Technical Report TR-CS_01_07

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

Hillol Kargupta, Kamalika Das, Kun Liu

Department of Computer Science and Electrical Engineering
University of Maryland-Baltimore County
1000 Hilltop Circle
Baltimore MD 21250

April 24, 2007

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

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

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 develops a game-theoretic framework for developing and analyzing 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. The data mining community has responded to this challenge by developing a new breed of distributed data mining algorithms that are privacy preserving. These 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; they perform all the necessary computation, communicate the way they are supposed to, do not collude and do not try to sabotage the process.

This paper suggests an alternate perspective for relaxing some of these assumptions. It argues that large-scale multi-party PPDM can be viewed as a game where each participant tries to maximize its 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 [8]. The paper also offers an equilibria analysis of this game and notes that the traditional version of secure sum computation does not offer a stable equilibrium since the protocol does not correspond to an optimal strategy for the players in a realistic scenario. The paper proposes a new version of secure sum algorithm that works based on “cheap talk" [10], a well known concept from game theory and economics. It presents results from simulation of this multi-party algorithm and shows that the algorithm converges to a stable equilibrium state that corresponds to no collusion. The contributions of this paper can be summarized as follows:

1. 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.

2. Equilibrium analysis and illustration of shifting equilibrium conditions in such algorithms.

3. A game theoretic analysis of the multi-party secure sum computation algorithm in terms of data privacy and resource usage.

4. A “cheap-talk”-based distributed variant of secure sum computation which offers a protocol that satisfy Nash [17] and other equilibrium conditions [1].

The remainder of this paper is organized as follows. Section 2 offers the motivation of the work. Section 3 discusses the related work. Section 4 describes multi-party PPDM from a game theoretic perspective. Section 6 illustrates the framework using multi-party secure sum computation as an example. Section 7 gives the optimal solution using a distributed penalty function mechanism. Section 8 presents the experimental results. Finally, Section 9 concludes this paper.

2 Motivation

Information integration in multi-party distributed environment is often an interactive process guided by the dynamics of cooperation and competition among the parties. The behavior of these parties usually depend on their own objectives. For example, consider the US Department of Homeland Security funded PURSUIT project[1] for privacy preserving distributed data integration and analysis of network traffic data from different organizations. The goal here is to detect "macroscopic" patterns from network traffic of different organizations for revealing common threats against those organizations. For example, this may allow us to identify a group of attacking nodes that are methodically targeting the nuclear energy companies of the country. Such a system would allow us to detect threats against the overall cyber-infrastructure of the country.

However, network traffic is usually very privacy-sensitive and no organization would be willing to share their network traffic with a third party. Privacy-preserving distributed data mining (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.

As such multi-party systems start gaining popularity and get deployed in large scale, we would start facing a new set of problems. Participating organization in a consortium like PURSUIT may not all be ideal. Some may decide to behave like a "leach"—exploit the benefit 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 argues that large-scale multi-party PPDM essentially looks like 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 also points out the need for analyzing such games on a solid analytical foundation and offers one possible route through game theory. In this paper we develop the foundation of this approach, develop a game-theoretic formulation of one such multi-party PPDM algorithm, and perform large-scale experiments to illustrate the validity of our framework.

[1] http://www.agnik.com/DHSSBIR.html

3 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 [2], perturbation [3], [7], [16], [9] or anonymization [21], [5], rule hiding [4], secure multi-party computation [19] and distributed data mining. The main objective of data hiding is to transform the data or to design new computation protocols so that the private data still remains private during and/or after data mining operations; while the underlying data patterns or models can still be discovered. The main objective of rule hiding, on the other hand, is to transform the database such that the sensitive rules are masked, and all the other underlying patterns can still be discovered. The Secure Multi-party Computation (SMC) [23] technique considers the problem of evaluating a function of two or more parties' secret inputs, such that no party learns anything but the designated output of the function. 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 [8]. 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. In this section we review some of the existing literature in game theoretic analysis of PPDM followed by a discussion on some algorithms in game theory that might be adapted to formalize a PPDM model.

Halpern and Teague [11] considered the problem of secret sharing and multiparty computation among rational agents. Abraham et al. [1] introduced the k-resilient Nash equilibria and offered a synchronous k-resilient algorithm for solving Shamir's secret sharing [20] problem. A proposal of using game-theoretic way for measuring the privacy of PPDM was proposed elsewhere [24].

The randomized secret share protocol presented by Halpern and Teague [11] is one way by which agents will refrain from the Nash equilibrium of not sending any messages. More recently, Kunreuther and Heal [15] and Kearns and Ortiz [14] proposed a practical security problem called the *Interdependent Security (IDS)*. The authors illustrated the problem is the following example of airline companies. Consider an airline agency (*A*) debating whether to invest money for screening of passenger baggages. It is well known that for transfer passengers there is not enough time for *A* to recheck the bags. So, for any flights operated by *A*, there are two hazards – one posed by the baggages screened by *A* itself and the other posed by the baggages of the transfer passengers. Naturally, *A* does not have any control over the screening methods adopted by any other airline. If the probability that the other airline is lenient is very high, it may become futile for *A* to invest. If every airline company thinks this way, the Nash equlibrium of such a game may be the point where none invests. This is contrary to the expected equilibrium, where we want everyone to invest in security screening of baggages. This problem is a real life illustration of the "free rider's problem" [12] of game theory. Kunreuther and Heal [15] propose several policy-based issues to deal with this problem. On the other hand, Kearns and Ortiz [14] deals with the computability of Nash equilibria of *IDS* games and presents several algorithms for the same.

The above model of IDS is closely related to privacy preserving data mining. Let us consider a simple distributed scenario where each agent/entity has a single number and we are interested in finding the sum of such elements securely. Each entity, knows that if it can get the sum of all the other entities from the network, it can calculate the total by simply adding it own. Therefore, it may refrain from sending is own data and clearly the Nash equilibria is when no one sends anything. We want to develop a game playing strategy such that the Nash equilibria is where everyone sends.

4 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 6). 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 may have the following objectives, intentions, responsibilities: (1) Perform or do not perform the local computation, (2) communicate or do not communicate with the necessary parties, (3) send misleading information instead of divulging correct local data (4) protect the local private data, (5) attack the messages received from other parties for divulging privacy-sensitive information regarding other parties, and (6) collude with others to change the protocols for achieving any of the above tasks.

Our goal is to view multi-party privacy-preserving data mining in a realistic scenario where the participating nodes are not necessarily assumed to be well-behaved; rather we consider them as real-life entities with their own goals and objectives which control their own strategies for dealing with each of the above listed dimensions. The following part of this section considers each of these dimensions seperately.

4.1 Gaming Strategies for Computation

Consider a PPDM algorithm that requires the i-th node to perform a sequence of m_i computing tasks: $M_{i,1}, M_{i,2}, \cdots M_{i,m_i}$. The t-th computing operation at node i is represented by $M_{i,t}$. At any given step t, node i can either perform the corresponding computation $M_{i,t}$ or not do that. Let $I_{i,t}^{(M)}$ be the corresponding indicator variable which takes a value of 1 if $M_{i,t}$ is performed or 0 otherwise. Let $I_i^{(M)}$ be the overall sequence of indicator variables $I_{i,1}^{(M)}, I_{i,2}^{(M)}, \cdots I_{i,m_i}^{(M)}$. The decision to compute $M_{i,t}$ or not to compute can be made either using a deterministic or a probabilistic strategy. Let $c_m(M_{i,t})$ be the utility of performing the operation $M_{i,t}$.

4.2 Gaming Strategies for Communication

Now consider the communication-related tasks in a PPDM algorithm. The PPDM 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}) from other nodes.

A less than ideal node may take either of the following steps when it comes to receiving a message: (1) receive the message using a deterministic strategy or a probabilistic strategy; (2) not receive the message either based on a deterministic or probabilistic strategy. Let $I_{i,t}^{(R)}$ be the corresponding indicator variable. Let $I_i^{(R)}$ be the overall sequence of indicator variables $I_{i,1}^{(R)}$, $I_{i,2}^{(R)}$, $\cdots$ $I_{i,r_i}^{(R)}$.

Similarly, when it comes to sending out a message, the node may do any of the following things: (1) Send out the correct message, (2) send out incorrect/misleading message, and (3) do not send out any message. $I_{i,t}^{(S)}$ be the corresponding decision variable which takes a value of 0, 1, and 2 respectively for the above three choices. Let $I_i^{(S)}$ be the overall sequence of indicator variables $I_{i,1}^{(S)}$, $I_{i,2}^{(S)}$, $\cdots$ $I_{i,s_i}^{(S)}$.

Let $c_r(R_{i,t})$ and $c_s(S_{i,t})$ be the utility of performing the receive and send operations $R_{i,t}$ and $S_{i,t}$ respectively. When the strategy is not deterministic, probabilities can be attached for sending out the correct message and incorrect/misleading message.

4.3 Privacy Attacks

Node i may or may not launch attack on each of the r_i messages ($R_{1,1}, R_{1,2}, \cdots R_{1,r_i}$) it receives from other nodes. Possible strategies that a node can take are the following: successfully divulge third party data in a deterministic or probabilistic manner; unsuccessful attempt in doing the former or not divulge any data (honest node). Let us assume that the i-th node perform a sequence of a_i attacks ($A_{i,1}$, $A_{i,2}$,$\cdots$ A_{i,a_i}) on the messages received from other nodes. Let $I_{i,t}^{(A)}$ be the corresponding indicator variable. Let $I_i^{(A)}$ be the overall sequence of indicator variables $I_{i,1}^{(A)}, I_{i,2}^{(A)} \cdots I_{i,a_i}^{(A)}$.

4.4 Colluding with Other Parties

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 $\sum_{j \in G | j \neq i} g_{i,j}$. Similarly in the probabilistic scenario we can define an expected total benefit of each node in a collusion.

4.5 Overall Game

A multi-party PPDM process can be viewed as a game among the participating parties. Each game involves a set of actions by these parties. The actions 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, privacy protection, privacy-breaching attack, and collusion with other parties. A strategy σ_i for player i essentially generates the tuple $(I_i^{(M)}, I_i^{(R)}, I_i^{(S)}, I_i^{(A)}, I_i^{(G)})$.

Let $\overline{\sigma} = (\sigma_1, \sigma_2, \cdots \sigma_n)$ be the joint strategy for n players. In the probabilistic scenario, this defines a joint distribution over the paths in the game tree. 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.

5 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. Therefore, the utility function can also be either deterministic or stochastic. The following sections discuss these two possibilities.

Deterministic Games

This is the case when the moves in the game are deterministic. 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

$$u_i(\overline{\sigma}) \quad = \quad f(I_i^{(M)}, I_i^{(R)}, I_i^{(S)}, I_i^{(A)}, I_i^{(G)})$$

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:

$$\begin{aligned} u_i(\overline{\sigma}) \quad = \quad & w_{m,i} \sum_t c_m(M_{i,t}) + w_{r,i} \sum_t c_r(R_{i,t}) \\ & + w_{s,i} \sum_t c_s(S_{i,t}) + w_{a,i} \sum_t c_a(A_{i,t}) + w_{g,i} \sum_{j \in G} g_{i,j} \end{aligned}$$

where $w_{m,i}, w_{r,i}, w_{r,i}, w_{s,i}$ 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:

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

This basically points out that in general a participant may face a constrained optimization problem for designing the best strategy to deal with a privacy-preserving distributed data mining process.

Non-Deterministic(Probabilistic) Games

In this case the strategies in the game are stochastic and are chosen at every step of the game based on the current state of the player, information retained about previous moves and a random function. Therefore the utility value is a probability distribution defined by the random variables corresponding to the different dimensions. Using $p_{i,t}^{(m)}$ as the probability of node i correctly performing local computations at stage t of the game (and similarly for communication and collusion), we can denote the expectation of the utility function as follows:

$$\begin{aligned} E[u_i(\overline{\sigma})] &= w_{m,i} \sum_t p_{i,t}^{(m)} c(M_{i,t}) + w_{r,i} \sum_t p_{i,t}^{(r)} c(R_{i,t}) \\ &\quad + w_{s,i} \sum_t p_{i,t}^{(s)} c(S_{i,t}) + w_{g,i} \sum_{j \in G} p_{i,j}^{(g)} g_{i,j} \end{aligned}$$

Now that we have formulated a generalized privacy preserving distributed data mining problem as a multi-player game, we would proceed to 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.

6 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 $0, 1, \ldots, N-1$. The basic idea of secure sum is as follows. Assuming sites do not collude, site 1 generates a random number R uniformly distributed in the range $[0, N-1]$, which is independent of its local value v_1. Then site 1 adds R to its local value v_1 and transmits $(R + v_1) \bmod N$ to site 2. In general, for $i = 2, \ldots, n$, site i performs the following operation: receive a value z_{i-1} from previous site $i-1$, add it to its own local value v_i and compute its modulus N. In other words,

$$z_i = (z_{i-1} + v_i) \bmod N = (R + \sum_{j=1}^{i} v_j) \bmod N,$$

where z_i is the perturbed version of local value v_i to be sent to the next site $i+1$. Site n performs the same step and sends the result z_n to site 1. Then site 1, which knows R, can subtract R from z_n to obtain the actual sum. This sum is further broadcasted to all other sites.

6.1 Utility of Computation and Communication Strategies

The secure sum computation algorithm expects each party to perform some local computation. This involves generating a random number (for the initiator only), one addition, and one modulo operation. The site may or may not choose to perform this computation. This choice will define the strategy of a node for computation.

The secure sum computation algorithm also expects a party to receive a value from its neighbor and send out the modified value after the local computation. This party may or may not choose to do so. This will define the strategy for communication.

The utility of computation and communication strategies may depend upon the following factors:

1. Computational cost of performing the local operations on the data.
2. Communication cost for receiving and sending data.
3. Impact of sending wrong or misleading information on the overall outcome of the PPDM computation. For example, if a party injects a randomly generated number to its neighbor as if it is the true outcome of the local computation then what is the effect on the overall accuracy of the secure sum?

One may construct different utility functions based on such parameters.

6.2 Utility of Launching a Privacy-Attack

This section explores the possibility of launching a privacy-breaching attack on secure sum by a party trying to extract the private information of other parties. This section shares some analytical results and points out that this may be a futile exercise. We present the analysis here just to illustrate the approach and the methodology since similar techniques will be used for analyzing the privacy-breaching properties of other existing PPDM (including secure multi-party computation) techniques.

For simplicity, let us consider the situation at site 2. A similar analysis can be done at each of the other sites. Let V be the value at site 1, R be the random number uniformly distributed over $[0, N-1]$, independent of V, where $N-1$ is the maximum possible value of the sum. We consider V (and R) as a random variable taking values in $\{0, 1, \ldots, N-1\}$. Let f_V, f_R be the probability mass functions of V and R, respectively. Note that f_V and f_R are zero outside range $[0, N-1]$ and $f_R(r) = 1/N$ for $r = 0, 1, \ldots, N-1$. Define random variables W and Z as follows:

$$W = V + R, \qquad Z = W \bmod N,$$

where W is an additively perturbed version of data V, Z (the modulus of W) is the result to be sent to site 2. Now we will investigate the relation between Z and V to find whether it is possible to recover V from Z. Notice that W can take values in $\{0, 1, \ldots, 2N-2\}$, and Z can take values in $\{0, 1, \ldots, N-1\}$. Since V and R are independent, the probability mass function f_W of W can be written as the convolution of f_V and f_R:

$$\begin{aligned} f_W(m) &= f_V(m) * f_R(m) = \sum_{s=0}^{N-1} f_V(s) f_R(m-s) \\ &= \begin{cases} \frac{1}{N} \sum_{s=0}^{m} f_V(s) & m = 0, 1, \ldots, N-1; \\ \frac{1}{N} \sum_{s=m+1-N}^{N-1} f_V(s) & m = N, \ldots, 2N-2. \end{cases} \end{aligned}$$

Note that modulus operation moves values in $[N, 2N-2]$ back into the interval $[0, N-1]$. Therefore the probability mass function f_Z of Z can be obtained from that of W as follows:

$$
\begin{aligned}
f_Z(m) &= f_W(m) + f_W(m+N) \\
&= \frac{1}{N}\sum_{s=0}^{m} f_V(s) + \frac{1}{N}\sum_{s=m+1}^{N-1} f_V(s) \\
&= \frac{1}{N}\sum_{s=0}^{N-1} f_V(s) = \frac{1}{N}, \;\; m = 0, 1, \ldots, N-1.
\end{aligned}
$$

It is clear that no matter what distribution f_V the data V has, Z is always uniformly distributed in the range $\{0, 1, \ldots, N-1\}$. In order to find the relation between W and V and the relation between Z and V, let us compute their joint probability mass functions. First we compute the joint probability mass function of W and V:

$$
\begin{aligned}
f_{WV}(w, v) &= P\{W = w, V = v\} = P\{R = w - v, V = v\} \\
&= P\{R = w - v\}P\{V = v\} \\
&= \begin{cases} \frac{1}{N} f_V(v) \; 0 \le v \le w \le N-1; \\ \frac{1}{N} f_V(v) \; N \le w \le 2N-2, w+1-N \le v \le N-1. \end{cases}
\end{aligned}
$$

Note in particular that $f_{WV}(w, v) \neq f_W(w) f_V(v)$ and hence random variables W and V are not independent, which means that there is a statistical relationship between them. This can be used to estimate V from W (perhaps using the conditional expectation $E[V \mid W]$ of V given W). Our proposed spectral filtering technique [13] can indeed be an approach in that direction. However, can we say the same thing about Z and V? Probably not. We show in the following that V and Z are statistically independent, which makes it a different scenario altogether. Note that both Z and V can take values in $\{0, 1, 2, \ldots, N-1\}$. The joint probability mass function of Z, V is:

$$
\begin{aligned}
f_{ZV}(z, v) &= P\{Z = z, V = v\} \\
&= P\{(W = z \text{ or } W = N + z), V = v\} \\
&= P\{R = z - v, V = v\} \\
&\quad + P\{R = N + z - v, V = v\} \\
&= \begin{cases} \frac{1}{N} f_V(v) + 0 & 0 \le v \le z \le N-1; \\ 0 + \frac{1}{N} f_V(v) & 0 \le z \le N-2, z+1 \le v \le N-1. \end{cases} \\
&= \frac{1}{N} f_V(v), 0 \le z, v \le N-1.
\end{aligned}
$$

Therefore, $f_{ZV}(z, v) = f_Z(z)f_V(v)$. This shows that Z and V are statically independent, and hence, Z does not contain any information about V. In other words, a privacy-breaching attack on secure sum by a single party may not be very successful. The following section considers the effect of collusion and presents a sample analysis for secure sum computation.

6.3 Utility of Collusion

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.

We have the following possible privacy breaches:

- 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 explanations. Now let us investigate the third case. Without loss of generality, we can arrange the sites in the following order:

$$\underbrace{v_1 v_2 \ldots v_{n-k-1}}_{\text{honest sites}} v_i \quad \underbrace{v_{i+1} \ldots v_{i+k}}_{\text{colluding sites}}$$

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 6.1 *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 [22] 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 6.2 *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: The probability generating function of Λ is

$$G_\Lambda(z) = E[z^\Lambda] = \frac{1}{m+1}(z^0 + z^1 + \cdots + z^m).$$

Therefore, the probability generating function of Θ is

$$\begin{aligned} G_\Theta(z) = (G_\Lambda(z))^s &= \frac{(z^0 + z^1 + \cdots + z^m)^s}{(m+1)^s} \\ &= \frac{(1-z^{m+1})^s}{(1-z)^s(m+1)^s}. \end{aligned}$$

The probability mass function (PMF) of Θ is computed by taking derivatives of $G_\Theta(z)$:

$$Pr\{\Theta = \theta\} = \left.\frac{G_\Theta^{(\theta)}(z)}{\theta!}\right|_{z=0,}$$

where $G_\Theta^{(\theta)}(z)$ is the θ-th derivative of $G_\Theta(z)$.

In practice, it is probably not easy to compute $G_\Theta^{(\theta)}(z)$. Instead, we can expand $G_\Theta(z)$ into a polynomial function of degree ms. The coefficient of each term z^t, $t = 0, \ldots, ms$ in the expanded polynomial gives the probability that $\Theta = t$.

To expand $G_\Theta(z)$, let us first leave out the factor $\frac{1}{(m+1)^s}$. Newton's generalized binomial theorem tells us that $\frac{1}{(1-z)^s} = \sum_{t=0}^{\infty} C_{s+t-1}^t z^t$. Hence,

$$\frac{(1-z^{m+1})^s}{(1-z)^s} = \left(\sum_{j=0}^{s} C_s^j z^{(m+1)j}(-1)^j\right)\left(\sum_{t=0}^{\infty} C_{s+t-1}^t z^t\right).$$

The above equation can be written as follows:

$$\begin{aligned} \frac{(1-z^{m+1})^s}{(1-z)^s} &= \sum_{t=0}^{\infty} C_{s+t-1}^t z^t - C_s^1 \sum_{t=0}^{\infty} C_{s+t-1}^t z^{(m+1)+t} \\ &\quad + C_s^2 \sum_{t=0}^{\infty} C_{s+t-1}^t z^{2(m+1)+t} - \ldots \end{aligned}$$

Therefore, the coefficients of the above polynomial have the following properties: for $t = 0, 1, \ldots, m$, we have

- the coefficient of z^t is C_{s+t-1}^t,
- the coefficient of $z^{(m+1)+t}$ is $C_{s+(m+1)+t-1}^{(m+1)+t} - C_s^1 C_{s+t-1}^t$,

- the coefficient of $z^{2(m+1)+t}$ is $C_{s+2(m+1)+t-1}^{2(m+1)+t}$ $-C_s^1 C_{s+(m+1)+t-1}^{m+1+t} + C_s^2 C_{s+t-1}^t$,
- etc.

In general, for $t = 0, 1, \ldots, m$ and $r = 0, 1, \ldots$, the coefficient of $z^{r(m+1)+t}$ is

$$\sum_{j=0}^{r} (-1)^j C_s^j C_{s+(r-j)(m+1)+t-1}^{(r-j)(m+1)+t}.$$

Given the above results, the probability mass function (PMF) of Θ is:

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

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

According to Theorem 6.2, the probability mass function (PMF) of X is

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

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

Combing Eq. 1, 2 and 3, we get the posterior probability of v_i:

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

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 privacy breach:

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

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 privacy-breach for defining the objective function.

6.4 Overall Objective Function

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

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

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 where the utilities due to communication and computation are constant and hence can be neglected for determining the nature of the function.

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

Figure 1 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.

7 What is the Solution?

In the previous section we pointed out that traditional no-collusion assumption is unlikely to hold true in a realistic multi-party distributed environment for computing a sequence of secure sum operations. Our goal is to design a technique for

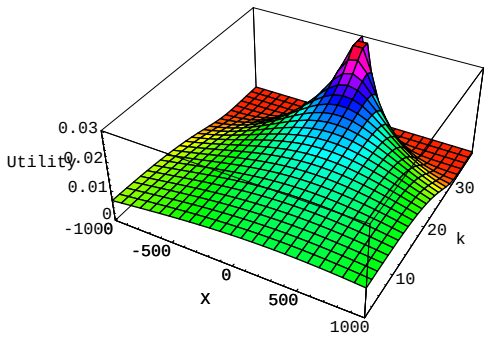

Figure 1: Plot of the overall objective function for SMC. The optimal strategy takes a value of $k > 1$.

multi-party secure sum computation that does not rely on unrealistic assumptions about the nodes in the network. So we want an algorithm that is privacy preserving sum computation, which means that we want a set of strategies that offer optimal utility at $k = 1$. In other words, 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 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. If party A wants to collude with another party B, then the former needs to contact the latter party and get its consent. If party A contacts party B, but the latter does not agree to collude then party A may become vulnerable to a penalty. Either party B directly decide to penalize party A for its inappropriate behavior and/or inform others (other participating parties or a central authority) about the intention of party A. Therefore, any party must be concerned about a penalty before contacting anyone else for a partner in collusion.

7.1 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

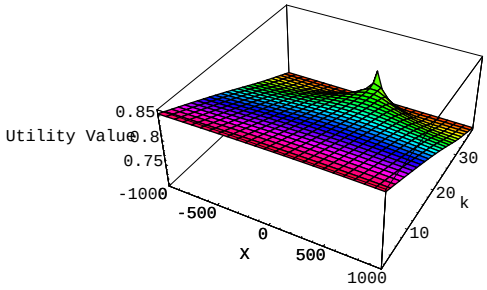

Figure 2: Plot of the modified objective function with penalty. The globally optimal strategies are all for $k = 1$.

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 shows the penalty due to excess computation and communication as a result of collusion.

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

Figure 2 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 Nash equilibrium to the case where $k = 1$.

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

7.2 Nash Equilibrium Illustration for 3-node Network

The idea described in the last section can be further explained using a simple example, illustrated in Table 1. Consider a game with a mediator where each party first contacts the mediator and declares their intention to be a good node (follow protocol) or bad party (willing to collude). When there is no penalty for misbehavior, everyone will benefit by colluding with others. This will result in all bad nodes colluding with each other in order to violate the privacy of the good nodes. Now, let us consider the scenarios where the mediator will penalize using either Policy I or Policy II. The mediator can enforce Policy I since everyone reports their intentions to the mediator. It can also easily enforce Policy II by simply counting k, the total number of bad parties.

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 [11]). Also we observe that the payoff for a 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 node (the initiator) who wants to protect the privacy of its data.

Table 1 also shows the pay-offs for both policies I and II. In both cases, 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 bu 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).

The following section presents some practical ways to implement Policy II for asynchronous environments.

7.3 How to Implement?

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. In this section we explore some practical ways of implementing these policies in a real life distributed setting.

7.3.1 Centralized Control

In this scheme there is a central authority who is always in charge of implementing the penalty policy. In practice, it can be a variant of the mechanism used in the example described in the previous section. Unlike the 3-node example in the last section, in this mechanism not everyone registers their intentions with a central node. Instead, nodes approach other nodes with proposals to collude. If a good node receives a proposal from a bad node then it reports to the central authority which in turn penalizes the perpetrator. This scheme is relatively easy to implement. However, it requires global synchronization. Since the penalty affects the total number of iterations needed for completing the secure sum computation, the process cannot start until the central authority makes a decision about the bad nodes. This requires that all the nodes synchronize with this deciding node. Such

global synchronization may create a bottleneck and limit the scalability of a distributed system.

7.3.2 Asynchronous Distributed Control

Fortunately, in games like this whenever there is a solution with a mediator, there is also a solution without one. Ben-Porath [6] showed that it is possible to simulate Nash equilibrium without a mediator as long as there is a proper strategy to penalize lack of compliance. For a total asynchronous distributed control we borrow the concept of $cheaptalk$ from game theory and economics [10] in order to develop a distributed mechanism for penalizing policy violations. Cheap talk is simply 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. For example, in the well known Prisoner's Dilemma game [18] one might add a round of pre-play communication where each player announces the action they intend to take. Although cheap talk may not effect the outcome of Prisoner's Dilemma, in many other games the outcome may be significantly influenced by such pre-play communication. 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. Either Policy I or II can be used for the penalty threat. Algorithm 1 (Secure Sum with Penalty (SSP)) describes a variant of the secure sum computation technique that offers a distributed mechanism for penalizing policy violations using a cheap talk-like mechanism.

Consider a network of n nodes where a node can either be *good* or *bad*. *Bad* nodes collude to reveal other nodes' information; while *good* nodes follow the correct secure sum protocol. Before the secure sum protocol starts, the colluding (*bad*) nodes send invitations for collusions randomly to nodes in the network. If such a message is received by a *good* node, then it knows that there are colluding nodes in the network. To penalize nodes that collude, this *good* node splits its local data into k' random shares where k' is an estimate of the size of the largest colluding groups. One possible way to estimate this could be based on the number of collusion invitations a good node receives. On the other side, the *bad* nodes, on receiving such invitation messages, form a fully connected networks of colluding groups. This initial phase of communication is cheap talk in our algorithm. After this the secure sum protocol starts. As in the traditional secure sum protocol, nodes forward their own data after doing the modulus operation and random number addition. However, good nodes do not send all the data at one go; rather they

send one random share at each round of the secure sum. Hence, it takes several rounds for the secure sum to complete. The total number of rounds needed for one complete secure sum computation is equal to $Max(k')$ where k' is the estimate of k (the size of the colluding group) that every node has.

The following Lemma (Lemma 7.1) shows that the SSP algorithm converges to the correct result.

Lemma 7.1 ***Correctness and Convergence:*** *SSP protocol 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 protocol, 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 an 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 protocol, the total rounds of computation is k, therefore the overall time required is bounded by $O(nk)$. ∎

8 Experimental Results

We have implemented the cheap talk-based secure sum with penalty protocol and study the nature of the Nash equilibrium. This experiment assumes that the nodes are rational in the sense that they choose actions that maximize their utility function, which in other words would mean that they minimize their cost.

8.1 Overview of the Simulation Set-Up

We set up a simulation environment comprised of a network of n nodes where a node can either be *good* or *bad*. We have experimented with a ring topology of 500 nodes (*n*=500) using the use the Distributed Data Mining Toolkit (DDMT) [2] developed by the DIADIC research lab at UMBC. Our game consisted of both

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

good and *bad* nodes where *good* nodes split the data into shares to thwart the possibility of a privacy breach by the *bad* nodes. We have assumed that each node has an integer vector of size l and it wants to compute the vector sum using the secure sum protocol for each of these integers. Therefore, there are going to be l rounds of secure sum computation. For calculating the vector sum we have used individual rounds of the secure sum protocol. As mentioned earlier, in order to penalize the *bad* nodes, *good* nodes split each of these data into random non-zero shares. Therefore one round of the secure sum (sum computation for one of the l vector elements) consists several iterations where in every iteration only one of the k' shares of every node is added to the secure sum. After every round of the secure sum protocol (that is after one of the l sum computations), we measure three quantities:

- Messages sent : For *good* nodes, number of messages is equal to the number of iterations needed for one round of secure sum computation. For *bad* nodes the number of messages sent is higher because of the additional messages exchanged for colluding.
- Computational power used : In this case also the *good* nodes only require computation necessary for the standard secure sum protocol. However, the *bad* nodes need additional computation for extracting other nodes values by colluding with partners. We consider every mathematical operation (in this case addition and modulus operation) as a unit of computation.
- Utility of collusion: This is the posterior probability that a node will guess the correct distribution of the value a node has given the total sum of the collusion. This quantity measures zero for the *good* nodes and varies among the different *bad* nodes depending on the size of the collusion that particular node is part of and the distribution of the values at the different nodes.

8.2 Results

Initially we start with a fixed percentage (30%) of the nodes to be *bad*. After every round each node measures the cost (or penalty) it incurred due to collusion. If the penalty sustained is too high (a dynamic threshold currently set by the user), some of the bad nodes decide not to collude again. Once these *bad* nodes turn into *good* ones, they send deallocate messages to their colluding groups and also set their estimates of collusion size k' same as the size of the collusion to which they belonged.

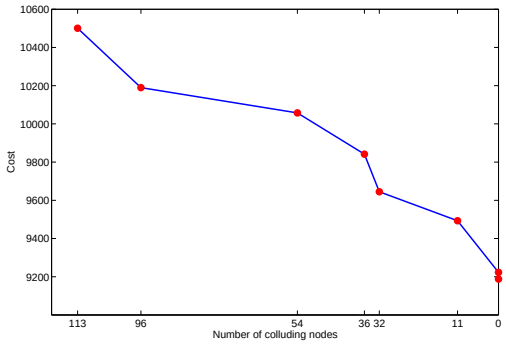

Figure 3: Variation of cost with changes in number of colluding nodes.

We observe in Figure 3 that for subsequent rounds of the secure sum computation the cost or overall penalty assigned decreases as the number of *bad* nodes decreases. When the ratio of *bad* to *good* nodes is significantly low, we can observe that the cost almost reaches an equilibrium. This is because the contribution of the penalty function becomes negligible and the total cost is governed mainly by the computation and communication costs that remain almost constant over successive rounds of secure sum with hardly any collusion.

In Figure 4 we have shown how the number of *bad* nodes decrease with successive rounds of secure sum computation. The *bad* nodes in the network start any round of secure sum with the intention to collude. However, some of them do not end up in any collusion since their invitations for collusion are not reciprocated by the *good* nodes. So at any round if b denotes the number of *bad* nodes (nodes with intentions to collude), the actual number of colluding nodes k is less than or equal to b. The plot with circular markers demonstrate the decreasing values of b in consecutive rounds of secure sum whereas the one with square-shaped markers presents the decreasing values of k. In either case, we see that as b or k decreases, the rate of their convergence to zero gradually falls due to the significantly low ratio of *good* to *bad* nodes in the network. The third plot in Figure 4 represents the decrease in the number of *bad* nodes in a network with an initial count of 60% bad nodes. We observe that even if the number of *bad* nodes in the network be double, the algorithm still converges to the same state where the number of colluding nodes in the network tend to zero.

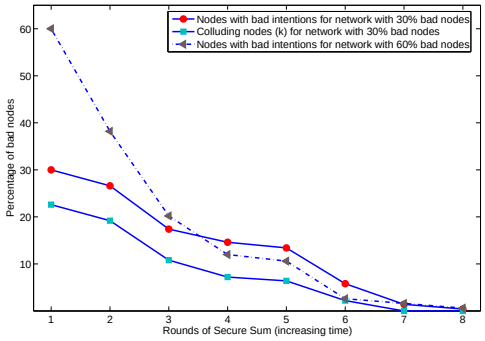

Figure 4: Change in the number of bad nodes in the network over time. It shows that both the number of nodes interested in violating the policy and the number of nodes that form collusions decrease to zero due to the penalty scheme.

9 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. 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 argued that most of these nice assumptions fall apart in real-life applications of privacy-preserving distributed data mining. The paper offered a more realistic formulation of the PPDM problem as a multi-party game where each party tries to maximize its own objective or utility.

The paper considered the multi-party secure sum computation problem for illustrating the game theoretic formulation. It pointed out that the expected strategy of every party in a traditional secure sum algorithm does not correspond to a Nash equilibrium state. The paper analyzed a variant of this algorithm which assign penalty to the policy-violators in a distributed manner. Experimental results point out that the equilibrium behavior of the algorithm corresponds to the desired no-collusion strategy.

The paper opens up many new possibilities. It offers a new approach to study the behavior of existing PPDM algorithms and invent new ones. We plan to consider other popular PPDM algorithms for computing inner product, clustering, and association rule learning in order to study those using the game theoretic framework developed here.

References

[1] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, Denver, Colorado, USA, July 2006.

[2] Charu C. Aggarwal and Philip S. Yu. A condensation based approach to privacy preserving data mining. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT'04)*, pages 183–199, Heraklion, Crete, Greece, March 2004.

[3] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, TX, May 2000.

[4] M. J. Atallah, E. Bertino, A. K. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the IEEE Knowledge and Data Engineering Workshop*, pages 45–52, 1999.

[5] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 217–228, Tokyo, Japan, April 2005.

[6] E. Ben-Porath. Cheap talk in games with incomplete information. *Journal of Economic Theory*, 108(1):45–71, 2003.

[7] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 589–592, Houston, TX, November 2005.

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2), 2003.

[9] A. Evfimevski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the ACM SIGMOD/PODS Conference*, San Diego, CA, June 2003.

[10] J. Farrell and M. Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996.

[11] Joseph Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: extended abstract. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 623 – 632, Chicago, IL, USA, 2004.

[12] R. Hardin. Collective action as an agreeable n-prisoners' dilemma. *Journal of Behavioral Science*, 16:472–481, September 1971.

[13] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining*, Melbourne, FL, November 2003.

[14] M. Kearns and L. Ortiz. Algorithms for interdependent security games. *Advances in Neural Information Processing Systems*, 2004.

[15] Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2-3):231–249, 2003.

[16] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(1):92–106, January 2006.

[17] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of the USA*, 36(1):48–49, 1950.

[18] M. Osborne. *Game Theory*. Oxford University Press, 2004.

[19] B. Pinkas. Cryptographic techniques for privacy preserving data mining. *SIGKDD Explorations*, 4(2):12–19, 2002.

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

[21] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[22] 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.

[23] A. C. Yao. How to generate and exchange secrets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[24] Nan Zhang, Wei Zhao, and Jianer Chen. Performance measurements for privacy preserving data mining. In *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005*, pages 43–49, Hanoi, Vietnam, May 2005.

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