Multi-Party, Privacy-Preserving Distributed Data Mining using a Game Theoretic Framework

Hillol Kargupta[1,3], Kamalika Das[1], and Kun Liu[2]
{hillol, kdas1}@cs.umbc.edu, kun@us.ibm.com

[1] University of Maryland, Baltimore County, Baltimore MD 21250, USA
[2] IBM Almaden Research Center, San Jose CA 95120, USA
[3] Agnik, LLC, 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's sensitive data. This paper argues that most of these assumptions fall apart in real-life applications of privacy-preserving distributed data mining (PPDM). This 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 to analyze the behavior of each party in such games and presents detailed analysis of the well known secure sum computation as an example.

1 Introduction

Advanced analysis of privacy-sensitive data plays an important role in many multi-party, cross-domain applications. For example, the US Department of Homeland Security-funded PURSUIT project[4] involves analysis of network traffic data from different organizations. Network traffic is usually privacy sensitive and no organization would be willing to share their information with a third party. PPDM offers one possible solution which would allow comparing and matching multi-party network traffic to detect common attacks and compute various statistics for a group of organizations that are not willing to share the raw data. However, many of the existing PPDM algorithms make strong assumptions about the behavior of the participants, *e.g.*, they are semi-honest and not colluding with others. Unfortunately, participants of a real application like PURSUIT may not all be ideal. Some may try to exploit the benefit of the system without contributing much; some may try to sabotage the computation; and some may try to collude with other parties for exposing the private data.

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 as a game where each participant tries to maximize its benefit

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

by optimally choosing the strategies during the entire PPDM process. Applications of game theory in secure multi-party computation and privacy preserving distributed data mining is relatively new [1, 4, 2]. This paper develops a game-theoretic framework for analyzing the rational behavior of each party in such a game, and presents detailed equilibrium analysis of the well known secure sum computation [7, 3] as an example. A new version of the secure sum is proposed. It works based on well known concepts from game theory and economics such as "cheap talk" and mechanism design. This paper also describes experiments on large scale distributed games and illustrates the validity of the formulations.

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

2 Multi-Party PPDM As Games

A game is an interaction or a series of interactions between players, which assumes that 1) the players pursue well defined objectives (they are *rational*) and 2) they take into account their knowledge or expectations of other players' behavior (they *reason strategically*). For simplicity, we start by considering the most basic game - the *strategic game*.

Definition 1 (Strategic Game). *A strategic game consists of*

- *a finite set P: the set of players,*
- *for each player $i \in P$ a nonempty set A_i: the set of actions available to player i,*
- *for each player $i \in P$ a preference relation $\succeq_i$ on $A = \times_{j \in P} A_j$: the preference relation of player i.*

The preference relation $\succeq_i$ of player i can be specified by a utility function $u_i : A \to \mathbb{R}$ (also called a payoff function), in the sense that for any $a \in A, b \in A$, $u_i(a) \geq u_i(b)$ whenever $a \succeq_i b$. The values of such a function is usually referred to as utilities (or payoffs). Here a or b is called the *action profile*, which consists of a set of actions, one for each player. Therefore, the utility (or payoff) of player i depends not only on the action chosen by herself, but also the actions chosen by all the other players. Mathematically, for any action profile $a \in A$, let a_i be the action chosen by player i and a_{-i} be the list of actions chosen by all the other players except i, the utility of player i is $u_i(a) = u_i(\{a_i, a_{-i}\})$.

One of the fundamental concepts in game theory is the Nash equilibrium:

Definition 2 (Nash Equilibrium). *A Nash equilibrium of a strategic game is an action profile $a^* \in A$ such that for every player $i \in P$ we have*

$$u_i(\{a_i^*, a_{-i}^*\}) \geq u_i(\{a_i, a_{-i}^*\}) \text{ for all } a_i \in A_i.$$

Therefore, Nash equilibrium defines a set of actions (an action profile) that captures a steady state of the game in which no player can do better by unilaterally changing her action (while all other players do not change their actions).

When the game involves a sequence of interactive actions of the players, and each player can consider her plan of action whenever she has to make a decision, the *strategic game* becomes an *extensive game*. In that situation, the *action* a_i for player i, is replaced by σ_i, the *strategy* for that player, which is a complete algorithm for playing the game, implicitly including all actions of that player for every possible situation throughout the game. The utility function also assigns a payoff to player i for each joint strategies of all the players, *i.e.*, $u_i(\{\sigma_i, \sigma_{-i}\})$.

Armed with the basic knowledge of game theory, we are now ready to formulate multi-party PPDM as a game. In a multi-party PPDM environment, each node has certain responsibilities in terms of performing their part of the computations, communicating correct values to other nodes and protecting the privacy of the data. Depending on the characteristics of these nodes and their objectives, they either perform their duties or not, sometimes, they even collude with others to modify the protocol and reveal others' private information. Let M_i denote the overall sequence of computations node i has performed, which may or may not be the same as what it is supposed to do defined by the PPDM protocol. Similarly, let R_i be the messages node i has received, and S_i the messages it has sent. Let G_i be a subgroup of the nodes that would collude with node i. The strategy of each node in the multi-party PPDM game prescribes the actions for such computations, communications, and collusions with other nodes, *i.e.*, $\sigma_i = (M_i, R_i, S_i, G_i)$. Further let $c_{i,m}(M_i)$ be the utility of performing M_i, and similarly we can define $c_{i,r}(R_i)$, $c_{i,s}(S_i)$ and $c_{i,g}(G_i)$. Then the overall utility of node i will be a linear or nonlinear function of utilities obtained by the choice of strategies in the respective dimensions of computation, communication and collusion. Without loss of generality, we consider an utility function which is a weighted linear combination of all of the above dimensions:

$$u_i(\{\sigma_i, \sigma_{-i}\}) = w_{i,m}c_{i,m}(M_i) + w_{i,r}c_{i,r}(R_i) + w_{i,s}c_{i,s}(S_i) + w_{i,g}c_{i,g}(G_i),$$

where $w_{i,m}, w_{i,r}, w_{i,s}, w_{i,g}$ represent the weights for the corresponding utility factors. Note that we omitted other nodes' strategies in the above expression just for simplicity. In the next section, we would illustrate our formalizations using one of the most popular PPDM algorithms, the secure sum computation.

3 Case Study: Multi-Party Secure Sum Computation

Secure sum computation [7, 3] computes the sum of n different nodes without disclosing the local value of any node. It has been widely used in privacy preserving distributed data mining as an important primitive, *e.g.*, privacy preserving association rule mining on horizontally partitioned data [5], k-means clustering over vertically partitioned data [8] and many else.

Secure Sum Computation Suppose there are n individual nodes organized in a ring topology, 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 $[0, N-1]$.

The basic idea of secure sum is as follows. Assuming nodes do not collude, node 1 generates a random number R uniformly distributed in the range $[0, N-1]$, which is independent of its local value v_1. Then node 1 adds R to its local value v_1 and transmits $(R+v_1)$ mod N to node 2. In general, for $i = 2, \ldots, n$, node i performs the following operation: receive a value z_{i-1} from previous node $i-1$, 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 node $i+1$. Node n performs the same step and sends the result z_n to node 1. Then node 1, which knows R, can subtract R from z_n to obtain the actual sum. This sum is further broadcasted to all other sites.

Collusion Analysis It can be shown that [6] any z_i has an uniform distribution over the interval $[0, N-1]$ due to the modulus operation. Further, any z_i and v_i are statistically independent, and hence, a single malicious node may not be able to launch a successful privacy-breaching attack. Then how about collusion?

Let us assume that there are k ($k \geq 2$) nodes acting together secretly to achieve a fraudulent purpose. Let v_i be an honest node who is worried about her privacy. We also use v_i to denote the value in that node. Let v_{i-1} be the immediate predecessor of v_i and v_{i+1} be the immediate successor of v_i. The possible collusion 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 nodes 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 nodes contain neither v_{i-1} nor v_{i+1}, or only one of them, then v_i is disguised by $n-k-1$ other nodes' values.

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

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

where W is a constant and is known to all the colluding nodes. Now, it is clear that the colluding nodes will know v_i is not greater than W, which is some extra information contributing to the utility of the collusions. To take a further look, the colluding nodes can compute the posteriori probability of v_i and further use that to launch a maximum a posteriori probability (MAP) estimate-based attack. It can be shown that, this posteriori probability is:

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

where $v_i \leq W$, $r = \lfloor \frac{W-v_i}{m+1} \rfloor$ and $t = W - v_i - \lfloor \frac{W-v_i}{m+1} \rfloor (m+1)$. When $v_i > W$, $f_{posterior}(v_i) = 0$. Due to space constraints, we have not included the proof of this result here. Interested readers can find a detailed proof in [6].

Note that, when computing this posteriori probability, we model the colluding nodes' belief of each unknown v_j $(j = 1, \ldots, n-k-1)$ as a uniform distribution over an interval $\{0, 1, \ldots, m\}$. This assumption has its roots in the principle of maximum entropy, which models all that is known and assumes nothing about what is unknown, in that case, the only reasonable distribution would be uniform.

Overall Utilities The derived posteriori probability can be used to quantify the utility of collusion, *e.g.*, $g(v_i) = Posteriori - Prior = f_{posterior}(v_i) - \frac{1}{N}$. We see here that this utility depends on $W - v_i$ and the size of the colluding group k. Now we can put together the overall utility function for the game of multi-party secure sum computation:

$$u_i(\{\sigma_i, \sigma_{-i}\}) = w_{i,m} c_{i,m}(M_i) + w_{i,r} c_{i,r}(R_i) + w_{i,s} c_{i,s}(S_i) + w_{i,g} \sum_{j \in P - G_i} g(v_j),$$

where P is the set of all nodes and G_i is the set of nodes colluding with node i.

Let us now consider a special instance of the overall utility where the node performs all the communication and computation related activities as required by the protocol. This results in a function: $u_i(\{\sigma_i, \sigma_{-i}\}) = w_{i,g} \sum_{j \in P - G_i} g(v_j)$, where the utilities due to communication and computation are constant and hence can be neglected for determining the nature of the function. Figure 1(Left) shows a plot of the overall utility of multi-party secure sum as a function of the distribution of the random variable $W - v_i$ 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, the optimal solution corresponds to the Nash equilibrium. This implies that in a realistic scenario for multi-party secure sum computation, nodes will have a tendency to collude. Therefore the non-collusion $(k = 1)$ assumption of the classical secure multi-party sum is sub-optimal. Next section describes a new mechanism that leads to an equilibrium state corresponding to no collusion.

4 Achieving Nash Equilibrium with No-colluding Nodes

To achieve a Nash equilibrium with no collusions, the game players can adopt a punishment strategy to threaten potential deviators. One may design a mechanism to penalize colluding nodes in various ways:

1. Policy I: Remove the node from the application environment because of protocol violation. Although it may work in some cases, the penalty may be too harsh since usually the goal is to have everyone participate in the process and faithfully contribute to the data mining process.
2. Policy II: Penalize by increasing the cost of computation and communication. For example, if a node suspects a colluding group of size k' (an estimate of k), then it may split the every number used in a secure sum among $\alpha k'$ different parts and demand $\alpha k'$ rounds of secure sum computation one for each

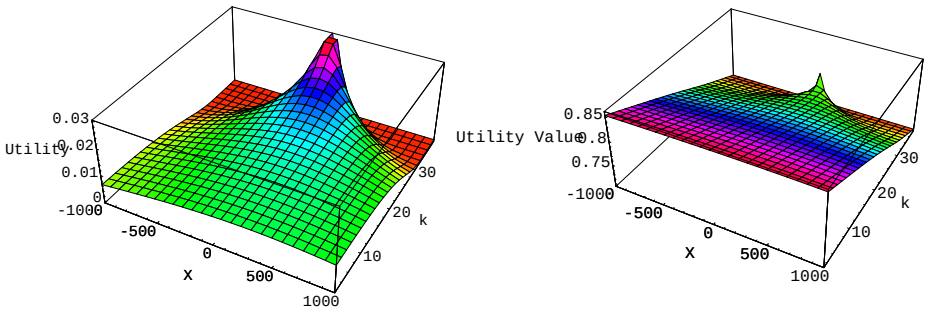

Fig. 1. Overall utility for classical secure sum computation (Left) and secure sum computation with punishment strategy (Right). The optimal strategy takes a value of $k > 1$ in the first case and $k = 1$ in the second case.

of these $\alpha k'$ parts, here $\alpha > 0$ is a constant factor. This increases the computation and communication cost by $\alpha k'$-fold. This linear increase in cost with respect to k', the suspected size of colluding group, may be used to counteract possible benefit that one may receive by joining a team of colluders. The modified utility function is given by $\tilde{u}_i(\{\sigma_i, \sigma_{-i}\}) = u_i(\{\sigma_i, \sigma_{-i}\}) - w_{i,p} * \alpha k'$. The last term in the equation accounts for the penalty due to excess computation and communication as a result of collusion.

Figure 1(Right) shows a plot of the modified utility 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 $k = 1$.

As a toy example, consider a three-party secure sum computation with the payoff listed in Table 1. When there is no penalty, all the scenarios with two bad nodes and one good node offer the highest payoff for the colluding bad nodes. So the Nash equilibrium in the classical secure sum computation is the scenario where the participating nodes are likely to collude. However, in both cases with penalty, no node can gain anything better by deviating from good to bad when all others remain good. Therefore, the equilibrium corresponds to the strategy where none of the nodes 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.

Implementing the Penalty Mechanism without Having to Detect Collusion: In order to implement the penalty protocol, one may use a central mediator who can monitor the behavior of all nodes (see, *e.g.*, [2]). However, this is usually very difficult, if not impossible in a real application environment. Moreover, it requires global synchronization which might create a bottleneck in the distributed system. Instead, we borrow the concept of *cheap talk*, a pre-play communication concept from game theory, to realize an asynchronous distributed control. The idea is based on the assumption that collusion requires consent from multiple parties. So a party with intention of collusion might get caught

A	B	C	Payoff (No Penalty)	Payoff (Policy I)	Payoff (Policy II)
Good	Good	Good	(3, 3, 3)	(3, 3, 3)	(3, 3, 3)
Good	Good	Bad	(3, 3, 3)	(2, 2, 0)	(2, 2, 2)
Good	Bad	Good	(3, 3, 3)	(2, 0, 2)	(2, 2, 2)
Good	Bad	Bad	(3, 4, 4)	(0, 0, 0)	(2, 2, 2)
Bad	Good	Good	(3, 3, 3)	(0, 2, 2)	(2, 2, 2)
Bad	Good	Bad	(4, 3, 4)	(0, 0, 0)	(2, 2, 2)
Bad	Bad	Good	(4, 4, 3)	(0, 0, 0)	(2, 2, 2)
Bad	Bad	Bad	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Table 1. Payoff table for three-party secure sum computation.

while sending out collusion invitation randomly in the network if those invitations reach some honest parties. The new protocol will therefore have a pre-play phase where "lobbying agents" (well-behaved nodes or advocacy groups) will make participants aware of the fact that one will be penalized if any collusion is detected. This "lobbying" does not affect the utility function. It simply makes everyone aware of that. *It does not require a perfect collusion detection.* A real threat with an estimated high-enough value of the collusion-size (k') will do. The threat of a good node introducing penalty using a perceived value of k' will push everyone toward proper behavior.

The new secure sum with penalty (SSP) protocol we proposed is as follows. 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 $k' = 0$ 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 $\alpha k'$ random shares. This initial phase of communication is cheap talk in our algorithm. The secure sum phase consists of $O(\alpha k')$ rounds of communication for every complete sum computation. This process converges to the correct sum in $O(n\alpha k)$ time. Note that, the SSP protocol does not require detecting all the colluding parties. Raising k' based on a perception of collusion will do. If the threat is real, the parties are expected to behave as long they are acting rationally to optimize their utility.

5 Experimental Results

We empirically verify our claim that the SSP protocol leads to an equilibrium state where there is no collusion. The utility function used for the experiments is the one described in Policy II. The penalty in this case is the excess amount of communication and computation needed. In the first experiment we demonstrate for different sizes of the network (500 nodes and 1000 nodes) that the utility is maximum when the collusion is minimum, see Figure 2 (Left). The maximum utility in the figure corresponds to the classical secure sum computation without collusion. The second experiment shows that the number of bad nodes decreases

with successive rounds of SSP, see Figure 2 (Right). Each bad node has a random utility threshold that is assigned during the 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. The time taken to have a no collusion scenario depends on the initial number of bad nodes in the network.

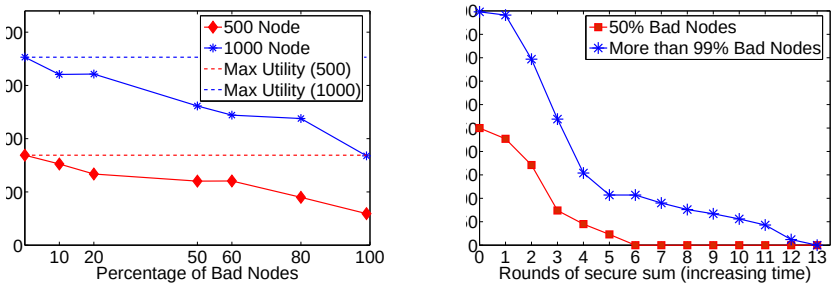

Fig. 2. (Left) Utility vs. Collusion-size. (Right) Rate of decrease of bad nodes.

6 Conclusions

This paper questions some of the common assumptions in multi-party PPDM and shows that if nobody is penalized for cheating, rational participants tends to behave dishonestly. This paper takes a game-theoretic approach to analyze this phenomenon and presents Nash equilibrium analysis of a well-known multi-party secure sum computation. A cheap-talk based protocol to implement a punishment mechanism is proposed to offer a more robust process. The paper illustrates the idea using the secure sum problem as an example. Future work includes theoretical analysis of the existence of Nash equilibrium, as well as the relationship between the amount of penalty and the payoff from collusion.

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 *PODC'06*, pages 53–62, Denver, CO, 2006.
2. Rakesh Agrawal and Evimaria Terzi. On honesty in sovereign information sharing. In *EDBT'06*, pages 240–256, Munich, Germany, March 2006.
3. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2):1–7, 2003.
4. W. Jiang and C. Clifton. Transforming semi-honest protocols to ensure accountability. In *PADM'06*, pages 524–529, Hong Kong, China, 2006.
5. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *DMKD'02*, pages 24–31, June 2002.
6. 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.
7. B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1995.
8. J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *ACM SIGKDD'03*, pages 206–215, Washington, D.C., 2003.