Jamie Hayes and George Danezis

Guard Sets for Onion Routing

Abstract: "Entry" guards protect the Tor onion routing system from variants of the "predecessor" attack, that would allow an adversary with control of a fraction of routers to eventually de-anonymize some users. Research has however shown the three guard scheme has drawbacks and Dingledine et al. proposed in 2014 for each user to have a single long-term guard. We first show that such a guard selection strategy would be optimal if the Tor network was failure-free and static. However under realistic failure conditions the one guard proposal still suffers from the classic fingerprinting attacks, uniquely identifying users. Furthermore, under dynamic network conditions using single guards offer smaller anonymity sets to users of fresh guards. We propose and analyze an alternative guard selection scheme by way of grouping guards together to form shared guard sets. We compare the security and performance of guard sets with the three guard scheme and the one guard proposal. We show guard sets do provide increased resistance to a number of attacks, while foreseeing no significant degradation in performance or bandwidth utilization.

DOI 10.1515/popets-2015-0017
Received 2015-02-15; revised 2015-05-13; accepted 2015-05-15.

Introduction

Tor [8] is one of the most popular low-latency distributed anonymity networks [2]. Volunteer relays route encrypted network traffic to a desired destination, providing users with anonymity. Tor employs three layers of relays – guards, middle, and exit relays – to provide anonymity to user circuits by obscuring the correspondence between initiators of an internet connection and the services they access. A body of research [4, 6, 18, 21] studies how these relays should be selected in order to minimize user exposure to de-anonymization risks. The choice of the circuit guard is crucial both for security and performance. Tor's (pre-2015) design saw each user choose a small set of guards and use three of them for extended periods of time; post-2015 Tor's default behaviour is for each user to choose a single guard from a set of high-availability high-bandwidth relays and use it for every circuit for up to nine months.

Jamie Hayes: UCL, E-mail: j.hayes@cs.ucl.ac.uk
George Danezis: UCL, E-mail: g.danezis@ucl.ac.uk

Security. Both Tor's three guard design [19] and the proposed single guard [7] designs are not without controversy. The choice of guard selection strategy impacts on the susceptibility of an onion routing system to three attacks:

Direct Observation. A corrupt guard node can, with the help of a small set of corrupt exit relays, compromise the anonymity of a user. A compromise, in the context of this attack, means that the guard will be able to directly observe at least one circuit from a particular user. The use of relatively stable guards therefore aims to limit the number of users that may fall foul of such an attack over time, by limiting the users exposed to the direct observation of corrupt relays.

Guard Fingerprinting. A number of attacks [1, 3, 21] aim to identify the set of guards used by a user. The set of such guards acts as a fingerprint for the user and in case it is unique it may be used to track and de-anonymize a user's action within the Tor network. The three guard design was susceptible to this attack, since each user chooses a different set of guards, the combination is likely to be unique. The single guard design is less susceptible to this attack (but not entirely, as we discuss in Section 5.1).

Statistical Disclosure Attacks. Even if the identity of the guards does not in itself uniquely determine the user, a bigger possible set of users is preferable to a smaller set of users [3, 7], as those act as an anonymity set. In case a guard, or set of guards, are only used by a small number of users, it is possible to link their patterns of actions to long term identifiers, and degrade the anonymity they enjoy through disclosure attacks. We examine in Section 5.1 how both the three guard, and the newer single guard proposals exhibit variants of this flaw.

The current three guard selection policy has been shown to provide less security than was originally thought [5, 10], and with that in mind the proposal of moving from three to one guard has been formulated, which solves some but not all security problems [7]. In terms of direct observation a single guard leads to a user having all

their circuits potentially compromised. Given the very conservative definition of compromise this is not significantly worse than in the case of using three guards, where some circuit would be quickly compromised even by a single corrupt guard in the set. In terms of fingerprinting, a single guard should no more identify a user to the same extent as three guards. However, users of smaller or newer guards enjoy a smaller "anonymity set" than users of larger or older guards. In this context the main contributions of this work are:

1. To present a design for "guard sets", sets of relays providing a certain amount of bandwidth, that are used – as a group – by multiple users. We show the scheme provides near optimal spread of load while protecting against attacks that the current entry guard schemes are susceptible to.
2. We design an algorithm based on a binary tree structure to automate the assignment of guards to guard sets and to users, while taking into account the dynamic conditions of the real Tor network where routers join and leave continuously.

This is work intended as a first examination of guard sets, and the security and performance analyses we provide leave important questions whose answers may require fundamentally different algorithms. Similarly many practical questions will need to be answered in order for any guard set approach to be deployable.

The remainder of this paper is organised as follows: In section 2 we introduce guard sets and explain the threat model we will work under. In section 3 we develop the guard set scheme and introduce a binary tree based algorithm for matching users to guard sets. Section 4 introduces the security metric by which we will measure the success of our proposal against other schemes. We evaluate the performance and security properties of our proposal in section 5. Finally, we conclude and discuss open issues in section 6.

Tor Guards & Guard Sets

Guards are relays that are fast and stable compared to other relays in the Tor network. More precisely, a relay has traditionally (pre-2014) been assigned a guard flag if the following requirements are met:

1. The relay must have been online longer than 12.5% of relays, or for 8 days.
2. The relay must advertise at least the median bandwidth in the network, or 250KB/s.
3. The relay needs to have at least the median weighted fractional uptime (WFU) of relays in the network, or 98% WFU. The WFU metric measures the fraction of uptime of a relay in the past. WFU values are discounted by factor 0.95 every 12 hours including the current uptime session.

The one guard proposal (post-2014) imposes similar requirements but increases the bandwidth threshold 250KB/s to 2MB/s.

Performance. Under the three guard selection policy clients rotate their guard every 30 days - 60 days. This serves both security and performance purposes. If clients never rotate guards then these relays accumulate more and more users and are under a heavier load compared with guards that are new to the network and have not had the opportunity to accumulate users. Conversely, new guard relays, that have only recently been assigned a guard flag, are under-utilized; the bandwidth weights allocate a large fraction of the relay's bandwidth for use as a guard, but only few clients have rotated to this new guard. Increasing the guard rotation period makes this worse: clients rotate guards less frequently, and new guards acquire users at a slower rate. Rotating guard relays frequently spreads the load on bandwidth efficiently, but also leads to a faster rate of compromise as [7, 10] note – leading to a decision to favor guard stability over load balancing.
The one guard solution relies on using new guards as "fractional" guards where their available bandwidth is used as middle and exit nodes when it is under-utilized initially. This ensures efficient use of bandwidth but does not increase the anonymity set behind a newer guard. Ideally we would like new guards to be well populated and used as soon as they are assigned a guard flag – a key feature of the proposed "guard set" mechanism.

Threat Model. Like the Tor system itself, we consider an adversary that may control directly, or observe, a fixed fraction of the Tor routing infrastructure.
Due to the time and bandwidth costs for becoming a guard relay we do not consider the attack where an adversary can leave and re-join the network whenever they wish – as this will lead to their guard status being lost. Thus those attacks may be neutralized through setting appropriate periods before a relay may become a guard. We consider that end-to-end correlation attacks are possible if an adversary controls a proportion of the the to-

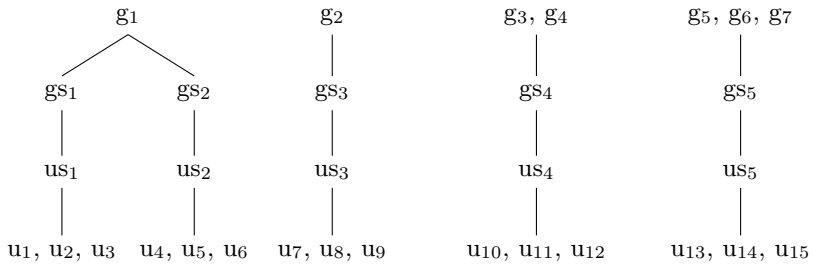


Fig. 1. An example of guard sets consisting of seven guards and fifteen users. The first guard has enough bandwidth to create two guard sets, which serve two user sets each with three users. The second guard has enough bandwidth to create one guard set. The third and fourth guards have enough combined bandwidth to create one guard set and similarly the fifth, sixth and seventh guard create one guard set.

tal available guard bandwidth and a malicious exit node (under new Tor specifications a relay can now simultaneously be a guard and exit node, though not on the same circuit). We make the assumption that a compromised guard leads quickly to a full compromise of the user. Those attacks are the reason to favor stable assignments of users to guards – but we do not aim to protect against those in any additional way (through padding, etc).

The proposal for "guard sets" is therefore primarily concerned with guard selection to safeguard against end-to-end correlation through direct observation, fingerprinting and statistical disclosure attacks.

Guard Sets. To alleviate those problems, the concept of "guard sets" was proposed [7, 15], but never extensively pursued or developed, due to intricacies of accommodating guards joining and leaving.

The core idea behind "guard sets" is that many users are assigned the same guard set, out of a small selection. When it comes to building a Tor circuit clients chose a first relay out of their assigned guard set – uniformly at random. Thus users at the same guard set are indistinguishable when it comes to fingerprinting, since multiple users will be using the same set of guards. Subject to carefully crafting rules to assign users to new guard sets, the population of users served by any guard set are plentiful, foiling statistical disclosure attacks. Added to this most guard sets consist of multiple guards, so if one guard fails users will not need to switch to a new guard set.

To summarize, "guard sets", instead of either the three guard proposals, or the single guard proposals, promise the following advantages:

- Better protection against the three identified attacks.
- Improved reliability and security when single guards are temporarily offline.
- The provision of more, and more uniform, bandwidth to each client as compared with the single guard proposal.

The remainder of this paper examines a possible construction for such "guard sets", dynamic allocation of users and guards to "guard sets" and compares the security of the new proposals with previous approaches. [1]

A Guard Sets Proposal

Our proposal consists of arranging guards and users in three layers, each being a bipartite graph:

1. Guards to guard sets.
2. Guard sets to user sets.
3. User sets to users.

We propose algorithms to determine and maintain guard sets, and show that the security of any guard selection scheme may be analysed on the basis of the quadripartite graphs linking guards to guard sets, to users sets and ultimately to users. An example of such a graph is shown in Figure 1: it demonstrates how clients connect to their guards via user sets – notionally the set of all users attached to a guard set – and guard sets. When a client initially joins Tor the assignment algorithm randomly chooses a guard set to use, which is

1 Unless stated otherwise, data used in experiments use consensuses beginning January 2013 in order to allow for a direct comparison with the one guard proposal [7].

used by possibly many user sets. The client is then randomly assigned to a user set via means of an identifier shared among all clients in that set.
Guard sets are initially created by grouping together guards with similar bandwidth via the current consensus report. Users are split into groups and uniformly distributed among these guard sets. The main premise of our proposal is that the relation between guard sets and user sets is stable, unless there is a need to add in extra bandwidth or to delete the guard set. This maintains a grouping of users with the same guard history leading to a robust anonymity set, resistant to fingerprinting, and statistical disclosure attacks. Each client assigned to a guard set then randomly selects a guard within the set when they wish to build a Tor circuit.

Guards and Guard Sets. Guard sets are initialized as follows. We split guards into available bandwidth quanta, the maximum bandwidth a guard can provide before the threshold for guard sets is reached. A single guard therefore may split into multiple quanta. We sort quanta in descending order of their guard bandwidth, and we then cycle through each quantum appending them to a set. Once the sum of quantum bandwidths in the set exceeds a threshold we define it as a guard set. We then proceed to define another guard set from the remaining quanta. If available quanta run out before exceeding the threshold, the quanta are discarded – and the bandwidth is wasted by not being allocated to a guard set.
We consistently use a threshold for becoming a guard set of 40 MB/s, and set a threshold for deleting and replenishing guard sets at half the creation threshold. Through experiments we observe this threshold creates a large number of guard sets and is large enough to ensure guard sets are deleted infrequently. We do not claim this threshold to be optimal, and will need to be adapted according to the Tor network conditions.
The benefits of allocating quanta to guard sets by order of guard bandwidth are:

1. If a guard has enough bandwidth it can serve multiple guard sets and therefore serve many user sets. This helps spread the load of bandwidth efficiently and facilitates adding new guards.
2. No guard with bandwidth smaller than the threshold can belong to more than one guard set, this limits the number of guard sets an adversary with control of a modest fraction of the total guard bandwidth can corrupt.
3. Alternative formation strategies see an adversary with control of a small fraction of total available bandwidth, through small guards, corrupt a larger number of guard sets. Assigning quanta to guard sets from guards with similar bandwidth prevents smaller adversaries infiltrating a large number of guard sets.

Client and Guard Set Dynamics. The dynamic nature of Tor requires our proposal to manage the leaving and joining of both guard sets and users in an efficient manner. We do this by considering guard sets and user sets as nodes on a finite binary tree. This (full) finite binary tree, also referred to as a finite Cantor tree, has nodes consisting of all binary sequences up to some length, as show in Figure 2.

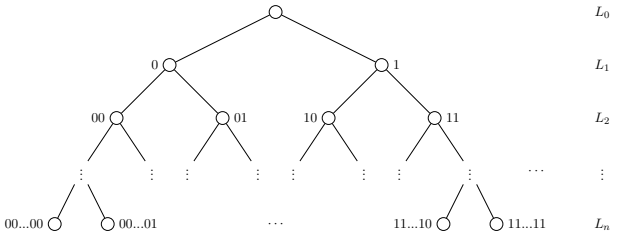


Fig. 2. An example of a finite Cantor tree with n layers.

Guard sets sit on some intermediate layer of the tree covering all leaves (the exact layer determined by how many guard sets are created upon initialization), with clients occupying leaves. Each client is assigned a stable identifier, the binary sequence assigned to that node. Note that the prefix of the clients identifier is also the identifier for the guard set to which it belongs.

Guard sets dynamics are based on the following rules:

1. On initialization we assign guard sets random identifiers of some fixed length.
2. Any path from the root to a leaf must pass through exactly one node occupied by a guard set.
3. When a new guard set is created we flip a fair coin at each branch until we get to a guard set node. We then force this node down a level creating room for the new guard set on the layer below. By convention the new guard set branches to the right, or in other words, all new guard set identifier sequences will end with 1.
4. If a guard set is deleted and the sibling node in the tree is also a guard set we elevate one layer, letting the sibling guard set occupy the parent node.

5. If a guard set is deleted and the sibling node is not a guard set we find the lowest layer guard set with a common ancestor and use the guard set which was created most recently to replace the deleted guard set.

Ideally clients should be split equally among all guard sets, with a uniform occupied / unoccupied leaf spacing. To accomplish this we simply assign each new user a random identifier which creates an even expected distribution of users among leaves. Initially this may be bad for load distribution between guard sets; ideally we would like to direct more users to a guard set handling 500 users than to a guard set handling 8000, but we assign users to guard sets in this fashion because (a) insertion attacks become easier when there is a large user assignment bias to certain guard sets (b) biasing could create swells of users among areas of leaves, if a guard set is deleted this would create a large disparity between user numbers for guard sets at different layers.

The benefits of using this tree structure to manage guard set and user set assignments can be summarised as:

- Clients with similar identifier prefixes will have similar guard histories, therefore constituting stable anonymity sets foiling fingerprinting and statistical attacks.
- It provides an automatic method for guard set deletion and creation, with back-ups selected. Because guard sets are assigned positions in the tree based on fair coin flips there is no deterministic way to forecast where a new guard set will be placed. In the event that a guard set is deleted, users in the same user set will use the same back-up guard set. Unlike other schemes, it is not possible for an adversary to uniquely identify users through back-up guard histories.
- Compromising a specific target client is made more difficult due to the random guard set placement in the tree. Fingerprinting attacks are made more difficult due to users with similar identifiers have similar guard histories, while not compromising fair distribution of load on guard sets.

Joining / Leaving Guards. In 2013 total guard bandwidth increased from 5283.52 MB/s to 14471.923 MB/s, illustrating the importance of an effective method for adding new guards to guard sets. Algorithm 1 outlines how we introduce guards to existing guard sets that have dropped below a bandwidth threshold at each consensus. In a nutshell, we detect guard sets with bandwidth below a threshold, and attempt to assign to them available bandwidth quanta from guards with similar bandwidth. The algorithm previously listed all available quanta in descending order of bandwidth and cycled through them appending them to guard sets until a threshold has been reached, and so each guard set will consist of quanta that have similar bandwidth capacity to one another. Then for each quantum the algorithm finds the most suitable guard set - in terms of similar quanta bandwidths. If enough bandwidth quanta are found to increase the bandwidth above the threshold the guard set is considered fine, otherwise it is deleted and all its quanta made available to build new guard sets. We consider a guard set broken if it has bandwidth capacity half of the creation threshold.

```
Data: consensus, guard sets, fix threshold,
Result: updated guard sets
broken guard sets ← guard sets with
bandwidth < fix threshold;
fixed guard sets ← guard sets;
for broken guard sets do
    compute candidate guards from available
    guards based on bandwidth similarity to
    broken guard sets;
    insert candidate guards in to broken
    guard sets;
    if broken guard set bandwidth > fix
    threshold then
        add to fixed guard sets;
    else
        return
    end
    fixed guard sets
end
```

Algorithm 1: How to add in guards to depleted guard sets

Algorithm 2 describes the process of creating, fixing and deleting guard sets. At each consensus we remove guard sets that have dropped below a bandwidth threshold, which we set as half the threshold for guard set creation. We redistribute the unused and new guards to existing guard sets, and create new guard sets with the remaining unused guards. With every consensus, or at fixed intervals, current guard sets are fixed or deleted, and the remaining available quanta are used (in the or-

der of their guards' bandwidth) to build new guard sets. Those guard sets are then inserted into the tree to take charge of a branch of users.

Data: consensus, guard sets, kill threshold,
Result: updated guard sets
for *every consensus* **do**
  retrieve guard sets positions in tree;
  guard sets ← **Algorithm 1**(guard sets);
  killed guard sets ← guard sets below kill threshold;
  remove killed guard sets from tree;
  new guard sets ← guard sets created from leftover quanta;
  guard sets + = new guard sets;
  add new guard sets to tree;
  return guard sets
end

Algorithm 2: How to compute guard sets given a consensus document

Remove Guard Rotation. Johnson et al. [14] showed one of the driving forces behind end-to-end correlation attacks was guard rotation, where a client would eventually rotate to a corrupt guard. We therefore propose to never rotate guards as this would change the entire guard set graph (displayed in Figure 1) leading to a faster compromise rate. Therefore guard rotation only occurs when new guards are added to existing guard sets to replenish bandwidth that has fallen below a certain bandwidth (Algorithm 1), or to insert new guard sets (Algorithm 2) in case a critical mass of bandwidth has become available.

Analysis

The guard set proposal arranges guards and user in three bipartite graphs illustrated in Figure 1: The first layer consists of edges between guards and guard sets. The second layer maps guard sets to user sets. Initially this is an injective assignment and as time goes by we allow the creation of new edges. We only delete an edge if a user set starts using a new guard set and we require a user set to have exactly one edge leading to a guard set at any time. The third layer maps user sets to users. For each user set this is a many users to one user set assignment. Note we never move an edge from an existing user to a new user set, we only delete (in the case of a user leaving Tor) or add in a new edge (in the case of a user joining Tor). Clearly moving edges in the third layer from existing users to user sets other than the group they were originally assigned to will eventually create users with unique guard histories.

Security Metric. We consider a Tor user compromised by an adversary if the user ever uses a guard set containing a guard under the adversary's control. Meaning once a user has been compromised we considering all future uses of Tor compromised. In terms of the graph representation a user is compromised if there exists or has ever existed an edge path from a corrupt guard to the user. Under this security metric any path from a guard to a user is a route for potential compromise, adding a new edge anywhere can only create the same number of potential paths or more. We therefore wish to keep the number of edges in our graph to a minimum while maintaining the load balance across guard sets.

One guard is optimal in a static environment. Due to the dynamic nature of Tor we will be forced to add and remove edges due to bandwidth and user fluctuations. If Tor were a static environment in which we knew that present and future bandwidth requirements were equal our security metric implies that the optimal set up of the graph would be an injective assignment in the guard to guard set layer, as it can be represented with the minimum possible number of edges. Keeping the requirement of optimal bandwidth load on guards and allowing for non-uniform sized user sets implies that, in the static case, the one guard proposal [7] is optimal. Since if a corrupt guard belongs to multiple guard sets it compromises them all, it is optimal to have each guard be it's own guard set. This is true in a static network where all users and guards are assigned initially to guard sets and user sets and never join or leave. In a dynamic case, it is no longer true: new guards are used by fewer users opening the way for statistical attacks; failing guards force users to fall back on others, leading to a unique fingerprint per user, and the need to rotate single failing guards increasing the likelihood of direct observation by an adversary. Therefore it is imperative to measure the quality of anonymity in a dynamic setting, where guards join and leave.

Evaluation

Tree algorithm experiment results. Our tree based algorithm is only applicable if layers on which guard set nodes sit do not diverge over time, and so each maintain a similar load. To evaluate the efficacy of the proposed tree structure scheme we evaluated how mappings between guard sets and user sets change under creations and deletions of guard sets throughout 2013. Our Python scripts return the difference in layers of guard sets per day throughout 2013.

Over a year period beginning 1 January 2013 we calculated, using a daily consensus document, that the lowest difference in guard set layers is three and the highest difference in guard set layers is six. This means in the lowest difference scenario we have at least one guard set hosting 2^3 times as many users as at least one other guard set, and in the highest difference scenario we have at least one guard set hosting 2^6 times as many users as at least one other guard set. The difference in guard set layers does not grow with time and so there will not be large differences between number of clients on guard sets. At any point in time user sets will therefore be plentiful and there will not be a large disparity of load on guard sets. Unlike guard sets, in the one guard scheme there is large disparity of number of users on large bandwidth guards and new smaller guards.

The introduction of a new guard set splits an existing user set in half. As a result the size of user sets is variable and depends on the depth of the corresponding guard set in the binary tree. This is preferred to user sets of a fixed size since this strategy can accommodate the dynamic nature of Tor. Since we cannot make guarantees about the number of Tor users at any one time, or make confident predictions about numbers of future users the scheme must adapt and enforce the similarity of user guard histories for any cardinality of global users.

5.1 Anonymity

Security. Elahi et al. [10] propose a security measure of "time to first compromise". Given an initial set of guards, an adversary adds in one corrupted guard. Under a no rotation policy in the current three guards scheme, 10% of clients will use the corrupted guard after 8 months. Under the three guard rotation policy (every 30 - 60 days) 14% of clients will be compromised after three months.

Under the guard set proposal, we assume that if a guard is corrupt in a guard set then all clients using that guard set are compromised. For comparison we ran our experiment with a random single guard chosen as an adversary guard from a random guard set and repeated 100 times. Figure 3 shows the fraction of users compromised. We see that on average we can expect less than 2% of users to be compromised through injecting one corrupted guard in to a guard set. At worst an adversary will compromise less than 6% of Tor users.

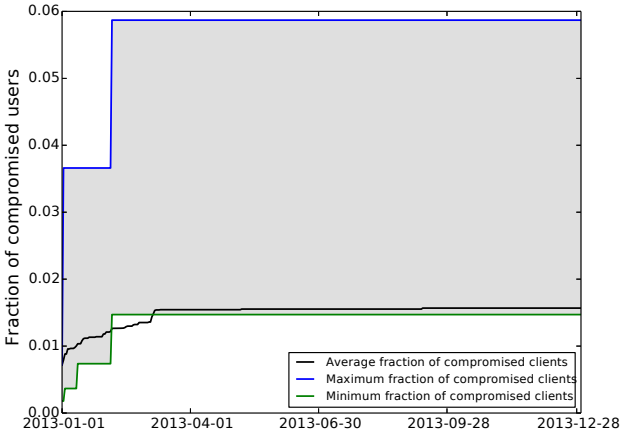
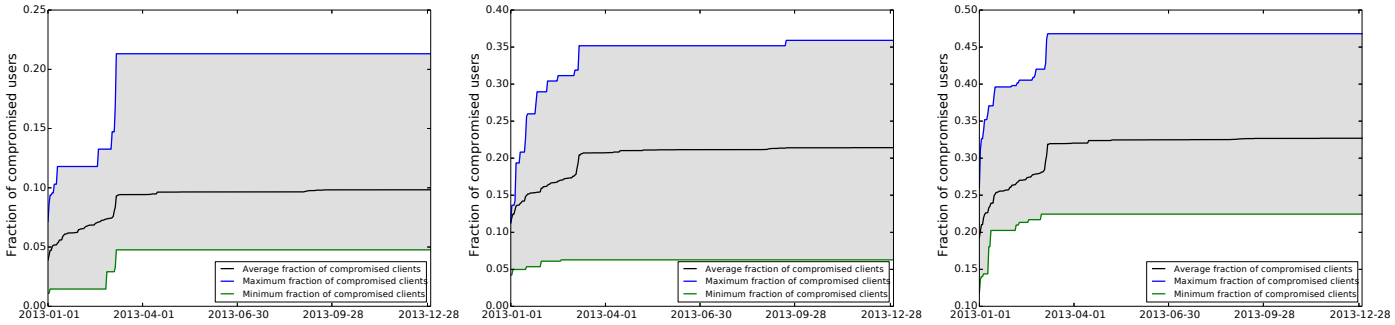


Fig. 3. Fraction of compromised users given an adversary controlling one guard. With 1 million users and 100 repeats.

We amend this metric by considering an adversary that controls a fixed fraction of total guard bandwidth instead of one guard. Initializing guard set creation on 1 January 2013 we chose at random adversary guards whose bandwidth would sum to 1%, 5% and 10% percent of total guard bandwidth. We then observe the fraction of the user base compromised over the course of one year. We repeat the experiment one hundred times for each of the scenarios to also estimate the variability of anonymity provided.

Experimentally, we observe that the probability of an adversary guard serving only in a single guard set, until it goes offline, is 0.95. The probability an adversary guard serves two guard sets, before going offline, is 0.03 and three guard sets is 0.02. We did not observe any guards observing more than two guard sets.

The average number of controlled adversary guards for 1%, 5% and 10% was calculated to be 4, 14 and 25, respectively. Assuming an adversary will maximize the number of guard sets under control, the probability that the adversary controls the same initial number of guard sets throughout 2013 is 0.8145, 0.4877 and 0.2773 when



(a) Adversary controls 1% **(b)** Adversary controls 5% **(c)** Adversary controls 10%

Fig. 4. Fraction of compromised users given an adversary controlling a fixed fraction of guard bandwidth. With 1 million users and 100 repeats.

in control of 1%, 5% and 10% of bandwidth, respectively. We can therefore be confident that a corrupt guard will remain in its initial guard sets when an adversary controls 1% of guard bandwidth, and will control more than its initial guard sets when an adversary controls 10% of guard bandwidth.

Figure 4 shows for each adversary, the compromised fraction of users at any point in time during 2013 (note the different y-scales). For each controlling fraction the highest line shows, after 100 repeats, the maximum observed fraction of the Tor user base that is compromised. The quantized structure of lines is due to whole user sets being served by corrupt guard sets as adversary guards' assignments change. We observe the final fraction of compromised users at the end of the year is stable throughout the year, after an initial period of 2 to 3 months.

The average rate of compromise after a year for the 1%, 5% and 10% of the bandwidth is 9.83%, 21.42%, 32.69% of users respectively. However, those rates of compromise are contingent on the specific guards being corrupt and their movements across guard sets. Thus for 1%, 5% and 10% corrupt bandwidth we observe scenarios with rates of compromise as high as 21.31%, 35.91% and 46.80%, respectively.

We note the rate of compromise increases sharply in the first quarter of the 2013 year. Figure 5 explains the proliferation in number of compromised users at the beginning of year.

We see a dramatic decrease in the number of guard sets in March 2013, leading to a larger fraction of clients who use each guard set compared to previous months. We see a small increment in user compromise in September as the number of guard sets once again decreases. By this time there are a large number of established guard sets in the Tor network. Due to the diversity in guard sets,

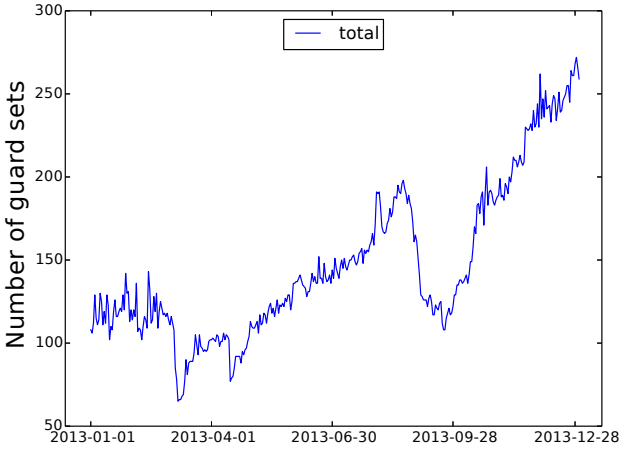


Fig. 5. Number of guard sets.

the fraction of users using each guard set is less than in the first few months of the year. We hypothesize this increment is not as drastic as in previous months due to the large number of guard sets.

By March a large proportion of adversary guards will still be online and so the fraction of users it can compromise will increase. Figure 5 shows that in following months the number of guard sets increase, thereby decreasing the fraction of users using each guard set, and adversary guards will also start to go offline. Hence we see a gradual decrease in the rate at which users are compromised.

Our metric assumes that once a user has been compromised we may consider future uses of Tor compromised. We have demonstrated that in the examined period guard and user sets are relatively stable, and that the number of daily deleted guard sets is a small fraction of the number of total guard sets. Guards with high likelihood, stay within the guard set they were originally assigned to. Our policy of not purposefully rotat-

ing guards limits the strategy an adversary controlling a small fraction of bandwidth can employ to corrupt Tor users. We expect an adversary to maximize the amount of guard sets under initial control. Our scheme therefore offer Tor clients greater anonymity than current schemes providing an adversary does not control a large fraction of guard bandwidth.

Susceptibility to Fingerprint Attacks. In the one guard proposal clients using new low bandwidth guards have a guard that is almost unique to them (in the three guard proposal things are much worse, virtually all users have a unique set of three guards). As of April 2014, Tor metrics estimates there to be 2.75 million global users of Tor, and the most likely set of three guards is 1.7×10^{-6} [7]. Therefore we can expect this set of guards to have 4.7 users, implying most users guards will uniquely identify them. Furthermore, every user has a guard fall back list, and using those leads to a unique guard history and fingerprint for the user.

Our proposal of keeping users in a group removes the chance that an attacker could identify a client based on the guards they used at different periods of time. Initialising on 1 January 2013 produces 108 guard sets, given 2.75 million users this creates 108 user sets of size 25463. Our experiment on tree dynamics showed that guard sets won't lie deeper than four or five layers from initialisation. So at worst there may exist some user sets of size 795, meaning there will always be at least 795 clients with the same guard history. All user sets will have sibling user sets that have the same guard histories prior to the last tree structure change that effected them, creating a hierarchy of user sets with similar guard histories. Guard sets provide a pool of guards to use and so provide resistance against unique guard histories through guard failure.

Therefore the guard sets scheme performs better than current schemes at protecting clients from being identified via their guard history, and provides a large anonymity set for all users to avoid disclosure attacks.

5.2 Diversity and Performance

In both the three and new one guard schemes guards are heavily under-utilized. This is due to bandwidth weights allocating a large fraction of bandwidth for use as a guard but few clients initially using newer guards. In the guard set design new guards are incorporated into an existing or new guard set and immediately used by a large number of users. Figure 6 shows the amount of

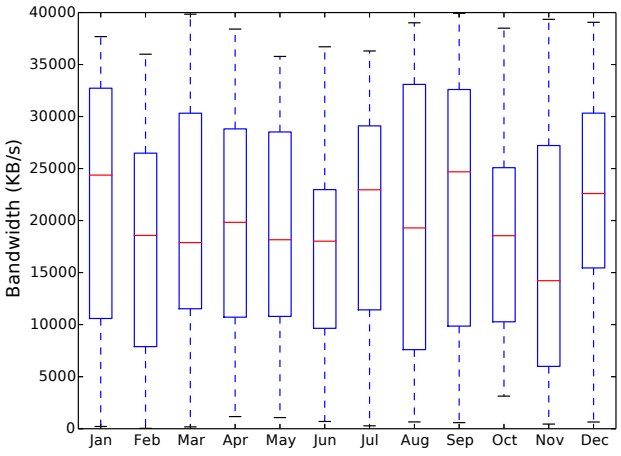


Fig. 6. Spare guard bandwidth

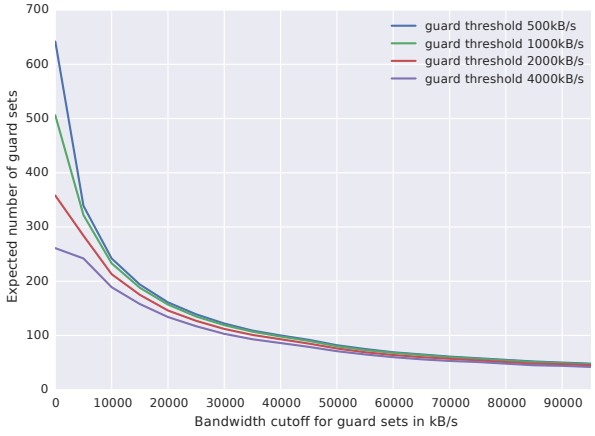
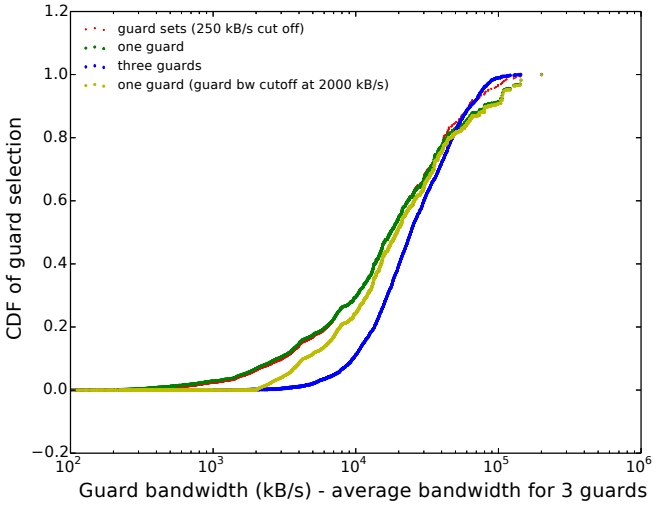


Fig. 7. Number of guard sets for varying cutoffs

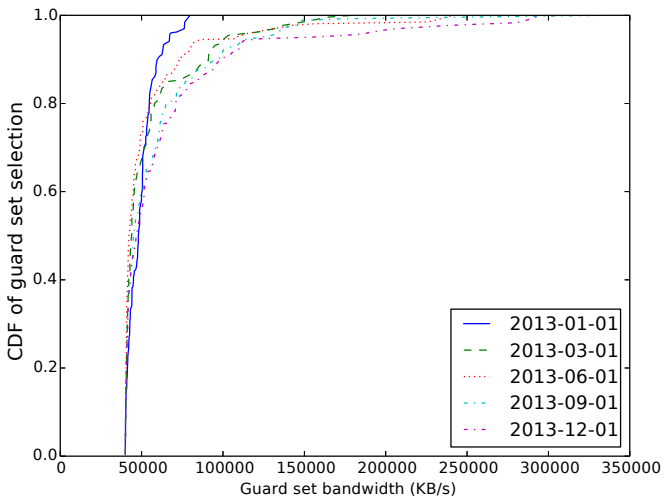
guard bandwidth that will left over[2] . As expected there is never more spare bandwidth than the threshold used to make guard sets, otherwise a new guard set would be created.

At the beginning of 2013 spare bandwidth accounts for 0.46% percent of the total guard bandwidth, and is as low as 0.031% by the end of the year. Since guards may also serve as middle and exit relays, this wastage is low enough to be of no concern. Due to the prioritization of bandwidth quanta from larger guards in constructing guard sets, spare bandwidth is also biased towards smaller bandwidth relays, which do not make a large contribution to total bandwidth on the network.

2 We redistribute available guards to new and existing guard sets at the end of each day instead of at each new consensus document (hourly).



(a) CDF of probabilities of guard bandwidth (consensus bandwidth)



(b) CDF of probabilities of guard set bandwidth (consensus bandwidth)

Fig. 8. CDF plots for guards and guard sets

Figure 7 shows the number of guard sets that would be created for various thresholds (guard and guard set thresholds). We observe that after a cutoff of around 25 MB/s the number of guard sets (and hence the spread of load) is roughly the same for guards pruned at different values. Indicating that raising the limit for becoming a guard has little impact on the number of guard sets created.

Figure 8a shows the fraction of clients using guards of various bandwidths (i.e. performance a client can expect on Tor) against both the one and three guard selection scheme. It was created upon initialisation of guard sets at which point all guard sets have an equal load of users. As we can see, 80% of clients have the same performance in the guard set and one guard proposal. The guard set solution is worse for a small fraction of clients; approximately 90% of clients have performance less than 75 MB/s whereas in the one guard proposal 85% of clients have performance less than 75 MB/s. Note that both proposals are worse than the three guards election scheme; only 10%, 20% of clients have better performance under the guard set proposal and one guard proposal, respectively. Upon initialisation our proposal performs slightly worse than the one guard proposal.

Figure 8b shows we can expect fluctuations in guard set selection statistics as guard sets move up and down the tree, but since guard sets never deviate from one another by more than a few layers these fluctuations will be small and temporary. Upon initialisation 80% of clients use guard sets with bandwidth 50 MB/s or less, and 100% of clients use guard sets with bandwidth 75 MB/s or less. At any time throughout the year 80% of clients use guard sets with bandwidth 75 MB/s or less, and 95% of clients will use guard sets with bandwidth 100 MB/s or less. Figure 8b implies that our guard set scheme is stable and does not create swells of users that can expect a degradation in performance as time progresses.

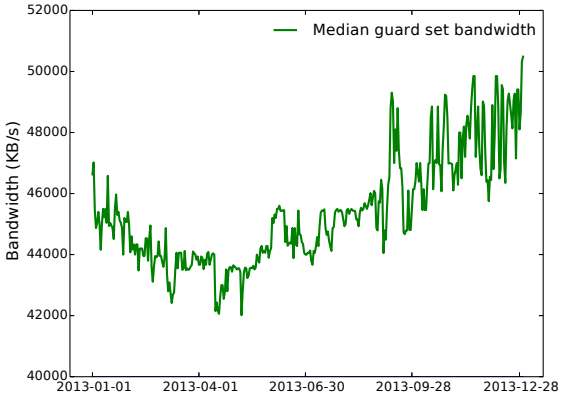
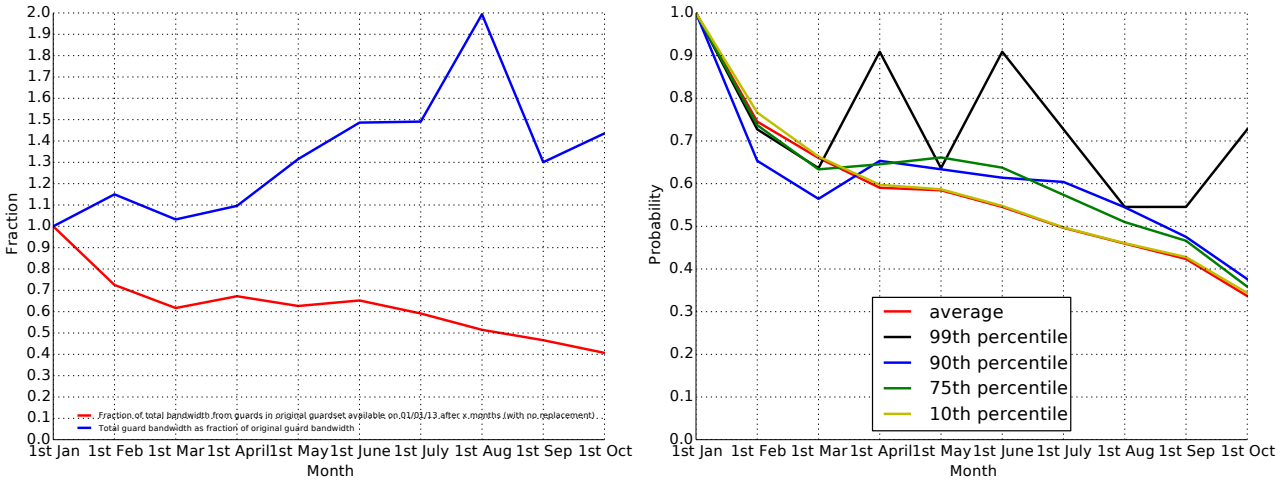


Fig. 9. Median bandwidth per guard set

Figures 8 and 9 imply that any Tor user can expect the same level of performance as any other Tor user. There is no hierarchy of performance level depending on which guard a client begins to use as there are with other schemes.



(a) Guard bandwidth as a fraction of the original guard bandwidth

(b) Probability that guard a client picked on 01/01/13 is still in the network

Fig. 10. Guard bandwidth and availability statistics

Figure 9 shows the median bandwidth per guard set throughout 2013. As expected nearly all guard sets throughout this period are between 40 and 50 MB/s. Cross referencing with Figure 11, we see that as the total guard bandwidth decreases during March so does the median bandwidth per guard set. The fluctuations in median bandwidth per guard set are directly mirrored by the fluctuations in total guard set bandwidth.

Ideally we would like to never change the initial edge structure between guards and guard sets leading to guard rotation and increasing the rate of compromise of clients [14]. So the question becomes, how often do we need to change the edge structure between guards and guard sets? Figure 10a shows what would happen to our total guard bandwidth if we decided to only use the guards from the initial creation and not use any new eligible guard relays. After 3 months the total available guard bandwidth has increased by 10% but only 68% of the initial guard bandwidth is still available. The trend continues: after 10 months the total guard bandwidth increases by over 40% but only 40% of the initial guard bandwidth is still available.

We conclude it is necessary for guards to be added in to guard sets to replace bandwidth lost over time. This is again highlighted in Figure 10b which shows the probability that a guard selected at the creation of guard sets is still available over a period of 10 months (for various percentiles). We see that even for Tor guards with bandwidth in the top 10% there is less than a 40% chance they are still available after 10 months

Although there is a need to replace dropped guards in guard sets regularly, permanent deletion of guard sets is an infrequent event for a guard set threshold of 40MB/s. Figure 5 and appendix B shows the daily number of total, deleted and added guard sets throughout 2013. The number of daily new and deleted guard sets remains small despite the number of total guard sets increasing. By the end of 2013 we have over double the initial number of guard sets. Experimentally we calculated that at 40 MB/s threshold roughly $< 10\%$ of guard sets will be deleted at the end of every day, with this dropping to $< 3\%$ near the end of 2013. We also found that with a 50 MB/s threshold the number of guard sets that warrant deletion at the end of each day is much higher than at 40 MB/s. This provided experimental support for our threshold. If we were to set a low threshold we create many guard sets, which limits the number of guard sets an adversary can corrupt, but we can expect a higher rate of churn contributing to an increase in compromise rate and possibly allowing for attacks via strategic injections of corrupt guards to guard sets. A higher threshold will provide faster guard sets but will create fewer in number, making it easier for an adversary to control a larger fraction of the total number of guard sets.

As shown by Figures 5 and 11, as total guard set bandwidth increases so does the number of guard sets, making it increasingly difficult to predict which back up guard sets a particular guard set may use. This increase in diversity of guard sets will simultaneously decrease the compromise rate of clients while increasing expected performance.

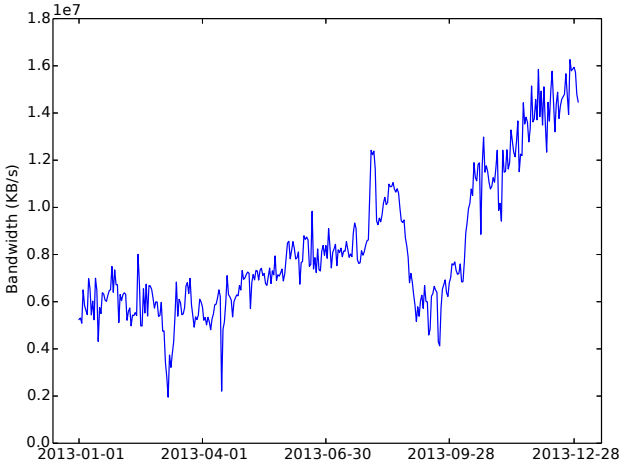


Fig. 11. Total guard set bandwidth

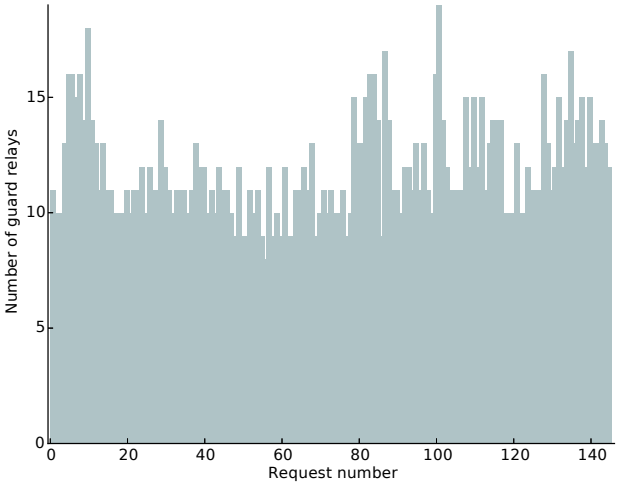


Fig. 12. Number of relays that were reported as down for each request. The request was made every 10 minutes over a 24 hour period. Using the 2014-10-25-11-00-00-consensus, the requests were made on 2014-10-25/26 from 11pm - 11pm.

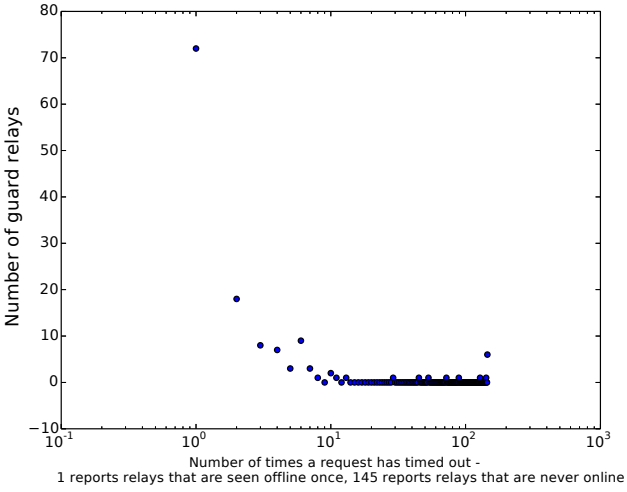


Fig. 13. Number of times requests have timed out over 24 hour period.

Guard Set Creation & Deletion due to Temporary Bandwidth Changes. To understand the dynamics and stability of guard sets we must know how often guards go offline temporarily and reappear online. Consensuses are issued every hour so it may be that guard sets are affected by higher frequency guard churn that is not being accurately captured. If guards were reported as online and offline frequently we may have to swap guards in and out of guard sets regularly which would increase the rate of compromise. Using a consensus from 25/11/14[3] we recorded how many guards are online and offline every 10 minutes over a 24 hour period. Every 10 minutes we sent an ICMP echo request to all guards and for all guards that timed out we sent a TCP scan to check if the relay was down or if they were not accepting echo requests.

Figure 12 shows the total number of relays that were reported as down every 10 minutes over the 24 hour period. The number of guards that are down during this period is small and remains nearly constant. During this period the lowest number of reported online guards was 1509 out of 1524, and the highest was 1515. Hence we can conclude that the ratio of offline to online guards remains stable.

Figure 13 shows the number of relays that were reported as offline. Over 70 relays out of 1524 we were reported as offline once over the 24 hour period. Following this the number of relays reported as offline for more than one request drops off; the number of relays reported as being offline for two requests (out of 145 requests) was below 20. The number of relays being seen as offline for more requests steadily decreases until 145 where there is a sharp increase due to some guards being offline for the entire 24 hour period. This implies that time outs are commonly due to an isolated incident or relays permanently going offline, with almost no middle ground.

Figure 14 shows the number of times a guard will be reported as online / offline given that it has been seen offline for x number of previous requests. We see that there is a 1.88% chance of a guard being reported as on-

3 Using an older consensus would result in the identification of many guards being offline for the entire request period due to them having dropped out of the network before the request was started.

line once it has been reported as offline for two requests. As expected the probability of a guard coming back online decreases the longer it is reported offline. This shows that while the number of guards reported offline is relatively small, for the unfortunate clients using a guard that drops offline there is little chance the guard will come back online. In the one guard proposal this leads to a possible fingerprinting attack, due to clients repeatedly using back-up guards, which eventually leads to unique guard histories for clients. In our proposal this is not a problem because nearly all clients will have multiple guards to choose from, and in the event that a guard set is deleted, users in the same user set will use the same back-up guard set.

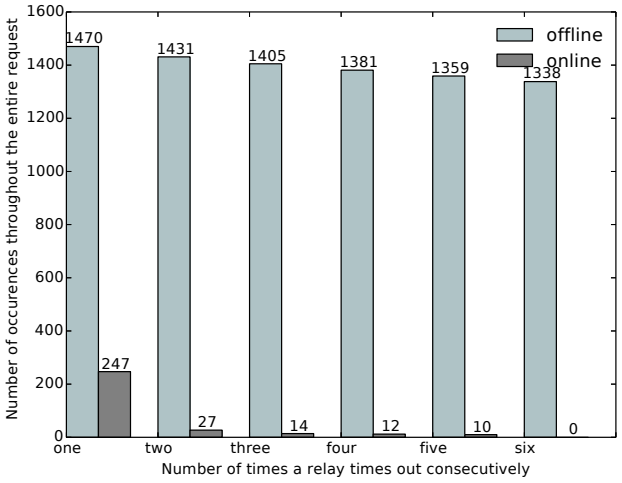


Fig. 14. This shows the number of times a relay may be seen offline and then in the following request is see offline again or is seen back online. For example we see there is a 14.4% chance your relay will be back online after timing out of one request, whereas there are no reported instances of a relay being down for six consecutive ping request and then coming back online.

Figure 15 shows the number of guard sets that dropped below various thresholds during a 24 hour period. We see that nearly all guard sets keep all their bandwidth, and there is very little churn. Only a few times does a guard set drop to 0 MB/s at which point we would move the user set to a back up guard set. At creation there were 468 guard sets created indicating guard set deletion due to bandwidth changes over short periods of time will happen infrequently.

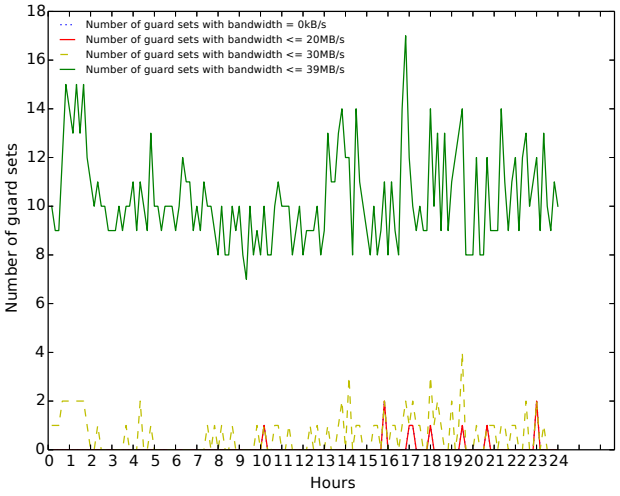


Fig. 15. Number of guard sets below bandwidth thresholds over a 24 hour period. Each request was made to over 1500 guard relays, creating 468 guard sets.

Future Work & Conclusion

This is a first study of how guard sets could work, and a number of practicalities need to be fleshed out before realistically considering deployment, which are beyond the scope of this work. Further security and performance analysis may lead to fundamentally different algorithms, and so we deliberately did not study in depth any specific route to deployment in this work.

More specifically, in our security analysis we considered only adversaries at Tor relays, ignoring the network between them. This has been an active area of research [9, 11, 14, 17, 20]; before deployment IX and AS-level adversaries should be considered. Seemingly, guard sets may suffer more than other guard designs; since guard sets are formed based entirely on bandwidth criteria it is more likely that guard sets with a larger number of guards, and so compromised of relays from a wide range of Autonomous Systems, would be more more vulnerable to an AS-level adversary. We note that Tor attempts to mitigate this to some extent; in a circuit relays need to be from different /16 subnets, and families of relays from the same operator can't be used in the same circuit.

Other security concerns for which we provide arguments against but would require formal proofs before deployment include:

1. **Adversarial manipulation of the algorithm.** Specifically, the optimal strategy an adversary will

use to corrupt guard sets. We show in section 5.1 the scheme provides strong security regardless of whether an adversary maximizes the number of controlled guard sets.

2. **DoS attacks on guard sets - targetting specific clients.**
   The tree structure attempts to mitigate these attacks via the random placement of new guard sets and making it difficult to predict the placement of back-up guard sets (section 3). These attacks are hard to totally prevent but expect the scheme to perform at least equally as well as the current guard schemes.
   We did not attempt to investigate how guard sets would cope under DoS attacks such as the Sniper Attack [13], where the intent is to simply observe more clients. We expect that guard sets may suffer more than other entry schemes if the attack is to maximize the number clients observed by the adversary.
3. **Resistance to fingerprinting attacks.**
   We currently do not formally prove resistance to fingerprinting attacks, but it would be possible to make an indistinguishability argument based on ID prefixes being clients' only used state and those being shared within an anonymity set.

A key question remains open: who manages the assignment of guards into guard sets? Answering this fully could easily be the subject of a separate work. We however note that Tor maintains a consensus of the set of all relays, and our algorithm can be formulated to be deterministic on the basis of this consensus document and a source of public but unpredictable randomness. Such randomness may be produced through a distributed fair coin flipping protocol [16] or through centralized mechanisms such as the NIST Beacon [12]. Thus given a sequence of consensus sets and a public but unpredictable random seed, any client may re-execute the algorithm that assigns guards to guard sets. However, due to the unpredictability of the seed an adversary may not game their position within guard sets. Thus, authorities only need to compute a consensus as they currently do, and execute the publicly verifiable assignment for each epoch. Clients may verify this assignment, but can also use it without verifying it - considering that a consensus and assignments signed by all authorities is trustworthy. We have outlined the first steps to a new entry guard scheme for the Tor network. The guard set proposal addresses many of the security concerns that both the three and one guard scheme are susceptible to while performing equally as well. The resulting guard sets are:

Plentiful. The number of guard sets provide enough diversity to prevent compromise of a large fraction of the Tor user base.

Stable. Due to relay stability and the way we created guard sets, guard set deletion is a rare occurrence. The low natural rotation of guards through guard sets limits the potential for direct observation of clients. The shared history, even under failure, eliminates fingerprinting attacks.

Fair. Most guard sets have equal bandwidth capacity. Clients can expect the same performance no matter their choice of guard set.

Large. All guard sets serve roughly equal sized user sets, and a large number of users at any time. This prevents statistical attacks on the basis of discovering a user's guards.

Guard sets allocate traffic on the Tor network so each relay is used efficiently. At any scale of network bandwidth capacity our proposal will offer every client the same level of performance as any other client. As network bandwidth increases, the number and stability of guard sets increase thereby decreasing the rate at which an adversary can compromise Tor users.

Acknowledgments. Work on this paper was supported by EPSRC Grant EP/M013286/1 on "Strengthening anonymity in messaging systems". The authors would like to acknowledge financial support from the UK Government Communications Headquarters (GCHQ), as part of University College London's status as a recognised Academic Centre of Excellence in Cyber Security Research. We thank the anonymous reviewers for their comments, particularly with regards to deployment and security. We thank Steven J. Murdoch for his helpful early comments and Paul Syverson for shepherding this work.

References

[1] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 92–102, 2007.

[2] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[3] George Danezis and Andrei Serjantov. Statistical Disclosure or Intersection Attacks on Anonymity Systems. In *Information Hiding, 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers*, pages 293–308, 2004.

[4] George Danezis and Paul F. Syverson. Bridging and Fingerprinting: Epistemic Attacks on Route Selection. In *Privacy Enhancing Technologies, 8th International Symposium, PETS, 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*, pages 151–166, 2008.

[5] Roger Dingledine. Improving Tor's anonymity by changing guard parameters. https://blog.torproject.org/blog/improving-tors-anonymity-changing-guard-parameters, October 2013.

[6] Roger Dingledine. Tor security advisory: relay early traffic confirmation attack. https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack, July 2014.

[7] Roger Dingledine, George Kadianakis, Nick Mathewson, and Nicholas Hopper. One Fast guard for Life (or 9 months). In *7th Workshop on Hot Topics in Privacy Enhancing Technologies, HotPETs, 2014, Amsterdam, The Netherlands*, 2014.

[8] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.

[9] Matthew Edman and Paul F. Syverson. AS-awareness in Tor path selection. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 380–389. ACM, 2009.

[10] Tariq Elahi, Kevin S. Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the guards: a framework for understanding and improving entry guard selection in Tor. In *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012*, pages 43–54, 2012.

[11] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In Vijay Atluri, Paul F. Syverson, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 66–76, 2004.

[12] National Institute for Standards and Technology. Beacon, http://www.nist.gov/itl/csd/ct/nist_beacon.cfm, 2015.

[13] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[14] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. Users get routed: traffic correlation on Tor by realistic adversaries. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 337–348, 2013.

[15] Nick Mathewson and Roger Dingledine. Brainstorm tradeoffs from moving to 2 (or even 1) guards. https://trac.torproject.org/projects/tor/ticket/9273, July 2013.

[16] Tal Moran, Moni Naor, and Gil Segev. An Optimally Fair Coin Toss. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[17] Steven J. Murdoch and Piotr Zielinski. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, volume 4776 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 2007.

[18] Lasse Øverlier and Paul F. Syverson. Locating Hidden Servers. In *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*, pages 100–114, 2006.

[19] The Tor Project. Frequently Asked Questions – Entry Guards. https://www.torproject.org/docs/faq#EntryGuards, 2015.

[20] Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. An Empirical Evaluation of Relay Selection in Tor. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society, 2013.

[21] Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004.

Guard set Layer Differences

The daily difference in tree layers for guard sets in 2013:

[4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 5, 5, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6,
5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4,
4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 4, 4, 4, 5, 5, 5, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 3, 3, 3, 4, 5, 4, 4,
4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 5, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 4, 5, 5, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Guard set creation & deletion rate

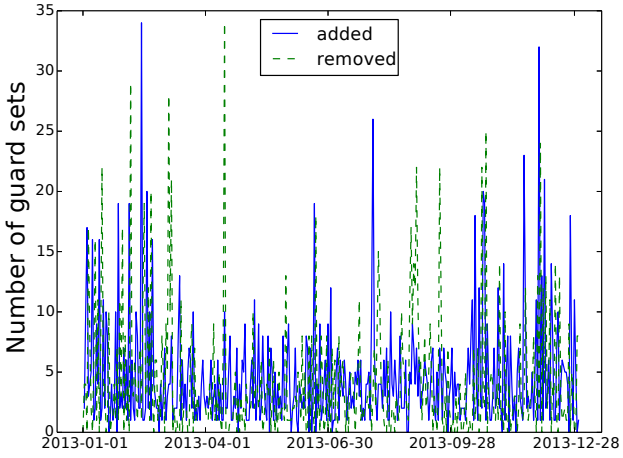


Fig. 16. Number of guard sets created and deleted.