Proceedings on Privacy Enhancing Technologies ; 2019 (2):88–104

Georgia Fragkouli, Katerina Argyraki, and Bryan Ford

MorphIT: Morphing Packet Reports for Internet Transparency

Abstract: Can we improve Internet transparency without worsening user anonymity? For a long time, researchers have been proposing transparency systems, where traffic reports produced at strategic network points help assess network behavior and verify service-level agreements or neutrality compliance. However, such reports necessarily reveal when certain traffic appeared at a certain network point, and this information could, in principle, be used to compromise low-latency anonymity networks like Tor. In this paper, we examine whether more Internet transparency necessarily means less anonymity. We start from the information that a basic transparency solution would publish about a network and study how that would impact the anonymity of the network's users. Then we study how to change, in real time, the time granularity of traffic reports in order to preserve both user anonymity and report utility. We evaluate with real and synthetic data and show that our algorithm can offer a good anonymity/utility balance, even in adversarial scenarios where aggregates consist of very few flows.

Keywords: Transparency, anonymity

DOI 10.2478/popets-2019-0021
Received 2018-08-31; revised 2018-12-15; accepted 2018-12-16.

1 Introduction

The Internet does not provide enough transparency: when traffic is lost, delayed, or damaged, there is no systematic way for the affected parties to determine where the problem occurred and who is responsible. This was a conscious design choice made by the Internet architects because, at the time, they did not intend for the Internet to be used commercially [16]. Indeed, lack of transparency has led to problems: Internet service providers (ISPs) sign service-level agreements (SLAs), committing to a certain packet delivery rate or latency [1, 3], that are impossible to verify; governments enact network neutrality regulations, requiring that ISPs treat all traffic the same independently from origin or application [5], that are impossible to enforce; content and network providers enter disputes that are impossible to investigate properly [4]. We are not arguing for SLAs or regulations; but from the moment Internet users care enough for them to exist, there should be a way to verify and enforce them.

To improve Internet transparency, researchers have proposed "transparency systems," where domains report on their own performance [9–12, 24, 39–41]. The common idea behind these proposals is that participating domains deploy special nodes at their boundaries, which collect samples or summaries of the observed traffic and report them to a ledger; based on these reports, one can accurately estimate each domain's performance with respect to various traffic aggregates. The challenges typically addressed are how to produce useful reports at low cost, and how to prevent domains from lying to exaggerate their perceived performance.

However, transparency systems face another significant challenge, which, to the best of our knowledge, has not been addressed: they interfere with anonymous communications. Anonymity networks enable a user to hide the fact that she is communicating with a particular destination. Transparency systems directly threaten this capability because they expose information about when given traffic is observed at given network points. For example, consider: a user that communicates with various destinations through Tor [6]; a government that monitors the user's Internet connection and observes the flows she generates; and a basic transparency system, where participating domains periodically group flows into aggregates and report per-aggregate packet counts. If the government gains access to the information stored in the ledger, it becomes equivalent to a passive adversary that observes the user's flows, on the one hand, and all Tor aggregates, on the other, and tries to de-anonymize the user's flows; low-latency anonymity networks like Tor are vulnerable to such adversaries [17, 27, 31].

Georgia Fragkouli: EPFL, E-mail: georgia.fragkouli@epfl.ch
Katerina Argyraki: EPFL, E-mail: katerina.argyraki@epfl.ch
Bryan Ford: EPFL, E-mail: bryan.ford@epfl.ch

Is it possible to design a transparency system that produces useful reports without interfering with anonymous communications? On the one hand, there exists a fundamental trade-off between report utility and flow anonymity: the finer the time granularity of the reports, the better one can compute loss burstiness or delay jitter between reporting nodes (so, higher report utility), but also the better one can match flow and aggregate patterns (so, lower flow anonymity). On the other hand, even time granularity of minutes can be useful: in the current Internet, downloading movies or kernel distributions can take from minutes to hours; hence, reports that help determine whether ISPs honor their SLAs or honor neutrality at such time granularity are still useful. The question then is: how coarse do reports have to be in order to preserve flow anonymity? If a transparency system reports at a time granularity of minutes, is that enough? How about seconds?

We take a first step toward answering this question: We consider a basic transparency system, where nodes report per-aggregate packet counts, which can be used to compute packet loss between pairs of nodes (§2). We first study how such a system would affect flow anonymity in the context of a low-latency anonymity network (§3). Then we propose MorphIT, an algorithm that takes as input a node's report and produces as output a modified version that improves flow anonymity (§4). Our algorithm does not introduce noise in the typical sense, i.e., does not introduce error in the reported packet counts; instead, it merges packet counts so as to obfuscate the flow patterns that stand out the most within each aggregate. We evaluate this approach with synthetic and real traffic traces obtained from CAIDA (§5). We show that, even in highly adversarial scenarios (e.g., only 64 flows per aggregate), merging packet counts across sub-second time intervals is enough to prevent an adversary from tracing any flow to a unique aggregate, and it also significantly reduces the number of flows that can be traced to few candidate aggregates (e.g., fewer than 5). Reporting packet counts of sub-second granularity means that we retain report utility, in the sense that we can use the reports to verify SLA or neutrality violations at such a fine granularity.

We close the paper by discussing the limitations of our approach (§6), related work (§7), and our conclusions (§8).

2 Setup

After defining the basic terms we use in the paper (§2.1), we state our threat model (§2.2) and problem (§2.3), and we summarize the most relevant anonymity metrics from related work (§2.4).

2.1 Definitions

A *domain* is a contiguous network managed by a single administrative entity, e.g., an enterprise network, an Internet service provider (ISP), an Internet exchange point (IXP).

A *transparency system* (Fig. 1) consists of *hand-off points* (HOPs) and a *report ledger*. A HOP is a network node that publishes periodic traffic reports. Each domain that joins a transparency system deploys a HOP at each point where traffic enters and exits its network; hence, a HOP is always collocated with a network interface of a gateway or border router. The report ledger collects the HOPs' reports and provides access to them. It is a logically centralized entity that could be owned and managed either by a third party (akin to how the root DNS servers are managed by companies) or by the participating domains themselves (in which case it would be implemented as a distributed, decentralized system).

A *time tick* t is a time interval of the smallest duration for which a HOP can produce statistics. A *time bin* T is a time interval of one or more time ticks. The *observation window* W, of size w, is the time interval for which our adversary (defined below) collects information.

A *flow* f is a sequence of packets exchanged between a unique source and destination. When we say that an adversary *knows a flow's pattern*, we mean that she knows the packet inter-arrival times, but not necessarily the packet contents (because they could be encrypted) or the flow's destination (because the source could be using an anonymity network). $N_f(t)$ ($N_f(T)$) denotes the number of f's packets observed at a HOP during time tick t (time bin T). λ_f denotes f's average packet count per time tick.

An *aggregate* A is a sequence of packets with a unique source and destination IP prefix that are observed at a particular HOP. $N_A(t)$ ($N_A(T)$) denotes the number of A's packets observed during time tick t (time bin T). λ_A denotes A's average packet count per time tick.

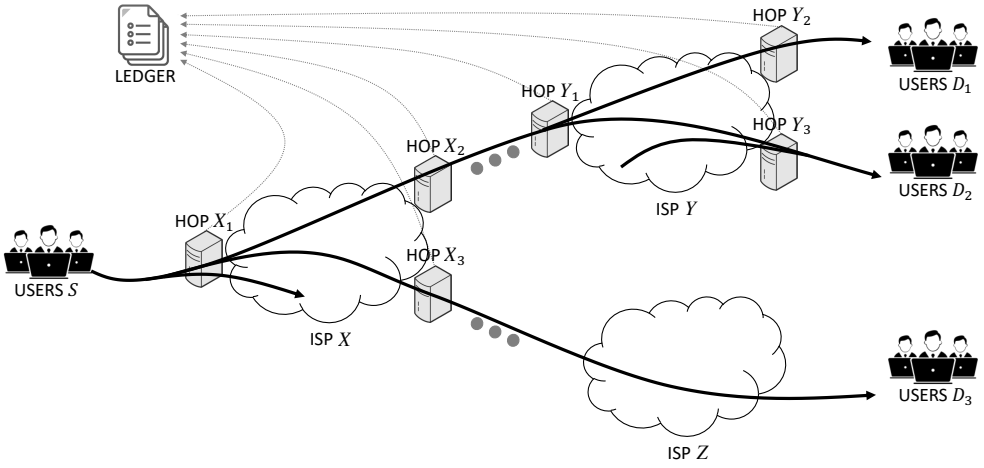


Fig. 1. A transparency system.

A *traffic report* $R(A)$ refers to a specific aggregate A and is a set of tuples:

$$R(A) \equiv \{ \langle t, N_A(t) \rangle \}.$$

The reports produced by two different HOPs for the same aggregate A are used to compute the packet-loss rate between the two HOPs with respect to A.

There exist many flavors of transparency systems [9–12, 24, 39–41], each one reporting slightly different statistics. We consider this particular flavor, because it produces minimal information (per-aggregate packet counts); if it poses a threat to flow anonymity, then any of the more sophisticated transparency systems that have been proposed will also pose a threat.

Table 1 lists the symbols used in the paper.

Symbol	Description
	Traffic units
f	A flow: a packet sequence exchanged between a unique source and destination
A	An aggregate: a packet sequence observed at a HOP
$\mathcal{A}$	A set of aggregates
	Time intervals
t	A time tick
T	A time bin: a set of consecutive time ticks
$\mathcal{T}$	A set of consecutive, non-overlapping time bins
W	The adversary's observation window
$w = \|W\|$	The size of the adversary's observation window (in time ticks)
	Traffic characteristics
$N_f(t)$	Number of packets in flow f in time tick t
$N_f(T)$	Number of packets in flow f in time bin T
λ_f	Average packet rate of flow f
$N_A(t)$	Number of packets in aggregate A in time tick t
$N_A(T)$	Number of packets in aggregate A in time bin T
λ_A	Average packet rate of aggregate A
ϕ	Maximum number of active flows per aggregate
	Algorithm parameters
ρ	Max flow burst size per time tick (in packets)
τ	Max bin size (in time ticks)
ω	Active window size (in time ticks)
	Differential Privacy
ϵ	Privacy loss
δ	Probability of exceeding ϵ

Table 1. List of symbols used in this paper.

2.2 Threat Model

In the context of a transparency system, the report ledger is trusted to collect the traffic reports and provide proper access to them, while domains may be honest (report true packet counts) or dishonest (report fake packet counts in an effort to exaggerate their performance). Like prior work, our transparency system by design prevents dishonest domains from exaggerating their performance. For example, consider Figure 1 and an aggregate A that crosses HOPs X_1, X_2, and Y_1. Suppose ISP X drops a packet from A and wants to lie about it. To this end, it makes X_2 report a fake packet count for A (increased by 1 relative to the true packet count). As a result, the report ledger thinks that the packet was lost on the inter-domain link between X_2 and Y_1 and attributes the loss to *both* ISPs X and Y. Hence, by lying, ISP X not only cannot hide its true packet loss, but it also falsely attributes packet loss to neighbor ISP Y, causing a dispute with that neighbor [11].

Moreover, we consider an adversary who is passive and aims to *trace* a target flow f: given a set of aggregates $\mathcal{A}$, one of which contains f, she wants to determine which aggregate is most likely to contain f. She has the following information: (a) f's pattern; (b) all the traffic reports for all the aggregates in $\mathcal{A}$ published within the observation window W. This adversary could be, for example, a government, who has subpoenaed an ISP (to gain access to a user's Internet connection), as well as the report ledger (to obtain the traffic reports).

We illustrate how this relates to low-latency anonymity networks through an example: Suppose user S in Figure 1 sends a flow f to some destination through Tor. Consider an adversary, Eve, who monitors S's Internet connection and learns f's packet-arrival pattern. Moreover, Eve knows that f is observed at HOP X_1 and then either X_2 or X_3. Without any extra information, Eve cannot determine f's destination. However, if she obtains the traffic reports published by HOPs X_2 and X_3, she can extract the patterns of the aggregates observed at these two HOPs, correlate them with f's pattern, and guess which one of these aggregates contains f. By repeating this process, she can guess at the sequence of HOPs that observe f. If she guesses correctly, she has narrowed down the flow's destination within an IP prefix. In some cases, this is all Eve needs to know, e.g., if the IP prefix belongs to a censored content provider.

It has already been shown that low-latency anonymity networks are vulnerable to similar adversaries. In one case, the adversary observes a target flow f and a set of aggregates $\{A_i\}$, one of which contains f; her goal is to guess which A_i contains f [17]. In another case, the adversary observes a random sample of a target flow f and a set of packet sequences, one of which is also a random sample of f; her goal is to guess which packet sequence is a random sample of f [31]. Our adversary is similar to these, in that she knows the pattern of the target flow, and she also has information about other traffic, observed at different points in the network, which may be correlated with the target flow.

The new aspect of our adversary is that she learns about traffic (other than the target flow) from transparency reports. We are interested in this particular adversary, because we want to assess the new anonymity risk that a transparency system would introduce. Our adversary would pose no threat to an anonymity network that explicitly makes aggregates indistinguishable by introducing latency and/or fake traffic [29]. We do not consider active adversaries [14, 25] or software exploits that target anonymity-network browsers[1].

2.3 Problem Statement

We want a transparency system that strikes a good balance between being useful (enabling accurate assessment of network performance) and obstructing flow tracing. Given that our adversary has access to the traffic reports published by the HOPs, any obfuscation of information must happen at the HOPs, before the reports are published.

Hence, we look for an algorithm that takes as input a traffic report and modifies it, such that the report makes flow tracing "as hard as possible" while meeting a target "utility level." Each HOP can then apply this algorithm to each traffic report it produces. We assume that each HOP has only a local view (the traffic it observes and the reports it produces), but not the adversary's global view (the reports produced by other HOPs). Moreover, from the HOP's point of view, any flow contained in an aggregate could be a target flow, which means that the algorithm cannot focus on making only specific flows hard to trace.

We put "as hard as possible" and "utility level" in quotes, because they are not meaningful until we define metrics that capture how much a report helps/obstructs flow tracing versus how much it helps transparency. Defining metrics was a key part of our work, and we state them later in the paper. In the next subsection, we summarize the most relevant privacy metrics from the literature that we used as basis and inspiration.

2.4 Anonymity Metrics

In general, anoymity metrics characterize an adversary's uncertainty about linking an item of interest to a target user's identity [33]. In our context, the item of interest is the target flow while the target user's identity is an aggregate observed at a HOP.

Traceability [17]. This metric is useful when one knows the packet-arrival patterns of a target flow f and two candidate aggregates, A_0 and A_1, one of which contains f; her goal is to trace f (decide which aggregate is

1 https://thehackernews.com/2013/08/Firefox-Exploit-Tor-Network-child-pornography-Freedom-Hosting.html

more likely to contain it). Traceability is defined as

$$TR \equiv \log \frac{\mathcal{L}\{H_0\}}{\mathcal{L}\{H_1\}},$$

where $\mathcal{L}$ denotes likelihood, and H_0 (H_1) is the hypothesis that A_0 (A_1) contains f. Traceability 0 means that the two hypotheses are equally likely, i.e., the packet-arrival patterns do not help trace f. The absolute value of traceability indicates the difference between the log-likelihoods of the two hypotheses, so, the larger it is, the more the packet-arrival patterns help trace the flow.

The paper that introduced traceability showed how to compute it assuming independent packet arrivals that follow a Poisson distribution for the target flow and a uniform distribution for all other traffic. Indeed, when we experimented with synthetic flows generated from a Poisson model, traceability worked, i.e., our adversary could use it to trace target flows. However, when we experimented with flows extracted from CAIDA traffic traces, the flows' packet-arrival patterns were not always Poisson, and computing traceability under the assumption that they were did not work. For instance, it would be the case that a target flow's packet-arrival pattern was clearly visible within one of the candidate aggregates, and yet traceability was close to 0.

Cross-correlation [35]. This metric captures the similarity between the packet-arrival pattern of a target flow f and a traffic report[2] $R(A)$ during a time interval $[t_1, t_2]$. It is defined as:

$$CC(f, R(A), [t_1, t_2]) \equiv \sum_{t \in [t_1, t_2]} (N_A(t) - \lambda_A)(N_f(t) - \lambda_f). \quad (1)$$

In our context, cross-correlation is more practical than traceability, because it does not require any assumptions about packet arrivals. Moreover, when we experimented with flows extracted from CAIDA traffic traces, cross-correlation worked, i.e., it did enable our adversary to trace target flows.

However, neither metric captures intuitively our adversary's power. For instance, it is not clear what values of traceability or cross-correlation we should target in order to argue that our adversary does not pose a threat to anonymity networks.

2 The original definition is for a flow and an aggregate. We define it for a flow and a traffic report, because, in our context, the adversary learns about the aggregate from a traffic report.

3 Approach

In this section, we describe first the metric that we use to capture our adversary's uncertainty (§3.1), then the measurements that motivated our approach (§3.2), and then the idea of using coarser time granularity for anonymization (§3.3).

3.1 Metric: T-Anonymity Set Size

Suppose our adversary wants to trace target flow f across a set of candidate aggregates $\mathcal{A} = \{A_1, A_2, \dots A_{|\mathcal{A}|}\}$. The ground truth is that aggregate A_x contains f. The adversary computes a likelihood distribution $\mathcal{L} = \{l_1, l_2, \dots l_{|\mathcal{A}|}\}$, where l_i is her estimated likelihood that A_i contains f.

We want a metric that captures the adversary's uncertainty in tracing the target flow to the correct aggregate, akin to an anonymity set size. In particular, we want our metric to have value:

$$\begin{array}{cc} 1, & \text{if } l_x = 1 \\ \in (1, |\mathcal{A}|), & \text{if } l_x \in \left(\frac{1}{|\mathcal{A}|}, 1\right) \\ |\mathcal{A}|, & \text{if } l_x \in \left[0, \frac{1}{|\mathcal{A}|}\right] \end{array}$$

The first row describes the scenario where the adversary knows the 1 aggregate that contains f. The last row describes the scenario where the adversary either has no information (she believes that all $|\mathcal{A}|$ aggregates contain f with the same likelihood $\frac{1}{|\mathcal{A}|}$), or has misleading information (she believes that aggregate A_x contains f with likelihood $< \frac{1}{|\mathcal{A}|}$). The middle row describes all other scenarios, where the adversary has some correct information.

At first we defined our adversary's anonymity set size as $2^{H(\mathcal{L})}$, where H denotes entropy, with the rationale that entropy is the standard way to quantify the uncertainty resulting from a likelihood distribution. This metric, however, does not work when the adversary has misleading information. For example, consider two scenarios: (a) the adversary is certain that aggregate A_x contains the target flow f, which is true; (b) the adversary is certain that aggregate A_y contains f, which is false. In both scenarios the entropy of her likelihood distribution is $H(\mathcal{L}) = 0$, and $2^{H(\mathcal{L})} = 1$, which indicates that the adversary has traced f to 1 aggregate, but ignores that, in scenario (b), the tracing is false.

We define our adversary's *T-anonymity set size* as:

$$\mathcal{S} \equiv \min\left\{\frac{1}{l_x}, |\mathcal{A}|\right\}.$$

Recall that: $\mathcal{A}$ is the set of candidate aggregates, the ground truth is that aggregate A_x contains the target flow f, and l_x is the adversary's estimated likelihood that A_x contains f. This metric satisfies our requirements and has an intuitive meaning. For instance, suppose the adversary knows that either A_x or A_y contain f, but has no further information; then, her T-anonymity set size is $\min\left\{\frac{1}{0.5}, |\mathcal{A}|\right\} = 2$, which indicates that she has correctly traced f to one of 2 aggregates. However, if the adversary believes that A_x contains f with likelihood 0.9, while A_y contains f with likelihood 0.1, then her T-anonymity set size is $\min\left\{\frac{1}{0.9}, |\mathcal{A}|\right\} \approx 1.12$, which indicates that she has correctly traced f to almost 1 aggregate.

Our metric is related to prior work as follows: Consider a slightly different context from ours, where: there is an item of interest, associated with one user from a set $\mathcal{A} = \{A_1, A_2, \dots A_{|\mathcal{A}|}\}$; a true distribution $\mathcal{P} = \{p_1, p_2, \dots p_{|\mathcal{A}|}\}$, where p_i is the likelihood that the item is associated with user A_i; and an estimated distribution $\mathcal{L} = \{l_1, l_2, \dots l_{|\mathcal{A}|}\}$, which is an adversary's estimate of $\mathcal{P}$. To quantify the adversary's uncertainty about the true distribution, it has been proposed to use the *relative entropy* or *KL divergence* from $\mathcal{P}$ to $\mathcal{L}$:

$$D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{L}) \equiv \sum_i p_i \log_2 \frac{p_i}{l_i}.$$

In our context (where the ground truth is that A_x contains the target flow f), we could say that the true likelihood distribution is $\mathcal{P} = \{p_i | p_{i=x} = 1, p_{i\neq x} = 0\}$, hence $D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{L}) = p_x \log_2 \frac{p_x}{l_x} = \log_2 \frac{1}{l_x}$. The standard way to convert this entropy to an anonymity set size would be $2^{\log_2 \frac{1}{l_x}} = \frac{1}{l_x}$. Hence, one can view our metric as the anonymity set size that corresponds to the relative entropy between the ground truth and our adversary's estimated likelihood distribution.

There remains the question of how to compute our adversary's likelihood distribution $\mathcal{L}$. First, the target flow f cannot belong to an aggregate A_i if f has more packets than A_i during any time tick t in the observation window W. Hence:

$$l_i = 0, \quad \textit{if } \exists t \in W, s.t.\ f(t) > N_{A_i}(t).$$

Otherwise, we compute l_i based on the cross-correlation between f and $R(A_i)$:

$$l_i = \frac{CC^+\left(f, R(A_i), W\right)}{\sum_{\forall A_j \in \mathcal{A}} CC^+\left(f, R(A_j), W\right)},$$

where

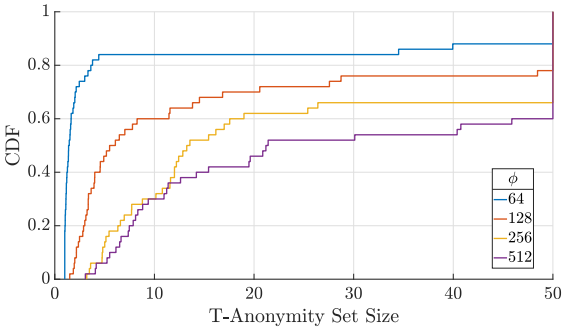
$$CC^+\left(f, R(A), [t_1, t_2]\right) \equiv \max\{\ CC\left(f, R(A), [t_1, t_2]\right),\ 0\ \}.$$

We count only positive cross-correlation between f and $R(A)$ as an indication that f belongs to A. We did experiment with an alternative approach, where we computed l_i as a normalized version of $|CC(f, R(A_i), W)|$, but we found that our adversary drew slightly worse conclusions that way. This is because we assume that our adversary correctly aligns target flow patterns to aggregate reports. As a result, negative cross-correlation between f and $R(A)$ is actually an indication that a flow does *not* belong to A.

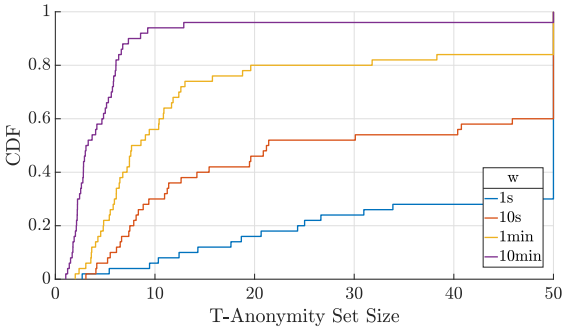
3.2 Would Transparency Affect Anonymity?

We use Internet backbone traces made available by CAIDA [2], collected at the equinix-nyc monitor, direction A, from March to November 2018. From each trace, we extract TCP and UDP flows; then we create aggregates by grouping flows together. We assume a time tick of 1ms (i.e., a traffic report contains an aggregate's packet count every 1ms). In each experiment, we emulate the scenario where: an adversary obtains traffic reports for 50 aggregates, $A_1, A_2, \dots A_{50}$ and wants to trace 50 target flows, $f_1, f_2, \dots f_{50}$; the ground truth is that aggregate A_i contains flow f_i. The number of flows per aggregate, ϕ, and the adversary's observation window, w, vary per experiment.

In the experiments we present, each target flow contributes a maximum of $\rho = 1$ packet during any given time tick (hence maximum rate 1.5 Mbps). In general, burstier, higher-rate flows are easier to trace. Low-latency anonymity networks do not aim to protect flows of arbitrary rate, for example, Tor's hidden-service statistics aim to protect flows that contribute up to 1MiB over 24 hours [18]. In our context, as stated above, a target flow f cannot belong to a candidate aggregate A if f has more packets than A during any time tick t in the observation window. The larger f's bursts, the more candidate aggregates the adversary can exclude with this rationale. Hence, we think the interesting question is whether transparency would affect the anonymity of relatively low-rate flows, which could not be trivially traced due to their bursts. For this reason, we cropped our target flows, such that each contributes up to $\rho = 1$ packet during any given time tick. To satisfy



(a) Observation window size $w = 10$s, varying number of flows per aggregate ϕ.



(b) Number of flows per aggregate $\phi = 512$, varying observation window size w.

Fig. 2. CDF of the adversary's T-anonymity set size as a function of flows per aggregate (left) and observation window (right).

this constraint, from the $50 \times 600\text{k}$[3] flow contributions per time tick that we considered, we had to crop 1%.

First, we look at how the adversary's uncertainty depends on the number of flows per aggregate ϕ. Recall that, in our context, an aggregate is all traffic observed at a HOP with a unique source and destination IP prefix pair. So, we expect most aggregates to contain at least hundreds of flows; however, it could happen that, for a short time window, an aggregate contains fewer active flows than usual, potentially making these flows more vulnerable to tracing. The question then is: what can our adversary do when ϕ is in the hundreds, and what can she do when ϕ drops to tens of flows per aggregate?

Fig. 2a shows the cumulative distribution function (CDF) of the adversary's T-anonymity set size $\mathcal{S}$ when her observation window size is $w = 10$s, while the number of flows per aggregate ϕ varies. When $\phi = 512$, the adversary traces no target flow to a unique aggregate, but she still traces 6% of the target flows to < 5 candidate aggregates. When $\phi = 64$, she traces 64% of the target flows to a unique aggregate, and 84% of the target flows to < 5 candidate aggregates.

Next, we look at how the adversary's uncertainty depends on her observation window size w. We expect that the adversary's uncertainty drops as w increases, and the question is how fast.

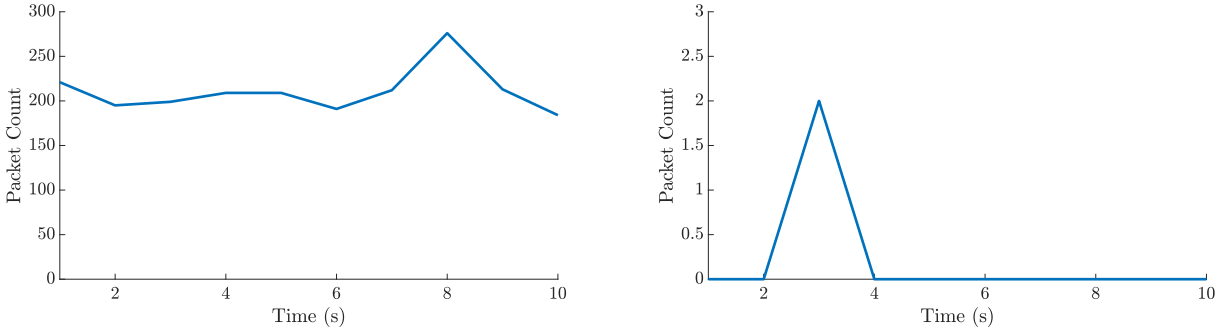
Fig. 2b shows the CDF of the adversary's T-anonymity set size $\mathcal{S}$, when there are $\phi = 512$ flows per aggregate, while the adversary's observation window size varies. As we saw in the previous graph, when $w = 10$s, the adversary traces no target flow to a unique aggregate, but she still traces 6% of the target flows to < 5 candidate aggregates. When $w = 10$min, she traces 16% of the target flows to a unique aggregate, and 62% of the target flows to < 5 candidate aggregates.

As expected, the adversary's uncertainty increases as ϕ gets bigger and w gets smaller ($\mathcal{S}$'s CDF shifts to the right); however, she always manages to trace a few target flows to a relatively small number of candidate aggregates. For example, even when $\phi = 512$ flows per aggregate, and the adversary's observation window is barely $w = 1$s (rightmost curve in Fig. 2b), there is still one target flow for which $\mathcal{S} < 5$. As expected, these are flows with peculiar packet-arrival patterns, e.g., bursts of unusual duration or period, that make them stand out even within 511 other flows. For example, the flow shown in Fig. 3a is vulnerable to tracing, because it is the only flow in our dataset to contribute such a high number of packets within a few seconds. In contrast, the flow shown in Fig. 3b is hard to trace, because it only contributes two packets in total—a pattern that is easily hidden within an aggregate.

3.3 Coarser Time Granularity as Noise

The first approach we tried was differential privacy [20]: Consider a HOP that observes two aggregates, A_1 and A_2, that differ in a single flow f; and publishes traffic reports $R(A_1)$ and $R(A_2)$. Ideally, if $R(A_1)$ contains a tuple $\langle t, N_{A_1}(t)\rangle$, and $R(A_2)$ contains a tuple $\langle t, N_{A_2}(t)\rangle$, it should be the case that probabilities of $N_{A_1}(t)$ and $N_{A_2}(t)$ taking any arbitrary value N are approximately equal. If we could guarantee this property for any A_1, A_2, and f, then we could say that the transparency system is "differentially private", in the sense that traffic reports never reveal any information that could help our adversary trace a target flow.

3 The maximum observation window we considered is 10min = 600k time ticks.



(a) A packet-arrival pattern that is easy to trace.

(b) A packet-arrival pattern that is hard to trace.

Fig. 3. Examples of actual packet flows that are easy (left) and hard (right) to trace.

We explored this approach but could not guarantee any meaningful differential privacy without making the traffic reports useless: We tried adding noise to the packet counts of a traffic report so as to guarantee ϵ-differential privacy; we considered both standard Laplace noise and a more recent Fourier-based variant [34]. When we set ϵ to any typical value (e.g., $\in (0, 1)$), the modified packet counts were so noisy that any statistic computed from them was arbitrarily unreliable. When we bounded the difference between the original and modified packet counts to any reasonable value (e.g., relative error 10%), the ϵ for which we could guarantee ϵ-differential privacy was so large (tens of thousands) that we could not reason about its meaning any more.

In retrospect, given the purpose of a transparency system, additive noise does not make sense as a first counter-measure (although it could be useful as an enhancement): If a traffic report contains packet counts at a granularity of 10ms, and we decide that that reveals too much information, it does not make sense to add noise to the packet counts while keeping the same, fine time granularity.

Hence, we explore the following idea: instead of adding noise to the packet counts of traffic reports, we coarsen their time granularity. This is another form of noise, and it has two benefits:

(1) It allows us to control report utility: Suppose traffic reports have time granularity τ, i.e., each HOP publishes a packet count per aggregate every τ time ticks. By coarsening the time granularity of the reports (increasing τ), we do not make them less reliable: if the packet counts are perfectly accurate, the packet-loss rates computed from them will also be perfectly accurate, albeit averaged over longer time intervals.

(2) It allows us to preserve the incentive structure of the transparency system (§2.1): By coarsening the time granularity of the reports, we do not change the fact that reports are expected to contain exact per-aggregate packet counts. Hence, as long as we can align the reports produced by subsequent HOPs for the same aggregate, a domain cannot escape the blame for a lost packet—it can only shift it from an internal path to one of its own inter-domain links, which does not improve its perceived performance and causes a dispute with a neighbor domain.

4 Algorithm

We now present our algorithm: first an overview (§4.1), then an "idealized" version (§4.2), and then a more practical online version (§4.3).

4.1 Overview

Given an aggregate A and an observation window W, our algorithm takes as input:

▷ A traffic report[4]

$$R(A) \equiv \{ \langle t,\ N_A(t) \rangle,\ \forall t \in W \}.$$

▷ The patterns of A's flows

$$\{ \langle t,\ N_f(t) \rangle,\ \forall t \in W,\ \forall f \in A \}.$$

4 Technically, this traffic report can be reconstructed from the second input (the patterns of A's flows). We state it as a separate input because we think that helps make the algorithm description clearer.

It produces as output a traffic report $R_o(A)$, which contains packet counts not per time tick (as the input report), but per time bin:

$$R_o(A) \equiv \{ \langle T,\ N_A(T) \rangle,\ \forall T \in \mathcal{T} \}, \tag{2}$$

where $\mathcal{T}$ is a set of consecutive, non-overlapping time bins that cover the observation window W.

Our algorithm takes the following configuration parameters:

- ▷ The maximum flow burst size ρ. Flows with bigger bursts are easier to trace: The adversary knows that a target flow f does not belong to a candidate aggregate A_i if she knows that f has more packets than A_i during any given time interval. The burstier f is, the more aggregates the adversary can exclude with this rationale. Our algorithm tries to protect flows that contribute up to ρ packets during any single time tick.
- ▷ The maximum bin size τ. Our algorithm produces time bins that contain up to this many time ticks. This parameter is our way of ensuring that the output report retains a certain utility level.

To capture the similarity between a target flow f and the output traffic report $R_o(A)$, we define the cross-correlation between f and $R_o(A)$ and its positive-only version as:

$$CC\left(f, R_o(A), [t_1, t_2]\right) \equiv \sum_{t \in [t_1, t_2]} \left(\frac{N_A(T_t)}{||T_t||} - \lambda_A \right) (N_f(t) - \lambda_f),$$

$$CC^+\left(f, R_o(A), [t_1, t_2]\right) \equiv \max\{\ CC\left(f, R_o(A),\ \ [t_1, t_2]\right),\ 0\ \}. \tag{3}$$

where $T_t \in \mathcal{T}$ is the time bin that contains time tick t. This is essentially the same definition as in Eq. 1, with the difference that $N_A(t)$ has been replaced by $\frac{N_A(T_t)}{||T_t||}$, because the adversary does not see the original packet counts $N_A(t)$ any more, but only the modified, coarser packet counts $N_A(T_t)$.

We do not try to design an optimal algorithm: Recall that our adversary collects traffic reports for a set of candidate aggregates observed at different HOPs, and tries to trace a target flow f; the metric for her success is her T-anonymity size $\mathcal{S}$ for f. An optimal algorithm would either maximize $\mathcal{S}$ subject to some minimum report utility; or maximize report utility subject to some minimum $\mathcal{S}$. Either approach requires knowledge of f's pattern as well as the patterns of all the candidate aggregates; whereas our algorithm runs at a single HOP and takes as input only $R(A)$ and the patterns of A's flows.

The simplest solution would be to choose time bins of fixed duration τ; as we show in our evaluation, this protects most realistic flows, but has two disadvantages: First, it introduces unnecessary noise, because it coarsens the report's time granularity uniformly, even during time intervals when the report does not improve the adversary's knowledge. Second, it can be bad for flows whose pattern happens to align in an unlucky way with the time bins. For instance, consider a flow that is active for a window of x time ticks, then inactive for a window of x time ticks, and the pattern repeats; if each time bin happens to align with an active or inactive window, then coarsening time granularity from 1 to x time ticks may not increase at all the adversary's uncertainty (depending on the other traffic). Such "on-off" flows do exist, albeit rarely, in the CAIDA traces.

Our solution relies on the concept of a *virtual flow* v: given an output traffic report $R_o(A)$, v is a synthetic flow that, during any time bin T, has the same pattern as the real flow that has the highest cross-correlation with $R_o(A)$ during time bin T. More precisely,

$$N_v(t) = N_{f_T}(t),\ \forall t \in T,$$

where

$$f_T = \arg\max_f CC^+(f, R_o(A), T). \tag{4}$$

Our algorithm tries to choose the time bins of the output traffic report, such that the report reveals as little information as possible about v. The rationale is that v consists of the most vulnerable pieces of A's real flows; so, if the output report hides v's pattern, we expect that it will also hide the patterns of A's real flows.

4.2 Idealized Algorithm

We first designed an idealized version of our algorithm (that we call MorphIT$_{id}$), which finds the time bins $\mathcal{T}$ that cover the observation window W while minimizing the following metric:

$$\sum_{\forall T \in \mathcal{T}} \max_f CC^+(f, R_o(A), T). \tag{5}$$

This is simply the positive-only cross-correlation (Eq. 3) between the virtual flow v (Eq. 4) and the output traffic report $R_o(A)$ (Eq. 2) during the observation window W.

Our algorithm relies on dynamic programming and defines three matrices:

$S[j+1,i]$ specifies how much our optimization metric (Eq. 5) will increase if the output report already covers time interval $[0,j]$ and we add time bin $[j+1,i]$:

$$S[j+1,i] = \begin{cases} \infty, & \text{if } i-j > \tau \text{ OR} \\ & N_A([j+1,i]) < (i-j)\rho, \\ \max_f CC^+(f, R_o(A), [j+1,i]), & \text{otherwise} \end{cases}$$

Notice that $S[j+1,i] = \infty$ if time bin $[j+1,i]$ exceeds the maximum bin size τ, or if aggregate A contributes fewer than $(i-j)\rho$ packets in time bin $[j+1,i]$ (in which case we could not protect flows with such burst sizes).

$S_{opt}[k,i]$ keeps track of the minimum value of the optimization metric when the output report covers time interval $[1,i]$ divided in k time bins:

$$S_{opt}[k,i] = \begin{cases} S[1,i], & \text{if } k=1, \\ S_{opt}[k-1,j^*] + S[j^*+1,i], & \text{if } k>1, \end{cases}$$

$$j^* = \underset{k-1 \le j \le i-1}{\arg\min} \{S_{opt}[k-1,j] + S[j+1,i]\}.$$

$T_{opt}[k,i]$ keeps track of the time bins that lead to $S_{opt}[k,i]$ (in particular, it specifies the beginning of the last time bin that leads to $S_{opt}[k,i]$).

Once the algorithm has filled S_{opt} and T_{opt}, it picks the time bins that lead to $S_{opt}[1,w]$ by backtracking from $T_{opt}[k^*,w]$, where:

$$k^* = \underset{\lceil w/\tau \rceil \le k \le n}{\arg\min} \{S_{opt}[k,w]\}.$$

The complexity of the idealized algorithm is

$$\mathcal{O}(w^3) + \mathcal{O}(\phi w^2),$$

where w is the size of the observation window covered by the input traffic report, and ϕ is the number of flows in aggregate A. The first term comes from filling S_{opt}: to fill each position, the algorithm examines $\tau = \mathcal{O}(w)$ entries; this is done for each of the $w \times w$ positions of the matrix, leading to $\mathcal{O}(w^3)$. The second term comes from filling S: to fill $S[j,i]$, the algorithm computes $CC(A,f,[j,i])$ for each of the ϕ participating flows; this is done for each of the $w \times w$ positions of the matrix, leading to $\mathcal{O}(\phi w^2)$.

Given that we want HOPs to run our algorithm in real time, this complexity becomes prohibitive when w extends beyond a few tens of seconds.

4.3 Online Algorithm

To make our algorithm practical, we designed an "online" version (that we call MorphIT$_\omega$), which divides the observation window into smaller windows of size ω, applies the idealized algorithm to each of them, and combines all the resulting outputs into one that covers the entire observation window.

Intuitively, as the size of the active window ω approaches the size of the observation window w, the performance of the online version improves and approaches that of the idealized one; however, we cannot guarantee that the performance gap between the two closes smoothly as ω increases. Figure 4 illustrates an extreme scenario where any $\omega < w$ yields bad results: Aggregate A_x has packets only during time tick 1, while aggregate A_y has packets only during the last time tick w. If the adversary is trying to trace a flow to one of these two aggregates, she will clearly succeed, unless the output traffic reports consist of a single time bin of length w—in which case the two aggregates become indistinguishable. In this scenario, any $\tau < w$ and any $\omega < w$ would have no impact on the adversary's T-anonymity set size.

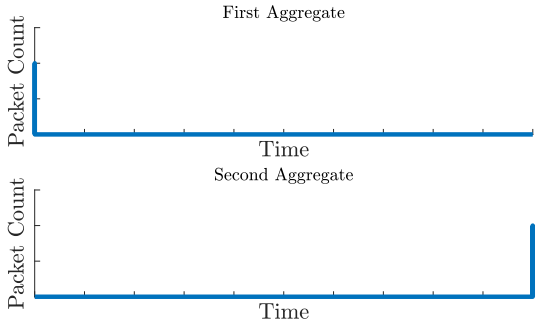


Fig. 4. Example of two aggregates that require $\tau = \omega = w$.

The question is: are there values of ω that make the algorithm deployable while maintaining most of the benefit of the idealized algorithm? The answer depends on flow duration and patterns, and we study it in our evaluation section.

5 Evaluation

After describing our setup (§5.1), we present our algorithm's performance (§5.2) and compare it to a simpler alternative (§5.3) and a state-of-the-art anonymization tool based on differential privacy (§5.4). We discuss processing overhead in the Appendix A.

5.1 Setup

The basic experimental setup is the same as in §3.2: We assume 1ms time ticks. In each experiment, we consider: 50 aggregates, $A_1, A_2, \ldots A_{50}$, and an adversary who wants to trace 50 target flows, $f_1, f_2, \ldots f_{50}$; the ground truth is that aggregate A_i contains flow f_i. The number of flows per aggregate ϕ and the adversary's observation window size w vary per experiment.

We experiment with three types of traffic:

1. **Poisson**: We generate flows with Poisson packet arrivals of average rate $\lambda_f = 1$ packet per time tick. We create an aggregate by grouping ϕ of these flows. This is similar to the experimental setup used in [17], the work that is closest in spirit to ours. There is also evidence that real traffic flows can have Poisson behavior [28].
2. **Real**: We extract TCP and UDP flows from Internet backbone traces as stated in §3.2. We create an aggregate by grouping ϕ randomly chosen flows. We crop the target flows such that each of them contributes up to $\rho = 1$ packet during any given time tick. So, we use the traces to obtain realistic flows, but assume that an aggregate may consist of any random subset of these flows; we do not rely on the traces to draw any conclusion about the nature of aggregates.
3. **On-off**: We craft a target flow with 10 packets at time tick 31, 10 packets at time tick 32, and no other traffic. Then we craft another target flow with the same pattern, but shifted by 2ms, i.e., 10 packets at time tick 33, 10 packets at time tick 34, and no other traffic. We craft 24 more such pairs of target flows, where each flow has traffic during only 2 time ticks, and one flow is shifted by 2ms relative to the other. We create an aggregate by grouping 1 on-off flow with $\phi - 1$ real flows (extracted from traces). The purpose of this traffic pattern is to illustrate the limitations of the Uniform algorithm.

We consider the following algorithms:

1. **MorphIT**$_{100}$: This is the online version of our algorithm with an active window of $\omega = 100$ms.
2. **MorphIT**$_{id}$: This is the idealized, non-implementable version of our algorithm. We run it whenever possible (when the observation window is small enough for the algorithm to finish in reasonable time), to give a sense of how much better it performs than our online algorithm.
3. **Uniform**: This is a simpler alternative (§4.1) that always picks fixed time bins of equal size τ. If it works well, then there is no reason for a more complex algorithm like MorphIT.
4. **PrivCount** [26]: This is a state-of-the-art system that uses Gaussian noise to provide (ϵ, δ)-differential privacy [19] for a number of statistics aggregated across the Tor network and over time (e.g. number of TCP connections exiting Tor within 24 hours).

We adjust the computation of the adversary's likelihood distribution $\mathcal{L}$ in a straightforward manner: Consider a target flow f, a candidate aggregate A_i, and the time bins $\mathcal{T}_i$ from A_i's report $R_o(A_i)$. First, f cannot belong to A_i if, during any one time bin in $\mathcal{T}$, f has more packets than A_i:

$$l_i = 0, \quad \textit{if } \exists T \in \mathcal{T}, s.t.\ f(T) > N_{A_i}(T).$$

Otherwise, we compute l_i based on the cross-correlation between f and $R_o(A_i)$:

$$l_i = \frac{CC^+\left(f, R_o(A_i), W\right)}{\sum_{\forall A_j \in \mathcal{A}} CC^+\left(f, R_o(A_j), W\right)}.$$

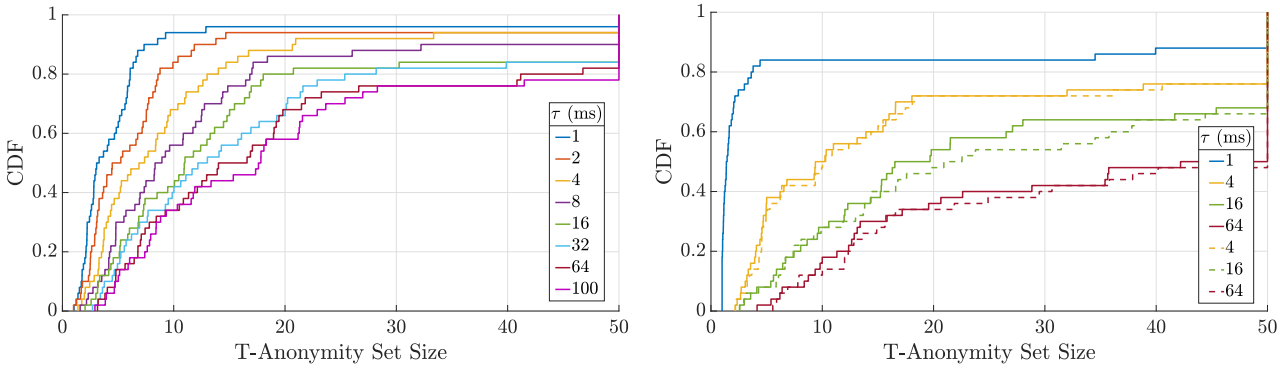
5.2 MorphIT Performance

We consider two scenarios from Section 3.2 where the adversary had little uncertainty:

1. **Long observation**: There are $\phi = 512$ flows per aggregate, and the adversary's observation window is $w = 10$min.
2. **Sparse aggregates**: There are only $\phi = 64$ flows per aggregate, and the adversary's observation window is $w = 10$s.

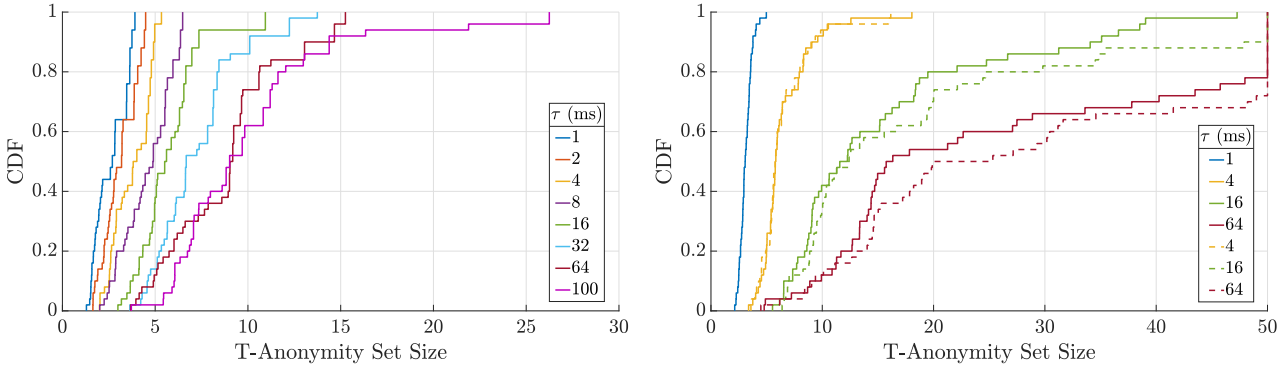
Fig. 5 shows the CDF of the adversary's T-anonymity set size $\mathcal{S}$ given real flows, in the "long observation" scenario (5a) and in the "sparse aggregates" scenario (5b). The solid curves were obtained with MorphIT$_{100}$, while the dotted curves were obtained with MorphIT$_{id}$ (we do not show MorphIT$_{id}$'s performance for $w = 10$min, because the algorithm is infeasible for such a large w). Different curves correspond to different max bin sizes τ.

Consider Fig. 5a. Without any modification to the traffic report ($\tau = 1$ time tick, leftmost curve), the adversary traces 16% of the target flows to a unique aggregate and 62% of the target flows to < 5 candidate aggregates. As τ increases, we allow our algorithm to modify more the traffic report, and the adversary's uncertainty



(a) "Long observation" scenario: $\phi = 512$ flows/aggr., $w = 10$min. **(b)** "Sparse aggregates" scenario: $\phi = 64$ flows/aggr., $w = 10$s.

Fig. 5. CDF of the adversary's T-anonymity set size given real flows. The solid curves are achieved by MorphIT$_{100}$, while the dotted curves are achieved by MorphIT$_{id}$. The max bin size τ varies.



(a) "Long observation" scenario: $\phi = 512$ flows/aggr., $w = 10$min. **(b)** "Sparse aggregates" scenario: $\phi = 64$ flows/aggr., $w = 10$s.

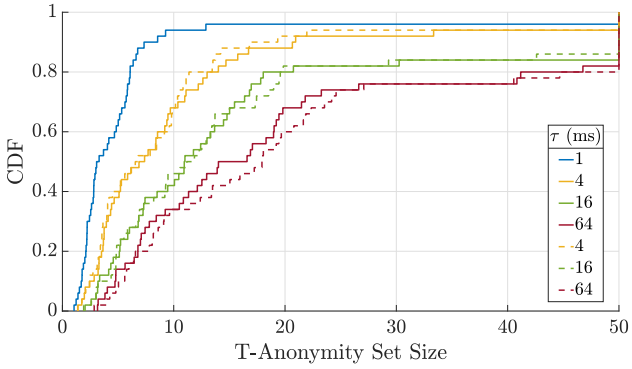
Fig. 6. CDF of the adversary's T-anonymity set size given Poisson flows. The solid curves are achieved by MorphIT$_{100}$, while the dotted curves are achieved by MorphIT$_{id}$. The max bin size τ varies.

increases ($\mathcal{S}$'s CDF shifts to the right). For $\tau = 16$ms, the adversary traces no target flow to a unique aggregate; for $\tau = 64$ms, she traces 14% of the target flows to a set of < 5 candidate aggregates. Regarding report utility, as long as $\tau < 1$s, users of the transparency system can still use the reports to compute domain performance at a sub-second time granularity.

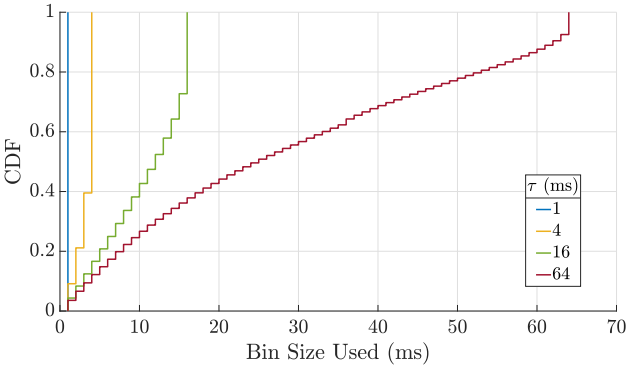
Consider Fig. 5b. Recall that this is a particularly adversarial scenario, with only $\phi = 64$ flows per aggregate. Without any modification to the traffic report ($\tau = 1$ time tick, leftmost curve), the adversary traces 66% of the target flows to a unique aggregate and 84% of the target flows to < 5 candidate aggregates. For $\tau = 64$ time ticks, she traces no target flow to a unique aggregate and 1 target flow to < 5 candidate aggregates. Regarding the relative performance of the two algorithms, MorphIT$_{100}$ closely follows the T-anonymity set sizes achieved by MorphIT$_{id}$.

Fig. 6 shows the CDF of the adversary's T-anonymity set size $\mathcal{S}$ given Poisson flows, in the "long observation" scenario (Fig. 6b) and in the "sparse aggregates" scenario (Fig. 6a).

Comparing our results given Poisson versus real flows: The adversary's uncertainty is, in general, lower with Poisson flows. We think that this is due to the fact that all Poisson flows are active throughout the observation window, whereas real flows are typically active for significantly shorter time intervals. On the other hand, the adversary's uncertainty is more stable with Poisson flows ($\mathcal{S}$'s CDF is closer to vertical), which is not surprising given that their packet arrivals follow the same distribution.

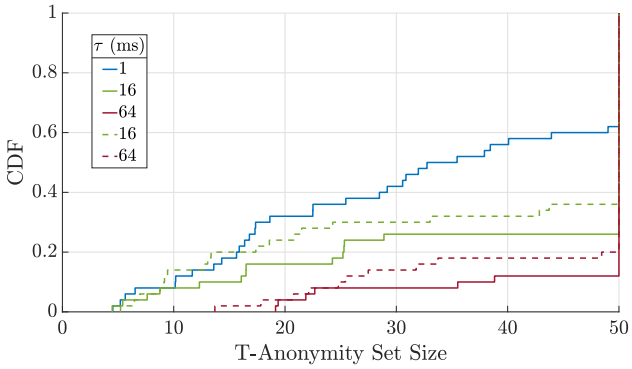


(a) CDF of the adversary's T-anonymity set size.

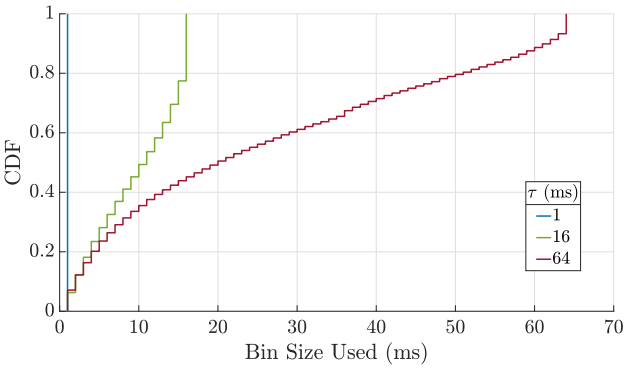


(b) CDF of the bin sizes picked by MorphIT$_{100}$.

Fig. 7. MorphIT$_{100}$ (solid curves) versus Uniform (dotted curves) performance given real flows. "Long observation" scenario: $\phi = 512$ flows/aggr., $w = 10$min. The max bin size τ varies.



(a) CDF of the adversary's T-anonymity set size.



(b) CDF of the bin sizes picked by MorphIT$_{100}$.

Fig. 8. MorphIT$_{100}$ (solid curves) versus Uniform (dotted curves) performance given on-off target flows. "Sparse aggregates" scenario: $\phi = 64$ flows/aggr., $w = 10$s. The max bin size τ varies.

5.3 Comparison to Uniform

We expected Uniform to achieve adversary uncertainty similar to MorphIT$_{id}$, but introduce a significant amount of unnecessary noise (because it coarsens the entire traffic report to time granularity τ, whether that helps anonymity or not).

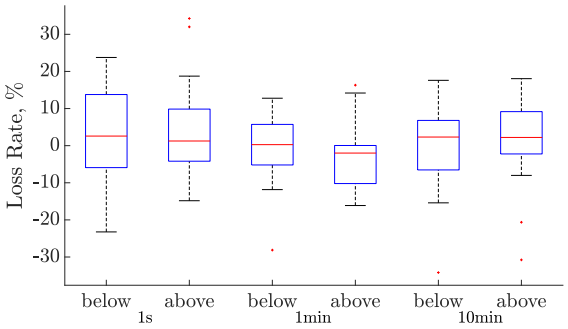
Fig. 7 compares Uniform to MorphIT$_{100}$ in the "Long observation" scenario, given real flows. Uniform achieves similar adversary uncertainty (Fig. 7a), but introduces significantly more noise: Fig. 7b show the CDF of the bin sizes picked by MorphIT$_{100}$ in each scenario. We see that it resorts to the max bin size sparingly, especially for the larger values of τ. For instance, when $\tau = 64$, 50% of the bins have size < 25ms, while 80% of the bins have size < 53ms.

Fig. 8 compares Uniform to MorphIT$_{100}$ in the "Sparse aggregates" scenario, given on-off target flows. Uniform achieves slightly worse (lower) adversary uncertainty (Fig. 8a), while it still introduces significantly more noise (Fig. 8b).

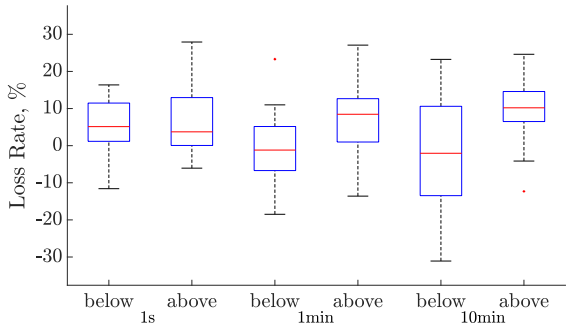
5.4 The Cost of Differential Privacy

Any mechanism providing differential privacy (like PrivCount) would provide better anonymity than MorphIT; the question we examine is at what cost to report utility.

We simulate the following scenario: An ISP has signed SLAs with 50 customer networks, promising packet loss below 0.1%. During some time interval, each customer network generates a traffic aggregate consisting of $\phi = 512$ real flows. During this time interval, the ISP honors half the SLAs and violates the other half: it introduces packet loss 0.01% to half of these aggregates (we call them "below" aggregates) and packet loss 1% to the other half (we call them "above" aggregates). Each aggregate A crosses the ISP at one ingress and one



(a) SLA: 0.1%. Real packet loss: 0.01% for "below" aggregates, 1% for "above" aggregates.



(b) SLA: 5%. Real packet loss: 1% for "below" aggregates, 10% for "above" aggregates.

Fig. 9. Packet-loss rate estimated from traffic reports anonymized with PrivCount. Time granularity is 1s, 1min, or 10min.

egress HOP; the corresponding traffic reports produced by the two HOPs are used to compute the ISP's packet loss rate with respect to A.

The HOPs use PrivCount to anonymize their traffic reports, with privacy budget $\epsilon = 1, \delta = 10^{-3}$, equally split between the two HOPs, and the sensitivity value (the maximum contribution of a flow to the total aggregate packet count) fixed to 1 packet per ms. We chose these values according to the recommendations in [30].

Fig. 9a shows the ISP's packet-loss rates as estimated from the anonymized traffic reports: The first (second) boxplot shows the estimated packet-loss rates for the "below" ("above") aggregates when the reports contain 1s packet counts. The next two boxplots correspond to 1min packet counts, and the last two boxplots to 10min packet counts.

Fig. 9b shows similar results for the scenario where the ISP promises packet loss below 5% and introduces packet loss 1% to the "below" aggregates and 10% to the "above" ones.

In both scenarios, the noise needed to guarantee (ϵ, δ)-differential privacy destroys the utility of the traffic reports. Even when the HOPs report a single per-aggregate packet count every 10min (last two boxplots in both figures), the estimated packet-loss rate can differ from the actual one by *tens* of percentage points. With such accuracy levels, it is impossible to determine whether the ISP has honored or violated an SLA for a given aggregate, or whether the ISP is discriminating in favor or against some of the aggregates.

MorphIT also affects report utility but in a controlled manner: A HOP that uses MorphIT reports exact per-aggregate counts. Hence, as long as an ISP's entry and exit HOP report counts for the same time interval, the ISP's packet-loss rate (and SLA compliance) can be accurately computed during that time interval. Even though HOPs pick their time bins independently from each other, we found that two HOPs that observe the same aggregate pick mostly the same time bins for that aggregate, even in the presence of packet loss. As a result, it is easy to align their traffic reports and compute accurate packet-loss statistics between them.

These results are not surprising, because PrivCount was not designed for anonymizing statistics that are meant to be used for verifying SLA or neutrality compliance. We included them to make the point that a straightforward application of differential privacy would not work in our context.

6 Discussion

We now discuss the limitations of our approach and opportunities for future work.

Privacy guarantees. Our approach does not provide any guarantees about how much a flow can be protected (even if the flow contributes no more than ρ packets during any given time tick). This is because HOPs produce their traffic reports without any coordination, hence there is no guarantee about how much cover their reports provide to each other's flows. Our results indicate that it is possible to protect most flows without coordination. To provide any kind of guarantees, however, we expect that coordination between HOPs is necessary.

Report correlation. In our evaluation, the adversary assumes that each target flow belongs to *one* of a set of candidate aggregates. The adversary does not leverage the fact that a target flow may cross multiple HOPs, hence belong to multiple aggregates. Preliminary experiments indicate that doubling the number of aggregates that a flow belongs to (and allowing the adversary

to leverage this in her analysis) is roughly equivalent to doubling the adversary's observation window, however, more work is required to draw reliable conclusions.

Transparency/privacy trade-off. Transparency systems are useful only if domains cannot arbitrarily lie about their performance, and prior work has shown how the report ledger can identify dishonest traffic reports and/or render them ineffective [11]. At the same time, to protect flow privacy, domains need to conceal information from the report ledger. Whether this enables domains to cheat depends on the nature of the obfuscation. For example, an obfuscation mechanism that naïvely adds high-variance noise to the traffic reports would enable domains to cheat. Our obfuscation mechanism is to coarsen the time granularity of the reports. Hence, we do not interfere with the ability of the report ledger to identify dishonest reports, but we do force the report ledger to detect dishonesty at a coarser time granularity. That said, we think that this trade-off between transparency and privacy can be explored much more deeply and deserves the attention of the research community.

7 Related Work

MorphIT shares a common vision with Network Confessional (NC) [11]: give ISPs an interface for supplying quality-of-service feedback to end users. Both proposals are different from approaches to Internet accountability that either resort to alternative Internet protocols, such as AIP [8] and APIP [32], or alternative designs of the whole Internet architecture, such as SCION [38]. However, unlike our system, NC does not target anonymity guarantees and may interfere with existing anonymous communication systems, Tor [6], by enabling global passive de-anonymization attacks [17, 27, 31].

Many before us have proposed designs for privacy-preserving data collection in networks [7, 13, 15, 22, 23, 26]. Among them, SEPIA [13] uses secure multiparty computation (MPC) [37], which allows learning of aggregate network statistics without disclosing local input data, but assumes that learning is secure in itself. Using the optimized SEPIA primitives, one can possibly devise protocols tailored to the applications we consider, i.e. SLA and neutrality verification, however we do not want to restrict a richer class of analysis on the generated traffic reports. Moreover, the SEPIA approach, applied to our context, would impose communication overhead and require coordination among domains.

State-of-the-art results in gathering privacy-preserving statistics on anonymity networks were obtained by PrivCount [26]. However, as seen earlier in this paper (Section 5.4), PrivCount is far from providing the report utility we seek. In the same vein, Pan-Private BLIP [7] provides theoretical privacy guarantees (ϵ-*differential pan privacy* [21]) and collects data over long periods of time (e.g. on a daily basis).

To balance anonymity and transparency, we also explored solutions based on *traffic morphing* [36]. By employing convex optimization techniques, HOPs could obfuscate features of traffic reports (e.g. the packet count distribution or the exact pattern of aggregates) while limiting the loss of utility (e.g. minimizing the L1 norm of the difference between the original and noisy aggregate sequences). Nevertheless, we decided not to include it in our final evaluation, as it makes the reports less accurate without the benefit of theoretical privacy guarantees provided by PrivCount.

8 Conclusion

We assessed the risk that a basic transparency system would pose for low-latency anonymity networks like Tor. We found that there is indeed a risk, in the sense that the traffic reports published by a transparency system can help a passive adversary deanonymize flows. We also found that adding noise to the traffic reports so as to ensure differential privacy would destroy report utility, i.e., make them unusable in the context of a transparency system. Instead, we proposed MorphIT, an algorithm that coarsens the time granularity of traffic reports in order to obfuscate the flow patterns that are most vulnerable to tracing. We experimented with Poisson and real flows and showed that MorphIT significantly improves flow anonymity even in highly adversarial scenarios where there are as few as 64 flows per aggregate.

Acknowledgments: We would like to thank Carmela Troncoso, Aaron Johnson, and the anonymous reviewers for their deep, constructive feedback, and Pavlos Nikolopoulos for his help with the PrivCount comparison. This work was not supported by any grant.

References

[1] AT&T SLA. http://cpr.att.com/pdf/se/0001-0003.pdf.
[2] CAIDA Traces. http://www.caida.org/data/.

[3] Comcast SLA for Wholesale Dedicated Internet. https://www.comcasttechnologysolutions.com/sites/default/files/2016-09/Service%20Level%20Agreement.pdf.

[4] Comcast vs. Netflix: Is this really about Net neutrality? https://www.cnet.com/news/comcast-vs-netflix-is-this-really-about-net-neutrality/.

[5] Net neutrality by country. https://en.wikipedia.org/wiki/Net_neutrality_by_country.

[6] Tor: Anonymity Online. https://www.torproject.org/.

[7] Mohammad Alaggan, Mathieu Cunche, and Sébastien Gambs. Privacy-preserving Wi-Fi Analytics. *Proceedings on Privacy Enhancing Technologies*, 2018(2):4–26, 2018.

[8] David G Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable internet protocol (aip). In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 339–350. ACM, 2008.

[9] Katerina Argyraki, Petros Maniatis, David Cheriton, and Scott Shenker. Providing packet obituaries. In *ACM HotNets-III*, 2004.

[10] Katerina Argyraki, Petros Maniatis, Olga Irzak, Subramanian Ashish, and Scott Shenker. Loss and delay accountability for the Internet. In *2007 IEEE International Conference on Network Protocols(ICNP)*, pages 194–205. IEEE, 2007.

[11] Katerina Argyraki, Petros Maniatis, and Ankit Singla. Verifiable network-performance measurements. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 1:1–1:12, New York, NY, USA, 2010. ACM.

[12] Boaz Barak, Sharon Goldberg, and David Xiao. Protocols and lower bounds for failure localization in the Internet. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 341–360. Springer, 2008.

[13] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 15–15, Berkeley, CA, USA, 2010. USENIX Association.

[14] Sambuddho Chakravarty, Marco V Barbera, Georgios Portokalidis, Michalis Polychronakis, and Angelos D Keromytis. On the effectiveness of traffic analysis against anonymity networks using flow records. In *International conference on passive and active network measurement*, pages 247–257. Springer, 2014.

[15] Ruichuan Chen, Alexey Reznichenko, Paul Francis, and Johanes Gehrke. Towards statistical queries over distributed private user data. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 169–182, San Jose, CA, 2012. USENIX.

[16] David Clark. The design philosophy of the DARPA Internet protocols. *ACM SIGCOMM Computer Communication Review*, 18(4):106–114, 1988.

[17] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50. Springer, 2004.

[18] Goulet David, Johnson Aaron, Kadianakis George, and Loesing Karsten. Hidden-service statistics reported by relays. *Tech. rep., The Tor Project, Inc.*, 2015.

[19] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology (EUROCRYPT 2006)*, volume 4004, page 486–503, Saint Petersburg, Russia, May 2006. Springer Verlag.

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

[21] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Proceedings of The First Symposium on Innovations in Computer Science (ICS 2010)*. Tsinghua University Press, January 2010.

[22] 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, pages 1068–1079, New York, NY, USA, 2014. ACM.

[23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[24] Sharon Goldberg, David Xiao, Eran Tromer, Boaz Barak, and Jennifer Rexford. Path-quality monitoring in the presence of adversaries. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '08, pages 193–204, New York, NY, USA, 2008. ACM.

[25] Amir Houmansadr and Nikita Borisov. The need for flow fingerprints to link correlated network flows. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 205–224. Springer, 2013.

[26] Rob Jansen and Aaron Johnson. Safely Measuring Tor. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1553–1567, New York, NY, USA, 2016. ACM.

[27] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348. ACM, 2013.

[28] Thomas Karagiannis, Mart Molle, Michalis Faloutsos, and Andre Broido. A nonstationary Poisson view of Internet traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1558–1569. IEEE, 2004.

[29] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 303–314. ACM, 2013.

[30] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[31] Steven J Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *International*

Workshop on Privacy Enhancing Technologies, pages 167–183. Springer, 2007.

[32] David Naylor, Matthew K Mukerjee, and Peter Steenkiste. Balancing accountability and privacy in the network. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 75–86. ACM, 2014.

[33] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity – a proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[34] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 735–746. ACM, 2010.

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

[36] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9, 2009.

[37] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[38] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. SCION: Scalability, control, and isolation on next-generation networks. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 212–227. IEEE, 2011.

[39] Xin Zhang, Abhishek Jain, and Adrian Perrig. Packet-dropping adversary identification for data plane security. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 24:1–24:12, New York, NY, USA, 2008. ACM.

[40] Xin Zhang, Chang Lan, and Adrian Perrig. Secure and scalable fault localization under dynamic traffic patterns. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 317–331. IEEE, 2012.

[41] Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, and Patrick Tague. Shortmac: Efficient data-plane fault localization. In *NDSS*, 2012.

A Processing Overhead

The online version of our algorithm operates on active windows of size ω, and its runtime should scale linearly with the number of active windows within the observation window.

Fig. 10 shows the average runtime of MorphIT$_{100}$ as a function of the maximum bin size τ. There are $\phi = 512$ flows per aggregate, and the adversary's observation window size is $w = 10$s. The implementation is in Matlab, and it is running on an Intel Core i7-7700 3.6GHz CPU.

The figure confirms our algorithm's scalability. Moreover, given that our implementation is far from optimized for performance, and it is running on a single core, these numbers indicate that our algorithm is deployable.

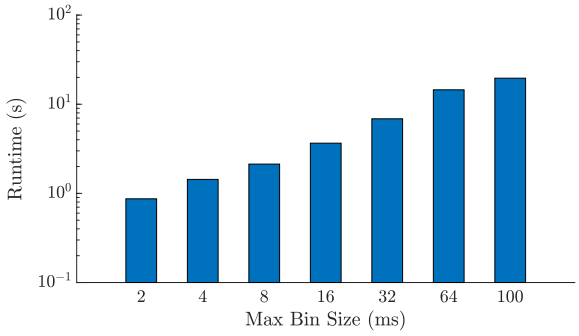


Fig. 10. Average runtime of MorphIT$_{100}$. $\phi = 512$ flows per aggregate. $w = 10$s.