Hidden-service statistics reported by relays

David Goulet
The Tor Project
dgoulet@torproject.org

Aaron Johnson
U.S. Naval Research Laboratory
aaron.m.johnson@nrl.navy.mil

George Kadianakis
The Tor Project
asn@torproject.org

Karsten Loesing
The Tor Project
karsten@torproject.org

Tor Tech Report

Abstract

In order to understand how hidden services are being used in Tor, relays can collect and report statistics about the hidden-service activity they observe. However, such statistics might harm the privacy of individual Tor users. We describe how relays can be used to collect statistics about hidden services, how they might be useful, and how they can be published in a way that protects individual privacy.

1 Introduction

Tor hidden services [4, 23] are a means to provide a network service while hiding the location of the server. To provide this anonymity to the server, client connections are set up using a different process than is used by connections to known servers. This process comes with a performance cost. Also, servers must be configured explicitly to be accessed as a hidden service. As a result, the way that Tor hidden services are being used may be quite different from the way that Tor is used to anonymize traffic to normal, non-hidden Internet services.

Therefore, in addition to the general statistics that Tor collects about its network [22], it is interesting and useful to collect statistics specific to hidden services. For example, hidden-service statistics could show how popular they are, how well they perform, and if they are under attack. Publishing such data could help Tor to manage and improve its network, Tor advocates to argue more effectively in support of Tor, and researchers to understand how Tor is being used.

Tor relays are frequently in a position to learn some information about hidden services and their usage. Relays have long collected and reported information for many *general* statistics of interest (i.e. those that are not specific to hidden services), such as the amount of total traffic

that they have relayed. They do this by including these data in "extra-info documents" [21], which are uploaded to the Directory Authorities. These local observations can be combined to give a global view of Tor's hidden service activity.

However, Tor's privacy protections depend on relays not sharing all their local observations. Therefore, we must carefully consider how to publish data in a way that doesn't hurt Tor's privacy goals. Tor has adopted some privacy protections for the general statistics it gathers. These protections include reporting data aggregated over a period of time and rounding some measurements to a desired level of accuracy. There is a set of privacy objectives and risks that are specific to hidden services, however, and so these techniques may not apply directly. For example, hidden services have a primary privacy goal of hiding the server's location and a secondary goal of allowing the existence of the service itself to be hidden.

Thus, to address the particular challenge of studying Tor hidden services, this report describes how relays can safely collect useful statistics about them. It includes the following contributions:

- A description of the types of observations that relays can make about hidden services
- A description of the privacy objectives specific to hidden services
- A description of obfuscation techniques that can be used to safely release hidden-service statistics

2 Hidden-service protocol and measurement points

The hidden-service protocol consists of multiple steps for a service to become available through the network and for a client to connect to a service [23]. Each step gives some relays the opportunity to gather statistics about their role in the hidden-service protocol. Hidden service clients and servers also are in a position to gather statistics, but in this report we focus on collection by relays. The major roles of relays in the hidden-service protocol are: a) Hidden Service Directory (HSDir), b) Introduction Point (IP), and c) Rendezvous Point (RP). Figure 1 gives an overview of protocol steps.

The protocol steps for making a service available through the network are as follows:

1. The service *establishes an Introduction Point* on one or more relays. A relay first receives a circuit extension request from another relay to become the next relay in a circuit. At this time the relay does not recognize that it will become an Introduction Point. Then the relay receives a request to establish an Introduction Point on behalf of the service that built the circuit. However, the Introduction Point does not know which service it's serving, because the service creates a fresh identity key for each Introduction Point. The circuit, now called introduction circuit, will be kept open until the service closes it. From that time on the relay accepts client introductions for the service coming in via other circuits.

2. The service *publishes a descriptor* to a total number of six Hidden Service Directories, which are the set of relays with high-enough uptime as indicated by the "HSDir" flags in the network consensus. Each of those relays receives a circuit extension request followed by a request to publish the descriptor. The relay can read most parts of the

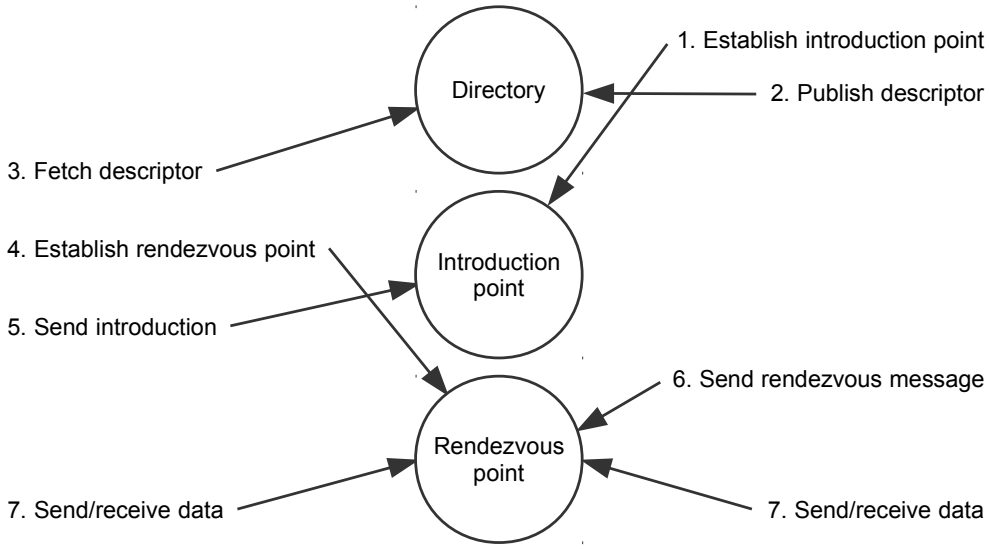

Figure 1: Hidden-service protocol steps as observed by relays

descriptor, including the service identity (this may change as part of a proposed protocol update [17]) and selected Introduction Points. The circuit that was built by the service is closed immediately after transmitting the descriptor.

The protocol steps for a client connecting to a service are as follows:

3. The client *fetches a descriptor* from a Hidden Service Directory. The relay that acts as an HSDir receives a circuit extension request followed by a request to fetch the descriptor. Once the descriptor is returned, or a response is returned saying that the descriptor does not exist, the circuit is closed immediately.

4. The client *establishes a Rendezvous Point* on a relay. The relay observes a circuit extension request, followed by a request to become the client's Rendezvous Point for connecting to a service. The relay does not learn which service the client attempts to connect to, only a random identifier. The circuit, now called rendezvous circuit, is kept open until either the client or service closes it.

5. The client *sends an introduction* to one of the service's Introduction Points. The Introduction Point observes a circuit extension request followed by the introduction message, which it forwards to the service via its introduction circuit. The client's circuit is torn down by the relay after receiving the introduction and responding with an acknowledgement.

6. The service *sends a rendezvous message* to the client's Rendezvous Point. The Rendezvous Point sees a circuit extension request followed by the rendezvous message, which it forwards to the client via its rendezvous circuit. Both service-side and client-side circuits are kept open until either side decides to close it.

7. Both *client and service send and receive data* along their part of the rendezvous circuit. The Rendezvous Point sees cells coming in from either side and forwards them to the

other side. All cells are padded and end-to-end encrypted between client and service, so that the Rendezvous Point only sees encrypted cells of the same size.

All of these protocol steps constitute potential measurement points for hidden-service related statistics. For example, some Tor relays have been reporting the following statistics about hidden services, which were proposed by Kadianakis et al. [1] and the results analyzed by Kadianakis and Loesing [16]:

1. NumRPCells: The number of hidden-service data cells seen at a Rendezvous Point, reported every 24 hours
2. NumHSDirIDs: The number of unique hidden identifiers seen at an HSDir, reported every 24 hours

3 Evaluation criteria for statistics

A proposal to collect some hidden-service statistics must be evaluated according to several criteria. The main criteria are the benefits to Tor, the effect on privacy, the reliability of the statistics, the efficiency of the collection process, and alternatives to live collection.

3.1 Possible benefits from gathering statistics

There are several compelling reasons to collect information about hidden service statistics. We describe some of them to illustrate the possible benefits.

Learn about usage Understanding how hidden services are being used can help both to direct development efforts and to justify them. Usage statistics could provide insight into how many hidden services exist, how available they are, how many clients they have, how much Tor traffic is hidden-service traffic, and what types of applications use hidden services. These data could help Tor developers decide how much effort to spend improving hidden services and what types of changes would most benefit users. They can also be of use to researchers that analyze Tor and design improvements. Hidden-service statistics would also provide evidence that can be used in efforts to improve their media coverage, to persuade potential funders, and to inform government policy.

Improve performance Data about hidden services can help Tor developers improve performance. Such data might include speed and failure rates for the steps of the hidden-service protocol, resource utilization at the relays, and hidden-service availability. This information could be used to improve load balancing across relays, to optimize the values of various parameters in the hidden-service protocol, and to add relay resources where they would be most effective. They could be used to inform more ambitious redesigns of the hidden-service protocols. They could also be used to build models of client, network, and hidden-service behavior to use in network simulations done in support of performance improvements.

Identify bugs Statistics can help detect bugs in hidden services. This could be especially useful to identify problems those that occur infrequently or only under realistic conditions, such the network traffic load or the mix of Tor software versions used. Information that could be particularly useful for this purpose includes failures rates and the frequency of unexpected events.

Discover attacks Hidden-service statistics may uncover ongoing attacks in the network. Denial-of-service attacks in particular might be detected from spikes in usage and failure statistics. Statistics about the utilization of relay resources such as CPU, memory, and bandwidth might also reveal variations that are unexpected under normal use. In addition, data about the software versions may reveal client Sybil attacks or botnet abuse [13].

3.2 Hidden-service privacy goals

Gathering statistics on Tor has implications for its privacy because the data is determined from relays' observations of the private behavior of Tor users. Tor statistics are published and thus may be used by an adversary to determine the private activity of Tor users.

Tor has several privacy goals. Primary, of course, is to provide anonymity to the source of a connection and to hide the network location of a hidden service. However, there are several other related and secondary goals. Complementing user privacy, for example, is the goal of unlinkability among separate Tor connections, which prevents an adversary from creating a pseudonymous profile of a user's network activity. We focus on the various privacy goals specific to hidden services, which to our knowledge have not previously been explicitly articulated. Our listing of privacy goals is not exhaustive. Tor's default position for information about the use of its network is that it should be protected, and exceptions should be made only in specific, carefully-considered cases. The privacy goals we highlight are those for which choosing to contradict them is very unlikely.

Information about a specific hidden service The following information about any specific hidden service should remain private:

1. The network location of the hidden service
2. The exact number of hidden services (i.e. the existence of the hidden service)
3. The number of clients that have used the hidden service
4. The number of connections to the hidden service
5. The amount of traffic to or from the hidden service
6. The availability of the hidden service

Observe that including several of these amounts to a goal of allowing a hidden service to remain undiscovered or undistinguished. This allows a hidden service to maintain a low profile and not come under investigation or scrutiny by an observant adversary. Also note that these are not all currently well-protected by Tor. For example, the existence and popularity of a hidden service can currently be learned by an HSDir [2], although there are plans to close this information leak [17].

Information about a specific hidden-service client Related to the above, it is a goal to hide information that is specific to a given hidden-service client, including

1. The number of hidden-service connections of the client
2. The amount of the client's hidden-service traffic
3. When a client is active on hidden services

These privacy goals also exist more generally for clients' Tor activity that doesn't use hidden services.

Information about a specific hidden-service connection It is also a goal to hide information about any given hidden-service connection, including

1. The client and the service of the connection
2. The exact number of hidden-service connections
3. When a connection occurred
4. How much traffic was transferred over some connection
5. If the connection shares a client or hidden service with another given connection

As with specific client information, it is a privacy goal more generally to protect information about any specific client connection.

Information about ongoing activity Statistics should not reveal information relevant to the *ongoing* activity of clients, hidden services, or connections. This should be maintained even when the information is safe to release *after* the activity has ceased. For example, it should not be revealed that certain relays are currently being used as Introduction Points by some hidden service, although it may be reasonable to reveal that some relays have been used as Introduction Points in the past. The risk of revealing ongoing activity is that an active adversary can use that information to begin targeted surveillance of that activity.

3.3 Reliability

An important criterion for a statistics-gathering procedure is its reliability. To be reliable, data must be accurate, available, robust to network changes, and resilient to malicious attacks. This criterion becomes especially important when a statistic affects the operation of Tor in real time. For example, the measurements gathered by TorFlow [19] are used to determine relay weights in the hourly consensuses.

Robustness to future network changes is an important concern in the Tor network, which frequently improves in response to user needs and adversarial capabilities. For example, there are several current proposals to change hidden services in minor [15, 12] and major [17] ways. Statistics collection shouldn't be broken or precluded by such protocol changes.

Malicious attacks are another particular concern for Tor, because it operates in a highly-adversarial setting that is not typical of many other networks and services. When relays collect and report hidden-service statistics, the process is particularly vulnerable to manipulation by malicious relays, which can provide completely arbitrary data. An ideal procedure for gathering statistics would be robust to attempted manipulation by a small fraction of the network.

3.4 Efficiency

Statistics collection should not place an undue burden on Tor's existing infrastructure. It should be designed to minimize bandwidth consumption, CPU utilization, and memory requirements. Thus it may be necessary to limit the frequency at which statistics are updated or the complexity of a cryptographic aggregation procedure [9, 8].

3.5 Alternatives

Some of the possible benefits of measuring the live Tor network may also obtained by measuring private or simulated [14, 20] networks. For example, private networks can be used to find bugs by testing hidden services under various conditions and looking for unexpected behavior. Using these methods eliminates the privacy risk that accompanies measurement of the live network, and so such methods should be preferred when possible.

4 Obfuscation methodology

There is a variety of available methods to produce general statistics about a dataset without revealing specific sensitive details. We briefly discuss how this might be done in the context of hidden services, and then we provide two algorithms for revealing count and distributional statistics that can be applied to a variety of specific hidden-service statistics.

Some useful techniques for privately publishing statistics on hidden services are

- Releasing aggregate statistics over time, such as total counts or averages in a given period
- Adding noise (i.e. random inaccuracy)
- Limiting accuracy to a certain granularity via rounding (aka "binning")
- Adding time-delay to the release of statistics such that the output doesn't reveal information about ongoing activity
- Using cryptographic techniques to hide the source of information, such as anonymizing reports from individual relays

However, in general, publishing any information that is broadly informative might break privacy, because that information may be just what the adversary needs to know in order to learn a sensitive fact [5]. Therefore, a discussion of privacy on Tor should consider plausible background knowledge for an adversary. We can expect that the adversary may know things such as

- The addresses of a large number of publicly-available services (e.g. by crawling the Web)
- A minimum amount of traffic received by a given hidden service (e.g. due to sending that traffic himself)
- The Introduction Points of a service (e.g. by obtaining the descriptor)
- The availability of a service (e.g. by attempting to connect periodically)
- Roughly the number of client connections and amount of client traffic for a service (e.g. a web forum that reveals user activity on the site)

Given such information, it can be seen that certain statistics cannot be published by individual relays with reasonable accuracy without violating some privacy. For example, if each relay accurately reported the number of client introduction requests it received, then

an adversary that knows the `.onion` address of a hidden service (and thus can obtain its Introduction Points) could infer how many client connections the hidden-service received, especially if there were few other hidden services sharing its Introduction Points. The general problem is that, for certain types of hidden-service activities, different hidden services or clients will make use of different relays in a way that may be known by the adversary.

We give two algorithms for privately publishing statistics that do not have this problem, that is, that can be released by each relay. Two examples of such statistics are those described by Kadianakis et al. [1] (also mentioned in Section 2):

1. NumRPCells: The number of hidden-service data cells seen at a Rendezvous Point
2. NumHSDirIDs: The number of unique hidden identifiers seen at an HSDir.

The first algorithm provides a way to publish statistics that are simple running counts, including both NumRPCells and NumHSDirIDs. The second algorithm provides a way to publish statistics about a distribution of values.

Our algorithms will add noise in a way that provides differential privacy [7] for "single actions". What constitutes a single action will depend on the specific statistic. For example, Kadianakis et al. [1] use the following for their statistics:

1. For NumRPCells, a single action is transferring a fixed amount of data (specifically, 2048 cells). Thus the added noise obscures the activity of a single service, client, or connection up to that amount of traffic.
2. For NumHSDirIDs, a single action is publishing a fixed number of hidden services (specifically, 8). Thus the added noise obscures the existence of any eight or fewer hidden services.

To obtain differential privacy, our algorithms add noise using the Laplace distribution, which has a probability distribution function $\mathsf{Lap}(b)$ of $f(x) = e^{-|x|/b}/(2b)$. We will choose b such that altering a single action will change the probability of the total output by a factor of at most e^{ϵ}. Thus more privacy is provided the smaller that ϵ is.

4.1 Counts

Reporting a basic count is useful in its own right, and it also provides a simple setting in which to develop privacy-preserving methodology that we can use for more complicated statistics. Counts can be used to summarize many types of hidden-service activity, such as the number of hidden services, the number of hidden-service clients, and the amount of hidden-service traffic.

To release a single count, we will use ideas from differential privacy to provide strong protection for a single hidden-service action. However, we wish to continually publish statistics over time, and as a result differential privacy, using a per-user or per-service privacy notion that includes actions over time, does not provide an adequate solution. The reasons are (*i*) mechanisms using global sensitivity [7] that would apply over time require amounts of noise that grow with the reporting period (potentially years in our case), and (*ii*) improving the accuracy using local mechanisms [18] is not feasible because the Tor protocol by design hides which activities correspond to the same user or service.

There are several challenges to privately publishing statistics over time. One is that, although the effect of a single action may be made difficult to determine in any given statistic, the collective set of statistics may reveal some level of activity by the same user or service. This problem of a single sensitive fact influencing many published statistics was effectively exploited

by Homer et al. [10] to identify if an individual was a member of a diseased study group based only on per-gene statistics (where here data per-gene replace data per-time-period). Another challenge is that it may be possible to remove the noise added to the published values if "fresh" noise (i.e. noise generated using new randomness) is added to each statistic. For example, if a count stays the same for a while, and the adversary knows that (or can guess it with some confidence), then the adversary can get a good estimate of the count by taking the mean of the sequence of published noisy counts. Simply reusing the same noise isn't adequate, however, because in that case the statistics would reveal with certainty all changes in the statistic and thus all activity since the fresh noise was chosen.

To handle this problem, we will use simple rounding or "binning". This will hide changes to a count that keep it in the same bin. Of course, this won't hide the effects of an action if the count happens to be near the rounding threshold, and also the adversary can in most cases himself perform actions that alter the count to attempt to determine the location of the count within the bin. However, we will mitigate both of these issues by making the bin output itself noisy.

We suggest the following to privately publish a count c:

1. Repeatedly maintain a running count c over a time period of length t.
2. Choose a bin granularity δ_1, which should be large enough to contain the change in a single statistic due to a *repeated* action, that is, an action repeated over many reporting periods that we wish to hide any single instance of.
3. Round c to the nearest multiple of δ_1, that is, let $\hat{c} = \delta_1[c/\delta_1]$, where [] indicates the nearest integer function. Rounding up (as in Kadianakis et al. [1]) or down can be performed instead for a similar effect on privacy, although they introduce a bias that should be corrected for [16].
4. Choose a value ν from the $\mathsf{Lap}(\delta_2/\epsilon)$ distribution. ϵ is the privacy parameter discussed in Sec. 4. δ_2 is the amount by which a single (non-persistent) action that we wish to hide could change the bin center.
5. Let the noisy count be $\tilde{c} = \hat{c} + \nu$.
6. Wait until all activity contributing to count c should have completed, and then publish $\tilde{c}$.

Example 1: Kadianakis et al. [1] use this algorithm to publish NumRPCells over periods of length $t = 24$ hours. The waiting period after collection is implicit in the length of time until the next extra-info descriptor is published by the relay, which happens every 18 hours unless certain relay properties change [21]. They use $\delta_1 = 1024$, $\delta_2 = 2048$, and $\epsilon = 0.3$. This setting of δ_1 hides any repeated transfer of up to 1024 cells. Tor cells have up to 509 payload bytes [3], and so this is 509 KiB. This setting of δ_2 hides any group of up to 2048 transferred cells (1018 KiB) (e.g. a connection in which a client downloads 1 MB). One way of interpreting the privacy from $\epsilon = 0.3$ is from a Bayesian perspective [11], where if a priori, for any given set of other transfers, the adversary believed a given transfer of up to 2048 cells did (or didn't) occur, then the statistics can only increase his confidence by at most $1 - e^{0.3}$ (i.e. $\sim 35\%$).

Example 2: Kadianakis et al. [1] use this algorithm to publish NumHSDirIDs over periods of length $t = 24$ hours, waiting to publish with the relay's next extra-info descriptor. They use $\delta_1 = 8$, $\delta_2 = 8$, and $\epsilon = 0.3$. These settings for δ_1 and δ_2 hide any single or repeated

publication of any given group of at most 8 onion services (e.g. a set of 8 or fewer related onion addresses that are aliases for the same onionsite).

4.2 Distributions

For many statistics, it would be very helpful to understand the distribution of values. For example, such information about the number of cells on a single circuit could reveal if most hidden-service traffic is produced by relatively few connections.

Following are several potential ways to publish information about a distribution:

- Publish a histogram of possible values (e.g. the number of values in $[0,1)$, $[1,10)$, $[10,100)$, and $[100,\infty)$).
- Publish a subset of percentile values (e.g. quartiles).
- Publish standard summary statistics, (e.g. mean, variance, skew, and kurtosis).

To protect individual privacy when releasing these kinds of data, we would again like to protect activity over time and also provide particularly-strong protection for a single activity. This is quite straightforward to do for publishing histograms simply by applying the techniques that we have developed for counts to each count in the histogram. Thus we suggest using histograms in this way to report distribution data, as follows:

1. Choose a finite number k of *buckets* that cover the possible values of the statistic (we use the term "buckets" to distinguish these from bins that will limit the granularity of each bucket). Each extra bucket will result in a certain additional amount of noise being added, but including more values in a bucket (i.e. increasing its width) reduces its accuracy. Therefore, the number and width of the buckets should be balanced while also choosing buckets to capture the most useful distinctions for the statistic under consideration (e.g. deciding between relative and absolute accuracy).
2. Over repeated time periods of length t, maintain running counts c_i, where c_i is the count of observed values in the ith bucket.
3. For the ith bucket, the count c_i of values in that bucket should be rounded to a chosen granularity δ_1: $\hat{c}_i = \delta_1[c_i/\delta_1]$. δ_1 should be at least the amount by which a single instance of a repeated action that we want to hide could change the bucket count, where again the notion of a single action depends on the context. The δ_1-sized bins here serve the same purpose of protecting privacy over time that they did when publishing a single count.
4. Fresh Laplace noise ν_i with distribution $\mathsf{Lap}(2\delta_2/\epsilon)$ should be added to the center of the bin of the ith bucket, where δ_2 is at least the amount by which a single (non-repeated) action that we want to hide could change the bin center. Let the resulting value be $\tilde{c}_i = \hat{c}_i + \nu_i$. The value 2 appears in the Laplace parameter because modifying a single entry in the histogram can change two values: the value of the bucket it was changed from and the value of the bucket it was changed to.
5. Wait until the activity contributing to the histogram should have completed, and then publish each noisy bucket count, $\tilde{c}_i$, $1 \leq i \leq k$.

5 Future Work

There are many possible ways to expand the hidden-service statistics that we can safely gather and to improve their accuracy. One promising option is to securely perform aggregation such that no single party can learn more than the final aggregate value. This would allow collecting statistics that cannot be reported by each relay individually, as previously discussed.

In addition, secure aggregation could reduce the amount of noise that needs to be added. With per-relay reporting, rounding and noise is applied to each relay's statistics when using the algorithms described in Sections 4.1 & 4.2. However, the same noise would provide equivalent protection if applied only once to the statistics after network-wide aggregation. This would reduce the amount of added noise and rounding inaccuracy by a factor of m, where m is the number of relays.

There are some promising approaches to secure statistics aggregation including

- Have the relays provide inputs to a secure multiparty computation (SMC) protocol that produces the desired statistics with any privacy-preserving modifications included (e.g. added noise) [8].
- Use a customized aggregation protocol, such at that of Elahi et al. [9]
- Anonymize statistics reports, either via Tor itself or via a shuffle run over a separate set of servers. For statistics that could be attributable to a small set of relays by their values alone (e.g. a large number of rendezvous data cells is likely to come from a small set of large relays), break up the values into minimum amounts.

Adoption of these approaches faces some challenges specific to their use in Tor. Those issues and some suggestions for handling them are

1. Implementation difficulty: Making use of sophisticated cryptographic tools, such as non-standard cryptosystems or novel SMC protocols, often requires building secure implementations of them. This can require significant time and skill. It also creates future maintenance obligations. When choosing from the above solutions, we will have to consider what reliable software already exists for their various cryptographic components.
2. Manipulation of statistics: Adversarial relays may report incorrect statistics in order to affect the aggregate statistic. For example, a simple aggregate such as a sum could be trivially destroyed by a malicious relay reporting a much larger value than its true input. One way to handle this problem is to use "robust" statistics [6], which are not excessively influenced by outliers. For example, we could use a median instead of a mean as the basis for a sum.

6 Conclusion

Statistics about the usage and performance of Tor hidden services could serve many valuable purposes. However, doing so in a privacy-preserving manner is challenging. We have described the kinds of hidden-service statistics that Tor relays are in a position to collect and the main criteria on which collection procedures should be evaluated. In particular, we have provided a list of privacy goals for Tor hidden services that goes beyond simply hiding the server's location. Furthermore, we explored how statistics might be published in a privacy-preserving way, and we gave two specific algorithms for publishing broadly useful count and distribution statistics.

References

[1] Better hidden service stats from Tor relays. https://gitweb.torproject.org/torspec.git/plain/proposals/238-hs-relay-stats.txt. Accessed: 2015-04-18.

[2] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, 2013.

[3] Roger Dingledine and Nick Mathewson. Tor protocol specification. https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt. Accessed: 2015-04-21.

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[5] Cynthia Dwork. Differential privacy. In *in ICALP*, 2006.

[6] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, 2009.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, 2006.

[8] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, 2014.

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

[10] N. Homer et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet*, 4, 2008.

[11] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 2008.

[12] Virgil Griffith, Fabio Pietrosanti, and Giovanni Pellerano. Making tor2web mode faster. https://gitweb.torproject.org/torspec.git/plain/proposals/233-quicken-tor2web-mode.txt. Accessed: 2015-04-19.

[13] Nicholas Hopper. Challenges in protecting tor hidden services from botnet abuse. In *Proceedings of Financial Cryptography and Data Security (FC'14)*, March 2014.

[14] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'12*, 2012.

[15] George Kadianakis. Give out HSDir flag only to relays with Stable flag. https://gitweb.torproject.org/torspec.git/plain/proposals/243-hsdir-flag-need-stable.txt. Accessed: 2015-04-19.

[16] George Kadianakis and Karsten Loesing. Extrapolating network totals from hidden-service statistics. Technical Report 2015-01-001, The Tor Project, 2015.

[17] Nick Mathewson. Next-generation hidden services in Tor. https://gitweb.torproject.org/torspec.git/plain/proposals/224-rend-spec-ng.txt. Accessed: 2015-04-18.

[18] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, 2007.

[19] Mike Perry. Torflow: Tor network analysis. In *3rd Hot Topics in Privacy Enhancing Technologies (HotPETs 2010)*, 2010.

[20] Sukhbir Singh. Large-scale emulation of anonymous communication networks. Master's thesis, University of Waterloo, 2014.

[21] Tor directory protocol. https://gitweb.torproject.org/torspec.git/plain/dir-spec.txt. Accessed: 2015-04-18.

[22] Tor Metrics Portal. http://metrics.torproject.org/. Accessed: 2015-04-18.

[23] Tor rendezvous specification. https://gitweb.torproject.org/torspec.git/plain/rend-spec.txt. Accessed: 2015-04-18.