Practical Anonymity Sets in a Pseudonymous Forum Setting

Benjamin Chen

Abstract

Pseudonymous forums are online websites where users can post publicly visible content and participate in discussions under a pseudonym. Such forums are not perfectly private, as their privacy can be compromised to traffic analysis attacks. However, many methods of providing perfect privacy to such a system come with a heavy performance cost—whether in bandwidth or latency. We examine the practicality of anonymity sets, a defense against such attacks that can still provide a formal privacy guarantee with less performance losses, and attempt to simulate their implementation in a real-world setting using real data scraped from Reddit, a popular pseudonymous forum. We try various different methods of creating these anonymity sets, finding that K-means with some dimensionality compression yields decent results; we also propose a method of defining a common traffic budget for members of a set. We find that anonymity sets are a feasible defense against such attacks in the pseudonymous forum setting.

Contents

1 Introduction

Pseudoanonymous public forums such as Reddit allow users to post on the internet under pseudonyms, which can be unrelated to their real name. Any person with access to the internet can freely examine the content posted on the forum, the time it was posted, and by whom (which username) the content was posted under. Finally, each individual user can only post from pseudonyms that they "own". Some common examples of public forums include websites such as Reddit and Stack Overflow.

Despite the fact that users' real information is never necessarily accessible on the forum, such systems remain vulnerable to intersection attacks from a global adversary. There exists a large amount of literature describing intersection attacks [1] [2] [3], but they are generally based on finding links between the activity of users (i.e. monitoring a individuals' internet traffic) and accounts on the forum.

Such attacks can be prevented by anonymizing users' sending activity (i.e. making every user's activity look identical). While there have been some efforts to apply this solution to private messenging settings, they have not been previously applied to pseudonymous public forums due to a key difference between the two systems: the traffic of each pseudonym is completely public in pseudonymous public forums. Thus, it becomes necessary to make every single user's traffic look identical in order to maintian perfect privacy, which is unreasonable in practice due to the large range in how different users use the forum.

A compromise between privacy and performance can be found in the concept of anonymity sets. Each anonymity set consists of a number of users, and within a set, every users' traffic is identical. However, different sets are allowed to have different traffic patterns, allowing more flexibility within a set and potentially better performance. In theory, this drastically improves performance while still maintaining a formal privacy guarantee (you are indistinguishable from the other users in your anonymity set). Although the concept of anonymity sets has been explored in previous works [4], there have been few efforts to explore anonymity sets in a real-world setting.

In this paper, I examine the practicality of such attacks in an anonymous forum setting. I use real data scraped from the online forum Reddit to simulate the creation of anonymity sets and users' traffic over time. In the process, I also propose a method for forming anonymity sets and setting budgets for each set to optimize for performance while still maintaining a formal privacy guarantee. Although I make some simplifying assumptions, my simulation shows that anonymity sets are practically implementable in real-world online pseudonymous forums.

2 Background

I first assume the existence and implementation of a mix [5], a system that ensures that the input and output data cannot be linked. Despite this, mixes do not hide users' participation, and they remain vulnerable to statistical disclosure attacks, or intersection attacks [1]. These attacks use traffic analysis to, in my case, link users to their pseudonyms.

Such attacks can be prevented by forcing all users to have the same exact sending pattern, thus making every user indistinguishable to the adversary. This can be done through the

usage of dummy traffic and by delaying some posts for a later time. However, as individuals' activity on any given website vary widely across all users, this can lead to performance losses. For example, relatively inactive users may be almost constantly sending dummy traffic, which could be a major issue for bandwidth-limited connections, while more active users will have many of their messages postponed and face a great deal of latency.

Although some implementations accept these performance losses or use scaling of the entire network to improve performance [6], anonymity sets can also be used to improve performance. In such an implementation, I would place each user in a set with users with similar traffic patterns, then require that all users look the same within a set (the activities of two different sets can be different). This reduces the amount of overhead needed, thus improving performance at the cost of potentially leaking information and a smaller privacy guarantee (less individuals with identical traffic patterns).

One important difference between pseudonymous forums and standard messaging is that in a forum, the set of senders and recipients is disjoint; each user only "sends" their posts to their pseudonym. Thus, the set of senders is all users, and the set of recipients is all pseudonyms. In standard messaging, this is not necessarily true: one account or user may be able to both send and receive messages. The implementation of anonymity sets has been examined in the latter case previously [4]. In this paper, I examine anonymity sets in an pseudonymous forum.

3 System Model

3.1 Threat Model

As stated previously, due to the nature of a pseudonymous forum, only sending relationships are considered and a disjoint set of senders and recipients is assumed. I also assume the existence of a modified mixnet which does not require constant participation, does not require uniform traffic patterns, and only works within each anonymity set. Thus, the adversary does not gain any knowledge from the content of the messages so that their only available information is the timestamps and quantity of messages sent. All comments are also treated equally regardless of their length in the simulation; in practice, short comments would be padded up to a certain size and longer ones would be divided to travel in several packets. I also assume a one-to-one relation between users and pseudonyms. Finally, I assume that no information about the identity of a user is leaked by the content of their comments.

I first create discrete rounds during which users are allowed to send message. During a round, a user can try to post comments, but they are held and the data only leave the user's machine at the end of the round. Such rounds are defined to be the same for all members of the anonymity set so that every user in the set sends traffic at the same time. In our case, each round is always 1 hour long, but this is tunable depending on the specific needs of the network. After data leaves the users' machines, they travel to the shim.

The shim disconnects the comments from the users that post them (e.g. through a mix), and enforces the sending pattern required by the user's anonymity set. It does this through

the following:

- A queue is initialized for each user
- Whenever new comments are received from a user, they are added to that user's queue
- At the start of every round, for each user (let n be the number of comments in the user's queue and b be their budget for that round):
- If $n \geq b$, send the first b comments to the network.
- If $n < b$, generate $b - n$ dummy comments and send those along with the user's n real comments to the network.

Once comments are received by the network, they are immediately visible on the forum under the pseudonym they were sent under.

We define unlinkability as follows:

- Let the adversary pick any user U in anonymity set S.
- Generate a random bit $b = 0$ or 1.
  - If b is 1, give the adversary the pseudonym owned by U.
  - If b is 0, give the adversary a randomly selected pseudonym owned by a user in S that is not U.
- If the adversary cannot determine b with more than a negligible advantage over 50% accuracy, unlinkability holds.

Given the assumptions we have made, unlinkability will hold in our system, as the adversary will have the exact same data on U as they will on every other user in S.

3.2 Dataset

Throughout this project, I use the Reddit comments datasets available freely from Pushshift.io [7]. Specifically, I use around a years' worth of data from 2011-12 (due to computation and space limits). I ignore all content of each comment, only looking at the timestamp and pseudonym it was posted under. Due to the relatively small size of Reddit at the time, the number of active users was small, and I only examined the 100 most active users (defined by total comments over the year).

3.3 Simulation

The data was first divided into two period: a train period consisting of the first three months of the year, and a test period consisting of the following nine months. The train period was used to create the anonymity sets, and the performance of the sets was monitored over the rest of the year (the test period). This is both to emulate a real-world implementation of anonymity sets (as some data about the users would be needed to create the initial

sets) and to examine the performance of the sets over a longer period of time. Finally, a parameter for a minimum set size k was created. This guaranteed that each user would be in an anonymity set of at least k, meaning that there are always $k-1$ other users that have the exact same traffic patterns as they do.

4 Results

For the purposes of acquiring data and comparing different setups, I set the budget of anonymity set S at each round i to be $\max(u_i, u \in S)$. This is unrealistic, as this would require the system to know u_i before round i for every user u, but it facilitates comparing different setups by combining two dependent variables (postponed messages and dummy messages) into one.

4.1 Comparison of Different Grouping Methods

To account for randomness in some of the grouping methods, the averages over 1000 trials of each grouping method was taken. The methods for some of the clustering algorithms (k-means, Gaussian mixture, spectral) were also modified to enforce the minimum anonymity set size parameter. Specifically, after the clusters were formed initially, sets that were smaller than the minimum set size were combined with other sets until they reached the minimum. The entry titled "Mean" was simply sorting the users by their mean activity (in comments/hour), then dividing the list of users into continuous sets of equal size. The algorithms for k-means, Gaussian mixture, and spectral clustering were provided by the Python package Scikit-learn [8]. As shown, k-means and Gaussian mixture offered similar average performance, while the other three methods trailed behind. K-means was ultimately chosen due to the fact that it ran faster than the Gaussian mixture model.

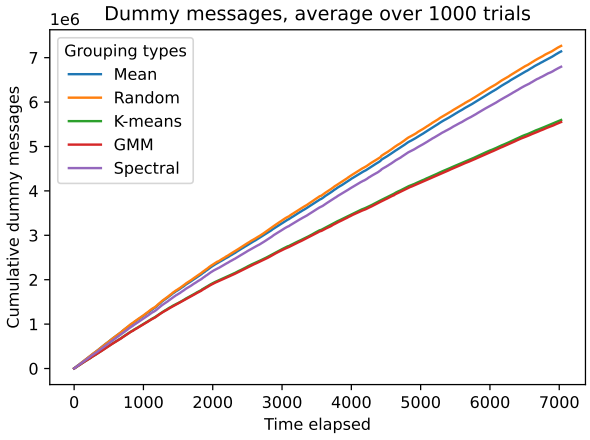

Figure 1: Performance of different set initialization methods

4.2 Relation Between Set Size and Performance

The implementation of anonymity sets leads to large improvements in performance over keeping perfect privacy in the model. Further, a higher minimum set size corresponds to worse performance. This is likely because with larger sets, the people in each set have less similar traffic patterns to each other, requiring more dummy or postponed traffic to enforce anonymity within the set. The graph shows this trend for k-means clustering, which was found to be the best algorithm for group creation.

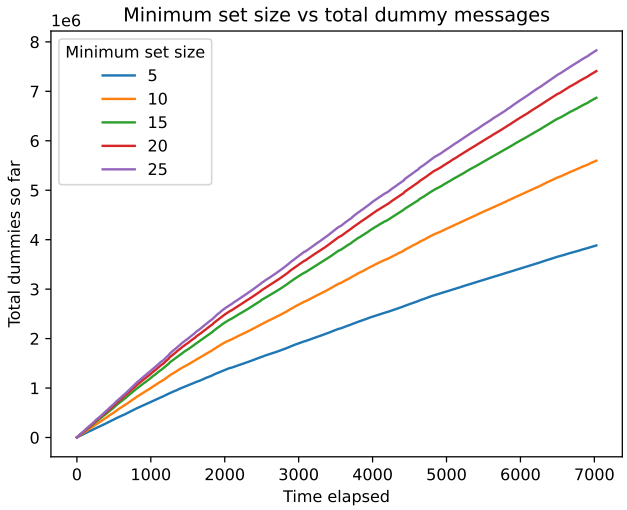

Figure 2: Performance of different minimum set sizes

4.3 Inertia as an Indicator of Performance

A high inertia in the anonymity sets roughly correlates with a higher amount of dummy messages and a lower amount of privacy, although there is a significant variation across clusterings with same total inertia. 4 shows the inertia of 1000 different simulations using k-means clustering as well as the total number of dummy messages sent over the test period; note the positive correlation between inertia and the amount of dummy messages sent. This implies that a high inertia (clusters with members that are far apart) corresponds to worse performance and shows the validity of inertia as an indicator of performance.

4.4 K-means Train Time

The clusters stabilized fairly quickly, with no significant performance or privacy differences from training for 10 to 500 iterations. Thus, the clusters can be created relatively quickly, although more testing is needed on larger datasets with more users.

4.5 Defining a Budget

Simply setting the budget for each set to be the mean fails in the long run, as postponed messages will accumulate and lead to extremely high latency. Thus, it becomes necessary to

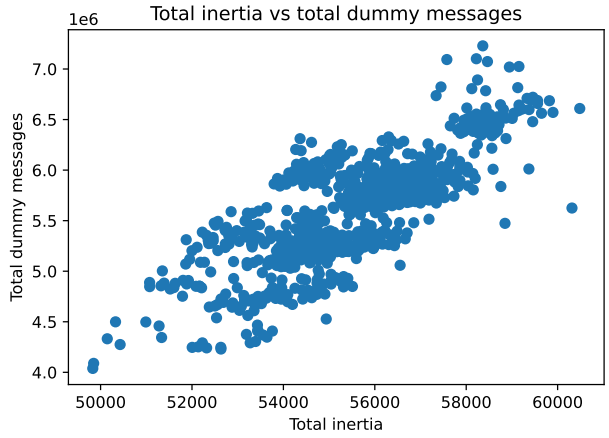

Figure 3: Inertia as an indicator of performance

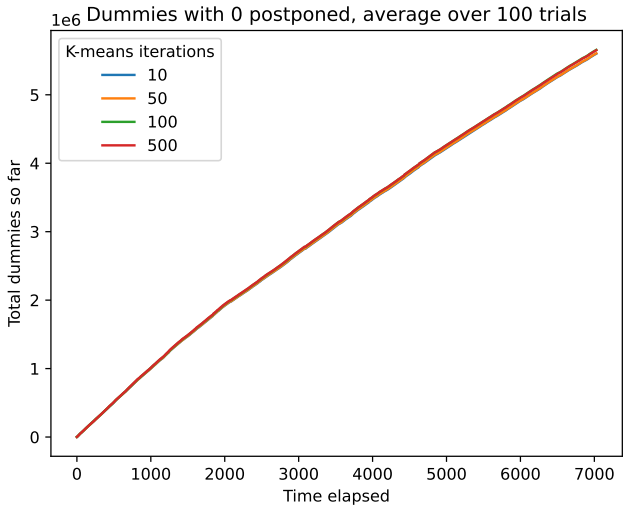

Figure 4: Performance with different k-means training iterations

set the budget slightly higher than the mean. In the simulation, I add a tunable multiple of the standard deviation of the activity of all users in the set. Here, I also adopt the test-train split as mentioned earlier to simulate a real implementation of this system on Reddit. Note that as the budget is increased, the amount of dummy traffic increases while the amount of postponed messages decrease.

5 Conclusions and Future Work

I have shown that anonymity sets are feasible in pseudonymous forums and proposed a method for creating them efficiently and effectively. However, future work remains to be done. First, using a larger and more recent dataset for the simulation would provide more reliable evidence and may display some different trends. Using machine learning methods to create the sets can also be experimented with, as there is a large amount of data available

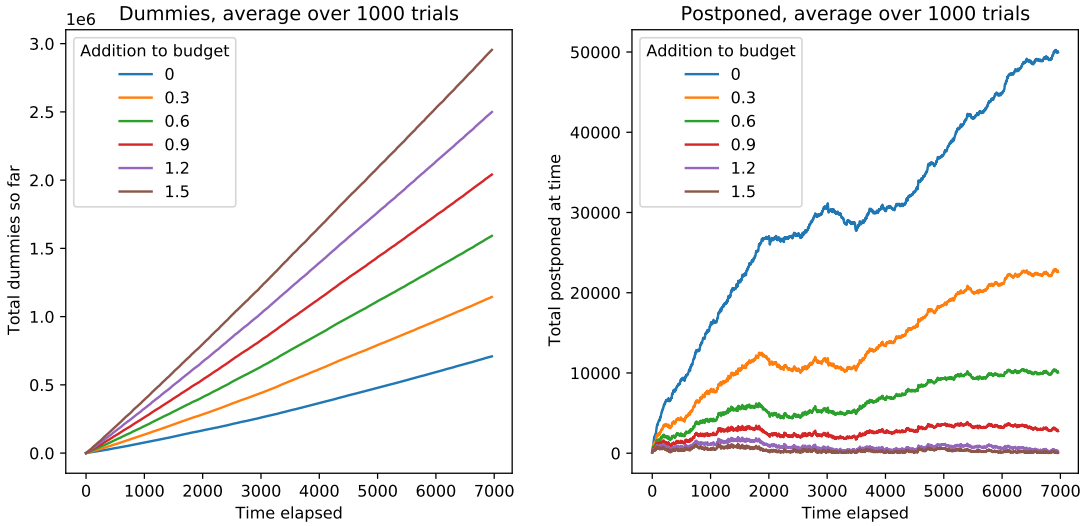

Figure 5: Performance with different additions to the budget

for a model to work with. Finally, the problem of changing sets can also be examined in the future. However, this is a challenging issue, as each set change can hurt the privacy of the users who were involved in the change.

6 Acknowledgements

I would like to thank Kyle Hogan for mentoring me and supporting me throughout the project. I would also like to thank the MIT PRIMES program for supporting this project and giving me this research opportunity.

References

[1] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. *Lecture Notes in Computer Science*, 2004.

[2] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided statistical disclosure attack. *Lecture Notes in Computer Science*, 2007.

[3] Nayantara Mallesh and Matthew Wright. The reverse statistical disclosure attack. *Lecture Notes in Computer Science*, 2010.

[4] David Isaac Wolinsky, Ewa Syta, and Bryan Ford. Hang with your buddies to resist intersection attacks. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[5] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 1981.

[6] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[7] Reddit comments. http://files.pushshift.io/reddit/comments/.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.