pyDiff: Differential Privacy as a Library

Zichuan Tian   Arjun Kashyap
University of Wisconsin-Madison
CS 839

Abstract:

Differential privacy is a group of formal mathematical mechanisms to allow privacy preserving data analysis. It allows a database owner to release statistics about a private database, while preserving the privacy of individual records. Following the popularity of big data, machine learning and artificial intelligence, it is essential to protect sensitive data leakage in the analysis. Since Python is the predominant language used in such analyses, we are providing a library to allow database owner to exercise differential privacy without worrying about the implementation. We experimented our library using network trace data. We are also providing an implementation to analyze live data stream.

Introduction:

I. Overview

The complexity of modern networks makes access to real-world data critical to networking research. Without this access it is almost impossible to understand how the network behaves and how well a proposed enhancement will function if deployed. But obtaining relevant data today is a highly frustrating exercise for researchers and one that can often end in failure. The released data is heavily sanitized (e.g., payloads are removed) and anonymized, limiting their research value. Research and real mishaps has demonstrated that anonymization is vulnerable to attacks that infer sensitive information. Because of this fear, many data owners today prefer the safer option of not releasing data at all.

We are providing an implementation of differential privacy to analyze real world network trace data, as they are essential for network research. Our code can also be adapted to analyze other datasets. An introduction about differential privacy can be found here [1]. Although big data, machine learning, artificial intelligence are gaining more popularity these days, data owners are reluctant to release their data because of privacy concerns and users are more concerned about the vast amount of data collected by applications. Un-randomized privacy mechanisms like k-anonymity provide good privacy guarantee when no inferences of the data set can be made. However, with auxiliary information, an attacker can reliably gain knowledge about individuals [2].

Differential privacy is group of privacy mechanisms that uses statistical methods to prevent individual records leakage from such attacks. It provides strong and formal privacy guarantees which are important prerequisite for data owners. However, the strong guarantees of differential privacy do not come for free. Privacy is preserved by adding noise to the output of the analysis, imposing on its accuracy. The added noise is scaled to mask the presence or absence of small sets of records. While the magnitude of the noise is typically small, and the distribution of the noise is known to the analyst, it can render sensitive analyses useless. While differential privacy mechanism provides the strongest privacy guarantee, some are not practical in real world because of computation complexity, space complexity or compromised result accuracy. We are providing a library to exercise differential privacy mechanisms on network trace data analysis. We tested our implementation in multiple approaches to validate its usefulness.

A network researcher maybe interested in both live and past data, including IP address, MAC address, port number, payloads, packet size, graph and etc. We implemented and evaluated the performance of several well known differential privacy mechanisms (randomized response, Laplace, stable transformations, exponential, two level counting counting [3], and etc.) on querying such kinds of data sets. We tested the utilities such as privacy, accuracy, and cost of each mechanism.

All the above analysis is on static data sets. There are scenarios where live network logs are needed to infer some statistics. Traditional differentially private mechanisms do not work for continuous data collection or streaming data. Hence, we applied the differentially private continual counter to answer queries like top-k hosts in the network generating maximum traffic/payload or top-k ports being currently used to determine which applications are using up the network resources/bandwidth. We studied the noise introduced by this mechanism and if any meaningful inferences could be drawn from it.

II. Background on Differential Privacy

Differential privacy requires that a computation exhibit essentially identical behavior on two data sets that differ only in a small number of records. Formally, let A and B be two datasets and A $\ominus$ B be the set of records in exactly one of them. Then, a randomized computation M provides ε-differential privacy if for all A and B and any subset S of the outputs of the computation:

$$Pr[M(A) \in S] \leq Pr[M(B) \in S] \times exp(\varepsilon|A \ominus B|)$$

That is, the probability of any consequence of the computation is almost independent of whether any one record is present in the input. For each record, it is almost as if the record was not used in the computation, a very strong baseline for privacy. The guarantee assumes that each record is independent of the rest and applies to all aspects of the record. So, if each record is a packet, differential privacy protects its IP addresses, payloads, ports, etc., as well as its very existence.

Differential privacy is preserved by adding “noise” to the outputs of a computation. Intuitively, this noise introduces uncertainty about the true value of the output, which translates into uncertainty about the true values of the inputs. The noise distributions that provide differential privacy vary as a function of the query, though most commonly we see Laplace noise (a symmetric exponential distribution). The magnitude of the noise is calibrated to the amount by which the output could change should a single input record arrive or depart, divided by ε. The value of a perturbed result depends greatly on the data, however; a count accurate to within ±10 may be useful over a thousand records but not over ten records. The noise distribution is known to the analyst, who can judge if the noisy results are statistically significant or not without access to the actual data.

The parameter ε is a quantitative measurement of the strength of the privacy guarantee. Lower values correspond to stronger guarantees, with $\varepsilon = 0$ being perfect privacy. Typically, $\varepsilon \leq 0.1$ is considered strong and $\varepsilon \geq 10$ is considered weak. We are not advocating specific levels of differential privacy as sufficient but are instead interested in understanding the trade-off between accuracy and privacy.

Unlike differential privacy, many alternative formulations do not provide a direct guarantee or are vulnerable to auxiliary information that the attacker might possess. Consider, for example, k-anonymity, which provides guidance on releasing data such that the identity of individual records remains private [7]. A release provides k-anonymity if the information for each record cannot be distinguished from at least k-1 other records. However, this definition provides no guarantee in the face of auxiliary information that may exist outside of the released dataset. Such information can break anonymization.

As another example, consider reducing information leakage as a way to preserve privacy. The reasoning is that the fewer bits of information that an analysis leaks about specific records, the more privacy is protected. However, this reasoning is indirect at best and fallacious at worst. Revealing even one bit can lead to significant loss in privacy. For example, revealing if hosts A and B communicate requires only one bit of information but may represent an unacceptable loss in privacy. Moreover, any such scheme always leaks at least one bit, in response to: “did the analysis reveal too many bits?” This response bit can encode very sensitive information, and is always revealed to the analyst.

III. Implementation

The complexity of modern networks makes access to real-world data critical to networking research. The networking logs contain IP addresses, MAC addresses, ports, payloads, packet lengths, and etc, from which the privacy of a user could be compromised. Previously the networking community released data which is heavily sanitized (e.g., payloads are removed) and anonymized, limiting their research value. Also, anonymization is vulnerable to attacks that infer sensitive information, many data owners today prefer the safer option of not releasing data at all. But, differential privacy (DP) guarantees that the presence or absence of individual records is hard to infer from the analysis output. DP is resilient to collusion, supports multiple interactive queries, and is also independent of any auxiliary information that an attacker might possess. Thus, earlier work has shown that it is possible to conduct network trace analysis with differential privacy guarantees [4].

In the study, we applied various differentially private mechanisms (randomized response, Laplace, Gaussian, and else to come) on network traces and observe the outcome as the strong guarantees of differential privacy do not come for free. Privacy is preserved by adding noise to the output of the analysis, imposing on its accuracy. Hence, we wish to observe how various DP mechanisms affect privacy and accuracy of network data.

Our first experiment observed to what extent the accuracy of aggregate statistics (in the network logs) could be affected by adding noise as the number of records in the dataset increase. To achieve this, we used the average packet length as the subject. We performed this experiment in both local and global privacy setting. In local setting, noise is added for each record and then the aggregate is computed, whereas in global setting, the noise is added at the end when the average packet is computed (just once and not for every record). The noise distribution used were Laplacian and Gaussian. For both these settings we kept increasing the number of records by a factor of 10,000 up to 100,000. At each step we measured the true average length and the noisy average length (the output from DP algorithm). Next we studied the accuracy privacy trade-off. We did this by varying epsilon values among 0.1, 1 and 3 for the Laplace Mechanism. Hereby an epsilon value of 0.1 indicates strong privacy (and hence more noise) and low accuracy. Again for this experiment we chose the average packet size.

Often, in addition to simple aggregates like counts and averages, we are interested in understanding the underlying distribution itself, which may have informative ranges or modes. In networking analyses, distributions are often studied using the cumulative distribution function (CDF). CDF(*x*) is the number of the records with value smaller or equal to *x*. One common packet-level analyses is measuring the distribution of ports. We compute the CDF of ports without noise and with different values of epsilons (0.1, 1 and 3).

In our last experiments, we implemented two level counting algorithm to reduce privacy leakage in online stream data querying. The idea is that when items from the stream come, we group them in blocks. Within a block, we use a simple counting mechanism, and on top of that, we run another simple counting algorithm, treating each block as a single element. Because each block has been sanitized, privacy leakage are reduced when queried multiple times. The algorithm is described in [3] 3.3. We are providing our library and experimental results here (https://github.com/ZtianWisc/pyDiff).

Experimentals:

Network datasets (exported to JSON format) are from the U.S. National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC). We applied Laplace and Gaussian mechanisms on querying mean packet (ip payload size) size to give differential private response. We parsed the input file and stored packet sizes in an array. Then both local and global versions of Gaussian and Laplace mechanisms are applied on the array. In the global version mechanisms, the noise is associated with epsilon (or sigma in Gaussian), sample size and maximum difference between two individual records. However in the local version, as the sample size increases, the noise of the output goes up to $\sqrt{n}$ of the global version. Thus making the result less accurate. This implies that in the real world, when running local version of differential private mechanisms, although it benefits to protect man-in-the-middle attacks, it is less accurate at the same privacy level. The results are shown in *figure 1* below.

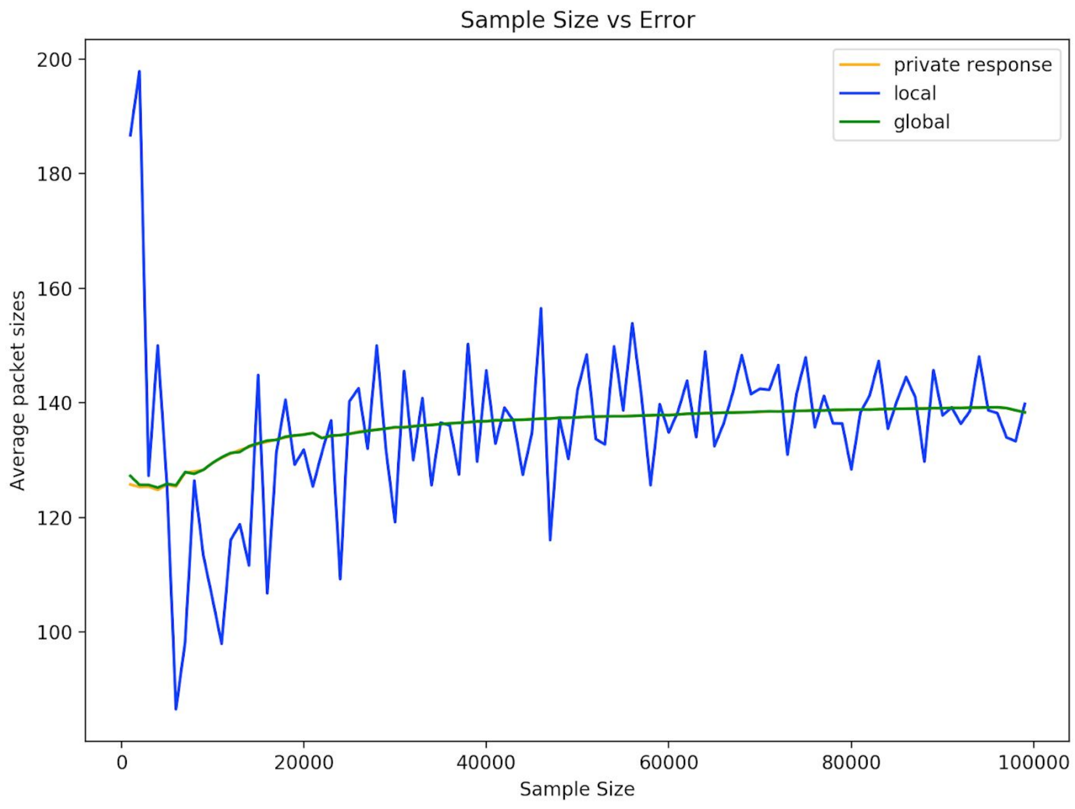


Figure 1. Average packet size versus input sample sizes. The noise in local version is about $\sqrt{n}$ *times higher than global version, where n is sample size. Here k, the max difference between individual records is around 1000, and epsilon is chosen to 1.0.*

The benefit of using local version mechanisms is that local differential privacy guarantees that it is difficult to determine whether a certain user contributed to the computation of an aggregate by adding slightly biased noise to the data. Even an attacker captured the response on-the-fly, s/he cannot determine the true value. This mitigates user's concern about data collected by government, or big companies leaking before transporting to the database.

We have also evaluated accuracy privacy tradeoff. Given a desired privacy level, noise parameter needs to be adjusted to maximize accuracy. In *figure 2*, as epsilon increases, noise goes down, thus meaning lower privacy, higher accuracy. Epsilon level of 0.1, 1, 3 are chosen. Batch sizes of 1000 are used, that is, each data point is average of adjacent 1000 packets captured in time ordering.

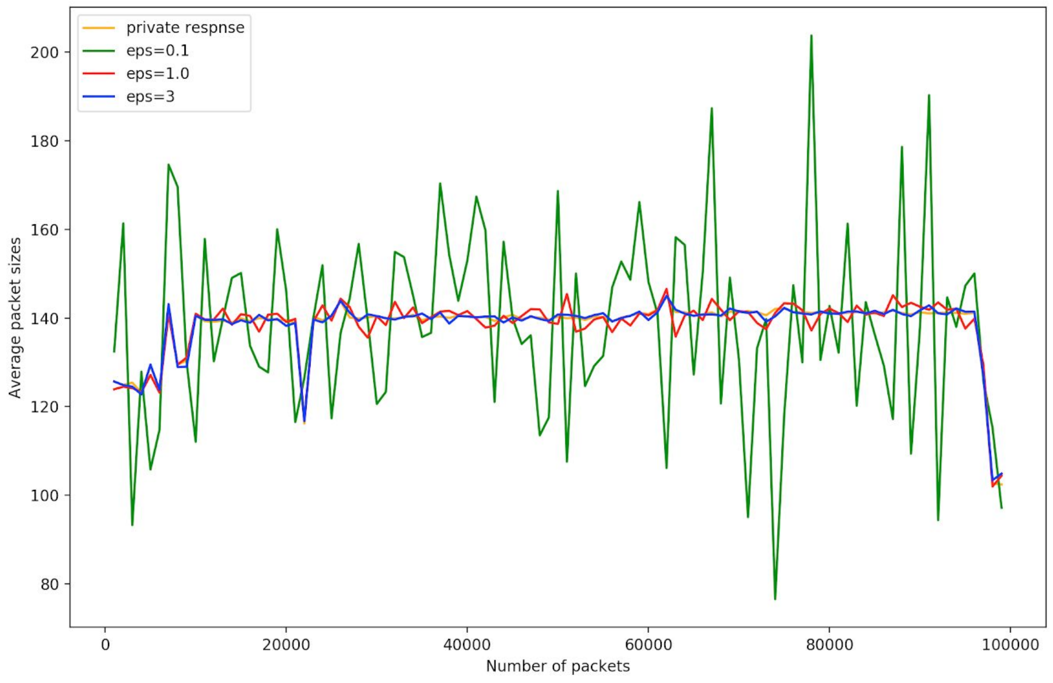


Figure 2. Average packet sizes versus sample sizes. Each line represents an epsilon value, with lower epsilon resulting in higher privacy.

Usually, in addition to simple aggregates like counts and averages, we are interested in understanding the underlying distribution itself, which may have informative ranges or modes. In networking analyses, distributions are often studied using the cumulative distribution function (CDF). CDF(*x*) is the number of the records with value smaller or equal to *x*. One common packet-level analysis measured the distribution of ports. Herein x is port number used in TCP connections. We compute the CDF of ports without noise and with different values of epsilons (0.1, 1 and 3). As we predicted, the noise level is ~10. It is enough to preserve privacy of individual records, while still useful for data analysis. In *figure 3* below, all lines aggregated to a single line because noise level is low ~ 0.1% even at low epsilon thus high privacy level. This means differential privacy mechanisms are useful in real world network trace analysis, and also many other data analysis scenarios.

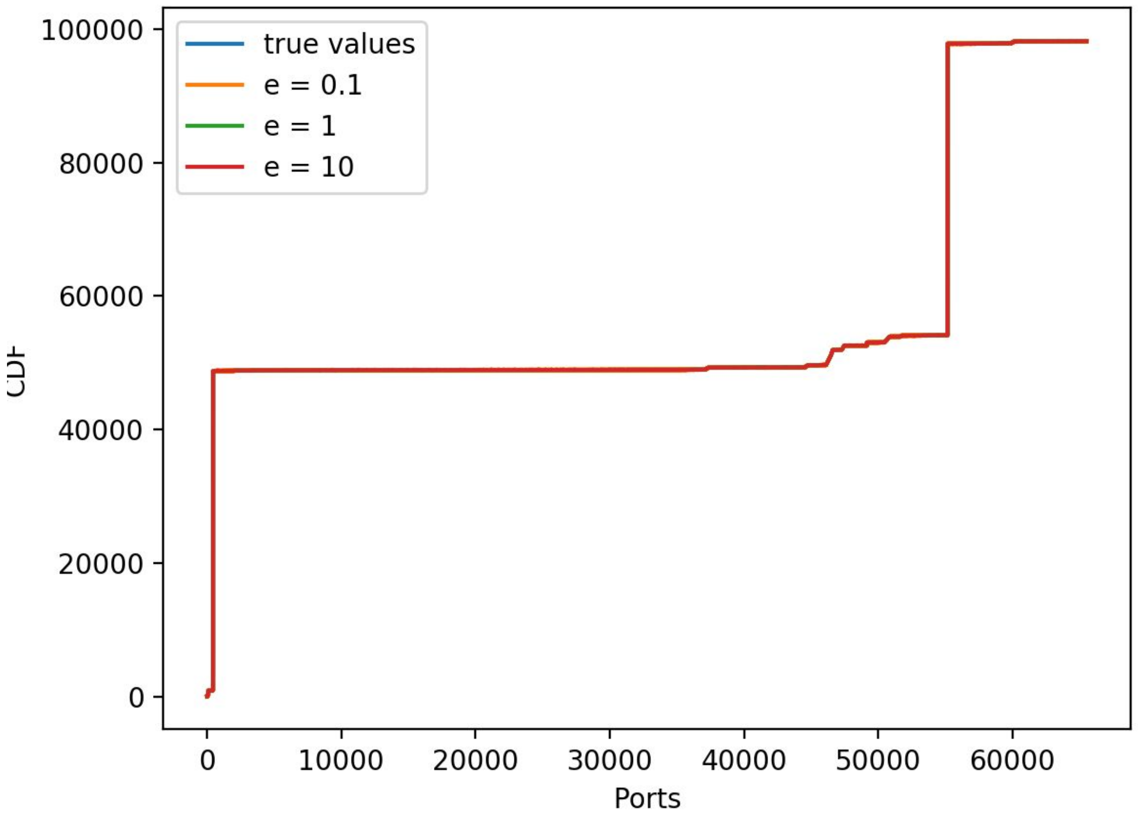
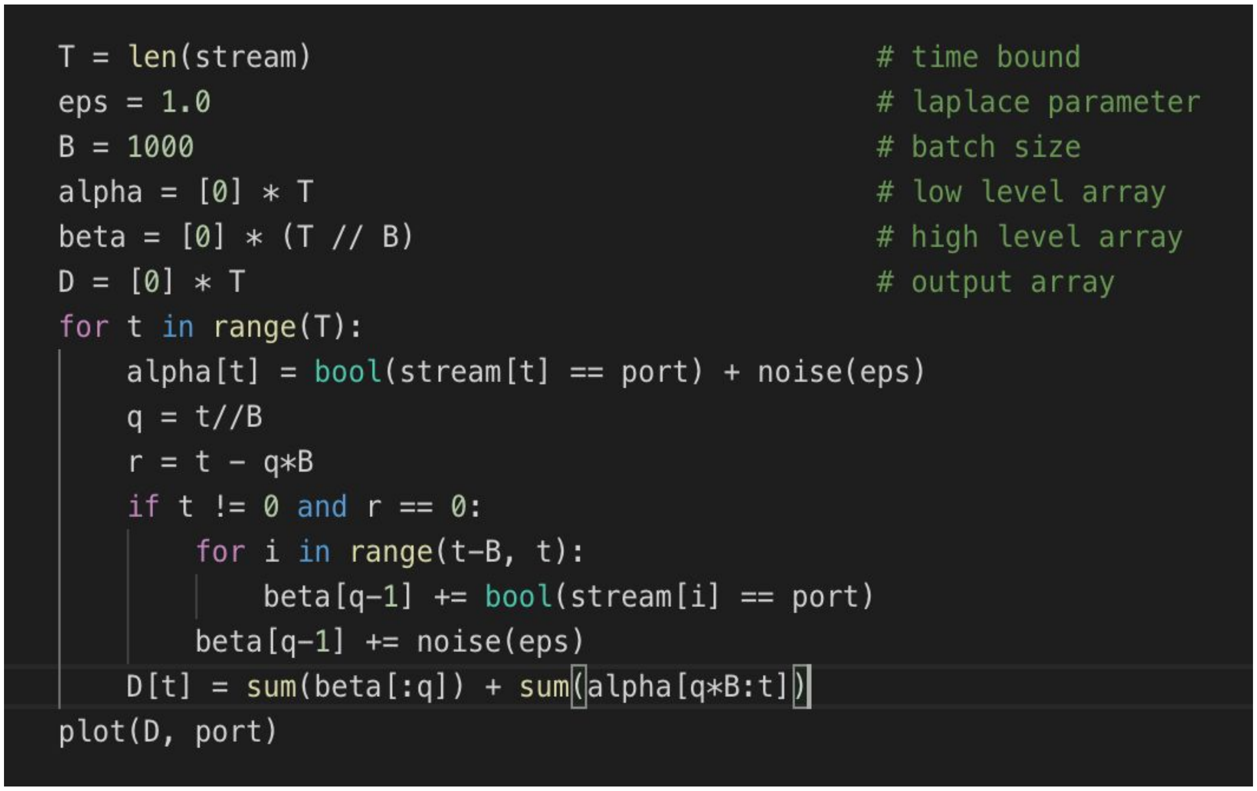


Figure 3. CDF(ports) vs ports. Even at low epsilon, high privacy level, the response is still very accurate, that is, very useful for database user. All lines aggregate to a single line because the noise is small compared with CDF values.

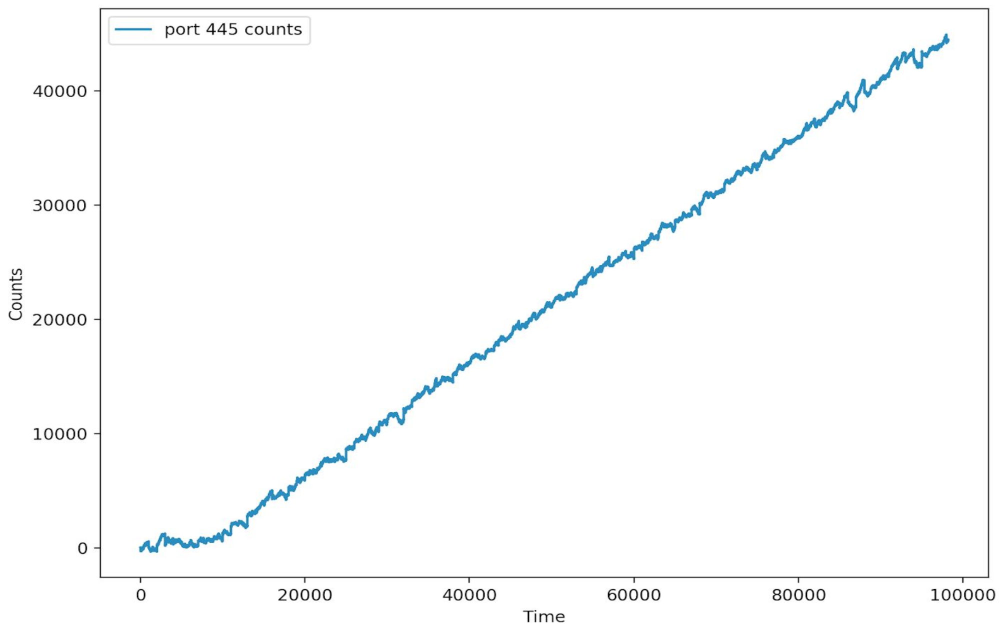
Lastly, we are providing an implementation of an online stream data analysis mechanism - two level counting mechanism. To analyze online data, one can simply add noise to q(t), where q(t) is the query result at time step t. However, privacy loss grows linearly with respect to the number of queries the fragment of data receives. A mechanism invented by Chan et al. [3] called *two level counting* can mitigate this disadvantage. The algorithm is described as follows: when items from the stream come, we group them in blocks. Within a block, we use a simple counting mechanism, and on top of that, we run another simple counting algorithm, treating each block as a single element. Because each block has been sanitized, privacy leakage are reduced when queried multiple times. Figure 4 is a snapshot of the implementation in python 3.



```
T = len(stream)                                         # time bound
eps = 1.0                                               # laplace parameter
B = 1000                                                # batch size
alpha = [0] * T                                         # low level array
beta = [0] * (T // B)                                   # high level array
D = [0] * T                                             # output array
for t in range(T):
    alpha[t] = bool(stream[t] == port) + noise(eps)
    q = t//B
    r = t - q*B
    if t != 0 and r == 0:
        for i in range(t-B, t):
            beta[q-1] += bool(stream[i] == port)
        beta[q-1] += noise(eps)
    D[t] = sum(beta[:q]) + sum(alpha[q*B:t])
plot(D, port)
```

Figure 4. Snapshot of the implementation of two level counting algorithm in python 3. Where *T* is the batch size. In real world, this would be the time interval size. *B* is block size. *Alpha* is an array of level 1 responses and *Beta* is an array of level 2, the block responses. *D* is an array of output versus time. The symbols used here are the same with the original paper [3].

We have experimented this implementation under various privacy levels. Under epsilon of 1.0 and 0.1 shown below in figure 5. We observe that lower privacy gives higher accuracy. However, at higher accuracy level (eps = 1.0), the results are still private enough to preserve the privacy of individual records. User of our implementation can choose their own privacy level based on how much accuracy and privacy s/he needs.



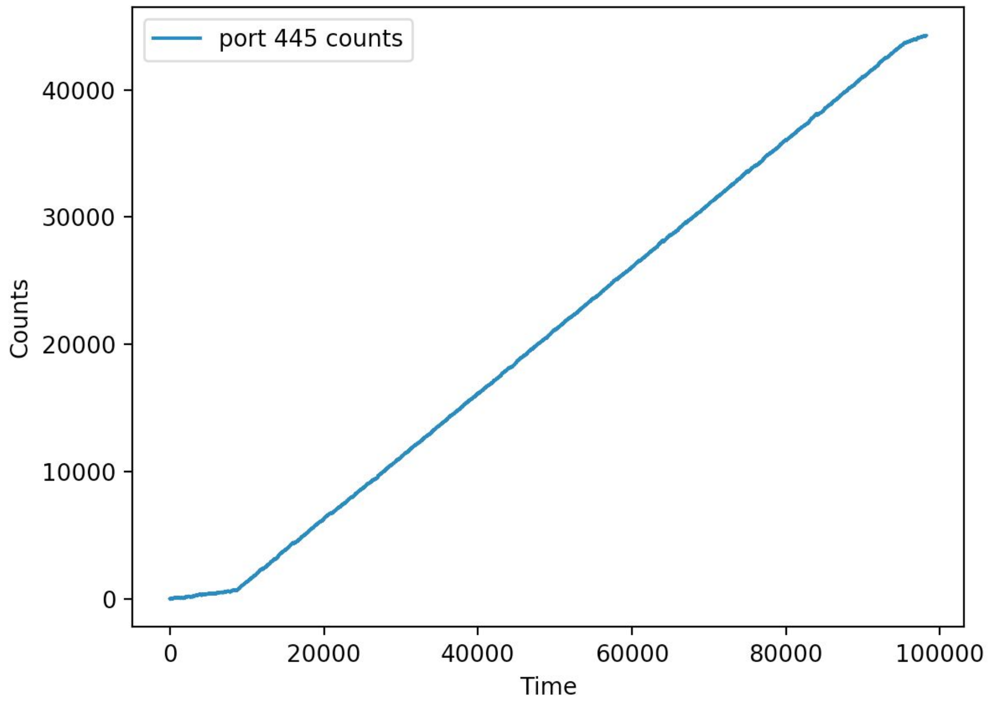


Figure 5. Counts of packets that use port 445 as TCP source port. The top level graph is at epsilon = 0.1 and the below one is at epsilon = 1.0. The below one is more accurate and useful but still private enough to preserve privacy.

Future work:

The guarantees of differential privacy are for the records of the underlying data set. Network data is interesting in that there are multiple possible privacy principles such as packets, flows, hosts, and services. If the underlying records are finer-grained than the intended principal (e.g., packets vs. hosts), no explicit guarantees are given for the principal. The finer-grained records that share the same higher-level principal could be aggregated into one logical record. Using this aggregated data will then provide guarantees as the level of the principal. The intuition is that analysis fidelity will decrease as fewer records are contributing to output statistics. In future, we wish to study how much worse the accuracy gets when aggregating data and applying differentially private mechanisms.

We need to find a way to manage privacy loss due to repeat analysis of the same data. Each use leaks some private information and successive uses leak more information. As the privacy cost of analyses could be added, data owners could limit the total privacy cost per analyst. This could be achieved by setting a privacy budget with respect to the kind of analysis which needs to be carried.

And there are multiple ways to implement an analysis, with different privacy costs, i.e., noise. The trade-off between algorithmic complexity and privacy cost needs to be considered by the data owner.

We need to carry out numerous other packet-level, flow-level analyses and graph-level analyses to be able to generalize the applicability of DP mechanisms to network trace data.

We were successfully able to apply two level counting mechanism to count ports in the stream of network data. In future, we would like to extend this to be able to output *top-k* items from the network data set. This would help network operators answer queries like top 10 ports currently in use, or top 10 users currently using the maximum bandwidth. Also, the two level counting mechanism is bounded, i.e., it needs to know how long it is supposed to run. We would like to apply unbounded mechanism like logarithmic counting mechanism and see if any useful network trace analysis could be conducted.

Conclusion:

We were able to show the variation of noise introduced by local/global differentially private mechanisms with dataset size. We observed that noise introduced by local mechanism was higher than global mechanism. This ensured better privacy guarantees with local mechanisms. The noise in local mechanism is approximately, $\sqrt{n}$ *times* higher than global mechanism. Hence, the accuracy of global mechanism was better.

We have implemented DP mechanisms (Laplace, Gaussian, Two level counting) in python language. We applied these mechanisms to network dataset collected from MACCDC to study whether mediated trace analysis of network data could be conducted in a differentially private manner. From our experiments, we were able to show that the packet-level analysis, basically port distribution, could be conducted with high privacy guarantees (epsilon = 0.1) without affecting the accuracy of the analysis. This also shows that aggregate analysis on network traces could be fruitfully done with strong privacy guarantees without loss of accuracy. The choice of epsilon, i.e., accuracy-privacy trade-off depends on the nature of analysis and the amount of privacy one wishes to protect.

We were also able to apply two level counting mechanism to a stream of network data to count ports at any given time. Again, we concluded that aggregate analysis could be conducted in manner which guarantees strong privacy and high accuracy.

Acknowledgements:

The authors would like to thank Prof. Hsu for his lectures and discussions.

References:

1. Mark Bun. *A Teaser for Differential Privacy.* 2017.
2. Arvind Narayanan, Vitaly Shmatikov. *Robust De-anonymization of Large Sparse Dataset*. 2008.
3. Hubert Chan, Elaine Shi, Dawn Song. *Private and Continual Release of Statistics.* 2011.
4. Frank McSheery and Ratul Mahajan. Differentially-Private Network Trace Analysis. In *SIGCOMM*, 2010
5. J.Reed, A. Aviv, D.Wagner, A. Haeberlen and B.C. Pierce. Differential Privacy for Collaborative Security. In *EuroSec*, 2010
6. D. Koukis, S. Antonatos, D. Antoniades, E.P. Markatos and P. Trimintzios. A Generic Anonymization Framework for Network Traffic. In *ICC,* 2006
7. L. Sweeney. *k-anonymity: A model for protecting privacy*. Int'l Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems, 10(5), 2002.