Secure Distributed Data-Mining and Its Application to Large-Scale Network Measurements

Matthew Roughan
School of Mathematical Science
University of Adelaide
SA 5005, Australia
matthew.roughan@adelaide.edu.au

Yin Zhang
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
yzhang@cs.utexas.edu

ABSTRACT

The rapid growth of the Internet over the last decade has been startling. However, efforts to track its growth have often fallen afoul of bad data — for instance, how much traffic does the Internet now carry? The problem is not that the data is technically hard to obtain, or that it does not exist, but rather that the data is not shared. Obtaining an overall picture requires data from multiple sources, few of whom are open to sharing such data, either because it violates privacy legislation, or exposes business secrets. Likewise, detection of global Internet health problems is hampered by a lack of data sharing. The approaches used so far in the Internet, *e.g.* trusted third parties, or data anonymization, have been only partially successful, and are not widely adopted.

The paper presents a method for performing computations on shared data *without any participants revealing their secret data*. For example, one can compute the sum of traffic over a set of service providers without any service provider learning the traffic of another. The method is simple, scalable, and flexible enough to perform a wide range of valuable operations on Internet data.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*network monitoring, network management*; H.2.8 [**Database Management**]: Database Applications—*data mining*

General Terms

Management, Measurement, Security, Algorithms

Keywords

Secure Distributed Data-mining, Secure Distributed Summation, Network Measurement, Network Management

1. INTRODUCTION

There is a common problem in many areas of science: obtaining adequate data. A particular frustration applies in some areas where the data exists, but is held in such a way that the data may not be accessed. Common examples arise in health science, where data may be held by multiple parties: commercial organizations (such as drug companies, or hospitals), government bodies (such as the Food and Drug Administration) and non-government organizations (such as charities). Each organization is bound by regulatory restrictions (for instance privacy legislation, *e.g.* [1]), and corporate requirements (for instance on distributing proprietary information that may provide commercial advantage to competitors). In such a case, an independent researcher may not receive access to data at all, while even members of one of these organizations sees an incomplete view of the data. However, data from multiple sources may be needed to answer some important questions. A classical example occurs for an organization like the CDC (Center for Disease Control and Prevention), who are mandated with detecting potential health threats, and to do so they require data from a range of sources (insurance companies, hospitals and so on), each of whom may be reluctant to share data.

The same problem has arisen in analysis of the Internet. The data of interest concerns Internet properties of global interest: for instance, traffic, network performance, routing, and topology. These data are of obvious interest to Internet researchers, and there are also commercial applications that require collection of data from disparate sources. For example:

- The Internet's growth has been often documented, but rarely with any accuracy [23], and this is a real problem. Odlyzko [23] provides convincing arguments that the lack of accuracy in reporting of Internet growth contributed substantially to the Internet bubble, which eventually lead to a major downturn in the telecommunications economic sector. It would be of advantage to companies and their investors to have accurate information about true traffic volumes. However, this requires summation of traffic volumes from many individual Internet Service Providers (ISPs), each of whom is typically reluctant to reveal this data.

- Regulators often require access to data to make informed decisions when writing or interpreting legislation. For example, in adjudicating anti-trust cases (for example see `http://www.usdoj.gov/atr/cases/f7100/7183.htm`), one needs to assess whether a company has a monopoly over some business, and therefore data is needed (in our case of Internet traffic volumes).

- Threats to Internet health, such as viruses, worms, and Distributed Denial of Service (DDoS) attacks are a serious problem. The detection of these is directly analogous to the problem the CDC faces. The CDC must determine threats to human health, based on collection of data from various sources. Problems in Internet health would be most easily detected and analyzed if data from many ISPs could be combined. Once again, however, ISPs are reluctant, or unable.

One approach to solving the above problem is to have an independent third party that acts to combine the data of interest. Such a party has to have the trust of all concerned parties, and legislative power to side-step privacy controls. A trusted third party has not emerged in the context of Internet measurements[1]. We argue that it is not likely to appear any time soon. The highly competitive nature of current ISPs, and the conservative nature of many network managers means that they would be unlikely to trust a third party with such data, unless mandated by further legislation. Furthermore, such a third party creates the need for processes for collecting data that are unlikely to be flexible or rapid, in order that a clear trust relationship is maintained at all times[2]. While this may be sufficient for some applications, it is clearly not useful for rapid detection of Internet health problems.

The problem comes under the heading of distributed data-mining, *i.e.* data-mining where the data of interest is distributed across several databases, and one cannot combine it centrally to perform operations on the data. More specifically, the problem under consideration is a problem sometimes called "secure multiparty computation", or "privacy preserving data-mining". Both terms carry the implication that the exchanges of information between databases cannot reveal the contents of the databases. Obviously, this is a challenging problem, but in the context of health sciences, there are a number of solutions [9, 21]. That is, there are techniques, whereby a distributed computation is made that reveals the answer to a question of interest, without any of the parties involved becoming aware of the specific information held in each other's databases. The methods are presented in the context of "honest but curious" participants, *i.e.* participants who are willing to honestly contribute their information, but who will try to find out whatever they can about other participants. In addition, given business relationships between companies are highly varied, some may choose to collude to determine additional information, and the methods presented can be made resistant to such collusion.

Given such a technique, we may then ask "How could we apply it to obtain useful information about the Internet?" This paper, does not solve all the problems of privacy and distributed data in the context of the Internet. However, we have found a few fruitful applications: calculating the sum of traffic across a set of networks, and detecting distributed Internet health problems. We believe that these could be implemented immediately, given support of network operators. This paper presents a roadmap to implementation of these applications, describing the details needed for the particular application to Internet data, given its peculiar requirements. In particular, we contribute a method for computing sketches in a secure distributed fashion in order to reduce the size of data transfers.

The paper is organized as follows. Section 2 presents related work on Internet data sharing. It is followed by Section 3, which describes the algorithms we plan to utilize, in the context of the applications of Internet traffic and performance measurement. We make a number of new points in this section, which we have not seen in the secure distributed computation literature, in particular, we note the extensions to time-series analysis, and to sketch based methods. Section 4 provides a description of the mechanisms and motivations need to get such a system working. This section concentrates on practical problems of implementation, and diverges once again from the existing secure distributed computation literature in considering a system with many more participants than typical. Section 5 concludes the paper, and describes some future directions for work on this topic.

[1]Except in some particular cases, for instance, the Australian Bureau of Statistics (ABS) has fulfilled this role in regard to traffic measurements in Australia [2].

[2]The ABS collected data on a quarterly basis, but is now reducing this to yearly collection.

2. RELATED WORK

One approach to allow sharing of Internet traffic data is *anonymization* [25, 26, 30]. In anonymization, some details of the traffic are obscured in order to maintain privacy of the data. This allows the data to be analyzed for a pertinent feature, without the features requiring privacy being available to a researcher.

However, anonymization in most of its forms is limited when one wants to compare data from multiple distributed datasets. In particular, the keys into the data are often the thing requiring the anonymization (for instance, IP addresses). When the keys are anonymized, one loses the ability to join data from more than one database, unless the anonymization is identical between different databases, in which case, any party to the anonymization may have enough information to reverse the process, and obtain some of the original data. Privacy-preserving data-mining can avoid this problem.

Furthermore, few providers are willing to release traffic data under under existing anonymization schemes. The particular reasons for this reticence are sometimes hard to tie down, but they lie in the fact that the exact data that a provider wishes to obscure is sometimes not known at the time of data capture. The important data may only become relevant after the fact, at which point the data is already public, and one cannot take it back. This is a conservative, but *valid* strategy. Network managers' goals are largely network reliability, and efficiency, not research.

There is now a substantial literature on secure distributed computation and data mining (*e.g.* see [3, 6, 7, 9, 21, 29, 31, 32] and the references therein). It is not the intention of this paper to review this literature, and so we restrict consideration to a few papers directly related to the methods of use here (see the following section), though note this is an active area of research. Of importance is the fact that it was shown early on [31] that any polynomial time function could be computed in a secure distributed manner, and so a great deal of this literature considers how to do so efficiently. This paper contributes in particular to this area by presenting a technique for using sketches (an efficient representation of high dimensional data) in secure distributed computation.

There is also a substantial literature on network security. While one of our applications is in anomaly detection, the methods presented here should be seen as a toolbox for constructing (global) data sets, to which we can apply arbitrary anomaly detection algorithms (for examples see [4, 8, 19, 20, 27]). We therefore will not focus on the anomaly detection component, except where its requirements might influence our algorithm design.

3. APPLICATIONS AND ALGORITHMS

In this paper we concentrate on two applications: calculating the sum of traffic across a set of networks, and detecting distributed Internet health problems.

3.1 Secure summation of Internet traffic

As noted above, the total Internet traffic is unknown (though it is estimated in [23]). Highly optimistic traffic-growth estimates underlay the hype at the peak of the Internet boom. Given claims that traffic would double every three months (a common claim) it was not hard to justify almost any business on the grounds of getting in early. However, the growth claims were largely unsubstantiated, and turned out to be highly optimistic, with more realistic growth being of the order of doubling every year.

How can one value Internet assets, or evaluate company performance, in an environment where an error on the order of a factor of 10 in growth every year is possible? As it turned out, one could not, and this led (in part) to the burst of the Internet bubble [23], and the collapse of many Internet companies.

Hence we wish to provide accurate, up-to-date estimates of total Internet traffic. In principle the calculation of such totals is trivially simple: assume ISP i carries total traffic v_i, then we wish to compute

$$V = \sum_{i=1}^{N} v_i.$$

There are of course practical considerations: for instance, what time interval are the data collected over? Time intervals for measurements must clearly be agreed in order to allow commensurate summations to proceed. In addition there is the problem of what is the total of interest?

1. total traffic carried by all providers?
2. total traffic accessing the Internet?

Sum (1) includes some double-counting, as traffic crossing the Internet would typically cross several providers. Sum (2) specifies that traffic should only be counted at the "edge" of the Internet, so that traffic is not double counted. However a major transit network might carry little or no edge traffic. The metric (2) might make such a backbone appear to be a very small network, despite large transit traffic, and there are other problems in defining the edge of the Internet in any case. Clearly both views provide information, and we suggest that one may wish to create several different summations:

- *total access traffic:* total traffic coming into and out of a network via access links such as DSL, cable modem, or dial-up links;
- *total backbone traffic:* total traffic crossing a network;

Together the two sets of data provide additional information about the efficiency of inter-domain routing. In many cases, a clear division between access links, and downstream providers may not be easy to obtain, and so it might be more practical to divide traffic entering a network by BGP-customers, non-BGP-customers, and peers (a classification that is relatively easy to make in most large networks [17]). We can then employ the method described here to find sums across all BGP speaking networks

Given agreement on exactly which quantity one wishes to sum, there is a well-understood process for performing secure distributed summation [9]. Assume the value V to be calculated is known to lie in the interval $[0, n]$, start from a particular ISP (we will denote this ISP 1), and list the other ISPs in some order with labels $2, 3, \ldots, N$. ISP 1 generates a random number R uniformly on the interval $[0, n]$, *i.e.* $R \sim U(0, n)$. Then ISP 1 adds R to its traffic total $v_1 \mod n$, and sends this to the next ISP, which adds its traffic, and repeats, until we get to the end of the sequence. The last ISP returns the total to ISP 1, who can then subtract the original random number R, and compute the total (and then provide this number to the other ISPs). Formally we specify the algorithm by

```
ISP 1:  randomly generate $R \sim U(0,n)$
ISP 1:  compute $s_1 = v_1 + R \mod n$
ISP 1:  pass $s_1$ to ISP 2
for i=2 to N
   ISP i:  compute $s_i = s_{i-1} + v_i \mod n$
   ISP i:  pass $s_i$ to ISP $i+1 \mod N$
endfor
ISP 1:  compute $v_N = s_N - R \mod n$
```

Each ISP $i = 2, \ldots, N$ has only the information v_i and s_{i-1}, which can be written in full as

$$s_i = R + \sum_{i=1}^{i} v_i \mod n.$$

Since this value is uniformly distributed across the interval $[0, n]$, ISP i learns nothing about the other traffics v_j, $j \neq i$. At the last step, ISP 1 has s_N, and when it subtracts R away it gets the total traffic.

Note that where we wish to allow computation of quantities that may be negative, the condition that $V \in [0, n]$ can be easily replaced by $V \in [-n/2, n/2]$. The algorithm above requires only that we adjust the range of the initial sum (*i.e.* in the second step ISP 1 takes $s_1 = n/2 + v_1 + R \mod n$), and that in the last step ISP 1 reverses this addition (*i.e.* $v_N = s_N - n/2 - R \mod n$).

Any ISP, given V can compute $V - v_i = \sum_{j \neq i} v_i$, and so this approach only works for $N > 2$, and in reality, where one could make meaningful guesses about some values, it is only really secure for reasonable values of N[3].

This incredibly simple process can, unfortunately, be corrupted if ISPs collude. If ISP $l-1$ and ISP $l+1$ share information, they can compute v_l by taking $s_l - s_{l-1}$ (s_l is received by ISP $l+1$, and s_{l-1} is sent by ISP $l-1$). A simple fix to this problem is provided in [9]: each ISP randomly partitions its total traffic into M shares v_{im} such that

$$v_i = \sum_{m=1}^{M} v_{im}.$$

Note we explicitly allow negative shares v_{im}. Secure summation is then performed M times to calculate the sum for each share individually. However, the summation order of the ISPs is permuted for each share so that no ISP has the same neighbor twice. Standard cryptographic methods can be used to enforce the summation order for each share. To compute v_i, the neighbors of ISP i from every iteration would have to collude. With M shares, $2M$ colluding parties are therefore required to violate security.

There is an appealing alternative based on ideas originally expressed in [6] and extended in [3]. It has the advantage that no cryptography is needed.

1. As before each ISP i randomly partitions its total traffic into M shares v_{im}, and each partition is shared with a random participant. ISP i chooses $M-1$ ISPs $\{j_1, j_2, \ldots, j_{M-1}\}$ randomly from the list of N. The random partition is chosen by taking v_{ij_k} for $k = 1, \ldots, M-1$ as random variables (some negative and some positive), and choosing $v_{i0} = v_i - \sum_{k=1}^{M-1} v_{ij_k}$.
2. ISP i distributes share v_{ij_k} to ISP j_k ($k = 1, \ldots, M-1$).
3. After share distribution, each ISP privately adds up all the shares it has received, *i.e.* it computes $z_j = v_{j0} + \sum_{i \neq j}^{M} v_{ij}$, where we set $v_{ij} = 0$ when ISP i did not send a share to j.
4. The N ISPs sum the individual totals, *i.e.* $V = \sum_i z_i$. A secure summation is not needed here [6].

The above approach is guaranteed secure [3] against collusion for less than $M-1$ colluding parties. It has an advantage in that it can also be formally shown to provide bounds on the probability

[3]Any multiparty computation (secure or otherwise) gives out the result, and hence cannot be secure against revealing whatever can be deduced from the input of a party and the ultimate output.

of successful collusion for more than $M - 1$ colluding parties [3] that decay exponentially with M. In addition, the parties involved will not know whether collusion has been successful, reducing the value of any data gained this way.

An appealing alternative to dividing the traffic into simple shares (as above) is to use Shamir's Secret Sharing Scheme [28]. Shamir's Scheme breaks a secret into N shares such that from any M the secret can be reconstructed. The method works by choosing a random degree $M - 1$ polynomial

$$y(x) = a_0 + a_1 x + \cdots + a_{M-1} x^{M-1},$$

such that the value of interest is a_0. The shares consist of pairs (x_i, y_i) lying on the polynomial (for N different values of $x_i \neq 0$). It is simple to show that for $M-1$ such pairs we cannot recover the polynomial (and hence a_0). However, given M such data points, we can uniquely identify the polynomial.

Benaloh [6] shows that the approach of summation of the shares can be applied to the shares generated using Shamir's Scheme, and so we can break the secret (the ISPs traffic), into such shares, distribute these, and then perform the algorithm as above. Hence the communications complexity can be made linear in the number of nodes on the network using either approach. Using Shamir's Scheme combined with Beneloh has the advantage that it can provide robustness to missing data — the secret shares are such that at least M are needed to reconstruct the total (rather than exactly M in the previous approach). If one node becomes unavailable during the computation then we might loose the information it contains, however, if the secret was partitioned into N components, and M are still available, then we can still reconstruct the original secret. One can then tune between the required redundancy and the communications overheads of the protocol. Furthermore, the order of computation is not important in this method, and it requires only two rounds of communication, instead of N, reducing its latency. Thus this approach has some advantages, despite being somewhat more complex to implement.

There is still one limitation of the algorithm in that it assumes that the ISPs are semi-honest to the extent of inputing correct data into the algorithm. No method can avoid this limitation without detailed scrutiny of the ISP's networks, and the semi-honest assumption is common to many such secure computation problems.

In terms of practicality, nearly all providers collect Simple Network Management Protocol (SNMP) data sufficient to compute their total traffic (precisely because such figures are useful in the context of forward planning). SNMP data limits the time granularity of the traffic totals collected, but for forward planning, and economic analysis a granularity of one day should be amply sufficient, and is easily obtained using SNMP.

The above approach could be used in a highly flexible manner, by collecting traffic data at different time scales (minutes, hours, days, or months), or looking at different aspects of the traffic (packets, bytes, or number of flows). We could also consider creating network metrics such as

- number of routers,
- number of links, or number of links of each type (*e.g.* OC48, Gig-Ethernet)
- kilometres of fiber,
- bandwidth-miles of network capacity,
- traffic-miles for carried traffic,

computing each using distributed summation. The method could also be focussed on particular groups of traffic, for instance, traffic on specific TCP ports, or to a particular destination prefix, or performance measurements. That brings us to our second application for privacy preserving data-mining: finding Internet problems.

3.2 Detecting distributed network problems

The previous application was motivated by economic concerns (as well as research interest). Such concerns typically act over long time scales (weeks, to months, to years). In some cases such data has been collected and analyzed by a trusted third party, and one might wish this to be the case everywhere (though it seems to us unlikely).

On the other hand, security threats act on the short term (seconds to hours). It seems unlikely that a centralized organization can build the type of trust required to have almost instantaneous access to traffic or performance data that they might need to analyze an arbitrary security threat. On the other hand, the flexibility, and ease of the above approach makes possible some quite powerful analyses without any third party being needed.

Data of interest in detecting Internet health problems include:

- traffic data at various granularities (*e.g.* per prefix, or port);
- performance data (such as collected from active probes).

Note that the data would now be a set of time series, so that various anomaly detection algorithms can be applied (*e.g.* see [4, 8, 19, 20, 27]). Initially, let us consider detecting anomalies in a few commonly used statistics, such as average volumes of packets, bytes, and flows, or average network performance. These would allow detection of larger events. In addition, a mechanism could be provided to allow queries to be created on the fly for more specific traffic data to be collected. For instance, suppose a DDoS attack is suspected to be targeting a particular customer, one might create a query to find a time series of the traffic that has been directed towards that customer's prefix from all participating ISPs.

SNMP measurements are not detailed enough for a prefix based query. Flow-level aggregation might be, and although not all providers collect such data it is not as important that all providers participate for such an application. A sample of providers may well provide enough data to answer a particular question. At the very least this is better than the view a single ISP has currently.

These applications are more demanding in their data requirements, and also in the algorithms one might apply. For instance, while the method for computing distributed sums is directly applicable when we wish to compute traffic data for anomaly detection, performance data is somewhat different. In the first analysis, one is often interested in the average (the arithmetic mean) of the performance measurements, which we could compute easily using a distributed sum as above (dividing by N after the last step). However, such a simple average may give unwanted weight to smaller ISPs whose performance measurements have little impact on the wider Internet. In such a case, it may make more sense to report a *traffic weighted average, i.e.* we wish to know

$$V = \frac{\sum_{i=1}^{N} t_i d_i}{\sum_{i=1}^{N} t_i},$$

where, for instance, t_i is the total traffic for ISP i, and d_i is the average delay across their network. The above weighted average can be easily computed by taking two steps, in the first we computer the total traffic $\sum_{i=1}^{N} t_i$ by taking $v_i = t_i$, and in the second we compute $\sum_{i=1}^{N} t_i d_i$ by taking $v_i = t_i d_i$.

The arithmetic mean suffers from a second problem in that it loses a large part of the detail of the distribution of delays. For instance, in many cases we might be more interested in the higher percentiles of the distribution of delays. We can compute an approximation to a distribution by simply partitioning the distribution into K bins, and then computing

$$V_k = \frac{1}{N}\sum_{i=1}^{N} p_{ik},$$

where p_{ik} is the probability that the delay distribution of ISP i falls into the kth bin. The V_k then form average probabilities for the distribution of delays. Such a distribution may prove more useful than a simple mean.

3.3 Time-series algorithms

The approaches above are based on the idea that one would perform summation over the ISPs to first obtain a time series, on which one might perform an arbitrary anomaly detection algorithm (for examples see [5, 8, 27]). However, it is worth noting that one might prefer to perform the algorithm on individual time series, and then aggregate the results, the assumption being that averaging prior to anomaly detection may "wash out" the anomalies.

However, note that many anomaly detection algorithms are based on a linear transformation of the time series, for example the ARIMA, Fourier and Wavelet methods [33]. Given the anomaly detection is based on a linear transform $\mathcal{L}(\cdot)$ of the data, then given each ISP observes time series $\mathbf{x}^{(i)}$ we note that by linearity

$$\mathcal{L}\ \ \sum_i \mathbf{x}^{(i)}\Bigg) = \sum_i \mathcal{L}\ \ \mathbf{x}^{(i)}\quad ,$$

and so it makes no difference whether we perform the transform on the individual data sets or the summation.

Most anomaly detection transforms are followed by a threshold operation: anomalies are signaled to an operator when $|\mathcal{L}(\mathbf{x})| > T$. This is a decidedly non-linear operation, and so this is where the washing out of anomalies might occur, though note that in many cases the aggregation actually should have the effect of washing out the *noise*, and making the anomalies stand out more clearly (by improving the signal-to-noise ratio).

If the order of thresholding is of concern, then we may apply the following approach to the problem. Each ISP creates a bitmap time-series, where each bit indicates the presence or absence of an anomaly at each time interval. Formally, we define

$$y^{(i)}(t) = \begin{cases} 1, & \text{if } \ \mathcal{L}\ \ \mathbf{x}^{(i)}; t \quad > T \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{L}\ \ \mathbf{x}^{(i)}; t$ is a linear transformation of the time series $\mathbf{x}^{(i)}$ at time point t (we leave choice of thresholds to future work). We then perform a distributed summation of the bitmaps, *i.e.* we calculate $\mathbf{y} = \sum_i \mathbf{y}^{(i)}$. Finally, we threshold the distributed sum to find time points where it is greater than 0. Thus we perform a distributed `OR` on the individual ISPs' anomalies. We can generalize this approach to other logical operators by obvious extensions.

Note that the operations above are not strictly privacy preserving as one gains access to some intermediate results. For example, in the case of the `OR` operation above, one also gains access to the number of providers for which the threshold is exceeded. While the secrecy of these intermediate results are not necessarily important, it is noteworthy that considerable effort has gone into developing approaches that allow strict privacy. For example, Atallah *et al.* [3] show that time-series algorithms (such as detecting a linear trend) can be performed, without revealing intermediate values. In the case of detecting a trend, the algorithm will reveal the slope to all parties, without revealing the absolute values.

3.4 Generalization to sketches

Now, it is perhaps too much to hope to have all possible views of the data (by port, or prefix, etc.) being global distributed. The volume of data would become tremendous. There are several methods for building more compressed views of this type of data (in order to find anomalies), *e.g.* [14, 19], but it is unclear how one could apply these methods in a distributed setting (such as above), though this may be a productive avenue for future research.

However, one more sophisticated anomaly detection mechanism that can be easily generalized is that of sketch-based change detection [18]. Instead of performing anomaly detection on the original data, we first build compact summaries of the traffic data using a *sketch*, a small-space data structure that allows one to approximately reconstruct the value associated with any given key. We can then implement a variety of time-series forecast models (ARIMA, Holt-Winters, etc.) on top of such summaries and detect significant anomalies by looking for flows with large forecast errors. Being able to compute significant differences in the list of top flows quickly can point towards *potential* anomalies. Depending on the length of the time period for which we compute forecasts and the duration of significant changes, we can accurately identify the presence of an anomaly.

Besides change detection, sketches are the basis for answering a number of fundamental queries on massive data streams, such as range queries, heavy hitters, and quantiles [22]. More recently, sketches have also been used for tracking distributed queries [10] and mining multigraph streams [12]. Sketches have many variants, such as counting Bloom filters [16], multi-stage filters [15], and the Count-Min sketch [11]. Below we introduce one such variant, the Count-Min sketch, and show how we can generalize it in a privacy preserving manner.

Data. We wish to study a data stream consisting of updates (a, u), where $a \in \{1, \ldots, n\}$ is a key, and $u \in \mathbb{R}$ a value. The signal that results from these updates is a high-dimensional vector $\mathbf{v} \in \mathbb{R}^n$, which records the total value for each key, *i.e.* for each update (a, u), we perform $v_a \mathrel{+}= u$.

Sketch. A Count-Min sketch consists of a $d \times w$ array of counts: $c[1, 1] \ldots c[d, w]$. Each entry of the array is initialized to zero. In addition, d hash functions $h_1 \cdots h_d : \{1 \cdots n\} \rightarrow \{1 \cdots w\}$ are chosen uniformly at random from a pairwise independent family.

Update. When an update (a, u) arrives, each row i of the sketch is updated by adding quantity u to the counter corresponding to index $h_i(a)$. That is, for all $1 \leq i \leq d$, update $c[i, h_i(a)] \mathrel{+}= u$.

Query. When a point query $\mathcal{Q}(a)$ arrives, an approximation of v_a is given by $\hat{v}_a = \min_i c[i, h_i(a)]$.

From the above specification, it is clear that the sketch data structure is linear in that we can add or subtract two sketches by adding or subtracting them on a per-entry basis. As a result, in order to generalize to provide privacy preserving summaries of the global data, we just require that each party uses the same hash functions, and then we can directly apply the secure summation protocol on each individual entry $c[i, j]$ of the local sketches.

There are other approaches to privacy preserving distributed data mining that might also be useful in this context (for instance, one may use set intersection methods to compute medians of distributions), but we believe that the simple summation approach provides significant value with maximal simplicity, and should be the first approach tested.

4. HOW TO MAKE IT WORK

There are three parts to this section, the first, on motivation, or "why should I participate", the second on mechanisms for building a real instantiation of the above, and the third considers some practical issues.

4.1 Motivation

To make such an approach work in practice, ISPs must be motivated to participate. Legislative approaches to enforce participation should be seen as a last resort, primarily because we wish companies to provide reliable data into the system, which we believe would be easier if the companies are positively motivated, rather than coerced.

Motivations for researchers in this context are obvious: the data would have a high intrinsic appeal for many such researchers. However, the motivation to join such a system cannot be research based. The motivation should be primarily company self-interest. It does not appear hard to provide such motivations, *e.g.*

- better planning for all ISPs;
- less internal rhetoric about company success: companies can objectively measure success, which should be appealing to senior management;
- better security Internet wide;

and the method would do this all without leaking any competitively sensitive information to other ISPs.

Making the cost to join such a system negligible would be a big plus for participation (especially for small ISPs). In order to make the method as cheap as possible for an ISP to implement, we intend it to be all open-source code, using common standards and tools, for instance RRDtool [24] is very commonly used by ISPs to record their SNMP traffic data, and building our system around the use of such a tool would be a good starting point. The computational requirements should also be such that it could be run on just about any server someone has lying around available for this service (note the algorithms described above are generally light on computation requirements). Relying (in the first instance) on SNMP data is also a good start, as such data is nearly universally available, and collected by most ISPs.

4.2 Mechanisms

There are several parts to an implementation:

- initialization: setting up communication between all participants;
- request a query: this includes the standard types of queries used to form, for example, the total traffic on the Internet.
- distributed calculation: performance of the privacy preserving distributed calculation for a particular query;

We discuss details of each in turn below.

4.2.1 Initialization

Setting up communications between an arbitrary N participants is non-trivial. Luckily, it has been much studied. In particular, Peer-2-Peer (P2P) systems appear to have many appealing features. They allow arbitrary joining and leaving of participants, and build mechanism for sharing data efficiently between these participants.

We suggest that such a P2P system should be used (reducing our need to design specialized protocols) for all communication between participants.

Additional authentication may be required, in order that arbitrary (non-ISP) participants do not join, and attempt to damage the system. Authentication is again non-trivial, but frequently studied, and we consider it to be a solved problem (within our context).

The P2P system is responsible for allowing properly authenticated participants to join or leave (and keeping track of those who might unintentionally leave the system); for allowing file transfers between the participants, and for maintaining a visible list of participants. We shall remain agnostic about which P2P system should be used, conditional on these features being available.

4.2.2 Queries

A second component of the system would be a mechanism to request a query. A general query might be of the form

```
sum all traffic over time  [t,t+dt]
```

Obviously, query specifications are important here, but again this is an area much studied in Internet measurement, for instance see [13] and the references therein. Note that we do not require real-time responses to queries — the approach we propose works asynchronously, and so, there is no need for all participants to be "on schedule". We simply wish the calculation to proceed as quickly as possible.

We envisage that initially only standard queries (such as total traffic) would run, and so a special node might be given responsibility for requesting this query. Note that one could easily create redundancy by delegating multiple alternate nodes to request the standard queries, should the originally nominated node disappear from the system.

4.2.3 Distributed calculation

Given each participate has access to a list of participants, and the desired value to sum, it is simply a matter of implementing the algorithms from Section 3. We describe here the most complicated algorithm (*i.e.*, the one based on Shamir's Secrete Sharing Scheme), as implementation of the other schemes is somewhat obvious. First each ISP creates its own random polynomial

$$y_i(x) = v_i + a_1^{(i)} x + a_2^{(i)} x^2 + \cdots + a_{M-1}^{(i)} x^{M-1},$$

Note that all arithmetic is performed modulo p, where p is a prime number greater than N and V, and the random coefficients $a_k^{(i)}$ are chosen uniformly on $[0, p)$. The ISPs must agree on a set of distinct points $x_j \neq 0$ for $j = 1, \ldots N$, at which each ISP i generates the values $y_i^{(j)} = y_i(x_j)$

Initially, let us consider the case where ISP i then distributes $y_i^{(j)}$ to ISP j. ISP j sums the terms it receives to get

$$y^{(j)} = \sum_{i=1}^{N} y_i^{(j)},$$

which are then distributed to all ISPs. From any M of these we may determine the polynomial $y(x) = \sum_i y_i(x)$, from which we obtain $V = y(0)$, though note that only ISP i has access to $y_i(x)$.

This approach is simple, highly robust to missing data (ISPs that drop out), and immune to collusion by up to $M - 1$ parties. It takes only two rounds of computation. However, its communications overhead is $O(N^2)$. To reduce this the first step of distributing the $y_i^{(j)}$ should be performed for a limited set of $K \geq M$ of the points x_j. Obviously, this concentrates computation and communications overheads on a limited subset of the ISPs, however, this set can be changed for each calculation (as can the polynomials in case coefficients are leaked via other insecure pathways).

Alternative approaches may be possible where the partitioning of computation in a single computation is more evenly shared, but this seems to be a topic for future research.

The second step of sharing the $y^{(j)}$ can be reduced to $O(N)$ by designating one (or a few) ISPs to perform the final computation (these ISPs must then share the results with the other ISPs). Using these approaches the communications overheads can be reduced to $O(KN)$, with resistance to collusion by up to $M-1$ parties, and redundancy to (at least) $K-M$ ISPs dropping out after the first stage of the computation (dropping out before this simply removes their traffic from the results).

4.3 Practical issues

Being wildly optimistic we naturally assume all ASes (some 20,000) might join this system. Most examples of distributed datamining in the literature are not this distributed (they typically use at most a dozen databases). Is it possible to make distributed calculations on such a scale? In fact the Internet already involves a distributed calculation on this scale, namely, BGP (the Border Gateway Protocol), which already uses a distributed computation between all of these ASes to compute global routes. It does so continuously. So the scale of the problem is feasible. However, in reality, it is likely that many fewer will join. In practical terms this is not a problem — the vast majority of traffic traverses a few large networks. Getting the majority of these involved would provide a substantial, and nearly authoritative data set. For instance, according to the ABS, in March 2005 there were 689 ISPs in Australia, however only 25 were classified as large, and only 10 as very large. The top ten ISPs carried 63% of the traffic, and the top 35 carried 87% of the traffic, thus, a relatively small number of ISPs could provide large amount of value (obviously the numbers will be larger in the United States and Europe, but it is likely that similar ratios hold).

A second practical issue is that participants may come and go from the system (hopefully more of the former), either through choice, or because of failures of their node, or network. Hence the system must live with changing numbers of participants. As noted the approach proposed can be made robust to ISPs leaving during a computation, but what about longer term growth estimation where the number of participants is changing as fast as the total traffic? This is a more thorny problem — how can we compare sums from different time intervals if the sum are based on data from a different set of participants. Given this possibility the number of ISPs included in each step of the computation should be recorded, so that at least one can look back across data, and test whether the number of participants was a significant factor in the results. Additionally, the use of RRD files (described below) can alleviate some of the issues regarding changing numbers of participants.

An important aspect of the problem is how to efficiently store and transfer data. RRDtool [24], formally the Multi-Resolution Traffic Grapher (MRTG), is one of the most common tools used by ISPs to collect and manage their SNMP data. It is popular for several reasons: it is essentially free, it is easy to use and configure, and it uses a clever method to maintain its database at constant size. This is what interests us here: the method used is a Round-Robin Database (RRD). A RRD stores data on fine times scales for a limited amount of time removing older values for newer ones, but values that are removed are not lost. When data is removed, it is in fact aggregated over a longer time interval, and stored in this form (again in a RRD). So, for instance, we might keep 5 minute granularity traffic data for 1 day, 1 hour averages, and maxima of this data for 1 week, and 1 day averages and maxima for one year. In this form, the database does not grow over time (as more data is collected) but we can see a view of the data over multiple time scales. This is ideal for network operations, because the fine data is typically only needed for recent data, and the longer term data is used to find trends, which one can see with the courser granularity data just as well.

This is directly applicable here. Instead of computing single sums for a time interval $[t, t+dt]$ for each query we propose that the queries use data as a RRD. That is, data to be transfered, and summed is held in a RRD, both in initial form, intermediate form, and final form. The result is (1) a finite, fixed file size to be transfered at each step, and (2) we can compute an entire time series for all current participants, for each query. Even if participants change over time, the current calculation will be with respect to all currently participating ISPs, and will go back as far in time as the set of ISPs have been collecting data.

The RRD files being transferred[4] can be chosen with parameters suitable to the calculation of interest. For example, for traffic sums (which are likely to be used in long term economic analysis) we need longer term data, but at a courser resolution, so perhaps 1 day averages over several years would be appropriate (making the calculation on $O(1000)$ terms each time). For another example, anomaly detection, we might wish to have several weeks of data at a 5 minute time scale (the smallest interval typically collected by ISPs), resulting in $O(4000)$ data elements. So we are not talking about very large volumes of data. Given reasonable compression, we might achieve something better than 5 bytes per data point, so we are talking of the order of 5-20kB transfers. Given $N = 1000$, and $K = 20$, this would result in traffic of 400MB for one query. Clearly we do not wish to run too many such queries (though note that this traffic distributed over the ISPs).

As a result, we additionally suggest that a throttling mechanism is needed, to prevent the system being used to launch a DoS attack via a malicious user. Such a throttling mechanism would avoid one ISP trying to trawl through the data of the others to obtain some commercial advantage. We suggest a simple leaky bucket mechanism to reduce the rate at which a single node can originate queries.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented a road map towards implementing a method of collecting meaningful Internet wide statistics of great use to providers and researchers alike. The method is needed, and the components required to make it work all exist. We plan in the future to implement the approach to demonstrate its main features: security, flexibility, and scalability.

There are many avenues for future research on this topic. Apart from obvious possibilities such as collecting statistics for each geographic region, or performance metrics between regions, one might like to consider some more sophisticated algorithms and how they might be used in this context. For instance, measuring traffic matrices is non-trivial, and so much effort has gone into their inference from other data (for example see [34] and the references therein). It would be interesting if such inverse problems can be solved using distributed summation methods.

6. ACKNOWLEDGEMENTS

We would like to thank one of the anonymous reviewers for many valuable comments.

[4]We need to use RRDtool's export format to avoid big/little-endian data representation problems. This format is XML, so it would also be beneficial to apply some compression to the database files before transfer.

7. REFERENCES

[1] Data-mining moratorium act of 2003. Introduced in Senate of the United States in January 2003. `http://thomas.loc.gov/cgi-bin/query/z?c108:S.188:`.

[2] Internet Activity, Australia. `http://www.abs.gov.au/Ausstats/abs@.nsf/0/6445f12663006b83ca256a150079564d?OpenDocument`, 2005.

[3] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES'04)*, Washington, DC, USA, October 2004.

[4] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Nov 2002.

[5] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *ACM SIGCOMM Internet Measurement Workshop*, Marseilles, France, November 2002.

[6] J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proc. Advances in Cryptology (CRYPTO '86)*, pages 251–260, 1987.

[7] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT, LNCS*, pages 236–252, 2005.

[8] J. D. Brutag. Aberrant behavior detection and control in time series for network monitoring. In *Proceedings of the 14th Systems Administration Conference (LISA 2000)*, New Orleans, LA, USA, December 2000. USENIX.

[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2), December 2002.

[10] G. Cormode and M. arofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Proceedings of the International Conference on Very Large Data Bases*, 2005.

[11] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Proceedings of Latin American Theoretical Informatics (LATIN)*, pages 29–38, 2004.

[12] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of ACM Principles of Database Systems*, 2005.

[13] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: a stream database for network applications. In *SIGMOD*, June 2003.

[14] C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[15] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. ACM SIGCOMM '2002*, Pittsburgh, PA, August 2002.

[16] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proc. ACM SIGCOMM '98*, Vancouver, British Columbia, CANADA, 1998.

[17] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, March/April 2000.

[18] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *ACM SIGCOMM Internet Measurement Conference*, Miami, Florida, USA, October 2003.

[19] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.

[20] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, 2004.

[21] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3), 2002.

[22] S. Muthukrishnan. Data streams: Algorithms and applications, 2003. Manuscript based on invited talk from *14th SODA*. Available from `http://www.cs.rutgers.edu/~muthu/stream-1-1.ps`.

[23] A. M. Odlyzko. Internet traffic growth: Sources and implications. In B. B. Dingel, W. Weiershausen, A. K. Dutta, and K.-I. Sato, editors, *Optical Transmission Systems and Equipment for WDM Networking II*, volume 5247, pages 1–15. Proc. SPIE, 2003.

[24] T. Oetiker. RRDtool. `http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/`.

[25] R. Pang and V. Paxson. A high-level programming environment for packet trace. In *Proc. ACM SIGCOMM*, August 2003.

[26] M. Peuhkuri. A method to compress and anonymize packet traces. In *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA, November 2001.

[27] M. Roughan, T. Griffin, M. Mao, A. Greenberg, and B. Freeman. IP forwarding anomalies and improving their detection using multiple data sources. In *ACM SIGCOMM Workshop on Network Troubleshooting*, pages 307–312, Portland, OR, September 2004.

[28] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[29] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[30] J. Xu, J. Fan, M. Ammar, and S. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *ICNP*, Paris, November 2002.

[31] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

[32] A. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1986.

[33] Y. Zhang, Z. Ge, M. Roughan, and A. Greenberg. Network anomography. In *Proceedings of the Internet Measurement Conference (IMC '05)*, Berkeley, CA, USA, October 2005.

[34] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *ACM SIGMETRICS*, pages 206–217, San Diego, California, June 2003.