A Game Theoretic Approach toward Multi-Party Privacy-Preserving Distributed Data Mining

Hillol Kargupta⋆, Kamalika Das, Kun Liu
{hillol,kdas1,kunliu1}@cs.umbc.edu

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD-21250, USA

Abstract. Analysis of privacy-sensitive data in a multi-party environment often assumes that the parties are well-behaved and they abide by the protocols. Parties compute whatever is needed, communicate correctly following the rules, and do not collude with other parties for exposing third party sensitive data. This paper argues that most of these assumptions fall apart in real-life applications of privacy-preserving distributed data mining (PPDM). The paper offers a more realistic formulation of the PPDM problem as a multi-party game where each party tries to maximize its own objectives. It offers a game-theoretic framework for developing and analyzing new robust PPDM algorithms. It also presents equilibrium-analysis of such PPDM-games and outlines a game-theoretic solution based on the concept of "cheap-talk" borrowed from the economics and the game theory literature.

1 Introduction

Advanced analysis of multi-party privacy-sensitive data plays an important role in many cross-domain applications that require large-scale information integration. Privacy preserving data mining algorithms attempt to analyze multi-party data for detecting underlying patterns without necessarily divulging the raw privacy-sensitive data to any of the parties. However, many of these privacy-preserving distributed data mining (PPDM) algorithms make strong assumptions about the behavior of the participating entities. For example, they assume that the parties are semi-honest; that is they follow the protocol without deviation. For example, the US Department of Homeland Security funded PURSUIT project[1] involves privacy preserving distributed data integration and analysis of network traffic data from different organizations. However, network traffic is usually privacy sensitive and no organization would be willing to share their network traffic with a third party. PPDM offers one possible solution which would allow comparing and matching multi-party network traffic for detecting common attacks, stealth attacks and computing various statistics for a group of organizations without necessarily sharing the raw data. However, participating organization in a consortium like PURSUIT may not all be ideal. Some may decide to behave like a "leach" exploiting the benefit

⋆ The author is also affi liated to Agnik, LLC, MD, USA
[1] http://www.agnik.com/DHSSBIR.html

of the system without contributing much. Some may intentionally try to sabotage the multi-party computation. Some may try to collude with other parties for exposing the private data of a party.

This paper suggests an alternate perspective for relaxing some of the assumptions of PPDM algorithms. It argues that large-scale multi-party PPDM can be thought of a game where each participant tries to maximize their benefit or utility score by optimally choosing the strategies during the entire PPDM process. The paper develops a game-theoretic framework for analyzing the behavior of such multi-party PPDM algorithms and offers a detailed analysis of the well known secure multi-party sum computation algorithm as an example [2]. The paper also offers an equilibrium analysis of the algorithm. The paper proposes a new version of secure sum algorithm that works based on “cheap talk” [3], a well known concept from game theory and economics. It also describes experiments on large scale distributed systems and illustrates the validity of the formulations.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes multi-party PPDM from a game theoretic perspective. Section 4 illustrates the framework using multi-party secure sum computation as an example. Section 5 gives the optimal solution using a distributed penalty function mechanism. Section 6 presents the experimental results. Finally, Section 7 concludes this paper.

2 Related Work

Recent interest in the collection and monitoring of data using data mining technology for the purpose of security and business-related applications has raised serious concerns about privacy issues. There exists a growing body of literature on privacy preserving data mining. Next we present a brief overview of the various techniques that have been developed in this area.

Existing techniques for privacy preserving data mining include data hiding using microaggregation, perturbation, or anonymization, rule hiding, secure multi-party computation (SMC) and distributed data mining. A large body of cryptographic protocols including circuit evaluation protocol, oblivious transfer, homomorphic encryption, commutative encryption serve as the building blocks of SMC. A collection of SMC tools useful for privacy preserving data mining (e.g., secure sum, set union, inner product) are discussed in [2]. The distributed data mining (DDM) approach supports computation of data mining models and extraction of ”patterns” at a given node by exchanging only the minimal necessary information among the participating nodes.

Game theory has been used extensively in economics and finance and security or defense related applications to come up with policies and governing rules. However, applications of game theory in privacy analysis of data mining algorithms in distributed scenarios is an area that is still in its nascent stage. Halpern and Teague [4] considered the problem of secret sharing and multiparty computation among rational agents. Abraham et al. [1] introduced the k-resilient Nash equilibrium (defined in Section 3) and offered a synchronous k-resilient algorithm for solving Shamir’s secret sharing [8] problem. A proposal of using game-theoretic way for measuring the privacy of PPDM was proposed elsewhere [10]. More recently, Kunreuther and Heal [7] and Kearns and Ortiz

[6] proposed a practical security problem called the *Interdependent Security (IDS)*. The model of IDS is closely related to privacy preserving data mining. Kearns and Ortiz [6] deals with the computability of Nash equilibria of *IDS* games and presents several algorithms for the same.

3 Multi-Party Privacy-Preserving Data Mining As Games

Let $D = \{d_1, d_2, \cdots d_n\}$ be a collection of n different nodes where each node represents a party with some privacy-sensitive data. The goal is to compute certain functions of this multi-party data using some PPDM algorithm. Most existing PPDM algorithms assume that every party cooperates and behaves according to the protocol. For example, consider a well-understood algorithm for computing sum based on the secure multi-party computation framework (details to be described in Section 4). Upon receipt of a message, a node performs some local computation, changes its states, and sends out some messages to other nodes. Most privacy-preserving data mining algorithms for multi-party distributed environments work in a similar fashion. During the execution of such a PPDM algorithm, each node has certain responsibilities in terms of performing their part of the computation, communicating correct values to other nodes and protect the privacy of the data. Depending on the characteristics of these nodes and their objectives, they either perform their duties or do not. Sometimes they even collude with other nodes to modify the protocol and reveal others' data.

3.1 Strategies for Multi-Party Games

Consider a PPDM algorithm that requires the i-th node to perform a sequence of m_i computing tasks, $M_{i,1}, M_{i,2}, \cdots M_{i,m_i}$. Also the algorithm requires a node to both send and receive messages. Let us assume that the i-th node sends out s_i messages ($S_{i,1}$, $S_{i,2}$,$\cdots S_{i,s_i}$) to other nodes and receives a total of r_i messages ($R_{i,1}$, $R_{i,2}$,$\cdots$ R_{i,r_i}). The different computational strategies the ith node might node adopt include performing the required computation or not. Similarly, a node can decide either to send or receive messages to and from other nodes or not. Let $I_{i,t}$ be the corresponding indicator variable which takes a value of 1 if at the tth step the ith node chooses the strategy defined by the protocol or takes 0 otherwise. The decision to choose a particular strategy is either deterministic or probabilistic. Let $I_i^{(M)}$ be the overall sequence of indicator variables $I_{i,1}, I_{i,2}, \cdots I_{i,m_i}$ for computations for the ith node. Similarly $I_i^{(R)}$ and $I_i^{(S)}$ are indicator variables for receiving and sending of messages. Let $c_m(M_{i,t})$ be the utility of performing the operation $M_{i,t}$ (similarly we can define utility for communication).

Collusion is nothing but a privacy attack launched on a third party's private data, only involving other nodes in the process. At any given step node i may come to an agreement with any subset (denoted by G) of n nodes (denoted by D) and decide to collude with them for exposing the privacy-sensitive data of another node. Let $I_{i,j}^{(G)}$ be the corresponding indicator variable which takes a value of 1 if party i decides to collude with party j or 0 otherwise. We shall use the symbol $I^{(G)}$ to denote the entire matrix where the (i, j)-th entry is $I_{i,j}^{(G)}$. Let $g_{i,j}$ be the benefit that node i may get by colluding with node j. For the sake of simplicity let us assume that collusions are permanent. Once a pair decides to collude, they stay faithful to each other. If every member of the set G agrees to collude with every other member in G then the total benefit that a node $i \in G$ receives is $\sum_{j \in G | j \neq i} g_{i,j}$.

3.2 Overall Game

The strategies chosen by a party at any step change the local state of the party. The entire play of the game by player i can therefore be viewed as a process of traversing through a game tree where each tree-node represents the local state described by player i's initial state and messages communicated with other nodes. Each run r represents a path through the tree ending at a leaf node. The leaf node for path (run) r is associated with a utility function value $u_i(r)$. A strategy σ_i for player i prescribes the action for this player at every node along a path in the game tree. In the current scenario, the strategy prescribes the actions for computing, communication, and collusion with other parties. A strategy σ_i for player i essentially generates the tuple $(I_i^{(M)}, I_i^{(R)}, I_i^{(S)}, I_i^{(G)})$.

Let $\overline{\sigma} = (\sigma_1, \sigma_2, \cdots \sigma_n)$ be the joint strategy for n players. Let $u_i(\overline{\sigma})$ be the utility (also known as pay-off) when the joint strategy $\overline{\sigma}$ is played. A joint strategy is a Nash equilibrium if no player can gain any advantage by using a different strategy, given that all other players do not change their strategies. However, Abraham et al. [1] remove the fragility of Nash equilibrium by allowing coalitions whose behavior deviate from the protocol. They define k-resilient equilibrium as a joint strategy $\overline{\sigma}$ such that for all coalitions C in the set of players D with $|C| <= k$, σ_C is the group best response for C to σ_{-C}, where σ_C is the set of strategies adopted by the members of the collusion, σ_{-C} is the set of strategies adopted by the players who do not belong to the collusion and $\sigma_C \in \overline{\sigma}$ and $\sigma_{-C} \in \overline{\sigma}$. Best response for a collusion σ_C to a strategy σ_{-C} implies that the utility of the σ_C is at least as high as the utility of σ_{-C}.

In order to formulate a PPDM process in a game theoretic framework, we need to define the search space of the strategies for the players and construct the utility functions for defining the pay-off that the players receive by playing a given strategy. Next section considers an example and constructs PPDM-games based on that.

3.3 Utility Function Representing the Game

Utility functions assign a score to a strategy or a set of strategies. Note that the search process in the strategy-space can either be deterministic or stochastic. In the deterministic case players design their respective strategies and take deterministic actions in order to maximize their utility scores. For games in PPDM the utility function will be a linear or nonlinear function (f) of utilities obtained by the choice of strategies in the respective dimensions of computation, communication, collusion and privacy attack. Mathematically we can write $u_i(\overline{\sigma}) = f(I_i^{(M)}, I_i^{(R)}, I_i^{(S)}, I_i^{(G)})$ A utility function which is a weighted linear combination of all of the above dimensions can therefore be expressed in terms of the individual utilities as follows:

$$u_i(\overline{\sigma}) = w_{m,i} \sum_t c_m(M_{i,t}) + w_{r,i} \sum_t c_r(R_{i,t}) + w_{s,i} \sum_t c_s(S_{i,t}) + w_{g,i} \sum_{j \in G} g_{i,j}$$

where $w_{m,i}, w_{r,i}, w_{s,i}$ and $w_{g,i}$ represent the weights for the corresponding utility factors. The overall utility function may also turn out to be constrained. For example, a participant may require bounds on the computation and communication cost over a segment of the entire process. A constraint like this may be captured in the following form:

$$\alpha_{m,i} \sum_{t=0}^{T} c_m(M_{i,t}) + \alpha_{r,i} \sum_{t=0}^{T} c_r(R_{i,t}) + \alpha_{s,i} \sum_{t=0}^{T} c_s(S_{i,t}) \leq \beta_i$$

In the next section we would illustrate our formalizations using one of the most popular PPDM algorithms, the secure sum computation. We first derive closed form expressions for each of the dimensions of PPDM with respect to secure sum and then do a Nash equilibrium analysis of the algorithm. We then propose a modified secure sum algorithm that gets rid of the semi honest requirement of the nodes and illustrate the change in equilibrium for the new algorithm.

4 Illustration: Multi-Party Secure Sum Computation

Suppose there are n individual sites, each with a value $v_j, j = 1, 2, \ldots, n$. It is known that the sum $v = \sum_{j=1}^{n} v_j$ (to be computed) takes an integer value in the range $0, 1, \ldots, N-1$. We want to compute this sum following the secure computation protocol described in [2].

4.1 Computation, Communication and Collusion Utilities

The secure sum computation algorithm expects each party to perform some local computation, receive data from its predecessor in the ring topology and send its own results to the next node. The site may or may not choose to perform either the computation or computation or both. This choice will define the strategy of that site.

Let us assume that there are k $(k \geq 2)$ sites acting together secretly to achieve a fraudulent purpose. Let v_i be an honest site who is worried about his/her privacy. We also use v_i to denote the value in that site. Let v_{i-1} be the immediate predecessor of v_i and v_{i+1} be the immediate successor of v_i. The possible collusions that can arise are:

- If $k = n - 1$, then the exact value of v_i will be disclosed.
- If $k \geq 2$ and the colluding sites include both v_{i-1} and v_{i+1}, then the exact value of v_i will be disclosed.
- If $n - 1 > k \geq 2$ and the colluding sites contain neither v_{i-1} nor v_{i+1}, or only one of them, then v_i is disguised by $n - k - 1$ other sites' values. As before, we shall use the symbol $\mathcal{C}$ to represent the set of colluding sites.

The first two cases need no explanation. Now let us investigate the third case. Without loss of generality, we can arrange the sites in the an order such that $v_1 v_2 \ldots v_{n-k-1}$ are the honest sites, v_i is the site whose privacy is at stake and $v_{i+1} \ldots v_{i+k}$ form the colluding group. We have

$$\underbrace{\sum_{j=1}^{n-k-1} v_j}_{\text{denoted by X}} + \underbrace{v_i}_{\text{denoted by Y}} = \underbrace{v - \sum_{j=i+1}^{i+k} v_j}_{\text{denoted by C}},$$

where v is the total sum of the n values. Now, the colluding sites can compute the posterior probability mass function (PMF) of v_i as follows:

$$f_{posterior}(v_i) = f_Y(y) = Pr\{Y = y\}, \tag{1}$$

where $Y = C - X$. X is a random variable and it is defined as $X = \sum_{j=1}^{n-k-1} v_j$. C is a constant and it is defined as $C = v - \sum_{j=i+1}^{i+k} v_j$. C is known to all the colluding sites. Because X is a discrete random variable, it is easy to prove that

$$f_Y(y) = f_X(x), \tag{2}$$

where $x = C - y$. To compute $f_X(x)$, we can make the following assumption about the adversarial parties' prior knowledge.

Assumption 1 *Each v_j $(j = 1, \ldots, n-k)$ is a discrete random variable independent and uniformly taking non-negative integer values over the interval $\{0, 1, \ldots, m\}$. Therefore, X is the sum of $(n-k-1)$ independent and uniformly distributed discrete random variables.*

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 [9] where a Bayes intruder model was proposed to assess the security of additive noise and multiplicative bias. Now let us compute $f_X(x)$.

Theorem 2. *Let Λ be a discrete random variable uniformly taking non-negative integer values over the interval $\{0, 1, \ldots, m\}$. Let Θ be the sum of s independent Λ. The probability mass function (PMF) of Θ is given by the following equations:*

$$Pr\{\Theta = \theta\} = \frac{1}{(m+1)^s} \sum_{j=0}^{r} (-1)^j C_s^j C_{s+(r-j)(m+1)+t-1}^{(r-j)(m+1)+t},$$

where $\theta \in \{0, 1, \ldots, ms\}$, $r = \lfloor \frac{\theta}{m+1} \rfloor$, and $t = \theta - \lfloor \frac{\theta}{m+1} \rfloor (m+1)$.

Proof. Due to space constraints, we have not included the proof of this theorem here. Interested readers can find a detailed proof of this theorem in [5].

According to Theorem 2, the probability mass function (PMF) of X is

$$f_X(x) = Pr\{X = x\} = \frac{1}{(m+1)^{(n-k-1)}} \times \sum_{j=0}^{r} (-1)^j C_{(n-k-1)}^j C_{(n-k-1)+(r-j)(m+1)+t-1}^{(r-j)(m+1)+t}, \tag{3}$$

where $x \in \{0, 1, \ldots, m(n-k-1)\}$, $r = \lfloor \frac{x}{m+1} \rfloor$, and $t = x - \lfloor \frac{x}{m+1} \rfloor (m+1)$. Combining Eq. 1, 2 and 3, we get the posterior probability of v_i:

$$f_{posterior}(v_i) = \frac{1}{(m+1)^{(n-k-1)}} \times \sum_{j=0}^{r} (-1)^j C_{(n-k-1)}^j C_{(s-k-1)+(r-j)(m+1)+t-1}^{(r-j)(m+1)+t},$$

where $x = C - v_i$ and $x \in \{0, 1, \ldots, m(n-k-1)\}$. $r = \lfloor \frac{x}{m+1} \rfloor$, and $t = x - \lfloor \frac{x}{m+1} \rfloor (m+1)$. Note that here we assume $v_i \leq C$, otherwise $f_{posterior}(v_i) = 0$. This posterior can be used to quantify the utility of collusion:

$$g(v_i) = Posterior - Prior = f_{posterior}(v_i) - \frac{1}{m+1} \tag{4}$$

We see here that the utility of collusion depends on the random variable x and the size of the colluding group k. Rest of this paper will use this quantitative measure of utility of collusion for defining the objective function.

4.2 Overall Objective Function

Now we can put together the overall objective function for the game of multi-party secure sum computation.

$$u_i(\overline{\sigma}) = w_{m,i} c_m U(I_i^{(M)}) + w_{r,i} c_r U(I_{i,t}^{(R)}) + w_{s,i} c_s U(I_i^{(S)}) + w_{g,i} \sum_{j \in D-\mathcal{C}} g(v_j)$$

where $c_m U(I_i^{(M)}) = \sum_t c_m(M_{i,t})$ denotes the overall utility of performing a set of computations $M_{i,t}$, indicated by I_i^M (similar definitions apply for communications like receive and send) and $w_{m,i}$ denotes the weight associated with computation.

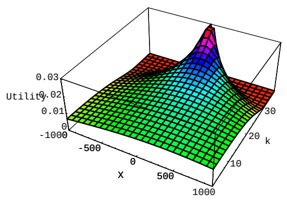
For illustrating the equilibrium state of this utility function, let us consider the simple unconstrained version of it. In order to better understand the nature of the landscape let us consider a special instance of the objective function where the node performs all the communication and computation related activities as required by the protocol. This results in an objective function

$$u_i(\overline{\sigma}) = w_{g,i} \sum_{j \in D-\mathcal{C}} g(v_j)$$

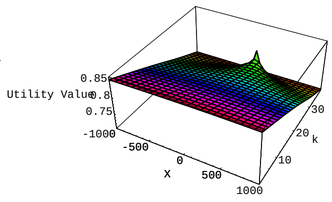
where the utilities due to communication and computation are constant and hence can be neglected for determining the nature of the function. Figure 1(a) shows a plot of the overall utility of multi-party secure sum as a function of the distribution of the random variable x and the size of the colluding group k. It shows that the utility is maximum for a value of k that is greater than 1. Since the strategies opted by the nodes are dominant(illustrated in the next section with an example), the optimal solution corresponds to the Nash equilibrium. This implies that in a realistic scenario for multi-party secure sum computation, parties will have a tendency to collude. Therefore the non-collusion ($k = 1$) assumption of the classical SMC-algorithm for secure sum is sub-optimal.

5 What is the Solution?

Our goal is to design a technique for multi-party secure sum computation that does not rely on unrealistic assumptions about the characteristics of nodes in the network. Therefore, we want an algorithm that creates a game where not colluding is an optimal strategy for everyone. One possible approach is to penalize the parties sufficiently enough so that the pay-off from collusion is counter-balanced by the penalty, if they



(a) Original Objective Function.



(b) Modified Objective Function.

Fig. 1. Plots of the overall objective functions for classical SMC (Figure 1(a)) and SMC with penalty (Figure 1(b)). The optimal strategy takes a value of $k > 1$ in the first case and $k = 1$ in the second case.

are caught. This approach may not work if the parties perceive that the possibility of getting caught is minimal. However, collusion requires consent from multiple parties. So a party with intention of collusion might get caught while sending out collusion invitations randomly in the network if those invitations reach some honest parties.

5.1 Penalty for Collusion

The possible options that we have for penalizing cheaters (colluding nodes) in a multi-party secure sum game can be enumerated as follows:

1. Policy I: Remove the party from the multi-party privacy-preserving data mining application environment because of protocol violation. Although it may work in some cases, the penalty may be too harsh since usually the goal of a PPDM application is to have everyone participate in the process and faithfully contribute to the data mining process.
2. Policy II: An alternate possibility is to penalize by increasing the cost of computation and communication. For example, if a party suspects a colluding group of size k'(an estimate of k) then it may split the every number used in a secure sum among k' different parts and demand k rounds of secure sum computation one for each of these k' parts. This increases the computation and communication cost by k'-fold. This linear increase in cost with respect to k', the suspected size of colluding group, may be used to counteract any possible benefit that one may receive by joining a team of colluders. The modified objective function is given below. The last term in the equation accounts for the penalty due to excess computation and communication as a result of collusion.

$$u_i(\overline{\sigma}) = w_{m,i} c_m U(I_i^{(M)}) + w_{g,i} \sum_{j \in D-G} g(v_j) - w_p * k'$$

Figure 1(b) shows a plot of the modified objective function for secure sum with policy II. It shows that the globally optimal strategies are all for $k = 1$. The strategies that adopt collusion always offer a sub-optimal solutions which would lead to moving the global optimum to the case where $k = 1$.

5.2 Nash Equilibrium Illustration for 3-node Network

Consider a game with a mediator where each party first contacts the mediator and declares its intention to be a good node (follow protocol) or bad party (willing to collude). Table 1 shows the pay-offs for different penalty policies. When there is no penalty, all the scenarios with two bad parties and one good party offer the highest pay-off for the colluding bad parties. Therefore, colluding with other nodes always becomes the highest paying strategy for any node in the network (this is called the dominant strategy [4]). Also we observe that the payoff for bad nodes always decreases if it becomes good, assuming the status of all other nodes remain unchanged. So the Nash equilibrium in the classical secure sum computation is the scenario where the participating nodes are likely to collude. Note that, the three-party collusion is not very relevant in secure sum computation since there are all together three parties and there is always a good node (the initiator) who wants to only know the sum.

A	B	C	Pay-Off (No Penalty)	Pay-Off (Policy I)	Pay-Off (Policy II)
G	G	G	(3, 3, 3)	(3, 3, 3)	(3, 3, 3)
G	G	B	(3, 3, 3)	(2, 2, 0)	(2, 2, 2)
G	B	G	(3, 3, 3)	(2, 0, 2)	(2, 2, 2)
G	B	B	(3, 4, 4)	(0, 0, 0)	(2, 2, 2)
B	G	G	(3, 3, 3)	(0, 2, 2)	(2, 2, 2)
B	G	B	(4, 3, 4)	(0, 0, 0)	(2, 2, 2)
B	B	G	(4, 4, 3)	(0, 0, 0)	(2, 2, 2)
B	B	B	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Table 1. Pay-Off table for three-party secure sum computation.

However, in both the cases with penalty, the Nash equilibrium corresponds to the strategy where none of the parties collude. In fact the Nash equilibrium in secure sum with policy II for penalization is strongly resilient, that is, it corresponds to a 2-resilient equilibrium for a 3-party game. In this case, if any node deviates from being good, the communication and computation cost increase k' ($O(k)$) fold due to the data being broken into shares. The penalty incurred due to this extra resource usage is not compensated by the benefit gained out of the collusion. Therefore the payoff is the highest when they don't collude. This is, in fact a strongly dominated strategy for this game. Since each player has a strictly dominated strategy, there is a unique Nash equilibrium which is the case in which none of the nodes collude. According to the definition of Nash equilibrium, in this case no player can gain anything more by deviating from good to bad when all others remain good. It also follows from the discussion that Policy II is *strongly resilient* since it can tolerate collusions of size up to n-1 (the payoff is highest when none colludes, even when compared to the case where $n - 1$ (n=3) collude).

Algorithm 1 Secure Sum with Penalty (SSP)

```
Input of node $P_i$: Neighbor $P_j$, $v_i$, $k'$ estimated from prior cheap talk
Output of node $P_i$: Correct secure sum
Data structure for $P_i$: NODETYPE (0 stands for good, 1 stands for bad, and 2 stands for initiator), colluding group
(colludeList), random shares of $v_i$ (randSharesList)
Initialization:
IF NODETYPE==0
  Initialize colludeList
  Exchange sum of elements in colludeList
ELSE IF NODETYPE==1
    Split the local data $v_i$ into at least ($k'$) random shares
    Initialize randSharesList
  ELSE IF NODETYPE==2
      Send its data $v_i$ after adding a random number and performing a modulo
      operation
    ENDIF
  ENDIF
ENDIF
On receiving a message:
IF NODETYPE==2
  IF randSharesList==NULL (for every node)
    End Simulation
    Send sum to all nodes
  ELSE
    Start another round of secure sum
  ENDIF
ELSE IF randSharesList!=NULL
    Select next data share from randSharesList
    Forward received data and new share to next neighbor
  ENDIF
ENDIF
```

5.3 Implementing the Protocol with Penalty

We observed in the last section that an appropriate amount of penalty for violation of the policy may reshape the objective function in such a way that the optimal strategies correspond to the prescribed policy. If we use ***centralized control***, then there is a central authority who is always in charge of implementing the penalty policy. If a good node receives a proposal from a bad node then it reports to the central authority which in turn penalizes the perpetrator. However, it requires global synchronization which might create a bottleneck and limit the scalability of a distributed system.

For a total ***asynchronous distributed control*** we borrow the concept of $cheaptalk$ from game theory and economics [3] in order to develop a distributed mechanism for penalizing policy violations. Cheap talk is a pre-play communication which carries no cost. Before the game starts, each player engages in a discourse with each other in order to influence the outcome of the game and form an opinion about the other players in the game. We would like to use cheap talk to communicate the threat of penalty. Cheap talk works when the parties depend on each other, their preferences are not opposite to each other, and the threat is real. Algorithm 1 (Secure Sum with Penalty (SSP)) describes a variant of the secure sum computation technique that offers a distributed mechanism for penalizing protocol violations (using Policy II) using a cheap talk-like mechanism.

Consider a network of n nodes where a node can either be *good*(honest) or *bad* (colluding). Before the secure sum protocol starts, the good nodes set their estimate of bad nodes in the network (k') to 0 and bad nodes send invitations for collusions randomly to nodes in the network. Every time a good node receives such an invitation, it increments its estimate of k'. Bad nodes respond to such collusion invitations and form collusions. If a bad node does not receive any response, it behaves as a good node. To penalize nodes that collude, good nodes split their local data into k' random shares. This initial phase of communication is cheap talk in our algorithm. The secure sum phase consists of k' (O(Max(k'')) rounds of communication for every complete sum computation. In every round each honest node adds one of it k' shares to the sum and forwards the result. The following Lemma (Lemma 1) shows that the SSP algorithm converges to the correct result.

Lemma 1. ***Correctness and Convergence:*** *SSP algorithm converges to the correct result in $O(nk)$ time. Here n is the total number of nodes in the network, and k is the maximum size of the colluding groups .*

Proof. (SKETCH) The basic idea behind this proof is that sum computation is decomposable, and the order of addition of individual shares does not change the total. In the SSP algorithm, each party P_i splits its number into $k_i(k_i \geq 0)$ shares and demands k_i rounds of secure sum computation. In each round, whenever a party receives a message, it adds one of its k_i shares. If all its shares have been added in, this party simply inputs a zero. Let $k = \max_i\{k_i\}$, after k rounds of computation, all the parties have added their numbers and the total sum is obtained. In the traditional secure sum computation, the message is passed to each node on the network sequentially. The convergence time is bounded by $O(n)$. In the SSP algorithm, the total rounds of computation is k, therefore the overall time required is bounded by $O(nk)$. □

6 Experimental Results

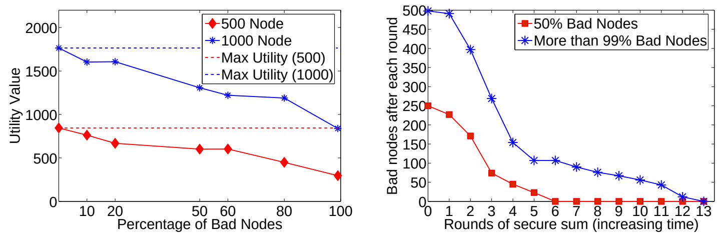
We set up a simulation environment comprised of a network of n nodes where nodes are randomly good or bad. We have experimented with ring topologies of 500 and 1000 nodes using the use the Distributed Data Mining Toolkit (DDMT) [2]. To demonstrate how the SSP algorithm forces the bad nodes in the network to adopt a no collusion strategy, each node in the network performs a series of secure sums on each of the l values in its data vector of size l(=13 for our experiments). Each secure sum computation consists of several rounds of communication since every integer is broken into k' random shares. After every round of secure sum computation (that is after each of the l sum computations), we measure three quantities: messages sent, units of computation performed and the benefit achieved by colluding (Section 4.1). The utility function used for the experiments is the one described in Section 4.2. The penalty in this case is the excess amount of communication and computation needed.

6.1 Results

We perform two experiments to verify our claim that the SSP algorithm leads to an equilibrium state where there is no collusion. In the first experiment we demonstrate

[2] DDMT - http://www.umbc.edu/ddm/wiki/software/DDMT/

for different sizes of the network that the utility is maximum when the collusion is minimum. We have experimented for 500-node and 1000-node networks for different percentages of bad nodes. We see in Figure 2(a) that when the percentage of bad nodes is very high(more than 99%), then the utility is minimum for all the nodes. The utility increases as this percentage decreases. The maximum utility for each network size is shown as the horizontal lines in the graph, which correspond to the classical secure sum computation algorithm. We observe that the maximum utility for the SSP algorithm corresponds to the case when there is no collusion.



(a) Utility vs. Collusion-size Function.

(b) Rate of decrease of bad nodes Function.

Fig. 2. Experimental Results

In Figure 2(b) we have shown the results of our second experiment which verifies how the number of bad nodes decrease with successive rounds of secure sum computation. At the end of each round of secure sum, the bad nodes that are part of any colluding group compute their utility values. Each bad node has a random utility threshold that is assigned during setup. If the computed utility falls below a node's threshold, the node decides to change its strategy and becomes a good node for the subsequent rounds. Once a node becomes good, it does not change its strategy again. We perform the second experiment for a 500-node network with 50% and more than 99% bad nodes as the two cases. We see that in subsequent rounds of secure sum the number of bad nodes decrease till they reach zero. The time taken to have a no collusion scenario depends on the initial number of bad nodes in the network.

7 Conclusions

This paper pointed out that many of the existing privacy-preserving data mining algorithms often assume that the parties are well-behaved and they abide by the protocols as expected. The contributions of this paper can be summarized as follows: (i)development of a game-theoretic foundation of multi-party privacy-preserving distributed data mining that attempts to relax many of the strong assumptions made by existing PPDM algorithms; (ii)analysis and illustration of shifting equilibrium conditions in such algorithms; (iii) game theoretic analysis of the multi-party secure sum computation algorithm in terms of data privacy and resource usage and (iv) a "cheap-talk"-based distributed variant of secure sum computation which offers a protocol that offers an equilibrium solution for no collusion.

The paper opens up many new possibilities in PPDM. In future we plan to study other popular PPDM algorithms for computing inner product, clustering, and association rule learning using the game theoretic framework developed here.

References

1. I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *ACM Symposium on Principles of Distributed Computing*, Denver, Colorado, USA, 2006.
2. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2), 2003.
3. J. Farrell and M. Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996.
4. J. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In *Proc. of ACM Symposium on Theory of Computing*, pages 623 – 632, Chicago, IL, USA, 2004.
5. H. Kargupta, K. Das, and K. Liu. A game theoretic approach toward multi-party privacy-preserving distributed data mining. Technical Report TR-CS-0701, UMBC, April 2007.
6. M. Kearns and L. Ortiz. Algorithms for interdependent security games. *Advances in Neural Information Processing Systems*, 2004.
7. H. Kunreuther and G. Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2-3):231–249, 2003.
8. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
9. M. Trottini, S. E. Fienberg, U. E. Makov, and M. M. Meyer. Additive noise and multiplicative bias as disclosure limitation, techniques for continuous microdata: A simulation study. *Journal of Computational Methods in Sciences and Engineering*, 4:5–16, 2004.
10. N. Zhang, W. Zhao, and J. Chen. Performance measurements for privacy preserving data mining. In *Advances in Knowledge Discovery and Data Mining*, pages 43–49, 2005.