Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

Efficient Privacy-Preserving Association Rule Mining: P4P Style

Yitao Duan and John Canny
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
Email: {duan, jfc}@cs.berkeley.edu

Justin Zhan
Carnegie Mellon University
Pittsburgh, PA, USA
Email: justinzh@andrew.cmu.edu

Abstract— **In this paper we introduce a new practical framework, called P4P (Peers for Privacy), for privacy-preserving data mining. P4P features a hybrid architecture combining P2P and client-server paradigms and provides practical private protocols for user data validation and general computation. The architecture is guided by the natural incentives of the participants and allows the computation to be based on verifiable secret sharing (VSS) where arithmetic operations are done *over small fields* (e.g. 32 or 64 bits), so that private arithmetic operations have the same cost as normal arithmetic. Verification of user data, which uses large-field public-key arithmetic (1024 bits or more) and homomorphic computation, only requires a small number (constant or logarithmic in the size of user data) of large integer operations. The solution is extremely efficient: In experiments with our implementation, verification of a million-element vector takes a few seconds of server or client time on commodity PCs (in contrast, using standard techniques takes hours). This verification can be used in many privacy-preserving data mining tasks to detect cheating users who attempt to bias the computation by submitting exaggerated values as their inputs. As an example, we demonstrate how association rule mining can be done in the P4P model with near-optimal efficiency and provable privacy.**

I. Introduction

Privacy has become an increasingly important issue in data mining. With the rapid growth of the Internet and the advance in microprocessors and algorithms, collecting and processing large amount data is becoming commercially viable. The potential for offering valuable services by mining information and knowledge from user data is great, so is the risk of privacy infringement.

Privacy-preserving data mining is an area of active research. Using [1]'s categorization, existing solutions basically fall into one of two models. The first is called *server-to-server* (S2S) where data is distributed across several servers. In the second model, on the other hand, data resides on each client and a server, or a data miner, performs mining tasks on the aggregate data from the clients. This model is denoted *client-to-server* (C2S) in [1].

In the S2S model, data records can be partitioned either horizontally or vertically. Horizontal partition refers to the cases where complete records for a subset of entities reside on each site (e.g. [2], [3]), while vertical partition means that each site holds a subset of the attributes for all entities [4], [5]. Existing privacy schemes in these models provide solutions for a number of data mining tasks such as classification [6], clustering [5], decision tree classification [3], etc.

In this paper we introduce a new private computation framework called Peers for Privacy (P4P) and show its potential in privacy-preserving data mining. In particular we demonstrate P4P's application in data mining with an efficient solution for privacy-preserving association rule mining over horizontally partitioned databases. P4P was introduced in [7] (A preliminary version can be found in [8]) as a practical framework for performing accurate computation with guaranteed privacy. Its architecture is similar to the C2S model. The difference is that a small subset of the users, called privacy peers, also participate in the computation. This arrangement not only accommodates a large number of existing server-based schemes, making their computation privacy-preserving with small changes to their existing infrastructure, but also allows us to take advantage of many powerful centralized tools such as linear algebra packages ARPACK, LAPACK, etc, which is not possible with a purely peer-to-peer approach. The involvement of privacy peers allows information to be shared among multiple participants, instead of residing on a single entity, and the computation can be based on verifiable secret sharing instead of homomorphic encryption as is done in [9] and [10] etc. The resulting scheme is much more efficient and practical because secret sharing can be done over *small* field where arithmetic operations are orders of magnitude faster than those performed in large field needed for homomorphic encryption. In any private computation scheme, to ensure correctness, it is necessary to verify, without leaking additional information, certain properties of the computation or data since now the raw data is private. Such verification typically involves cryptographic operations with large integers (e.g. public key encryption using keys with sufficient length, exponentiation in DDH groups, etc). P4P features novel zero-knowledge protocols for such verification that only require a small number (constant or logarithmic in the size of user data) of large integer operations. P4P supports general computation of any arithmetic circuit over finite field consisting of addition and multiplication gates. [1] However, this paper focuses on those algorithms that can be implemented via iterative vector

[1] P4P's support for multiplication is work in progress.

aggregation steps which involve only addition and are substantially more efficient than multiplication. Private vector addition in P4P is very efficient, actually as efficient as its centralized, non-private implementation. This remains true even with a zero-knowledge verification that checks that the L2 norm of user's vector is bounded by a predefined limit [2] because the this verification only incurs a small number of large integer operations and the cost is still dominated by the linear number of small field arithmetic operations. In this sense P4P provides privacy for vector addition-based algorithms almost for free and becomes a practical framework for many data mining tasks that can be built upon such algorithms.

A. Vector Addition-based Algorithms

Surprisingly, vector addition is a very general tool for many practical algorithms. Examples include linear algorithms like voting and summation, as well as non-linear algorithms such as singular value decomposition (SVD), link analysis, regression, analysis of variance (ANOVA) and several machine learning algorithms based on Expectation Maximization (EM) (such as [11]). These methods represent many of the most accurate algorithms for e-commerce personalization and targeted marketing etc. The non-linear algorithms aggregate user data only in certain steps, such as conjugate gradient, which are *linear* in per-user data thus can be built upon vector-addition. This has been demonstrated for SVD in [10] and link analysis in [12].

The rest of the paper is organized as follows. In Section II we give an overview of the P4P framework. Section III states P4P private computation model. We describe association rule mining problem in Section IV. In Section V, we show how association rule mining can be done in the P4P framework with high efficiency and guaranteed privacy. We further discuss our solution in Section VI. Section VII examines previous solutions and other related research. Finally we discuss current status and future directions in Section VIII.

II. P4P: An Overview

P4P supports mining of data collected from a group of users. In P4P, we assume there is a single computer called the *server*, which is operated by a service provider or a data miner. Unlike other server-based solutions such as [9], P4P also involves a (small) number of designated parties called *privacy peers* (PP) to participate in the computation. In contrast to previous work, privacy peers do not have to be dedicated servers and can belong to users in the community. The fundamental role of the privacy peers is to remove data control from the server by participating in the computation so that no one in the system, except for the owner herself, has access to user's data. Privacy peers are not required to be honest, and the protocol ensures that they cannot break the privacy of the protocol without the server's help.

[2]The verification is to prevent a user from exerting too much influence on the computation. It only uses a small number of cryptographic operations. The total cost is still dominated by the regular arithmetics.

This architecture is a hybrid of client-server and P2P. On one hand, the server shares the bulk of the computation and storage, and also synchronizes the protocol. This allows us to take advantage of its large computation/storage capacity and high availability. It also leads to practical, efficient protocols that are not possible with fully distributed architecture. On the other hand, the peers also participate in the computation and provide privacy. This particular architecture is motivated by several practical considerations:

- Client-server is the de facto business and service model for most existing service providers. It is unlikely that they will be willing to abandon their existing infrastructure for the sake of providing users with privacy.
- In a purely serve-based system, there is a tremendous power imbalance between the service provider, who possesses all the information and controls who can access to it, and the client users. This imbalance is prevalent in almost all aspects of their relationship including economic, information access, and computing etc.
- Any privacy enhancing technology will inevitably reduce the amount of user information that a service provider can collect and entail additional cost. Understandably, the service providers will be reluctant to adopt them. For any technological solutions to be viable, (1) it must be efficient: the cost for privacy protection should be small, ideally free, compared to the main computation; (2) the service provider's incentives must be taken into consideration.
- The power imbalance and the lack of incentive can be offset by the active participation of users/peers. At minimum, privacy protection requires user awareness and control of the collecting and use of personal information. Given the dominance and the reluctance on the server side, users must be involved in the loop to secure and verify the protection.
- Current distributed multiparty computation protocols have computation and communication complexity at least quadratic in the number of participants, with large hidden constant in the asymptotic complexity, and are not practical for large scale systems. But for many applications, the more users involved, the better quality the computation results are. Such applications are prevalent in data mining. The minimal scale required by such applications makes it infeasible to carry out the computations among the users.

P4P accommodates service provider's existing business model while taking advantage of participants' heterogeneity that exists in real-world systems. The server, which is powerful and highly available, is still responsible for most of the computation/storage but the privacy peers work with the users to provide privacy. This arrangement allows us to use a verifiable secret sharing (VSS)-based paradigm for computation which can be executed in regular-sized (32- or 64-bit) fields. Big integer field is used only for verification which involves only a small number (constant or $O(\log m)$ where m is the size of user data) of big integer operations. This avoids the pitfall

of generic MPC protocols (e.g. [13], [14], [15], [16]) which work in the big field all the time. On a typical computer today there is a six order of magnitude difference between the big integer cryptographic operations (order of milliseconds) and regular arithmetic operations (fraction of a nano-second). In P4P the cost is dominated by the *regular* arithmetic that one would have to pay even in a server-based, non-private scheme. In addition, P4P computes a function over data from all n users but it does not use an n-party protocol. Instead, computation in P4P consists of a number of independent, parallel pairwise two-party protocols between the server and one privacy peer. Each privacy peer services a small subset of users and each pairwise computation produces aggregate over this subset and the server further aggregate them to produce the final sum. Compared to a fully distributed multiparty scheme, by restricting the number of players to 2 in each pairwise computation, many expensive operations such agreement are made efficient. We can show that private computation in P4P is almost as efficient as regular, centralized arithmetic operations. In this sense, P4P adds privacy at almost no cost to the service provider and provides a practical solution for large scaled private computation tasks.

The candidates for the privacy peers may differ with applications. Since they are not trusted and cannot compromise user data, there are many options. It has been demonstrated that in practical P2P systems such as Gnutella and Napster, a small fraction of the users in the community provide most of the services to the others. The existence of such altruistic users is a pervasive phenomenon in communities and those users can provide the computation cycles for the privacy peers. In this case, we assume a much smaller ratio between peer and users, e.g. while a single server would normally serve millions of users, an individual privacy peer would support a few hundreds to perhaps tens of thousands of users. For workplace privacy, the peer would be a special employee (e.g. a union representative). The service may also be provided for a fee by a third-party commercial "privacy provider". In certain cases, it may be feasible to distribute the VSS among service providers. For example, two hospitals may wish to mine data collected from patients. However, it is not desirable to disclose patients' private information. Indeed, in many countries, legal privacy rules even prohibit health care providers from disclosing customers information, even to other providers. In this case P4P can be used to support this type of data mining by letting one of the service providers assume the role of a privacy peer in the protocol and the result is that both hospitals learn the accurate (aggregate) final computation but neither learns anything about users private data.

In addition to computation support, P4P also provides some highly efficient tools. All of them are zero-knowledge (i.e. they reveal no information other than what is being verified) protocols that involve only constant or logarithmic (in the size of user data) public-key operations. They include a user data validation protocol that verifies that the L2 norm of user vector is within a predefined bound and a zero-knowledge test of vector equality [17]. These are for dealing with cheating users who may wish to bias the computation by submitting exaggerated values as their inputs, which is a realistic threat in real-world applications but often ignored by many schemes such as [9].

P4P does not overthrow existing systems. Instead, it offers a way to "patch" them, with minimum impact on performance, to provide provable privacy for the users while maintaining the accuracy of the computation. This approach is applicable to a large number of existing and emerging real-world applications and is more realistic to guarantee adoption.

A. Security from Heterogeneity

The security of a P4P system is based upon the assumption that the server and the privacy peers won't collude (the corrupted privacy peers may collude with cheating users or other peers). We show why this is realistic.

Collusion comes in two forms:

1) The owners of the two machines conspire and share data and coordinate their actions;
2) An attacker (who can be one of the two players) corrupts both machines.

These two attacks, although often handled uniformly in most schemes, rely on different conditions to be successful: for two players to conspire, they must have the right incentive, which may come from anticipated benefit out-weighting foreseeable risk, and overcome the mutual distrust between them. For an adversary to corrupt both machines, it must penetrate the armor of the better protected one. P4P defends against them with different mechanisms, targeting at their individual conditions. Concretely, we make the following observations/assumptions:

1) The server is well protected against outside attacks.
2) The server and the privacy peers won't conspire and share data between each other.
3) The owner of the server won't attempt to break into peers' machines.

All three are provided by exploiting the asymmetry between server and privacy peer, which is an important source of security. While peer machines are not always well maintained, cooperations invest large amount of resources protecting their servers. On the other hand, collusion means exchange of data between the server and the privacy peer. Both will be aware of the cheating. In many situations there is a mutual distrust between the two which means neither can trust the other not to leak the plot. And the owner of the server, which is typically a big company, is usually in the spotlight of scrutiny, by the media, law enforcement, the competitors and the customers, etc., and the exposure of breach of its privacy policy is a big scandal that it cannot afford.

In essence, by exploiting the difference in their incentives, the heterogeneity of their individual protection, and the mutual *distrust* among the participants, P4P is made efficient yet secure. In a nutshell, our system relies on the server for defending against external attacks and uses the privacy peers to protect user privacy against a curious server. In practice the protection offered by this arrangement is very good compared to multi-server approaches. And economically, the P4P

approach is superior since it requires no additional resources on the server side.

III. The P4P Private Computation

A. Preliminaries

Let ϕ be a small integer [3] (e.g. 32 or 64 bits). Our goal is to support "normal"-sized integer (or fixed-point) arithmetics. This is what most applications need and the arithmetics are extremely efficient when each integer fits into a single memory cell. To provide information-hiding, we use a $(2,2)$-threshold secret sharing to embed this integer range in the additive group of integers modulo ϕ. To support signed values, which is what most applications require, we consider the specific coset representatives of the integers $\mod \phi$ in the range $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor$ if ϕ odd, or $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor - 1$ if ϕ even. We use $\mathbb{Z}_\phi$ to denote this field.

Let $d_i \in \mathbb{Z}_\phi^m$ be an m-dimensional data vector for user i. Throughout this paper, unless explicitly stated, all vectors are column vectors. We use $v[i]$ to denote the i^{th} element of a vector v. We also use $|a|$ to denote the bit length of an integer a in its binary representation.

Similar to [7], [8], we describe our protocols as 2-way multi-party computations carried out between the server and a privacy peer who are referred collectively to as *talliers*. In this paper we denote the server T_1 and the privacy peer T_2. The entire P4P computation is composed of a number of such independent pair-wise computations.

We assume the two talliers follow the protocol faithfully. This corresponds to a *semi-honest* model in cryptographic terms (e.g. [14]). In many applications both the server and the peers benefit from accurate computation. For example, the scheme can be a recommendation system hosted by an e-commerce vendor. The vendor can optimize its business strategy while the peers/users can obtain high-quality recommendations from accurate data mining computation. Or it can also be that two research institutes wish to perform computation on user data and they each participate in the protocol as a tallier. In both cases neither should have the incentive to bias the computation and a passive model is appropriate. The P4P framework can handle active cheating from privacy peers (without much loss of efficiency) but for simplicity of description we do not pursue this in this paper. Please refer to [7], [8] for details. Note that we allow the users to behavior arbitrarily and we provide a mechanism to detect and handle active user cheating.

B. Basic P4P Vector Aggregation

Let $Q = \{1, \ldots, n\}$ be the initial set of qualified users. The basic vector aggregation in P4P is carried out as follows:

1) User i generates a uniformly random vector $u_i \in \mathbb{Z}_\phi^m$ and computes $v_i = d_i - u_i \mod \phi$. She sends u_i to T_1 and v_i to T_2.
2) User i gives a ZK proof to both talliers that $\|d_i\|_2 < L$ using the protocol described in [7], [8]. If she fails to do so, both talliers exclude her from Q.
3) T_1 computes $\mu = \sum_{i \in Q} u_i \mod \phi$ and T_2 computes $\nu = \sum_{i \in Q} v_i \mod \phi$. T_2 sends ν to T_1, and T_1 sends μ to T_2.
4) T_1 publishes $s = \mu + \nu \mod \phi$.

The correctness is straightforward to verify. The privacy can be easily proven using a simulator who, given only the public sum, could generate random shares for the corrupted tallier, and produce the data received from the other tallier by simple subtraction. This is indistinguishable from the actual data that the corrupted tallier receives in a real execution of the protocol. This essentially shows that whatever information about user data an adversary can obtain from attacking the protocol can be generated by itself (by running the simulator) thus there is no leakage beyond the sum. For detailed proof please see [7], [8].

IV. Mining Association Rule Over Private Data

The association rule mining [18] is still one of most popular pattern-discovery methods in the field of knowledge discovery. Briefly, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items. The meaning of such rules is as follows: Given a database D of records, $X \Rightarrow Y$ means that whenever a record R contains X then R also contains Y with certain confidence. The rule confidence is defined as the percentage of records containing both X and Y with regard to the overall number of records containing X. The fraction of records R supporting an item X with respect to database D is called the support of X.

A. Distributed Association Mining

Let n be the total number of users and m be the total number of items. User i maintains a private data set D_i, $i = 1, 2, \ldots, n$ (which may contains e.g. her purchase records). The data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ forms a database, which is actually the concatenation of D_1, D_2, $\cdots$ and D_n. The database can be represented by a $n \times m$ boolean matrix x where $x_{ij} = 1$ if record i contains item j and 0 otherwise. We consider horizontally partitioned database [19] where each user's data set contains the same attributes. Without causing confusion, we also use D_i to denote the rows of the matrix x maintained by user i.

The goal is to conduct association rule mining on $[D_1 \cup D_2 \cup \cdots \cup D_n]$ and to find the association rules with support and confidence being greater than the given thresholds. We say an association rule (e.g., $X \Rightarrow Y$) has confidence $c\%$ in the data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ $c\%$ of the records which contain X also contain Y (namely, $c\% = P(Y \mid X)$). We say that the association rule has support $s\%$ in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if $s\%$ of the records in $[D_1 \cup D_2 \cdots \cup D_n]$ contain both X and Y (namely, $s\% = P(X \cap Y)$). Consequently, in order to learn association rules, one must compute the candidate itemsets, and then prune those that do not meet the preset confidence and support

[3] It does not have to be a prime since the computation only involves addition.

thresholds. In order to compute confidence and support of a given candidate itemset, we must compute, for a given itemset C, the frequency of attributes (items) belonging to C in the entire database (i.e., we must count how many attributes in C are present in all records of the database, and divide the final count by the size of the database which is m.) Note that association rule mining works on binary data, representing presence or absence of items in transactions. However, the proposed approach is not limited to the assumption about the binary character of the data in the content of association rule mining since non-binary data can be transformed to binary data via discreterization.

B. Association Rule Mining Procedure

The following is a fast algorithm for mining association rules on $[D_1 \cup D_2 \cdots \cup D_n]$, introduced in [20]:

1) L_1 = large 1-itemsets
2) **for** $(k = 2; L_{k-1} \neq \emptyset; k++)$ **do begin**
3) C_k = **apriori-gen**(L_{k-1})
4) **for** all candidates $c \in C_k$ **do begin**
5) **Compute *c.count***
6) **end**
7) $L_k = \{c \in C_k | c.count \geq min\text{-}sup\}$
8) **end**
9) Return L = $\cup_k L_k$

At line 5 in the above procedure, $c.count$ divided by the total number of records is the support of a given item set. We will show how to compute it in Section IV-C.

The procedure **apriori-gen** is described in the following (please also see [18], [20] for details).

apriori-gen(L_{k-1}: large $(k-1)$-itemsets)

insert into C_k
select $p.item_1$, $p.item_2$, ..., $p.item_{k-1}$,$q.item_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.item_1 = q.item_1$, ..., $p.item_{k-2} = q.item_{k-2}$
$p.item_{k-1} < q.item_{k-1}$;

Next, in the $prune$ step, we delete all itemsets $c \in C_k$ such that some $(k-1)$-subset of c is not in L_{k-1}:

$\forall$ itemsets $c \in C_k$ **do**
$\forall$ $(k-1)$-subsets s of c **do**
if($s \notin L_{k-1}$) **then**
delete c from C_k;

*C. Computing **c.count***

In the procedure of association rule mining for horizontal partition, each party computes the partial *c.count* based on their own data. Without loss of generality, let's assume that user i maintains $c.count_i$, $i = 1, \ldots, n$. The goal is to compute the $c.count$ which equals to $\sum_{i=1}^{n} c.count_i$ without disclosing $c.count_i$ to party j where $i \neq j$. In the next section, we will provide a privacy-preserving protocol to compute this summation under P4P framework.

V. Association Rule Mining in P4P

Algorithm **apriori-gen** is to generate a superset of possible candidate itemsets and then prune it to get C_k. We use the P4P vector aggregation and verification protocols to compute $c.count$ for all $c \in C_k$ as one P4P aggregation step.

Recall that D_i is the data set of party i. In the case of horizontal partition, D_i is essentially a number of rows of items. At step k of the above algorithm, let $m_k = |C_k|$ and $C_k = \{c_1, \ldots, c_{m_k}\}$. The protocol is as follows:

1) User i constructs an m_k-dimensional vector $d_i^{(k)} \in \mathbb{Z}_\phi^{m_k}$ such that

$$d_i^{(k)}[j] = \sum_{r \in D_i} (\prod_{l \in c_j} x_{rl})$$

2) User i then inputs $d_i^{(k)}$ into the P4P protocol. She also provides a zero-knowledge proof that her data is valid using the technique that will be elaborated later. If the user fails the proof, the server and the privacy peers exclude her data from subsequent computation.
3) Let $s^{(k)} = \sum d_i^{(k)}$ be the output of the P4P aggregation. The desired counts are simply $c_j.count = s^{(k)}[j], j = 1, \ldots, m_k$.

At the first step, for all the rows in her data set, for each itemset $c_j \in C_k$, the user computes the product of all the attributes in c_j. She then add these products across all the rows in her block. This is the j-th elements in d_i. Clearly, this is the number of rows in her data set containing the itemset c_j. By aggregating these vectors across all users, P4P outputs the total number of rows containing all itemsets $c_j \in C_k$ in the entire database $[D_1 \cup D_2 \cup \cdots \cup D_n]$.

At step 2, the user is required to prove, in zero-knowledge, that her data is valid. There are two ways to accomplish this that can be used in different situations. When k is small, there hasn't much pruning yet so the candidate set C_k tends to be quite large. For typical data such as web pages or vendor inventory, m_k can be in the range of billions or millions. In this case the validation protocol introduced in [7], [8] should be used with a bound

$$L = \alpha \sqrt{m_k} |D_i|$$

where $0 < \alpha \leq 1$.

This protocol verifies that the L2 norm of the user vector is bounded by L without leaking any information about the data. The P4p's protocol is extremely efficient and takes only a few seconds to validate a vector of a million elements on commodity PCs (in contrast, using standard techniques takes hours).

At the later steps of the algorithm when k is large, C_k can become small since it has gone through many iterations of pruning. In this case one can apply the ZK boundedness proof directly on each of d_i's elements to verify that all of them are bounded by $|D_i|$. Please see [7], [8] and [21] for details.

VI. Discussion

A. Privacy Analysis

The privacy of our solution is straightforward to analysis. As is proven in [7], [8], the P4P aggregation and validation do not leak any more information about individual user data other than what can be inferred from the final results. And the results are just the supports of the itemsets that we are computing. These have been treated as public data by many private association rule mining schemes (e.g. [4], [22]).

B. Near Optimal Efficiency

In a realistic association rule mining application, the scale of the problem is typically quite large. For example, WalMart sells over 100,000 items. And the web has billions of web pages. When mining association rules using the a priori algorithm over such datasets, C_k may contain millions of itemsets. For any solutions to be practical, they must be efficient enough to handle dataset of such scale.

The P4P solution is *near optimal* in that its cost is comparable to that of a distributed, non-private implementation of the a priori algorithm. For each iteration, computing $c.count$ for all $c \in C_k$ is simply one single P4P vector aggregation. And since the P4P's main computation (which has complexity $O(m_k|D_i|)$ for party i) is done over small field (32- or 64-bit) and has the same cost as regular, non-private arithmetic, compared to a server-based, non-private implementation of the algorithm, the computation overhead for each user in P4P is only doubled while that for the server remains the same. Verification is done in large field but the number of such operations is only $O(\log m_k)$. This is one of the important features provided by the P4P framework and it guarantees that the privacy mechanism causes little overhead for the server or the privacy peers, which are the bottlenecks of the system. Using the benchmark obtained by the P4P implementation in [7], [8], verification of a million element vector takes only a few seconds of user and server time. The same task takes hours using other solutions. In a sense P4P provides privacy to association rule mining applications almost for free.

C. Dealing with Malicious Users

A lot of private data mining schemes such as [9] cannot deal with active cheating from users (in cryptographic terms, their schemes are only secure passive adversary). However, in any realistic applications, users often pose active threats. A user may submit invalid data, or refuse to participate in the computation at all. A competitor of the data mining service provide can easily disguise as a user and disrupt the computation.

In our scheme we use P4P's efficient built-in zero-knowledge user data validation protocol to verify that a user cannot exert too much influence on the computation. The bound used in the protocol should be at most $\sqrt{m_k}|D_i|$ (corresponding to the case of setting $\alpha = 1$), but usually substantially smaller, determined by the application based on the estimate on the distribution of the raw data. The check is not tight in that a cheating user can still falsify her data without being detected. However, the effectiveness of her cheating is limited (by the bound) and assuming a reasonably large fraction of users are honest (otherwise there is no point running the data mining algorithm anyway), cheating users cannot cause too much change to the result.

In addition, our solution features a simple cheating user handling mechanism. When a user is detected cheating by P4P's validation protocol, we simply disqualify her from the computation for future steps. This is equivalent to setting all her entries to 0s for the rest of the computation. The results from previous steps may not be very accurate but generally there is no need to redo them. This is because the validation ensures that if a user tries to manipulate the support of an itemset by increasing her data in the corresponding entry by a large amount, she will be detected. So the only undetectable cheating is to increase the value by a small amount or decrease it. If the user only holds a few rows of the database, which typically should be the case for typical user-based mining, neither should have big influence on the computation since we are aggregating across a large number of users and most of them are honest. If, on the other hand, she holds a large trunk of the database, the server and the privacy peers can simply exchange the shares of her data and reconstruct her original vector $d_i^{(1)}$. They then correct the results by subtracting all cheating users data from the aggregation. The correction is done in public, in regular-sized field and should be very efficient.

VII. Related Work

Theoretically, the data mining tasks performed by P4P can be done with secure multiparty computation (MPC) protocols, which provide general solutions for computing any probabilistic n-ary function among n players while protecting each player's private data [23], [24], [13], [14], [15]. The problem with the MPC protocols is their prohibitive cost. These protocols make heavy use of public-key cryptosystems or one-way functions, applying verifications or ZKPs at most steps, and some even operate at bit-level (rather than on arithmetic values). Even those arithmetic protocols typically have at least $O(sn^3)$ computation and communication complexity, with large hidden constant, where s is a security parameter. [4]

Existing privacy-preserving data mining solutions use either randomization (e.g. [28], [29]) or cryptographic techniques (e.g. [3], [6], [5], [9], [30]) to protect privacy. The first approach, besides sacrificing accuracy, has been shown to provide very little privacy protection in many cases [31]. Most of the cryptographic schemes use some form of MPC (e.g. [3], [6], [5] or homomorphic encryption [9], [30]. They attempted to obtain reasonable efficiency by targeting at specific problems or restricting the number of players.

[4] For addition only functions some protocols only require $O(n)$ broadcasts (e.g. [25], [26]). However, broadcast has to be emulated with Byzantine agreement protocols in a realistic network such as the Internet. The most efficient broadcast protocols require $\Omega(n^2 l)$ bits to be sent for broadcasting an l-bit message [27].

P4P is also in the cryptographic category but it provides a more general primitive and much better efficiency. Its private vector addition protocol works in "normal" size (32 or 64 bits) fields so local computation is as efficient as regular arithmetics. All its zero-knowledge verifications [8], [17] involves only small (constant or $\log(m)$) number of big field operations thus adding very little cost.

VIII. Conclusion and Future Work

In this paper we described potential applications of a new private computation framework, Peers for Privacy, to the area of data mining. We demonstrated the application of P4P in privacy-preserving association rule mining over horizontally partitioned data. The scheme is near optimal in terms of efficiency and robust against active cheating from malicious users.

The P4P framework and the protocols described in this paper are being actively developed and its feasibility has been demonstrated by evaluations with current implementation on real-world datasets. In the near future, we plan to build more "middle tier" components to support more concrete applications. Our goal is to make P4P a useful tool for developers in areas such as data mining and others to build privacy preserving real-world applications.

References

[1] N. Zhang, S. Wang, and W. Zhao, "A new scheme on privacy-preserving data classification," in *KDD '05*. New York, NY, USA: ACM Press, 2005, pp. 374–383.

[2] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[3] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of cryptology*, vol. 15, no. 3, pp. 177–206, 2002.

[4] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD '02*. New York, NY, USA: ACM Press, 2002, pp. 639–644.

[5] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2003, pp. 206–215.

[6] W. Du, Y. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *SIAM International Conference on Data Mining*, 2004, pp. 222–233.

[7] J. Canny and Y. Duan, "Practical private computation of vector addition-based functions or: Can privacy be for free?" Submitted to 2007 IEEE Symposium on Security and Privacy, 2006.

[8] J. F. Canny and Y. Duan, "Practical private computation of vector addition-based functions or: Can privacy be for free?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-12, February 8 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-12.html

[9] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving classification of customer data without loss of accuracy," in *SDM 2005*, 2005.

[10] J. Canny, "Collaborative filtering with privacy," in *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 45–57. [Online]. Available: http://citeseer.nj.nec.com/canny02collaborative.html

[11] J. F. Canny, "Gap: a factor model for discrete data," in *SIGIR '04*. ACM Press, 2004, pp. 122–129.

[12] Y. Duan, J. Wang, M. Kam, and J. Canny, "A secure online algorithm for link analysis on weighted graph," in *Proceedings of the Workshop on Link Analysis, Counterterrorism and Security at the SIAM Data Mining Conference, 2005*, April 2005, pp. 71–81.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *STOC'88*. ACM, 1988, pp. 1–10.

[14] O. Goldreich, *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[15] S. Goldwasser and L. Levin, "Fair computation of general functions in presence of immoral majority," in *CRYPTO '90*, ser. LNCS, vol. 537. Springer-Verlag, 1991, pp. 77–93.

[16] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *PODC '98*. ACM Press, 1998, pp. 101–111.

[17] Y. Duan and J. Canny, "Zero-knowledge test of vector equivalence and granulation of user data with privacy," in *IEEE International Conference on Granular Computing*, 2006.

[18] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of ACM SIGMOD Conference on Management of Data*, P. Buneman and S. Jajodia, Eds., Washington D.C., May 1993, pp. 207–216.

[19] J. Zhan, "Privacy preserving collaborative data mining," Ph.D. dissertation, University of Ottawa, 2006.

[20] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[21] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology – EUROCRYPT 2000*, ser. Lecture Notes in Computer Science, vol. 1807. Springer-Verlag, 2000, pp. 431–444.

[22] J. Zhan, S. Matwin, and L. Chang, "Privacy-preserving collaborative association rule mining," in *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*.

[23] A. C.-C. Yao, "Protocols for secure computations," in *FOCS '82*. IEEE, 1982, pp. 160–164.

[24] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game — a completeness theorem for protocols with honest majority," in *STOC '87*, 1987, pp. 218–229.

[25] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *EUROCRYPT '01*. Springer-Verlag, 2001, pp. 280–299.

[26] I. Damgård and J. B. Nielsen, "Universally composable efficient multiparty computation from threshold homomorphic encryption," in *CRYPTO 2003*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2003, pp. 247–264.

[27] B. A. Coan and J. L. Welch, "Modular construction of a byzantine agreement protocol with optimal message bit complexity," *Information and Computation*, vol. 97, no. 1, pp. 61–85, 1992.

[28] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *PODS '03*. New York, NY, USA: ACM Press, 2003, pp. 211–222.

[29] W. Du and Z. Zhan, "Using randomized response techniques for privacy-preserving data mining," in *KDD '03*. New York, NY, USA: ACM Press, 2003, pp. 505–510.

[30] R. Wright and Z. Yang, "Privacy-preserving bayesian network structure computation on distributed heterogeneous data," in *KDD '04*. New York, NY, USA: ACM Press, 2004, pp. 713–718.

[31] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques," in *ICDM '03*. Washington, DC, USA: IEEE Computer Society, 2003, p. 99.