# Peer-to-Peer Data Mining, Privacy Issues, and Games

Kanishka Bhaduri, Kamalika Das and Hillol Kargupta*
kanishk1, kdas1, hillol@{cs.umbc.edu}

University of Maryland, Baltimore County
Baltimore, MD-21250, USA

**Abstract.** Peer-to-Peer (P2P) networks are gaining increasing popularity in many distributed applications such as file-sharing, network storage, web caching, searching and indexing of relevant documents and P2P network-threat analysis. Many of these applications require scalable analysis of data over a P2P network. This paper starts by offering a brief overview of distributed data mining applications and algorithms for P2P environments. Next it discusses some of the privacy concerns with P2P data mining and points out the problems of existing privacy-preserving multi-party data mining techniques. It further points out that most of the nice assumptions of these existing privacy preserving techniques 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 and points out some recent results.

## 1 Introduction

Peer-to-peer (P2P) systems such as Gnutella, Napster, e-Mule, Kazaa, and Freenet are increasingly becoming popular for many applications that go beyond downloading music without paying for it. P2P file sharing, P2P electronic commerce, and P2P monitoring based on a network of sensors are some examples. Novel data integration applications such as P2P web mining from the data stored in the browser cache of different machines connected via a peer-to-peer network may revolutionize the business of Internet search engines. A peer-to-peer clustering algorithm that clusters the URL-s visited by each user (with due privacy-protection) in to different subjects by exchanging information with other peers can be very useful for discovering web-usage patterns of users. This may help characterizing each user based on their browsing pattern, and forming clique of peers having similar interest. Also, this may help routing query about a particular topic to the most appropriate peer in a P2P network. There can be many other similar interesting information integration and knowledge discovery applications involving data distributed in a P2P network.

Privacy is an important issue in many of these P2P data mining applications. Privacy-preserving data mining offers many challenges in this domain. The algorithms must scale up to very large networks and must be asynchronous. Moreover, many of the assumptions (e.g. semihonest, abides by the protocol) that existing privacy-preserving data mining algorithms make may not be valid. We may have some peers trying to

* The author is also affiliated with Agnik LLC, MD, USA

sabotage the computation. This paper presents a high-level overview of an effort to address some of these problem using game theoretic framework for privacy-preserving data mining.

The rest of the paper is organized as follows. In the next section (Section 2), we present the related work on P2P computing and PPDM. A web-mining application that motivates the need for game theoretic PPDM algorithms is presented next in Section 3. The next few sections are devoted on describing the game theoretic approach for PPDM and the preliminary results. We conclude the paper in Section 6.

## 2 Related work

This section presents a very brief related work both on data mining and privacy preserving techniques in P2P networks.

### 2.1 Data Mining in P2P Networks

Knowledge discovery and data mining from P2P network is a relatively new field with little related literature. Some researchers have developed several different approaches for computing basic operations (*e.g.* average, sum, max, random sampling) on P2P networks, *e.g.* Kempe *et al.* [8], Boyd *et al.* [4], Jelasity and Eiben [9] and Bawa et al. [3]. Mehyar *et al.* [15] proposed a new approach for averaging on a P2P network using the Laplacian of a graph.

All the approaches mentioned so far require resources that scale directly with the size of the system. For more scalable approaches, researchers explored the paradigm of *local* algorithms for doing data mining in P2P network. *Local* algorithms [12, 16, 2, 11, 10] are ones in which the result is usually computed using information from just a handful of nearby neighbors. Still, it is possible to make definite claims about the correctness of the result. These algorithms are very scalable as resource requirements are independent of the size of the system and a good fit for P2P networks spanning millions of peers. Lately, simple thresholding based *local* algorithms have been used for complicated data mining tasks in P2P systems: majority voting [19], L2 norm thresholding [18] and possibly more. For a detailed survey interested readers are urged to look into [5].

### 2.2 Privacy Preserving Data Mining

Privacy-preserving data mining can be roughly divided into two groups: data hiding and rule hiding. 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. Techniques like additive perturbation [1], multiplicative perturbation [14], secure multi-party computation [20] all fall into this category. On the other hand, rule hiding tries to transform the database such that the sensitive rules are masked, and all the other underlying patterns can still be discovered [17]. For a detailed review of PPDM and game theory please refer to [7].

## 3 Motivational Application: Peer-to-peer Client-Side Web-usage Mining

Before we address the privacy issues in P2P data mining, we present a motivational application in which preserving privacy of a peer's data is important. We discuss an exciting application of web usage mining using the concepts of client-side web cache and P2P technology based on [13] and emphasize the need for privacy in such mining operation.

Traditional web mining has spent a considerable amount of effort on analyzing the server logs. However, since the results of these analysis are not accessible to the users, the later are deprived of their own generated knowledge which can potentially be used for better searching, routing, forming trust-based communities etc. Dynamically aggregating peers with similar interests could greatly enhance the capability of each individual, could facilitate knowledge sharing, and reduce the network load. In order to solve this problem, we present a framework where the users themselves can form implicit communities by sharing their own browsing behavior. Throughout the remainder of this discussion, we use the term 'peer' and 'user' interchangeably, to refer to the same physical entity – a user (peer) browsing the Internet and connected to other users (peers) in the network.

This application uses the frequency of the web domains a user has visited during a period of time as the user's profile vector. Each user maintains a profile vector that keeps the frequency of visit of common web-domains. To measure the similarity between two users' browsing patterns, we use inner product between their profile vectors. To do that, the application uses *order* statistics-based *local* algorithm to measure inner product between different users' profile vector and that information is used to form communities such that users with high similarity in profile vectors are placed in the same community. One of the big advantages of this framework is, any meta-level technique that can measure similarity in metric space (vectors, trees and the like) can be plugged into this framework , and help to form similarity based communities which will share common interest between each other to enhance browsing/online experience.

It is obvious that user privacy is a big concern in such a P2P applications. Since formation of these communities involves sharing the actual browsing data, it may violate the privacy of the users. For any user it is imperative that its browsing data is not revealed in its raw form while forming these communities – otherwise it is almost impossible to convince web users to take part in such P2P computation where every user shares some data, does some computation and finally gets benefitted from the aggregate result (by being part of a similar minded community). In order to safeguard each user's private data, Liu et al. [13] have used cryptographic secure inner product protocols to compute the inner product between two users' profile vectors. However, such secure multi-party protocols are based on honest/semi-honest third party assumptions, which assumes that a user or a group of users will follow the protocol as specified and will not form a malicious liaison or do anything to extract private information from other users. In real-life, however, such ideal assumptions fall apart since very little control exists on each user's behavior and there is no centralized coordinator or administrator to monitor and govern all user activity. Besides, experimental results reported by Liu et al. [13] show that this secure protocol is (1) computationally very intensive (2) ex-

pensive from communication point of view and (3) not scalable at all. It is evident from the results that privacy preserving techniques designed for standard data mining is not going to work well in a P2P setup. A completely different approach is necessary to ensure privacy in a P2P setup. That motivates us to introduce a game theoretic approach to privacy preserving data mining which does not suffer from above-mentioned issues and relaxes some of the assumptions regarding user behavior.

## 4 Game Formulation

In this section we present a high level overview of PPDM algorithms designed as games.

We model the large-scale multi-party data mining applications as games where each participant tries to maximize its benefit or utility score by optimally choosing the strategies during the entire PPDM process. 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 nicely.

For example, consider a well-understood algorithm for computing sum based on the secure multi-party computation framework (details to be described in Section 5). 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) protect the local private data, (4) attack the messages received from other parties for divulging privacy-sensitive information regarding other parties, and (5) 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 nodes in the system can adopt different strategies for communication, computation, collusion or launching of a privacy breach attack. Every such decision is motivated by the utility associated with the choice. The utility value represents the benefit that a node gets by performing (not performing) a necessary communication or computation step that is part of the protocol or by colluding (not colluding) with other nodes in the network. 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 $\sigma_i$ 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 $\sigma_i$ for player $i$ essentially generates the tuple $(I^{(M)}, I^{(R)}, I^{(S)}, I^{(A)}, I^{(G)})$ where the $I$s are indicator variables for a node's computation, communication (receive and send), privacy attack and collusion strategies. Now we can put together the overall objective function for the game of multi-party secure sum computation.

$$u_i(\overline{\sigma}) = w_{m,i}c_m U(I_i^{(M)}) + w_{r,i}c_r U(I_{i,t}^{(R)}) + w_{s,i}c_s U(I_i^{(S)}) + w_{g,i} \sum_{j \in D-G} g(v_j) \quad (1)$$

Here $c_m U(I_i^{(M)})$ denotes the overall utility of performing a set of operation $M_{i,t}$, indicated by $I_i^M$ (similarly for other notations), $w$'s are the weights of the corresponding computation, computation etc., *D* denotes the set of all nodes, *G* denotes the set of colluding nodes and $g(v_j)$ is the benefit of node $j$, due to its local value $v_j$.

## 5 Illustration: Multi-Party Secure Sum Computation

Suppose there are $s$ individual sites, each with a value $v_j, j = 1, 2, \ldots, s$. It is known that the sum $v = \sum_{j=1}^{s} v_j$ (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 $(R + v_1)$ mod $N$ to site 2. In general, for $i = 2, \ldots, s$, 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 $N$. Then site 1, which knows $R$, can subtract $R$ from $z_s$ to obtain the actual sum. This sum is further broadcast to all other sites. For this secure sum protocol one may construct different utility functions based on different parameters such as cost of communication, computation or the cost of launching a privacy attack. It can be shown that a privacy-breach attack on a secure sum protocol by a single node might not be very successful. Similarly it can be shown that the utility of collusion in secure sum protocol depends on the size of the network, the number of colluding nodes, and the range of values at the different nodes. [7].

### 5.1 Game Equilibrium

Let us consider this simple unconstrained version of the objective function given in 1. 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 related activities as required by the protocol resulting in the following objective function (neglecting the constant term contributed by the communication-related factors):

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

Figure 1 shows a plot of this function as a function of $c_m U(I_i^{(M)})$ and $k$, the number of colluding parties. It shows that the optimal solution takes a value of $k > 1$. This implies

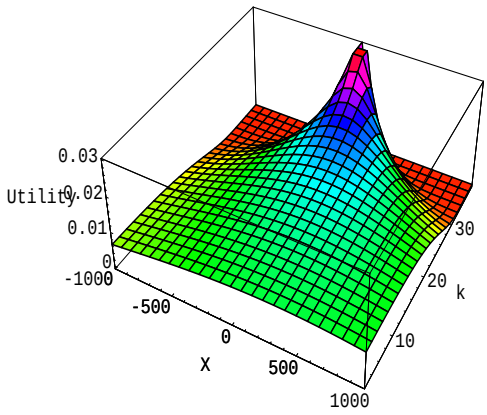


**Fig. 1.** Plot of the overall objective function. The optimal strategy takes a value of $k > 1$.

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 that the classical SMC-algorithm for secure sum makes is sub-optimal.

One way to deal with the problem 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 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 with the penalty term is

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

Here $w_p$ refers to the weight associated with the penalty. Figure 2 shows a plot of the modified objective function. It shows that the globally optimal strategies are all for $k = 1$. The strategies that adopt collusion always offer a sub-optimal solutions. 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. Our plan is to borrow the concept of Cheap Talk from game theory and economics [6] 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. For example, in the well known Prisoner's Dilemma game 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 game, in many other games the outcome may be significantly influenced by such pre-

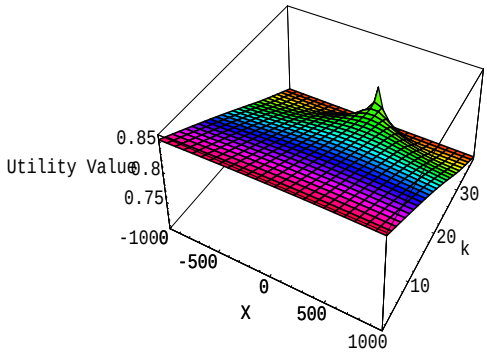


**Fig. 2.** Plot of the modified objective function. The globally optimal strategies are all for $k = 1$.

game 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. The algorithm in the following section describes a variant of the secure sum computation technique that offers a distributed mechanism for penalizing policy violations using a cheap talk-like mechanism.

### 5.2 Secure Sum with Penalty: Distributed Control

Consider a network of $s$ 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 groups 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. 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 random shares at each round of the secure sum. Hence, it takes several rounds for the secure sum to complete.

### 5.3 Results

We have implemented the above cheap talk-based solution without and performed multi-agent simulations in order to study the behavior of the agents. This experiment assumes

that the agents are rational in the sense that they choose actions that maximize their utility function. The details of the experimental setup are described in [7]. The results obtained from our simulation are represented by the following figures.

Initially we start with a fixed percentage (say 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.

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.

## 6 Conclusion

This paper presented an overview of a relatively new model of computation for distributed systems *viz.* P2P model of computing. It argued that we need highly scalable and efficient algorithms for doing data integration in such environments. Further it presented a web-mining application using some of the existing P2P technologies. It also 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 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 opens up many new possibilities. It offers a new approach to study the behavior of existing PPDM algorithms using a game theoretic approach and invent new ones *viz.* PPDM algorithms for computing inner product, clustering, and association rule learning.

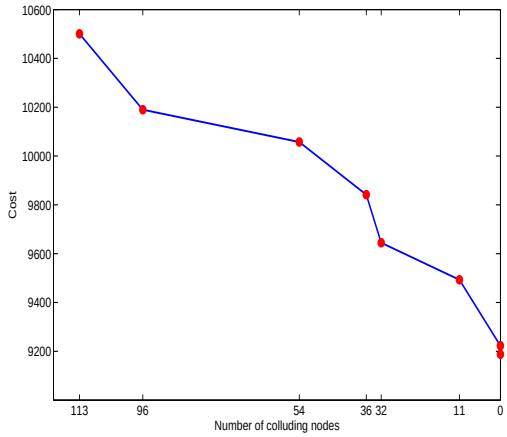


**Fig. 3.** Variation of cost with changes in number of colluding nodes.

## Acknowledgement

The authors would thank Souptik Datta for his valuable contributions.

## References

1. 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.
2. B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing. *Proceedings of the 21st ACM Symposium on the Theory of Computing (STOC)*, 1989.
3. M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The price of validity in dynamic networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 515–526, 2004.
4. S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proc. IEEE Infocom'05*, volume 3, pages 1653–1664, Miami, March 2005.
5. S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing special issue on Distributed Data Mining*, 10(4):18–26, 2006.
6. Joseph Farrell and Matthew Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996.
7. H. Kargupta, K. Das, and K. Liu. A game theoretic approach toward multi-party privacy-preserving distributed data mining. In *In Communication*, 2007.
8. D. Kempe, A. Dobra, and J. Gehrke. Computing aggregate information using gossip. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FoCS)*, pages 482–491, 2003.
9. Kowalczyk W., Jelasity M., and Eiben A. Towards Data Mining in Large and Fully Distributed Peer-To-Peer Overlay Networks. In *Proceedings of BNAIC'03*, pages 203–210, 2003.

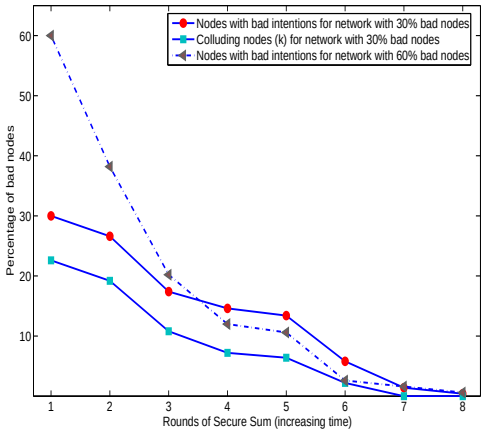


**Fig. 4.** Change in the number of bad nodes in the network over time. It shows that both the number of nodes interested in violating the policy and the number of nodes that form collusions decrease to zero due to the penalty scheme.


10. Kuhn F., Moscibroda T., and Wattenhofer R. What Cannot be Computed Locally! In *Proceedings of the 23rd Symposium on Principles of Distributed Computing (PODC)*, 2004.
11. Kutten S. and Peleg D. Fault-Local Distributed Mending. *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, 1995.
12. Linial N. Locality in Distributed Graph Algorithms. *SIAM J. Comp.*, 21:193–201, 1992.
13. K. Liu, K. Bhaduri, K. Das, P. Nguyen, and H. Kargupta. Client-side web mining for community formation in peer-to-peer environments. *SIGKDD Explorations*, 8(2):11–20, 2006.
14. 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.
15. M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray. Distributed averaging on a peer-to-peer network. In *Proceedings of IEEE Conference on Decision and Control*, 2005.
16. Naor M. and Stockmeyer L. What Can Be Computed Locally? *Proceedings of the 25th ACM Symposium on Theory of Computing (SToC)*, pages 184–193, 1993.
17. V. S. Verykios, A. K. Elmagarmid, B. Elisa, Y. Saygin, and D. Elena. Association rule hiding. In *IEEE Transactions on Knowledge and Data Engineering*, 2003.
18. R. Wolff, K. Bhaduri, and H. Kargupta. Local L2 thresholding based data mining in peer-to-peer systems. In *Proceedings of SIAM International Conference in Data Mining (SDM)*, Bethesda, Maryland, 2006.
19. Wolff R. and Schuster A. Association Rule Mining in Peer-to-Peer Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(6):2426–2438, December 2004.
20. A. C. Yao. How to generate and exchange secrets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.