A Local Distributed Peer-to-Peer Algorithm Using Multi-Party Optimization Based Privacy Preservation for Data Mining Primitive Computation

Kamalika Das, Hillol Kargupta*
University of Maryland Baltimore County
Baltimore, MD-21250
{kdas1,hillol}@cs.umbc.edu

Kanishka Bhaduri
MCT Inc. at NASA Ames Research Center
Moffett Field, CA-94040
Kanishka.Bhaduri-1@nasa.gov

Abstract

This paper proposes a scalable, local privacy-preserving algorithm for distributed peer-to-peer (P2P) data aggregation useful for many advanced data mining/analysis tasks such as average/sum computation, decision tree induction, feature selection, and more. Unlike most multi-party privacy-preserving data mining algorithms, this approach works in an asynchronous manner through local interactions and therefore, is highly scalable. It particularly deals with the distributed computation of the sum of a set of numbers stored at different peers in a P2P network in the context of a P2P web mining application. The proposed optimization-based privacy-preserving technique for computing the sum allows different peers to specify different privacy requirements without having to adhere to a global set of parameters for the chosen privacy model. Since distributed sum computation is a frequently used primitive, the proposed approach is likely to have significant impact on many data mining tasks such as multi-party privacy-preserving clustering, frequent itemset mining, and statistical aggregate computation.

1 Introduction

Privacy-preserving data mining (PPDM) is a requirement in increasing number of multi-party applications where the data is distributed among many nodes in a network. Web mining applications in Peer-to-Peer (P2P) networks [7][2] and cross-domain network threat management systems for analyzing cyber-terrorism trends[1] are some examples where data privacy is an important issue. In such large distributed environments, PPDM algorithms are unlikely to work unless they can offer scalability and heterogeneous privacy-models. Local algorithms are the ones in which the communication overhead is bounded by a constant or slowly growing polynomial [2] and are usually scalable for large asynchronous networks. Some of the existing methods for privacy preservation in multi-party environments include the SMC protocols [12] which fail to scale to large networks. More recently, Kargupta *et al.* [5] present a game theoretic solution for dealing with the collusion problems in large distributed environments. Since privacy is usually a social issue, different parties might have different privacy requirements. Therefore, a heterogeneous privacy model gives parties the autonomy to optimize their privacy cost requirements. This paper takes a step toward developing such a model for privacy preserving data aggregation in a P2P network. The main contributions of this work are two-fold: (1) multi-objective optimization-based heterogeneous privacy model, and (2) a local asynchronous algorithm for distributed data aggregation in a large network for client-side web mining [2, 7].

Data analysis in such heterogenous environments calls for a genre of algorithms which perform the analysis in a distributed fashion. One possibility is distributed data mining (DDM) which deals with the problem of data analysis in environments with distributed data, computing nodes, and users. This area has seen considerable amount of research during the last decade. For an introduction to the area, interested readers are referred to [6]. P2P data mining has recently emerged as an area of DDM research, specifically focusing on algorithms which are asynchronous, communication-efficient, and scalable. Datta *et al.* [3] presents an overview of this topic.

This paper explores the problem of computing the sum of a collection of numbers distributed in a P2P network in a privacy-preserving manner. We develop a distributed averaging technique that uses secure sum computation as a building block for scalable data aggregation useful for many advanced data mining tasks. The algorithm is provably correct. Unlike most secure multi-party computation protocols, our algorithm does not assume semi-honest adversary [1]. However, we prove that this algorithm, though not se-

*Also affiliated to AGNIK LLC, MD, USA

[1] `http://www.agnik.com/PursuitFlyer.pdf`

978-1-4244-5067-1/09/$26.00 ©2009 IEEE

cure, is privacy preserving in a well-defined way. This paper also proposes a new multi-objective optimization-based privacy model for a heterogenous distributed environment where each node defines its own privacy requirement. Each user can specify its own set of parameters for the chosen privacy model. Under this proposed model, each peer gets to choose its own privacy and the algorithm guarantees that the privacy requirement of each peer is satisfied at the end of the protocol. We discuss ranking a set of web advertisements as a client-side web mining application of the proposed algorithm.

The rest of the paper is organized as follows. In the next section (Section 2) we present an illustrative application followed by necessary background material in Section 3. Section 4 first describes the optimization-based privacy model and then presents the privacy preserving distributed sum computation algorithm. We analyze the algorithm in Section 5 and demonstrate its empirical performance in Section 6. Finally, we conclude the paper in Section 7.

2 Illustrative Application

Consider a designer shoe manufacturing company that wants to study the market in South Asia before finalizing their advertising campaign for that geographical region. They plan to use the web for aiding their market research. They buy some web-advertisements designed for collecting user preference-statistics at a popular web-portal and inks in a deal with a web-analytics company to provide business intelligence by combining the click-stream data from the web-advertisements along with user background information collected through the IP address-based mapping of the geographical location and other related techniques. Among other things, the business intelligence provider company counts the geographic distribution of clicks on different parts of the advertisements from different IP addresses. This is how it works today. However, growing concerns for online privacy protection is creating technology for protecting the identity of the user. For example, use of anonymizing networks such as TOR [2] may prevent web-servers to collect any meaningful information regarding any query involving the geographical location of the users. In this case, the IP address associated with a click may come from randomly selected node in the TOR network. As a result the web-analytics may give completely misleading information. How do we solve this problem—protect privacy of the user but still be able to rip the benefit of the web mining technology?

This paper offers a solution to this type of problems. It offers a P2P framework where the user identity is protected and the web-mining task is accomplished using distributed privacy-preserving data mining algorithms. It provides a decentralized client-side solution for distributed privacy preserving data aggregation.

The web mining problem of advertisement ranking discussed here is only a representative application scenario and the algorithm can be extended to solve a variety of data aggregation tasks. Since the Internet can be viewed as a connected network of users, we pose this as a data analysis problem in a large P2P network. Every user (peer) in the network has a predefined vector of fixed size where the j^{th} entry of the vector corresponds to the number of clicks for thej^{th} advertisement. In this environment, ranking the advertisements can be framed as a global sum computation problem. As the network of users converge to the global sum for every entry in the data vector, they can locally sort the vectors to get the correct global popularity based ranks of the advertisements. Since web browsing information can be privacy sensitive, it is important to do this sum computation in a privacy-preserving manner. This becomes particularly challenging in heterogeneous environments such as the Internet, since different users might have different requirements of privacy. Therefore the problem that this paper addresses is to compute the global sum of a data vector in a distributed, asynchronous, and privacy-preserving manner.

3 Background

In this paper we propose a privacy-preserving distributed sum computation technique. Since scalability is an important issue for any large distributed computing environment asynchronous solutions are preferred. To the best of the authors' knowledge, there does not exist any privacy-preserving asynchronous algorithm for sum computation. The secure sum protocol [1] solves a similar problem but is highly synchronous. There exist several solutions to asynchronous distributed averaging, but are not privacy-preserving such as [10]. The algorithm proposed here uses distributed averaging for privacy preserving sum computation in a locally synchronous fashion. Note that the average computation problem can be converted to a sum computation problem by scaling up the data of each peer by the total number of peers.

3.1 Notations

Let $P_1, P_2, \ldots, P_d$ be the set of peers connected to each other by an underlying communication infrastructure. The network can be viewed as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{P_1, P_2, \ldots, P_d\}$ denotes the set of vertices and $\mathcal{E}$ denotes the set of edges. Let $\Gamma_{i,\alpha}$ denote the set of neighbors of P_i at a distance of α from P_i and $|\Gamma_{i,\alpha}|$ denote the size of this set *i.e.* the number of neighbors in the α-neighborhood.

[2] http://www.torproject.org/

Further, let $\Phi_{d\times d}$ denote the connectivity matrix or topology matrix of $\mathcal{G}$ representing the network where

$$\phi_{ij} = \begin{cases} 1 \text{ if } i,j \in \mathcal{E} \ \& \ i \neq j \\ -|\Gamma_{i,1}| \text{ if } i,j \in \mathcal{E} \ \& \ i = j \\ 0 \text{ otherwise} \end{cases}$$

Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p$ denote an ordered set of advertisements common to all peers and let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p$ denote the real-valued data vectors of size p for each peer. For peer P_i, x_{ij} denotes the number of clicks of advertisement $\mathcal{A}_j$, *i.e.* x_{ij} is the jth element of the data vector $\mathbf{X}_i$. Let X be the random variable for the distribution of x_{ij}. Let S_j denote the global sum of the j-th data element x_{ij}. Finally, let n_i^* denote the size of the ring that peer P_i forms for the secure sum computation.

3.2 Distributed Averaging

In distributed averaging, the objective is to compute the global average $\Delta_j = \frac{1}{d}\sum_{i=1}^{d} x_{ij}$ where every peer P_i has a real number x_{ij} and d is the size of the network. In the naive solution, all the peers can exchange information to compute the correct sum. However, this solution is highly synchronous and does not scale well for large P2P networks. Distributed approaches include the LTI approach proposed by Scherber and Papadopoulos [10]. The basic idea of all these approaches is to maintain the current estimate of Δ_j ($z_i^{(t)}$) and exchange messages with its immediate neighbors to update $z_i^{(t)}$. As iteration $t \to \infty$, $z_i^{(t)} \to \Delta_j$, *i.e.* the system asymptotically converges to the correct average.

The distributed averaging problem, as proposed in [10], is not privacy-preserving. Moreover it works only for symmetric topologies. Our multi-objective optimization framework requires asymmetric network topology. To handle this, we present a modified protocol in Section 4.5.

3.3 Secure Sum Protocol

Secure sum computation [1] computes $S_j = \sum_{i=1}^{n} x_{ij}$ without disclosing the local value x_{ij} of any user. It has been widely used in privacy-preserving distributed data mining as an important primitive. The secure sum protocol requires the existence of a ring topology (or an overlay ring network) connecting the users *i.e.* for peers 2 through $d-1$, $\Gamma_{i,1} = \{P_{i-1}, P_{i+1}\}$, $\Gamma_{1,1} = \{P_d, P_2\}$ and $\Gamma_{d,1} = \{P_{d-1}, P_1\}$. Let each $x_{ij} \in \{0, 1, 2, \ldots m\}$. It is known that the sum $S_j = \sum_{i=1}^{d} x_{ij}$ (to be computed) takes an integer value in the range $[0, N-1]$. Assuming peers do not collude, P_1 generates a random number R uniformly distributed in the range $[0, N-1]$, which is independent of its local value x_{1j} and transmits $(R + x_{1j}) \bmod N$ to P_2. In general, for $i = 2, \ldots, d$, peer P_i executes:

$$y_{ij} = (y_{i-1j} + x_{ij}) \bmod N = (R + \textstyle\sum_{q=1}^{i} x_{qj}) \bmod N,$$

where y_{ij} is the perturbed version of local value x_{ij} to be sent to the next peer $i+1$. P_d performs the same step and sends the result y_{dj} to P_1. Then peer P_1, which knows R, can subtract R from y_{dj} to obtain the actual sum. This sum is finally broadcast to all other users.

The secure sum protocol is highly synchronous and is therefore unlikely to scale for large networks. Combining a newer variation of the distributed averaging (Section 4.5) with the secure sum protocol in a small neighborhood of a peer, we propose an privacy-preserving sum computation algorithm which (i) asymptotically converges to the correct result and (2) being only locally synchronous, scales well with the network size.

4 Privacy-Preserving Distributed Sum Computation

In this section we present the model of privacy in a heterogenous computing environment and show how the global sum can be computed while satisfying the different privacy requirements of different users.

4.1 Privacy Protection as Optimization

Privacy is a social concept. In a distributed data mining environment different peers have different notions and requirements of privacy. Due to sharing of private information in the process of computation, privacy of the users' data is threatened. Every user in the network has a prior belief (assumption) about the *threat* to its data privacy. The threat that a peer's data is exposed to can be considered as a measure of the lack of privacy of its data. Again, the amount of resources available to a peer varies across the network and hence, the cost (of computation and communication) a peer can bear to ensure its data privacy also varies. In this paper we assume that each peer has the same privacy model, but a different value of the parameters which satisfy its privacy. Therefore, every user in the network solves an optimization problem locally based on its cost and threat threshold, *i.e.* how much threat the user is willing to bear and how much resources it is willing to spend for ensuring that. Without loss of generality we consider a linear model for the objective function:

$$f_i^{obj} = w_{ti} \times threat - w_{ci} \times cost$$

where the $cost$ is the total cost of communication of peer P_i within a neighborhood of size n_i^* and $threat$ is the privacy breach that P_i assumes to be exposed to due to its participation in the data mining task. w_{ti} and w_{ci} are the weights (importance) associated with $threat$ and $cost$ respectively.

These parameter values are local to each peer and are independent of the values chosen by any other peer in the network. In order to measure threat, we need a way of measuring privacy in such heterogenous environments. One such model is discussed in the next section.

4.2 Bayes Optimal Privacy Model

The Bayes optimal model of privacy [8] uses prior and posterior distribution to quantify privacy breach.

Let X be a random variable which denotes the j-th data value at each node. The value at node P_i is denoted by x_{ij}. The prior probability distribution is $prior = P(X = x_{ij})$ Once the data mining process is executed, the participants can have some extra information. Given this, we define the posterior probability distribution as $posterior = P(X = x_{ij}|\mathcal{B})$, where $\mathcal{B}$ models the extra information available to the adversary at the end of computation. There are several ways for quantifying the Bayes optimal privacy breach.

Definition 1. *[$\rho_1 - to - \rho_2$-privacy breach[4]] Let f_{prior} and $f_{posterior}$ denote the prior and posterior probability distribution of X. The $\rho_1 - to - \rho_2$ privacy breach happens when $f_{prior} \leq \rho_1$ and $f_{posterior} \geq \rho_2$, where $0 < \rho_1 < \rho_2 < 1$.*

As noted in [8], any privacy definition which quantifies the privacy breach in terms of principle 1 or 2, is known as the Bayes optimal privacy model. However, this $\rho_1 - to - \rho_2$ privacy model is applicable only when there is a single node in the network. Below we extend this privacy framework for a distributed multi-party scenario.

Definition 2. *[Multi-party $\rho_1 - to - \rho_2$ privacy breach] For the i-th peer P_i, privacy breach occurs if $f^i_{prior} \leq \rho_{1i}$ and $f^i_{posterior} \geq \rho_{2i}$. Multi-party $\rho_1 - to - \rho_2$ privacy breach occurs when the constraints are violated for any peer in the network i.e. $\forall i,\ f^i_{prior} \leq \rho_{1i}$ and $f^i_{posterior} \geq \rho_{2i}$, where $0 < \rho_{1i} < \rho_{2i} < 1$.*

In the definition, the posterior probabilities of each peer can either be dependent or independent of each other. If the peers share the extra information ($\mathcal{B}$), their posterior distributions are also related. Since in our framework each peer solves the optimization problem locally, the dependence or the independence of the posterior probabilities does not change the privacy requirements.

Since in a distributed environment, different peers have different privacy requirements, it is difficult to achieve the distributed $\rho_1 - to - \rho_2$ privacy using a single secure sum since the $\rho_1 - to - \rho_2$ privacy is achieved in terms of the number of participants of the ring (as shown in Section 4.3). So our proposed algorithm uses multiple local sum computation protocols with different ring sizes, one for each node in the network. This approach addresses two issues: (1) it proposes a solution to privacy preservation in heterogenous environments and (2) it avoids creating a single large synchronous ring for sum computation which makes the algorithm scalable for large-scale distributed systems. The sum computation does not claim to be a secure protocol by getting rid of the semi-honest assumption, but still is privacy preserving. Before we describe the algorithm for doing the distributed averaging based local secure sum, we introduce a measure of the *threat* component in the objective function applicable to the secure sum protocol.

4.3 Threat Measure under Collusion

The secure sum computation algorithm assumes semi-honest parties who do not collude. However, it has been shown in the literature [5] that such an assumption is sub-optimal and that rational parties would always try to collude in the absence of a penalizing mechanism. In this paper we adapt the expression of threat developed in [5] to estimate the threat component in our objective function. Each peer forms a ring of size n_i^* (referred to as n in this section for sake of simplicity) in our algorithm. Let us assume that there are k $(k \geq 2)$ nodes acting together secretly to achieve a fraudulent purpose. Let P_i be an honest node who is worried about its privacy. Let P_{i-1} be the immediate predecessor of P_i and P_{i+1} be the immediate successor of P_i. We will only consider the case when $n - 1 > k \geq 2$ and the colluding nodes contain neither P_{i-1} nor P_{i+1}, or only one of them, then P_i is disguised by $n - k - 1$ other nodes' values. This can be represented as

$$\underbrace{\sum_{q=1}^{n-k-1} x_{qj}}_{\text{denoted by Y}} + \underbrace{x_{ij}}_{\text{denoted by X}} = \underbrace{S_j - \sum_{q=i+1}^{i+k} x_{qj}}_{\text{denoted by W}},$$

where W is a constant and known to all the colluding nodes. The posterior probability of x_{ij} is:

$$\begin{aligned} f_{posterior}(x_{ij}) &= \frac{1}{(m+1)^{(n-k-1)}} \sum_{q=0}^{r} (-1)^q \binom{n-k-1}{q} \\ &\times \binom{n-k-1+(r-q)(m+1)+t-1}{(r-q)(m+1)+t} \end{aligned}$$

where $z_j = W - x_{ij}$ and $z \in \{0, 1, \ldots, m(n-k-1)\}$. $r = \lfloor \frac{z_j}{m+1} \rfloor$, and $t = z_j - \lfloor \frac{z_j}{m+1} \rfloor (m+1)$. Note that here we assume $x_{ij} \leq W$, otherwise $f_{posterior}(x_{ij}) = 0$. This posterior probability can be used to measure the threat faced by a peer while participating in the secure sum computation protocol, if there is collusion:

$$threat = Posterior - Prior = f_{posterior}(x_{ij}) - \frac{1}{m+1} \quad (1)$$

Note that using uniform distribution as the prior belief is a reasonable assumption because it models the basic knowledge of the adversaries. This assumption was also adopted by [11] where a Bayes intruder model was proposed to assess the security of additive noise and multiplicative bias.

It can be observed from this threat measure that (1) as k increases, the posterior probability increases, and (2) as n increases, the posterior probability decreases. This implies that as the size of the network involved in the secure sum computation increases, the threat decreases for a fixed size of the colluding group. Therefore, the privacy of the data of the users in the secure sum depends on the initiator's choice of the size of the group (n). The choice of n can vary between 1 and the total number of nodes d. As the value of n increases, the threat to a user's data due to collusion decreases, assuming a constant percentage of colluding nodes in the network. However, increasing n increases the overall communication cost and synchronization requirements of the algorithm. Since the communication cost increases linearly with the size of the secure sum "ring", the objective function that is optimized by every peer in the network can be written as:

$$\max_n [w_{ti} \times threat(n) - w_{ci} \times cost(n)]$$

subject to the following constraints: $cost < c_i$ and $threat < t_i$ where $threat(n)$ is given by Equation 1 and $cost(n) = w_c \times g \times n$. g is the proportionality constant and c_i and t_i are constants for every peer and denote the cost threshold and privacy threshold that each peer is willing to withstand. This is a multi-objective optimization problem where the threat increases while the cost decreases with increasing n. Below is a solution to this optimization problem. Let

$$h(n) = \sum_{q=0}^{r} (-1)^q \binom{n-k-1}{q} \binom{n+t-k-2+(r-q)(m+1)}{(r-q)(m+1)+t}$$

It can be observed that $h(n) \geq 1$. Now, using the constraint $threat \leq t_i$ we get

$$\frac{w_t}{(m+1)^{(n-k-1)}} \times h(n) \leq t_i$$

$$\Rightarrow \quad n \geq 1 + k + \frac{\log(w_{ti}) - \log(t_i)}{\log(m+1)} \tag{2}$$

Similarly, using the constraint on cost, we get

$$w_{ci} \times g \times n \leq c_i \Rightarrow n \leq \frac{c_i}{w_{ci} \times g} \tag{3}$$

Using Equations 2 and 3, we get the optimal value of n (denoted as n_i^* in accordance with the rest of the paper):

$$1 + k + \frac{\log(w_{ti}) - \log(t_i)}{\log(m+1)} \leq n_i^* \leq \frac{c_i}{w_{ci} \times g} \tag{4}$$

Now, depending on its personal preference, each peer can choose the number of nodes (n_i^*) for computing the sum in a privacy preserving fashion, even in the presence of colluding parties. Recall from Section 4.2, for the posterior distribution, the extra information ($\mathcal{B}$) in the secure sum protocol is only the sum of the colluding nodes (W). Note that this is independent of the total sum since, secure multiparty computation protocol guarantees that no extra information is revealed by the sum other than its own inputs [1].

4.4 Threat Measure for Multiple Rings

The above expression for threat only gives us a measure of the same when there is only one ring. In the presence of multiple rings, a colluder can infer more knowledge about an honest node's data. In this section, we derive an expression for the threat in the presence of multiple rings. For simplicity, we consider the situation of only two intersecting rings. The case for multiple rings can be analogously derived.

Let there be m nodes in ring 1 and n nodes in ring 2. The values at the nodes for the two rings be arranged as follows:

Ring 1:

$$\overbrace{x_{1,j} \rightarrow \cdots \rightarrow x_{c-1,j} \rightarrow x_{c,j}}^{\text{common}} \rightarrow \overbrace{x_{a,j} \rightarrow x_{a+1,j} \rightarrow \cdots \rightarrow x_{g,j}}^{\text{not common}}$$

Ring 2:

$$\overbrace{x_{1,j} \rightarrow \cdots \rightarrow x_{c-1,j} \rightarrow x_{c,j}}^{\text{common}} \rightarrow \overbrace{x_{b,j} \rightarrow x_{b+1,j} \rightarrow \cdots \rightarrow x_{h,j}}^{\text{not common}}$$

For ring 1, let the colluding nodes be $x_{c-1,j}, x_{c,j}, x_{a,j}, x_{a+1,j}$. Similarly, for the other ring, $x_{c-1,j}, x_{c,j}, x_{b,j}, x_{b+1,j}$ are the colluding nodes. Denoting the sum of the rings by C_1 and C_2, and subtracting we can write,

$$x_{a,j} + \cdots + x_{g,j} - (x_{b,j} + \cdots + x_{h,j}) = C_1 - C_2$$

Moreover, since the sum of the colluding nodes is known to all colluders, we can write:

$$x_{a+2,j} + \cdots + x_{g,j} - (x_{b+2,j} + \cdots + x_{h,j}) = C$$

Without loss of generality, let the node whose value is at threat be $x_{g,j}$. Thus, we can write,

$$\underbrace{x_{g,j}}_{Z} = C + \left(\underbrace{x_{b+2,j} + \cdots + x_{h,j}}_{\text{denoted by} X} \right) - \left(\underbrace{x_{a+2,j} + \cdots + x_{g-1,j}}_{\text{denoted by} Y} \right)$$

Now since C is a constant, it can be shown that,

$$\begin{aligned} P(Z = z) &= P(X - Y = z) \\ &= \sum_{y=0}^{(g-a-2)m} P(X = y + z) P(Y = y) \end{aligned}$$

Using the expressions for $P(X = z + y)$ and $P(Y = y)$ from the previous section, we can easily write the expression for $P(Z = z)$.

4.5 Distributed Averaging for Asymmetric Topologies

In this section we present the iterative distributed algorithm for computing the global sum of a set of data vectors.

Our solution is inspired by the distributed averaging algorithms proposed in [10].

The distributed averaging technique that we are exploring asymptotically converges to the global average. It can easily be used to compute the sum if each peer multiplies its data by the total number of peers in the network. Therefore, for the given scenario, each peer P_i contains a real number $d \times x_{ij}$ where d is the size of the entire network and the objective is to compute $\Delta_j = \frac{1}{d}\sum_{i=1}^{d} d \times x_{ij}$ *i.e.* the sum of the numbers. There exist several techniques in the literature to estimate the network size. Examples include the capture-recapture method proposed by Mane *et al.* [9]. Moreover at any time, the number of nodes in the network can be estimated efficiently using heartbeat mechanisms or retransmissions. From now on we assume that each entry x_{ij} of the data has been multiplied by the total number of peers so that distributed averaging gives the global sum and not the global average.

Let x_{ij} denote the j-th data of peer P_i. $\mathbf{z}_j^{(t)} = \left[z_{1j}^{(t)} z_{2j}^{(t)} \dots z_{dj}^{(t)}\right]^T$ denotes the estimate of the global sum $\Delta_j = \frac{1}{d}\sum_{i=1}^{d} x_{ij}$ by d peers at the t-th iteration. The initialization is $\mathbf{z}_j^{(0)} = [x_{1j} x_{2j} \dots x_{dj}]^T$. The proposed algorithm works as follows: at any iteration, each peer P_i gets the estimate from all of its neighbors (the $z_{ij}^{(t-1)}$'s for $i \in \Gamma_{i,1}$) and then generates the estimate for round t (*i.e.* $z_{ij}^{(t)}$) based on those received estimates and its local data. This algorithm is asynchronous and local since each node gets update from its neighbors only. The update rule used is first order: $\mathbf{z}_j^{(t)} = \mathbf{W}\mathbf{z}_j^{(t-1)}$. Any choice of $\mathbf{W}$ guarantees asymptotic convergence if $\mathbf{W}$ satisfies the following properties: (i) $\mathbf{W}.\mathbf{1} = \mathbf{W}^T.\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ denotes a $d \times 1$ vector of all ones and (ii) the eigenvalues of $\mathbf{W}$, λ_i when arranged in descending order are such that $\lambda_1 = 1$ and $|\lambda_i| < 1$. Setting $\mathbf{W} = \mathbf{I} + \rho\Phi$ satisfies these conditions; where ρ is a small number which determines the stability of the solution and the convergence rate, and $\mathbf{I}$ denotes the identity matrix.

From Section 4.3, it is clear that depending on the solution to the optimization problem, each peer can have a different value of n_i^*, *i.e.* number of nodes it wants to communicate with. This means that if peer P_i chooses peer P_j to be part of its sum computation, it is not necessary that P_j would choose P_i to be part of its sum computation ring. This implies that even if P_j is a neighbor of P_i, P_i need not be a neighbor of P_i (in terms of adjacency matrix). This implies that the resulting topology matrix is asymmetric. Note that if we use $\mathbf{W} = \mathbf{I} + \rho\Phi$, the resulting $\mathbf{W}$ does not satisfy the requirements stated above. Therefore, asymmetric topology matrices cannot be directly used for generating the update matrix $\mathbf{W}$. Now, an asymmetric topology matrix can be converted to a symmetric one as follows: $\Phi^{''} = \Phi + \Phi^T$, where Φ^T is the transpose of Φ. Since Φ is a square matrix, $\Phi^{''}$, by definition, is a symmetric matrix. In order for $\mathbf{W}$ to satisfy the properties stated above, it can be generated using the transformation $\mathbf{W} = \mathbf{U} + \rho\Phi^{''}$ where each entry of $\mathbf{U}_{d\times d}$ is such that

$$u_{ii} = \begin{cases} 1 - \rho\sum_{j=1}^{d} \phi_{ij}^{''} \\ 0 \text{ otherwise} \end{cases}$$

In Section 5, we analyze the convergence and correctness of this proposed distributed averaging algorithm. Based on the above transformation, every peer updates its estimate of Δ_j using an update rule that depends on the ring it forms. The following lemma (Lemma 4.1) states the update rule for our proposed distributed averaging problem (proof omitted due to shortage of space).

Lemma 4.1. *The update rule for any peer can be written as* $z_{ij}^{(t)} = \{1 - 2\rho|\Gamma_{i,1}| - \rho(n_i^* - |\Gamma_{i,1}|)\} z_{ij}^{(t-1)} + 2\rho\sum_{q\in\Gamma_{i,1}} z_{qj}^{(t-1)} + \rho\sum_{q=1}^{n_i^*-|\Gamma_{i,1}|} z_{qj}^{(t-1)}$.

4.6 Overall Algorithm

In this section we present the overall algorithm. We have two different algorithms: namely, the local ring formation algorithm (**L-Ring**) which is executed only once, offline. The second algorithm is the iterative local privacy preserving sum computation algorithm (**L-PPSC**).

4.6.1 Local Ring Formation Algorithm (L-Ring)

For distributed averaging, peer P_i updates its current state based on the information it gets from its n_i^* neighbors. In order to preserve privacy, P_i does not get the raw data from its neighbors; rather a ring is formed among n_i^* neighbors and sum computation is performed in that ring. We call this ring the *local* ring since each ring is only formed in a peer's neighborhood. This has the advantage that (1) the algorithm is only synchronous in a peer's local neighborhood and (2) the communication is bounded due to local peer interactions.

L-Ring takes as input the predefined values of cost and threat threshold, *i.e.* c_i and t_i. When the algorithm starts, each peer solves a local optimization problem based on local constraints c_i and t_i to choose a value of n_i^*, the size of the ring for sum computation. It then launches n_i^* random walks in order to select n_i^* nodes uniformly from the network to participate in P_i's ring. The random walk we have used is the Metropolis-Hastings random walk which gives uniform samples even for skewed networks. We do not present the details here, interested readers are referred to [2]. Whenever one random walk ends at P_j, it first checks if $n_i^* < n_j^*$. If this is true, it poses a potential privacy breach for P_j. Hence P_j may choose not to participate in P_i's call by sending a **NAC** message along with its n_j^*. Otherwise

P_j sends an **ACK** message to P_i. If P_i has received any **NAC** message, it computes $max(n_j^*)$ and checks if it violates its cost constraint. If the constraint is violated, P_i chooses a different peer P_q by launching a different random walk. Otherwise, it then sends out all of the $max(n_j^*)$ invitations again which satisfies the privacy constraints of all the participants. The pseudocode is presented in Alg. 1.

Algorithm 1 L-Ring

Input of peer P_i:
Threat t_i and cost c_i that peer P_i is willing to tolerate
Initialization:
Find the optimal value of n_i^* using t_i and c_i.
If P_i initializes a ring:
Contact the neighbors as dictated by n_i^* by launching n_i^* parallel random walks
When a random walk ends in node P_j:
Fetch the value of n_i^* as sent by P_i
IF ($n_i^* < n_j^*$) Send (**NAC**,n_j^*) to P_i **ELSE** Send **ACK** to P_i
ENDIF
On receiving NAC, n_j^* from P_j:
IF replies received from everyone
IF n_j^* violates cost constraint
Contact different neighbor P_q
ELSE $max = argmax_j\{n_j^*\}$; Set $n_i^* = max$
Send invitation $I(n_i^*)$ to P_j with n_i^* value
ENDIF
ENDIF

Once the rings are formed offline, the local sum computations start.

4.6.2 Local Privacy Preserving Sum Computation Algorithm (L-PPSC)

In the local privacy preserving distributed sum computation algorithm (**L-PPSC**), initially all peers in the network have a data vector of size p which represents the number of clicks for each of the p advertisements under consideration. The j-th entry of this vector corresponds to the number of clicks of advertisement $\mathcal{A}_j$. Below we discuss the algorithm with respect to only one sum computation (a scalar quantity). In practice, the secure sum will be computed over a vector of size p, the number of advertisements. Assuming that each peer has agreed on a ring in its local neighborhood, each initiator peer starts a round of sum computation based on the secure sum computation. The message sent by the initiator node for any sum computation contains: (1) the ID of the initiator, (2) the data which needs to be added for the local sum, (3) the size of the local ring that it has constructed for the sum, and (4) which peer needs to multiply the data by 2 (according to Lemma 4.1).

This algorithm differs from traditional secure sum computation protocol in the update rule and the enforcement of the ring topology. In the traditional version, the initiator sends its data masked by a random number while all others in the ring add their numbers as is and pass the sum on. Here, however, the initiator specifies in its message the parameters of the update rule: the amount of scaling that some of the peers might need to do to their data before adding them to the received sum and passing them on. This is essential to guarantee convergence of the algorithm to the correct result, following Lemma 4.1.

These steps are executed by every peer in the system. The algorithm is locally synchronous in that, during every round of sum computation, the initiator has to wait for all peers in its rings to complete their previous round. This is essential since this algorithm is based on the working of first order LTI systems where the update in the t-th round uses data from the $(t-1)$-st round. Algorithm 2 lists the steps in a pseudo-code format.

Algorithm 2 Local Privacy Pres. Sum Comp. (L-PPSC)

Input of peer P_i:
Convergence rate ρ, local data x_i, *round*, set of n_i^*-local neighbors arranged in a ring or $\{ring_{i,n^*}\}$, random number R, and the max range of the sum N
Initialization:
Initialize $\{ring_{i,n^*}\}$, ρ, x_i; Set $round \leftarrow 1$
Set $j \leftarrow$ first entry of $\{ring_{i,n^*}\}$
$\{ring_{i,n^*}\} \leftarrow \{ring_{i,n^*}\} \setminus j$
Send $(R + x_i, \{ring_{i,n^*}\}, round)$ to j
On receiving a message ($data, \{ring\}, rnd$) from P_j:
IF $\{ring\} = \emptyset$
Update $z_i^{(round)}$ using ($data - R$) and Lemma 4.1;
$round \leftarrow round + 1$;
Set $j \leftarrow$ first entry of $\{ring_{i,n^*}\}$
$\{ring_{i,n^*}\} \leftarrow \{ring_{i,n^*}\} \setminus j$
Send $\left(z_i^{(round)}, \{ring_{i,n^*}\}\ round\right)$ to j
Check if any node is waiting on this peer
Send data to all such nodes
ELSE IF $round < rnd$ Wait
ELSE
Set $y = (data + z_i^{rnd}) \text{mod} N$; Set $j \leftarrow$ first entry of $\{ring\}$
$\{ring\} \leftarrow \{ring\} \setminus j$; Send $(y, ring, rnd)$ to P_j
END

Using **L-PPSC** algorithm the peers can compute the sum of the number of clicks for each advertisement in a privacy preserving (not secure) fashion. Once that is done, ranking them by popularity becomes a sorting problem which each peer can solve independently.

5 Algorithm Analysis

In this section we analyze the properties the **L-Ring** and **L-PPSC** algorithms. Due to shortage of space we do not

present the proofs.

5.1 L-Ring Running Time

The running time of **L-Ring** algorithm is $O(max(n_i^*, n_j^*))$, where n_i^* is the optimal value for node P_i and n_j^* is the value required by node P_j where P_i and P_j belong to the same ring for the sum computation. This can be easily proved by considering two cases: (1) $n_i^* < n_j^*$ and (2) $n_i^* > n_j^*$. In either case we need to count the maximum number of times a peer needs to contact other peers to satisfy its own privacy requirement.

5.2 L-PPSC Privacy

Lemma 5.1. *For any P_i, the ρ_{1i}-to-ρ_{2i} privacy is satisfied in the **L-PPSC** protocol.*

Lemma 5.1 proves that the privacy is satisfied for every node in the network. Hence, using Definition 2, this protocol is privacy-preserving for the entire network.

In the **L-PPSC** algorithm, it is assumed that each ring has fewer than $(n_i^* - 2)$ bad nodes. If this condition is violated, then we know that privacy breach will surely occur. Next we derive an expression for the probability of this happening and show that it is very low.

Lemma 5.2. *Let θ be the probability of a node being good. Then the probability that in a ring of size n_i^*, there are at most $(n_i^* - 2)$ bad nodes is given by $1 - (1-\theta)^{n_i^*-1}$.*

The above expression shows that the probability of selecting less than $n_i^* - 2$ bad nodes increases with increase in the (1) probability of a good node θ, and (2) ring size n_i^*. Figure 1 (left) shows how the probability varies as a function of θ and n_i^*. As shown, the probability increases with increasing θ. This is intuitive, since with increasing θ, there is a higher chance that each contacted node is good. Also for a fixed θ, as n_i^*, the ring size increases and the probability of contacting less than $n_i^* - 2$ bad nodes goes to 1 faster.

Now consider another scenario in which there is the possibility of a privacy breach. Consider two intersecting rings which contains only one honest node. Now the probability of this occurring is given by $\theta(1-\theta)^{n_i^*+n_j^*-1}$, where n_i^* and n_j^* are the sizes of the two rings. Figure 1 (right) demonstrates the variation of this expression with θ, n_i^* and n_j^*. As seen in the figure, the probability is very low and decreases with increasing size of the ring. Also, for a fixed ring size, as θ increases, the probability decreases.

5.3 Correctness and Convergence

L-PPSC protocol is based on the distributed averaging protocol proposed by Scherber and Papadopoulos [10]. The

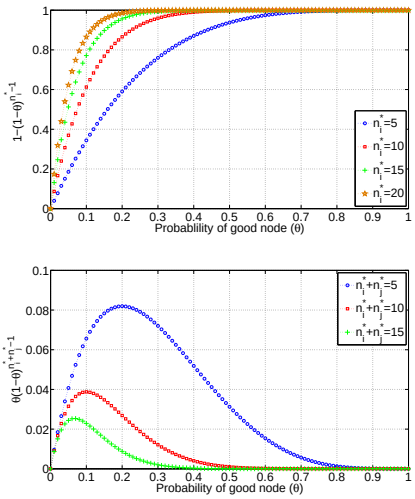

Figure 1. The left figure demonstrates the variation of $\theta(1-\theta)^{n_i^*+n_j^*-1}$ vs. θ, n_i^* and n_j^*. The right figure shows the probability that less than $n_i^* - 2$ nodes are bad in a ring of size n_i^*.

correctness proof of **L-PPSC** can be derived based on two observations analogous to [10]: (1) $\mathbf{W}.\mathbf{1} = \mathbf{W}^T.\mathbf{1} = \mathbf{1}$ and (2) the eigenvalues of $\mathbf{W}$, λ_i when arranged in descending order are such that $\lambda_1 = 1$ and $|\lambda_i| < 1$. We do not prove these here, but leave it for an extended version.

Following similar arguments in [10], we can show that the error (between the true average and the estimate at each peer) tends to zero exponentially fast as the number of iterations tend to infinity.

5.4 Locality

There are several definitions of locality proposed in the literature. The locality concept proposed by Das *et al.* [2] is characterized by two quantities — (1) α – which is the number of neighbors a peer contacts in order to find answer to a query and (2) γ – which is the total size of the response which a peer receives as the answer to all the queries executed throughout the lifetime of the algorithm.

In case of **L-PPSC**, the choice of α is guided by the optimal solution of the objective function defined earlier. In the worst case, a peer may choose α to be equal to the size of the entire network. Therefore, $\alpha = O(d)$ in the worst case. It can be shown that $\gamma \leq \frac{log(\epsilon)-log(d)}{log(\lambda_{max}^2)} \left[\log(z_{max}^{(t)}) + n_i^*\right]$, where $z_{max}^{(t)}$ is the maximum of data values at any peer in a ring at round t and ϵ is the error between the true sum (Δ) and the node estimates.

6 Experimental Results

To validate the performance of the proposed **L-PPSC** algorithm, we have conducted experiments on a simulated network of peers. The topology is generated using BRITE[3]. We have used the Barabasi Albert (BA) model in BRITE since it is often considered a reasonable model for the Internet. In all our experiments, we have used the following default values of the system and algorithm parameters: size of the network (d) = 1000, the maximum range of the sum for the secure computation (N) = $x_i \times d$ and $\rho = \max_i \frac{1}{|\Phi_{ii}|}$.

6.1 Experiments on Synthetic Dataset

In this section we first discuss about the dataset and then describe the convergence and scalability results.

As already noted, each peer agrees on a predefined set of advertisements. We assume that there are 5 advertisements A, B, C, D and E with arbitrary counts. The goal is to find the sum of all the clicks on advertisements over all the peers. A data set was generated consisting of tuples from different random distributions. Each advertisement is generated from a fixed uniform distribution (with a fixed range). Thus, there are as many different distributions as the number of advertisements. This centralized data set was then split among a fixed number of neighbors such that each peer has a fraction of the count of all the advertisements (0 if none exists). Note that this requires a separate privacy-preserving sum algorithm to be invoked for each advertisement/category. For the rest of this section we will present our results with respect to one sum computation only.

As shown in Figure 2 (top), the algorithm converges to the correct sum with respect to a centralized algorithm, where a centralized algorithm is one which has access to all the data of all the peers. In this figure we have plotted the estimate of all the peers at each time instance *i.e.* the $\mathbf{z}_j^{(t)}$ values for each t. To start with, each peer is assigned a data value which corresponds to the number of clicks of a particular advertisement. Hence, initially the estimate of each peer is close to its local data. As time progresses, the peers slowly converge to the correct sum. Figure 2 (bottom) demonstrates the number of messages per peer.

In Figure 3 (top), we show the correctness result of the **L-PPSC** algorithm (in triangle) when the number of peers vary from 100 to 2000. Also shown in the figure are the results computed by a centralized algorithm on the same data (using the circles). The graph shows that our algorithm converges to the correct result for varying sizes of the network. The cost of the algorithm with increasing network size is demonstrated in Figure 3 (bottom). It can be noted that the number of messages per peer is almost a constant. Hence, our algorithm is highly scalable.

[3] `http://www.cs.bu.edu/brite/`

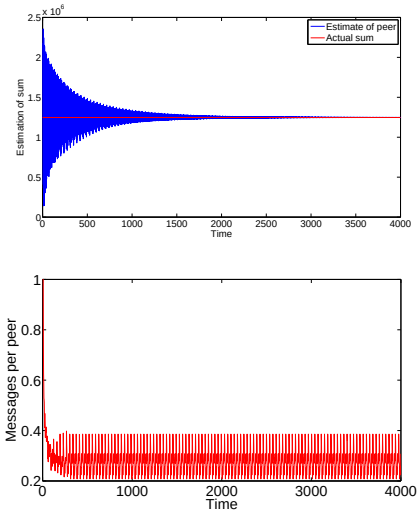

Figure 2. Convergence to global sum and communication cost per peer.

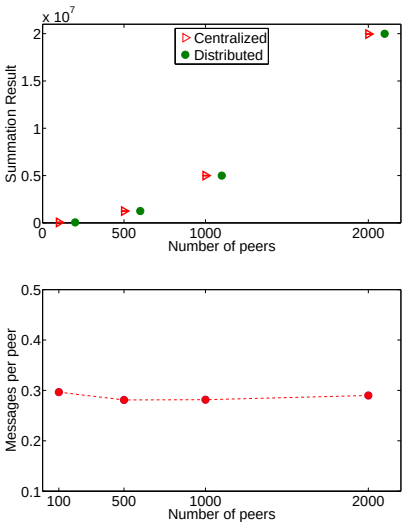

Figure 3. Scalability of the algorithm as the number of peers is increased: quality (top) and cost (bottom).

6.2 Results on Real Dataset

Finally in this section we describe the results of the experiments with a real data set. Volunteers at UMBC were asked to search for the following five categories in the popular search engines: (1) digital camera, (2) auto insurance, (3) cars, (3) laptop, and (4) gps systems. They were also asked to store the web urls which they found as the closest match for each of these categories. In the experimental

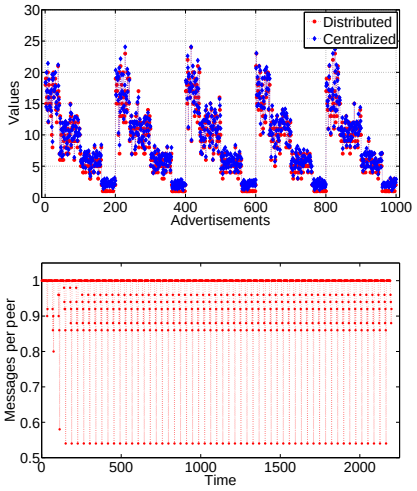

Figure 4. Results on the real advertisement data set: relative orderings (top) and messages exchanged (bottom).

setup, we list all these links in a single file (for all categories) and for each link, count the number of times it has been reported by the volunteer. In order to simulate the P2P setup, we then divide this data file randomly among 100 peers, such that each peer contains only fraction of the data — either links or count for each link. If a peer does not have a link, it may add a value of zero in order to participate in the **L-PPSC** protocol. In total there were 1000 links. Once the rings were formed using the **L-Ring** protocol, we ran 1000 sum computations in parallel. Figure 4 shows the results of the **L-PPSC** protocol on this data set. The x-axis in the quality figure (top) refers to the 1000 links grouped per category. The y-axis shows the total count per link for the **L-PPSC** protocol (circles). Also shown in the figure are the true counts per link (diamonds) which we call the centralized execution scenario. As easily verified, the counts of the links in the distributed experiments is very close to those found in the centralized situation. Similarly, the cost figure (bottom) shows the number of messages exchanged per peer per unit of time which varies between 0.5 and 1. A value of x at a particular time instance means that only $x\%$ of all the peers send message at that time instance.

7 Conclusion

In this paper we have presented a local privacy-preserving peer-to-peer data aggregation algorithm for doing data mining in a large P2P setting. Due to the constant communication complexity and locally synchronous nature of the algorithm, it is highly scalable. We have framed privacy and cost as a multi-objective optimization local to each peer and shown that our proposed algorithm is privacy-preserving according to this definition. To the best of the authors' knowledge, this is one of the first solutions which blends in the concept of local asynchronous distributed averaging with secure sum protocol to develop a scalable privacy preserving sum computation algorithm tailored to accommodate every participant's privacy and cost constraints. This algorithm is, therefore, applicable for large scale heterogeneous distributed systems such as the Internet and has various applications that require privacy preserving data mining.

Acknowledgement

This research is supported by the NASA Grant NNX07AV70G and the AFOSR MURI Grant 2008-11.

References

[1] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations*, 4(2), 2003.

[2] K. Das, K. Bhaduri, K. Liu, and H. Kargupta. Distributed Identification of Top-l Inner Product Elements and its Application in a Peer-to-Peer Network. *TKDE*, 20(4):475–488, 2008.

[3] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed Data Mining in Peer-to-Peer Networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[4] A. Evfimevski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of SIGMOD'03*, San Diego, CA, June 2003.

[5] H. Kargupta, K. Das, and K. Liu. Multi-Party, Privacy-Preserving Distributed Data Mining using a Game Theoretic Framework. In *Proc. of PKDD'07*, pages 523–531, 2007.

[6] H. Kargupta and K. Sivakumar. *Existential Pleasures of Distributed Data Mining. Data Mining: Next Generation Challenges and Future Directions.* AAAI/MIT press, 2004.

[7] K. Liu, K. Bhaduri, K. Das, P. Nguyen, and H. Kargupta. Client-side Web Mining for Community Formation in Peer-to-Peer Environments. *SIGKDD Explorations*, 8(2):11–20, 2006.

[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE'06*, page 24, GA, April 2006.

[9] S. Mane, S. Mopuru, K. Mehra, and J. Srivastava. Network Size Estimation In A Peer-to-Peer Network. Technical Report 05-030, University of Minnesota, September 2005.

[10] D. Scherber and H. Papadopoulos. Distributed Computation of Averages Over ad hoc Networks. *IEEE Journal on Selected Areas in Communications*, 23(4):776–787, 2005.

[11] M. Trottini, S. Fienberg, U. Makov, and M. Meyer. Additive Noise and Multiplicative Bias as Disclosure Limitation Techniques for Continuous Microdata: A Simulation Study. *J. Comp. Methods in Sci. and Eng.*, 4(1,2):5–16, 2004.

[12] A. C. Yao. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, pages 162–167, 1986.