+c-\frac{3}{2}gc](1-d)$ | $[g+c-\frac{3}{2}gc]d$ |
| Compromised | $\frac{1}{2}(gc+g)+[1-\frac{1}{2}(gc+g)](1-d)$ | $[1-\frac{1}{2}(gc+g)]d$ |

with $Re^3$ deployed.

# 6. EXPERIMENTAL EVALUATIONS

In this section, we present a series of simulation and real-world experimental results, all of which highlight the effectiveness of $Re^3$ in filtering unreliable/compromised relays under real-world settings. We first look at the false positive and false negative errors in our filtering scheme. Next, we evaluate the probability of selecting compromised circuits. Finally, we run experiments over the live Tor network and find that there is significant difference in the application and network level reliability of individual Tor relays. All these findings help us conclude that $Re^3$ can assist users in selecting more reliable and well-behaved Tor relays.

## 6.1 Simulation Results

### 6.1.1 Simulation Setup

We implemented a simulator that emulates the basic functionality of Tor circuit construction and active circuit dropping. We collected actual relay information (such as- IP address, advertised bandwidth and probability of the relay being selected for entry, middle and exit position) from the Tor compass project [6] and randomly tagged 20% of the bandwidth to be controlled by an adversary, i.e., in our threat model we assume $c = 0.2$. For our experimental setup we consider 3 guards, 23 middle relays and 23 exits where each relay belongs to a distinct /16 IP subnet. The reason behind using 23 middle and exit relays is that we assume a user uses Tor for three hours continuously[5] and since a given circuit is alive for only 10 minutes, a user would need at most 18 circuits in a 3 hour period, i.e., at most 18 different middle and exit relays. Since we assume 20% of the available bandwidth is controlled by a malicious entity, on average we need $18/0.8 \approx 23$ relays to build 18 non-compromised circuits. We create a total of $3 \times 23 \times 23 = 1587$ circuits (each tested exactly once) to determine the reputation of the selected relays.

Table 5 summarizes the parametric settings for our simulation. We vary two environmental parameters $(g, d)$ to analyze the robustness and effectiveness of $Re^3$ against active circuit dropping attacks. Here, 100% drop rate refers to selective DoS and 0% drop means no dropping at all. In the following evaluations we again give more emphasis to results for $g = 1/3, 2/3$, as $g = 0, 1$ are trivial scenarios. Regarding guard selection strategy, our default strategy is to use all guards that are not outliers (i.e., strategy 1 as described in Section 4). To approximate the circuit failure rate present in the current Tor network we utilize the TorFlow project [11]. The TorFlow project measures the performance of Tor network by creating Tor circuits and recording their failure rate. We run TorFlow's *buildtime.py* [11] python script to generate 10 000 Tor circuits and record their failure rate. We found the average failure rate over 10 runs to be approximately 21%. Thus, we set the circuit failure rate, $f$ to 0.21 in all our simulations. All simulation results are averaged over 100 000 runs with 95% confidence interval (with 100 000 runs the confidence intervals are too small to see in some cases).

**Table 5:** Simulation Parameters

| | Parameter | Description | Value/Range |
|---|---|---|---|
| Computation Setting | $K_p$ | Proportional gain | 0.5 |
| | $\mu$ | Rewarding factor | 2 |
| | $\nu$ | Punishment factor | 1 |
| | $\beta$ | Confidence co-efficient | 0.5 |
| Environment Setting | $g$ | Fraction of compromised guards | $[0, 1/3, 2/3, 1]$ |
| | $d$ | Drop rates for compromised relays | $0 \leq d \leq 1$ |
| | $f$ | Transient network failure | 0.21 |

### 6.1.2 Filtering Compromised Relays

To compare the ranking score of honest relays with that of compromised relays, we set $d = 1$ and compute the ranking score of all the relays by varying $g$. Figure 4 shows the ranking score of both honest and compromised relays for different numbers of compromised guards. As discussed in Section 4, we set cutoffs based on the average ranking score of the top $(1-c)$% or 80% ranked relays. To filter outliers we exclude relays that are further than $\sqrt{3}$ standard deviations away from the average (i.e., $k = \sqrt{3}$; we analyze the sensitivity of $k$ in Appendix-C). The dotted/dashed lines in the figure represent the boundaries for acceptable region $(\mu - k\sigma, \mu + k\sigma)$. Figure 4 shows that as the number of compromised guards increases the distinction between honest and compromised relay shrinks. This is understandable because as the number of compromised guards increase, the ranking score for compromised relays also start to increase because more and more circuits with compromised guards are created. However, since honest re-

[5]Tor users download the Tor consensus data every three hours so it would make sense to refresh the relay list every three hours.

lays dominate the total population, the average reputation score of the system lies close to the average reputation score of the honest relays. As a result, even with $g = 2/3$ we can successfully filter out a significant portion of the compromised relays.

### 6.1.3 *Evaluating Robustness*

In this section we present results that show the robustness of our reputation model in the presence of compromised relays. First, we look at false positive and false negative errors of our filtering protocol, and then we evaluate the probability of constructing compromised circuits under different strategic scenarios.

**False errors of our filtering scheme:** We define false negative (FN) and false positive (FP) error as follows:

- **FN:** Fraction of compromised relays in the accepted list.
- **FP:** Fraction of honest relays in the discarded list.

Figure 5 highlights the calculated FN and FP errors. Ideally you want both false errors to be low but since compromised relays are a minority and honest relays are plentiful, lowering FN is *more important* than lowering FP. Figure 5 shows that as the drop rate $d$ increases, FN decreases (except for $g = 1$, which is not interesting to look at as all the guards are compromised). We see a similar trend for FP, where the FP error decreases as the drop rate $d$ increases. This is expected because as the drop rate increases the distinction between compromised and honest relays becomes more clearer. So, whether honest relays are heavily penalized (for $g \geq 2/3$) or rewarded (for $g \leq 1/3$), the average ranking score of the relays in the system shifts towards the ranking score of honest relays as the majority of the relays are assumed to be honest. That is why we see FP error reduce to almost 5%. These results indicate that carrying out active attacks like selective DoS is a losing proposal for an attacker.

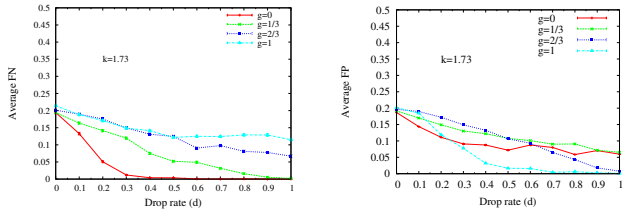



**Figure 5:** Average FN and FP with 95% confidence interval against drop rate $d$. Both FN and FP decrease as drop rate increases.

**Probability of constructing compromised circuits:** Next, we evaluate the probability of constructing a compromised circuit once outliers have been discarded. The probability of such an event is:

$$\frac{g_f c_f}{g_f c_f + (1-g_f)(1-c_f)^2 + (1-d)[1 - g_f c_f - (1-g_f)(1-c_f)^2]} \quad (9)$$

where $g_f c_f$ refers to the fraction of circuits with a compromised guard and exit, while $(1-g_f)(1-c_f)^2$ refers to the fraction of circuits with all honest relays at each position in the circuit ($g_f$ and $c_f$ represent the fraction of compromised guards and other relays in the accepted list respectively). Now we evaluate this probability under both guard selection strategies as outlined in Section 4. Figure 6 shows the probability of constructing compromised circuits against drop rates ($d$) under both strategies. For $g \leq 1/3$, we see that this probability quickly decreases to almost zero as the drop rate increases, which is significantly better than what conventional Tor guarantees (indicated by the dashed lines). Even for $g = 2/3$, we see a significant improvement but compared to strategy 1, strategy 2 performs much better. The main reason behind this is that with two guards out of the three being compromised we could potentially lower the probability of constructing compromised circuits by considering only the honest available guard. This is what strategy 2 does, as it considers only the top ranked guard relay while discarding the remaining two guards during actual circuit creation. For $g = 1$, we do not see much improvement, however, $g = 1$ is a hopeless case as all the guards are already compromised. Thus, for $g < 1$ the dominant attack strategy is to perform *no* dropping.

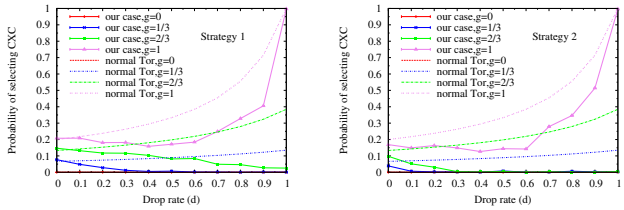



**Figure 6:** Probability of constructing compromised (CXC) circuits after filtering outliers using strategy 1 and 2. We see that our approach outperforms conventional Tor as drop rate increases.

Finally, we test our reputation model against an adversary who tries oscillate between building and milking its reputation. The adversary adopts the following strategy—*"Drop circuit only if its reputation is above a chosen threshold"*. Under this scenario we assume the adversary mimics a normal user in the system and keeps track of the reputation score of its relays so that it can optimally decide whether to drop a circuit or not. We vary the reputation threshold chosen by the adversary from -1 to +1 in increments of 0.1. In all the simulations we compute the drop rate carried out by the adversary along with the probability of constructing compromised circuits by clients. Figure 7 shows that as the reputation threshold increases the drop rate declines to zero. In other words, to obtain positive reputation the adversary cannot afford to to drop too many circuits. The probability of constructing compromised circuits ($Pr(CXC)$) rises to a stable value as drop rate declines to zero. This happens because with no dropping, the reputation score of relays remains unaffected. We therefore conclude that $Re^3$ discourages active circuit dropping by compromised relays.

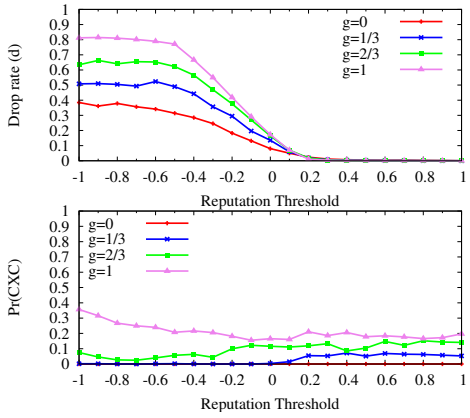



**Figure 7:** Evaluating drop rate and probability of constructing compromised circuits for various reputation thresholds. In this scenario compromised relays drop circuits only if its reputation is above the chosen threshold. We see that to retain positive reputation compromised relays cannot afford too many active circuit dropping.

## 6.2 Real World Experiments

In this section we perform a series of live experiments on the Tor network to show the effectiveness of our reputation system. First of all, we show that clients from different geographic regions successfully filter compromised relays. Next, we determine what kind

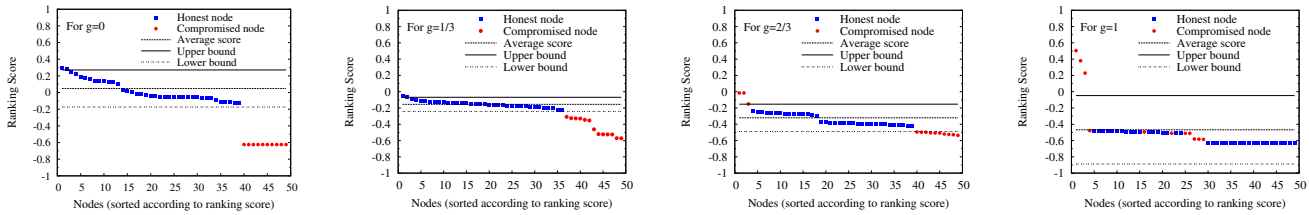



**Figure 4:** Ranking score of honest and compromised relays for various fractions of compromised guards. relays that are $k = 1.73$ standard deviations away from the mean are considered outliers. We can see that even with $g = 2/3$ a client can successfully filter out a significant portion of the compromised relays.

of benefit our reputation system provides in terms of reliability.

### 6.2.1 Filtering Compromised relays

We carried out our experiment by introducing our own relays into the Tor network, all of which acted as compromised relays. For this purpose we used 11 Emulab [2] machines, 10 of which were configured to act as Tor relays with a minimum bandwidth capacity of 20KBps. Note that all our relays belonged to the same /16 subnet, meaning that no user would (by default) choose two of our relays in the same circuit. Moreover, to prevent other users from using our relays as exit relays, we configured our relays with a fixed exit policy (allowing connection to only specific web sites). All these measures were taken to respect user privacy. To emulate real clients we used 10 PlanetLab [5] machines from 5 different continents and had them create circuits through the Tor network.

To implement selective DoS we take an approach similar to the one described by Bauer et al. [19]. Here, out of the 11 machines, we run Tor protocol (version *tor-0.2.2.35*) on 10 of them, and used the remaining machine as a server for gathering timing information about which relay is communicating with whom and at what time. The sever performs timing analysis and informs the other 10 machines when to drop communication to carry out selective DoS. We implemented our reputation model in the client side in Python (we used the open source Python library of the Tor Controller [8]).

In our experiments we first query the Tor directory server to retrieve a list of all available Tor routers and then filter this list by considering only those routers which are flagged as *running*, *stable* and *valid*, because we want our routers to be alive and running during our experiments. We selected 39 Tor relays (3 guards, 18 exits and 18 middle relays) at random with probability proportional to their bandwidth and added our own 10 relays to this set to get a total of 49 relays (for the same reasons as described in Section 6.1.1). This choice results in about 20% of the relays being compromised[6]. To emulate user traffic, we retrieve a random web file of 300 KB in size. Each client investigates a total of $3 \times 23^2 = 1587$ circuits[7] and builds its own reputation score for those relays. Clients first filter outliers and then compute FN and FP errors.

Table 6 summarizes our findings. Real world results are quite similar to what we found in our simulation. However, some of the false positive errors are somewhat higher than what we found in simulation. One plausible reason is that some of the relays were down during our experiment, which is understandable because Tor consensus is updated every 1 hour. Also, since not all the PlanetLab machines were of the same configuration, the total time required to complete all the probings varied from machine to machine. However, similar to our simulation results, we see that as the number of compromised guards increases, FN also increases. For $g \leq 2/3$ FN is almost zero. Another observation is that clients, though geographically distant, observe a similar ranking of Tor relays. Thus, $Re^3$ succeeds in capturing the selective dropping characteristics of our deployed Tor relays.

[6]Alternatively, we could set the bandwidth of our relays such that they make up 20% of the total bandwidth

[7]Without considering the same relay at different position, because guards and exits can be middle relays, too.

**Table 6:** Experimental results from live Tor network (all values are in %)

| Continent | Country (State) | g=0 | | g=1/3 | | g=2/3 | | g=1 | |
|---|---|---|---|---|---|---|---|---|---|
| | | FN | FP | FN | FP | FN | FP | FN | FP |
| North America | US(X) | 0 | 10.3 | 0 | 10.3 | 0 | 17.9 | 15.2 | 0 |
| | US(Y) | 0 | 7.69 | 0 | 10.3 | 0 | 15.4 | 11.4 | 0 |
| South America | Argentina | 0 | 5.13 | 0 | 10.3 | 0 | 17.9 | 11.4 | 0 |
| | Brazil | 0 | 7.69 | 0 | 5.13 | 0 | 10.3 | 15.2 | 0 |
| Europe | UK | 0 | 7.69 | 0 | 7.69 | 0 | 12.8 | 15.2 | 0 |
| | Russia | 0 | 7.69 | 0 | 10.3 | 2.9 | 15.4 | 11.3 | 0 |
| Asia | India | 0 | 7.69 | 0 | 10.3 | 2.8 | 10.3 | 11.4 | 0 |
| | Singapore | 0 | 7.69 | 0 | 10.3 | 0 | 15.9 | 9.3 | 0 |
| Australia | Australia | 0 | 5.13 | 0 | 12.8 | 0 | 17.9 | 9.3 | 0 |
| | NewZealand | 0 | 17.9 | 0 | 17.9 | 0 | 20.5 | 15.2 | 0 |

**Table 7:** Reliability result from the live Tor network

| Measurements | Our Model | | | Conventional Tor | | |
|---|---|---|---|---|---|---|
| | 1st hop (%) | 2nd hop (%) | 3rd hop (%) | 1st hop (%) | 2nd hop (%) | 3rd hop (%) |
| Mean | 0.00 | 0.00 | 2.55 | 0.00 | 4.00 | 9.73 |
| SD | 0.00 | 0.00 | 2.21 | 0.00 | 3.01 | 8.01 |
| Max | 0.00 | 0.00 | 6.00 | 0.00 | 9.00 | 22.00 |
| Min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

### 6.2.2 Reliability

In addition to protecting Tor against maliciousness, our reputation model can also improve Tor's resilience to poorly performing relays. To motivate this, we first show that there is a high degree of variance in reliability across Tor relays. For this experiment we build Tor circuits using our own entry and exit relay pairs while using existing Tor relays as the middle relay of a Tor circuit. We then record statistics of circuit construction success rate for each Tor relay. We use the *nmap* command to scan the Tor-port (OR-port) of all the available Tor relays to cross-check whether they are actually online. The whole experiment was rerun every 30 minutes for a period of one full day. Figure 8 shows the difference between application level and network level reliability. We see that a certain fraction of the Tor relays drop circuit more often than others even though they are reachable. Figure 8 also shows the ratio of the total number of relays that failed to the total number of relays that succeed in building our customized circuit over different time intervals. Again we see a significant deviation in the application and network level reliability across the full day inspection and it is not the case that the deviation alters significantly across the time axis. We also highlight the distribution of these failures against the Tor relay's observed and advertised bandwidth. We computed the correlation between the advertised bandwidth and reliability. We found the correlation coefficient to be less than $10^{-8}$ in magnitude. So we can say, bandwidth is not an indicator of reliability in Tor network. We also verified that *past performance is an indica-*

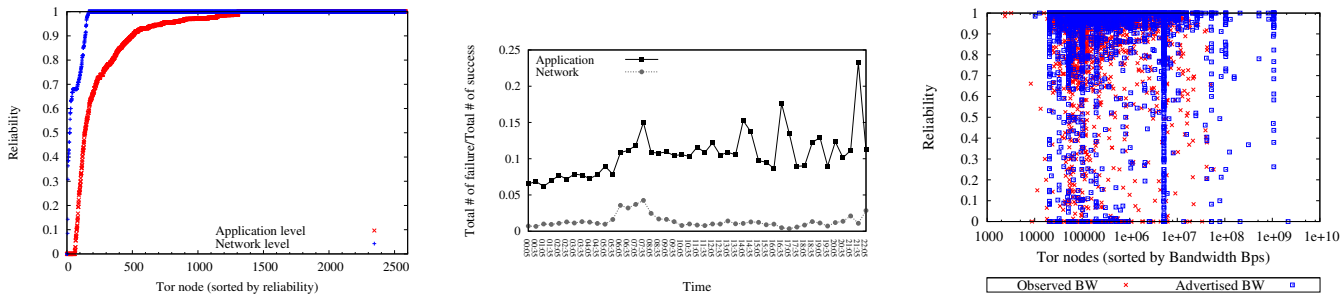



**Figure 8:** (a) Comparison of application and network level reliability of Tor relays. (b) Distribution of the ratio of total failure by success across different time intervals (c) Distribution of reliability against both observed and advertised bandwidth. We see that there is a significant difference between application level and network level availability.

*tion of future results*. For this we compute the "**Pearson Correlation**" between past observed failure probability and next observed outcome; we found the correlation co-efficient to be $\mathbf{0.72}$ which suggests that monitoring the reliability of relays is helpful for predicting their future performance.

From this, we can see that a client potentially benefits by keeping track of the reliability of the Tor relays he/she uses. To demonstrate this we run experiments on the live Tor network where a client creates circuits from a small set of Tor relays (3 guards, 23 middle relays and 23 exits). In one setting the client first generates a ranking of the Tor relays by probing each possible circuit once and then filters out potentially unreliable ones. Next, the client creates 100 circuits from the filtered list of relays. In the other setting the client just simply creates 100 circuits randomly from the full set of 49 non-filtered Tor relays. In both cases we record the percentage of circuits failing at different hops (we take the average of 10 runs). Table 7 shows the obtained results and we can see $\boldsymbol{Re^3}$ assists users to choose more reliable Tor relays for anonymous communications.

# 7. DEPLOYMENT CONSIDERATIONS

In this section we discuss the following two ways of deploying $\boldsymbol{Re^3}$ into the live Tor network.

- Localized: Individual clients run $\boldsymbol{Re^3}$
- Centralized: $\boldsymbol{Re^3}$ is run by directory authorities (DA servers).

We also study the convergence of $\boldsymbol{Re^3}$ as this is important for any practical deployment.

## 7.1 Localized Approach

The easiest way to incorporate our reputation model is to have it run in the client side. Each client can accumulate his/her experience to compute the reputation of Tor relays. Such a local approach does not require cooperation from any other relays in the system. However, this approach requires some incubation time to mature. That being said, clients can speedup the bootstrapping process by randomly browsing a small number of nonsensitive web sites or replaying some nonsensitive browsing history. The client could probe circuits periodically if needed and can also share reputation score with friends that they trust to speedup the convergence. Additionally, a client can initially concentrate on profiling higher bandwidth Tor relays before profiling other relays. For example, about 500 relays provide 80% of the available bandwidth in Tor [12].

## 7.2 Centralized Approach

In this approach the reputation model is run by the Tor directory authorities (DA servers). Tor authority servers already gather statistics of Tor relays (mainly observed bandwidth) to build a consensus database [7]. Each DA server can probe Tor circuits periodically and build its own reputation table. DAs can then include this information when they participate in the consensus protocol. The benefit of this approach is that it requires very few modifications to the existing protocol and the overhead of this approach is minimal. However, centralized probes need to be indistinguishable from real user traffic, otherwise malicious relays may alter their behavior during those probes. Randomizing both the entry relay of a probe and the time at which probes are made can make it harder for compromised relays to distinguish probes from actual user traffic.

## 7.3 Convergence of $\boldsymbol{Re^3}$

For any system to be practically deployable it must converge quickly to its stable state. In this section we investigate how the number of interactions affect the accuracy of $\boldsymbol{Re^3}$. For this purpose, we run our simulator where a client iteratively creates circuits following the default Tor path selection algorithm and then applies our filtering scheme. We then computes the probability of selecting compromised circuits after every 10 interactions. We vary drop rate $\boldsymbol{d}$ ($\mathbf{0} \leq \boldsymbol{d} \leq \mathbf{1}$) and consider the maximum probability that an adversary can achieve. Results are averaged over 100 000 runs.

Figure 9 shows the probability of selecting compromised circuits after every 10 interactions. We can see that as the number of interactions increases this probability dies down. This is expected because the more a client interacts with relays the more he/she becomes confident about those relays' reliability. However, we see that this probability quickly descends to a stable value after $\approx$200 interactions. We also look at the distribution of the number of unique relays against different number of interactions. Figure 10 shows that after 1000 interactions a client can roughly profile around 600 Tor relays (or 60% of Tor's bandwidth)[8]. Thus, we can conclude that a user does not need to accumulate too much experience to get a consistent view of the reliability of Tor relays.

# 8. RELATED WORK

Securing anonymity systems against active attacks is relatively a new research topic. Borisov et al. [20] first showed that a selective DoS attack can have devastating consequences for both high and low-latency anonymity systems.

More recently, Danner et al. [23] proposed a detection algorithm for selective DoS attack in Tor. Their algorithm probes each individual Tor relay in the network and requires $\boldsymbol{O(n)}$ probes to detect all compromised relays for a network comprising of $\boldsymbol{n}$ par-

[8]Alternatively, clients can profile higher bandwidth relays first.

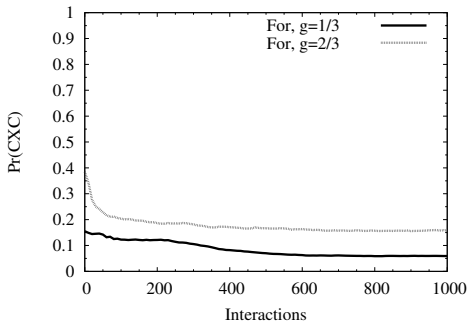



**Figure 9:** Probability of constructing compromised circuits after various number of interactions. This probability quickly converges to a stable value after only 200 interactions.

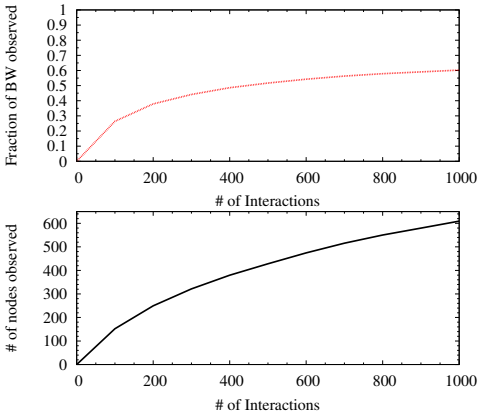



**Figure 10:** Average fraction of Tor bandwidth and Tor relays observed for different number of interactions.

ticipants. However, to handle transient network failures they proposed repeating each probe $r$ number of times, so their approach requires $O(nr)$ probes. So, at best their approach seems suitable for a centralized deployment. However, their algorithm assumes that compromised relays exhibit fixed characteristic of always dropping non-compromised circuits. They do not consider complex attack strategies where compromised relays may perform random dropping. Such dynamic malicious behavior could potentially increase the number of probes required to successfully identify compromised relays.

Mike Perry proposed a client-side accounting mechanism that tracks the circuit failure rate for each of the client's guards [10]. The goal is to avoid malicious guard relays that deliberately fail circuits extending to non-colluding exit relays. However, profiling only guards is not enough because it is less likely that an attacker will launch selective DoS at guard position, only to sacrifice the cost of obtaining a guard status (guards fulfill strong commitments like–minimum bandwidth, minimum uptime). Rather deploying a moderate number of cheap middle-only relays can boost the effect of the selective DoS [18].

Researchers have also leveraged incentive schemes [15, 30] to encourage good behavior from Tor relays. All incentive schemes basically encourage participants to be cooperative by providing the cooperating participants with something that they care about; however, incentive schemes do not enforce malicious participants to behave properly.

There are reputation based routing protocols for wireless adhoc networks [21, 34, 37] that try to identify selfish/malicious routers with the intent to avoid them during forward paths setup. While these protocols have similar goal as ours there are different challenges in directly using them for anonymity systems. For example, in all of these protocols routers maintain reputation information about their neighbors which they share with other routers in the network. This information sharing could potentially introduce new attack vectors where an adversary could figure out which relays certain users are using. Moreover, to the best of our knowledge none of these protocols handle strategic malicious behavior.

There are many papers on reputation systems for P2P networks [31, 40, 46]. TrustGuard [42] proposes a reputation framework which is capable of handling strategic malicious behavior. But TrustGuard is vulnerable to whitewashing attack, we introduce a confidence metric to hinder whitewashing attack. Moreover, most models focus on building distributed reputation systems, rather than worrying about privacy and anonymity as described in [39]. Dingledine et al. [24] described a reputation system for MIX-net environment [22]. But their approach relies on trusted witnesses which are hard to find in Tor network. Later on Dingledine et al. [28] restructured their reputation system [24] to avoid trusted witnesses and proofs in favor of self-rating groups of remailers. However, their approach does not reduce or prevent creeping death attack. They only propose to randomize the choice of node selection hoping that compromised nodes occupying the top positions in the reputation spectrum are not selected. In our case we propose a filtering scheme based on reputation score to discard such compromised relays, where the reputation metric itself can handle strategic oscialltions.

# 9. LIMITATIONS

Our work has several limitations. First, in the absence of attacks, a small fraction of honest relays are classified as outliers due to random network failures. For anonymity systems, it is much more critical to blacklist malicious relays than to ensure that all honest relays are included. Moreover, these discarded honest relays should reflect either low performing or highly congested relays in absence of attack. Thus discarding them might actually help in shuffling the overall network load. Second, for the local deployment approach, our model does not defend against the scenario where all of a user's guard relays are malicious. We note that for 20% of malicious relays, the probability of all three of a user's guard relays being malicious is less than 1%. Finally, new users benefit from our reputation model only after a certain amount of usage.

# 10. CONCLUSION

Anonymity systems are vulnerable to active attacks like selective denial-of-service. Such attacks, however, can be detected by profiling relay behavior. We proposed a generic reputation model that profiles relays based on their historical behavior. Our model takes adaptive malicious behavior into consideration and penalizes any participant exhibiting such behavior. We theoretically analyze our system under different attack scenarios, including probabilistic variants of selective DoS, targeted framing attack and the creeping-death attack. Our simulation and experimental results on the live Tor network suggest that our reputation model can effectively filter compromised relays mounting active attacks. We also show that our reputation model provides benefits even outside the context of active attacks; Tor clients using our model experienced significant improvement in the reliability of circuit construction.

Our reputation model has broad applicability, and can be used in domains such as P2P and recommendation systems, where users would benefit from profiling participants with dynamic behavior.

# 11. REFERENCES

[1] Dutch government proposes cyberattacks against... everyone. https://www.eff.org/deeplinks/2012/10/dutch-government-proposes-cyberattacks-against-everyone.
[2] Emulab. https://www.emulab.net.
[3] Known bad relays in Tor. https://trac.torproject.org/projects/tor/wiki/doc/badRelays/trotsky.
[4] NSA attacks Tor. http://gizmodo.com/the-nsas-been-trying-to-hack-into-tors-anonymous-inte-1441153819.
[5] PlanetLab. http://www.planet-lab.org/.
[6] Tor compass. https://compass.torproject.org/.
[7] Tor consensus. https://metrics.torproject.org/consensus-health.html.
[8] Tor controller. https://svn.torproject.org/svn/blossom/trunk/TorCtl.py.
[9] Tor directory authorities compromised. https://blog.torproject.org/blog/tor-project-infrastructure-updates.
[10] Tor proposal 209:. https://gitweb.torproject.org/user/mikeperry/torspec.git/blob/path-bias-tuning:/proposals/209-path-bias-tuning.txt.
[11] Torflow project. https://gitweb.torproject.org/torflow.git.
[12] Torstatus. http://torstatus.blutmagie.de/index.php.
[13] Trotsky IP addresses. https://trac.torproject.org/projects/tor/wiki/doc/badRelays/trotskyIps.
[14] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A low-latency AS-aware Tor client. In *IEEE S & P*, 2012.
[15] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin. Par: Payment for anonymous routing. In *Proceedings of the 8th international symposium on Privacy Enhancing Technologies*, PETS '08, pages 219–236, 2008.
[16] K. Ang, G. Chong, and Y. Li. PID control system analysis, design, and technology. 13(4):559–576, 2005.
[17] R. Aringhieri, E. Damiani, S. D. C. Di Vimercati, S. Paraboschi, and P. Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *J. Am. Soc. Inf. Sci. Technol.*, 57(4):528–537, 2006.
[18] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker, and D. Mccoy. On the optimal path length for Tor, 2010.
[19] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against Tor. In *WPES*, 2007.
[20] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? In *CCS*, 2007.
[21] S. Buchegger and J. Y. Le Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems (P2PEcon)*, 2004.
[22] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24, 1981.
[23] N. Danner, D. Krizanc, and M. Liberatore. Detecting denial of service attacks in Tor. In *FC*. 2009.
[24] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase mix-net reliability. In *IH*, 2001.
[25] R. Dingledine and N. Mathewson. Tor path specification. https://gitweb.torproject.org/torspec.git/blob/HEAD:/path-spec.txt.
[26] R. Dingledine and N. Mathewson. Tor project, Tor path specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt.
[27] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.
[28] R. Dingledine and P. Syverson. Reliable mix cascade networks through reputation. In *FC*, 2003.
[29] M. Edman and P. Syverson. AS-awareness in Tor path selection. In *CCS*, 2009.
[30] T.-W. "Johnny" Ngan, R. Dingledine, and D. S. Wallach. Building incentives into tor. In *Proceedings of the 14th international conference on Financial Cryptography and Data Security*, FC'10, pages 238–256, 2010.
[31] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *WWW*, 2003.
[32] H. Kwakernaak. *Linear Optimal Control Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1972.
[33] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems. In *FC*, 2004.
[34] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced Communications and Multimedia Security*, The International Federation for Information Processing (IFIP), pages 107–121. Springer US, 2002.
[35] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *HICSS*, 2002.
[36] S. J. Murdoch and P. Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *PET*, 2007.
[37] F. Oliviero and S. Romano. A reputation-based metric for secure routing in wireless mesh networks. In *IEEE GLOBECOM*, pages 1–5, 2008.
[38] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.
[39] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12), 2000.
[40] Z. Runfang and H. Kai. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4), 2007.
[41] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *ESORICS*, 2006.
[42] M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: Countering vulnerabilities in reputation management for decentralized overlay networks. WWW '05, pages 422–431, 2005.
[43] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In *International Workshop on Designing Privacy Enhancing Technologies*, 2001.
[44] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communications against passive logging attacks. In *IEEE S & P*, 2003.
[45] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *NDSS*, 2002.
[46] L. Xiong and L. Li. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.
[47] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *PET*, 2004.

# APPENDIX

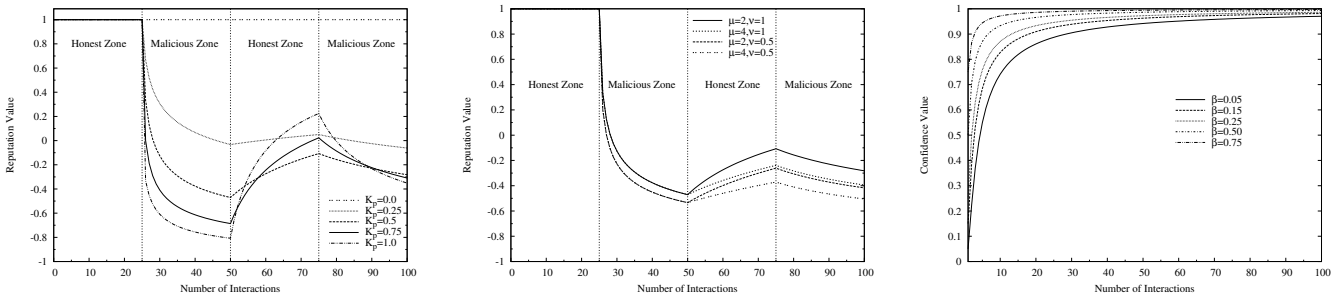
## A. STABILITY ANALYSIS

From control theory we know that for a discrete-time linear system to be stable all of the poles of its *transfer function* must lie inside the unit circle [32]. To determine this we first need to take the *Z-transform* of the transfer function and then determine its poles. We first rewrite equation (1) as the following first order discrete-time linear system.

$$\boldsymbol{y(n) = \alpha x(n) + (1 - \alpha) y(n - 1)} \tag{10}$$

where $\boldsymbol{y(n)}$ and $\boldsymbol{x(n)}$ denotes the $\boldsymbol{n}$-th output and input of the system respectively (i.e., $\boldsymbol{y(n)}$ refers to newly generated reputation value $\boldsymbol{R_n}$ of a relay while $\boldsymbol{x(n)}$ refers to the reference value $\boldsymbol{R_c}$). Taking the *Z-transform* of equation (10) yields:

$$\boldsymbol{Y(z) = \alpha X(z) + (1 - \alpha) z^{-1} Y(z)} \tag{11}$$




(a) (b) (c)

**Figure 11:** Sensitivity analysis of (a) controller gain $K_p$ (b) reward $\mu$ and punishment $\nu$ factor (c) confidence factor $\beta$.

From equation (11) we can compute the transfer function as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\alpha z}{z - (1 - \alpha)} \tag{12}$$

So the transfer function $H(z)$ has a pole at $z = 1 - \alpha$. From equation (3) we know that $0 < \alpha < 1$, so the pole will always lie inside the unit circle. Thus our closed-loop reputation system guarantees stability.

## B. TUNING MODEL PARAMETERS

In this section we look at how the parameters related to $Re^3$ affect the reputation score of compromised relays. As shown in Table 5, $Re^3$ has a total of four parameters. We will study each of their impact on reputation score. In the following study, we mainly want to see how our model reacts to dynamic behavioral change. For this purpose we assume that a compromised relay, with all the other relays being honest, participates in a total 100 circuits oscillating between honest and malicious nature every 25 interactions.

### B.1 Proportional constant $K_p$

The proportional constant, $K_p$, determines to what degree we want to react to the deviation between the reference value ($R_c$) and current system output ($R_{n-1}$). This should not be set either too high (near 1) or too low (near 0). If it is set too high it will oscillate too much and if it is set too low then it will discount most of the deviation and result in slow convergence. Figure 11(a) highlights what we have just discussed. As we can see from the figure for $K_p = 1$ it oscillates heavily and for $K_p = 0$, it totally discard the deviation between reference value and system output. We therefore conservatively set $K_p$ to 0.5, so that the model becomes neither too sensitive nor too insensitive to sudden deviation.

### B.2 Reward $\mu$ and Punishment $\nu$ Factor

We now investigate how our reputation model responds to failures and successes. We want the degree of punishment to be greater than that of reward. So whenever the model receives a negative feedback (in our case a rating of -1) we want our EWMA function (equation (3)) to give higher weight to the current feedback (i.e., larger value of $\alpha$). As $\alpha$ in dependent of $\delta$ (see equation (5)), we need to increase $\delta$ more for failure than success. So under our setting we require $\mu > 1$ and $\nu \leq 1$. Figure 11(b) highlight the reputation score for different combination of $\mu, \nu$. As long as $\mu > 1$ and $\nu \leq 1$ our model can effectively discourage selective DoS.

### B.3 Confidence Factor $\beta$

Now, we look at our confidence factor $\beta$. The confidence factor determines how confident a user is about the reputation score of a particular relay. As the number of experience with a particular relay increases, a user becomes more confident about his/her computed reputation score. $\beta$ controls how quickly we become confident about a reputation score. Figure 11(c) highlights how confidence factor increases as the number of interaction increases. It should be mentioned that any monotonically increasing function of the number of interactions can be used as a confidence metric.

## C. TUNING CUTOFFS FOR OUTLIERS

In filtering outliers we previously chose a deviation interval of $\sqrt{3}$ times the standard deviation (see Section 6) from the average. Here, we investigate what kind of impact other deviation intervals would have on the performance of $Re^3$. A tradeoff exists between the value of $k$ and false errors FN and FP. As we increase $k$ more relays become acceptable, so FP goes down but FN rises. We tested for $k = 1.3, 2.0$. Figure 12 illustrates the FN and FP errors for different values of $k$. From the figure we see that as we increase the allowed deviation from average, more and more relays become acceptable and as a result FP decreases while FN increases.

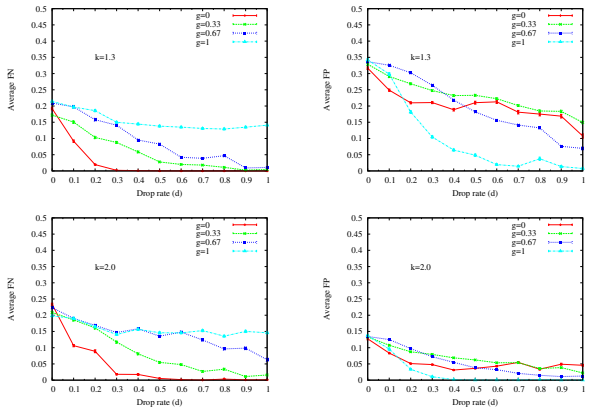



**Figure 12:** False Negative (FN) and False positive (FP) ratios for different values of $k$. As $k$ increases FN tends to rise and FP tends to fall.