Privacy-Preserving Data Mining: Why, How, and When

Data mining is under attack from privacy advocates because of a misunderstanding about what it actually is and a valid concern about how it's generally done. This article shows how technology from the security community can change data mining for the better, providing all its benefits while still maintaining privacy.

Jaideep Vaidya
Rutgers University

Chris Clifton
Purdue University

In today's information age, data collection is ubiquitous, and every transaction is recorded somewhere. The resulting data sets can consist of terabytes or even petabytes of data, so efficiency and scalability is the primary consideration of most data mining algorithms.

Naturally, ever-increasing data collection, along with the influx of analysis tools capable of handling huge volumes of information, has led to privacy concerns. Protecting private data is an important concern for society—several laws now require explicit consent prior to analysis of an individual's data, for example—but its importance is not limited to individuals: corporations might also need to protect their information's privacy, even though sharing it for analysis could benefit the company. Clearly, the trade-off between sharing information for analysis and keeping it secret to preserve corporate trade secrets and customer privacy is a growing challenge.

But is it even possible to perform large-scale data analysis without violating privacy? Given sufficient care, we believe the answer is yes. In this article, we'll describe why data mining doesn't inherently threaten privacy, and we'll survey two approaches that enable it without revealing private information: randomization and secure multiparty computation (SMC).

The problem

In 2002, concerns over government collection of data led to street protests in Japan.[1] In 2003, concern over the US Total Information Awareness program even led to the introduction of a bill in the US Senate that would have stopped any US Department of Defense data mining program. Such knee-jerk reactions don't just ignore the benefits of data mining—they display a lack of understanding of its goals.

Models

The goal of data mining is to extract knowledge from data. David Hand, Heikki Mannila, and Padhraic Smyth[2] categorize data mining into five tasks:

- *Exploratory data analysis (EDA)*. Typically interactive and visual, EDA techniques simply explore the data without any preconceived idea of what to look for.
- *Descriptive modeling*. A descriptive model should completely describe the data (or the process generating it); examples include models for the data's overall probability distribution (density estimation), partitions of the p-dimensional space into groups (cluster analysis and segmentation), and descriptions of the relationship between variables (dependency modeling).
- *Predictive modeling: classification and regression*. The goal here is to build a model that can predict the value of a single variable based on the values of the other variables. In classification, the variable being predicted is categorical, whereas in regression, it's quantitative.
- *Discovering patterns and rules*. Instead of building models, we can also look for patterns or rules. Association rules aim to find frequent associations among items or features, whereas outlier analysis or detection focuses on finding "outlying" records that differ significantly from the majority.
- *Retrieval by content*. Given a pattern, we try to find similar patterns from the data set.

The first three tasks output models that essentially summarize the data in various ways; the last two find specific patterns, but they're often generalized and don't reflect particular data items. Because these models generally don't contain individual data values, they don't present an immediate threat to privacy. However, there's still the issue of inference. A "perfect" classifier, for example, would enable discovery of the target class, even if the individuals' target classes weren't directly disclosed. In practice, though, probabilistic inferences are more likely, giving the model's possessor a probabilistic estimate of private values. We'll discuss some recent work that restricts the results of data mining in more detail later. The more immediate privacy problem with data mining is based not on its results, but in the methods used to get those results.

Methods

Most data mining applications operate under the assumption that all the data is available at a single central repository, called a *data warehouse.* This poses a huge privacy problem because violating only a single repository's security exposes all the data. Although people might trust some entities with some of their data, they don't trust anyone with all their data. This was the root of the protests in Japan—the national government wasn't collecting new information, but simply creating a single repository for information previously kept by the prefectures.

Mediators and federated databases don't solve this centralization problem—they just change the nature of attacks. Mediators provide access to distributed data, and it is this access—rather than the data's location—that creates the opportunity for misuse. Whether a data warehouse is real or virtual is irrelevant: if the data mining algorithm can access the data, the opportunity exists for an attacker to get it, too.

A naïve solution to the problem is de-identification—removing all identifying information from the data and then releasing it—but pinpointing exactly what constitutes identification information is difficult. Worse, even if de-identification is possible and (legally) acceptable, it's extremely hard to do effectively without losing the data's utility. Latanya Sweeney[3] used externally available public information to re-identify "anonymous" data and proved that effectively anonymizing the data required removal of substantial detail.

Better solutions are possible. One is to avoid building a centralized warehouse in the first place. Distributed data mining algorithms minimize the exchange of data needed to develop globally valid models. Later in this article, we'll show that such algorithms can be performed in ways that provably preserve privacy and prevent disclosure of data beyond the original sources.

Another approach is data perturbation—modifying the data so that it no longer represents real individuals. Because the data doesn't reflect real values, even if a data item is linked to an individual, the individual's privacy remains intact. (Such data sets must be known to be perturbed, though, so that anyone attempting to misuse the data knows it can't be trusted.) An example is the US Census Bureau's Public Use Microdata Sets. A primary perturbation technique is data swapping—exchanging data values between records in ways that preserve certain statistics but destroy real values.[4] An alternative is randomization—adding noise to data to prevent discovery of the real values. Because the data no longer reflects real-world values, it can't be used to violate individual privacy. The challenge is in obtaining valid data mining results from the perturbed data. Let's review some techniques that enable data mining of such noisy data.

Data perturbation

The problem with data perturbation is that doing it indiscriminately can have unpredictable impacts on data mining. Vladimir Estivill-Castro and Ljiljana Brankovic explored the impact of data swapping on data mining for decision rules (combinations of attributes that are effective at predicting a target class value).[5] Full swapping (ensuring that no original records appear in the perturbed data set) can prevent effective mining, but the authors concluded that limited swapping has a minimal impact on the results. The key for privacy preservation is that you don't know which records are correct; you simply have to assume that the data doesn't contain real values.

Randomization, adding noise to hide actual data values, works because most data mining methods construct models that generalize the data. On average, adding noise (if centered around zero) preserves the data's statistics, so we can reasonably expect that the data mining models will still be correct. Let's assume we're building a model that classifies individuals into "safe" and "unsafe" driver categories. A likely decision rule for such a model would state that drivers between the ages of 30 and 50 are likely to be safe. Now assume we add random noise to the ages to prevent discovery of an individual's driving record simply by knowing his or her age. Some safe drivers between the ages of 30 and 50 will be moved into other age brackets, and some unsafe drivers who are younger or older will be moved into the 30 to 50 bracket, but the 30 to 50 bracket will also lose unsafe drivers and gain safe drivers from other age brackets. On the whole, drivers in the 30 to 50 range will still likely be safer—even on the perturbed data.

The problem is that knowing the overall values is not sufficient for building a decision tree. Data mining must also help us figure out where to make the decision points, not just the decision for those ranges—for example, why do we use 30 to 50 as the range? Why not 35 to 55 or 25 to 60? Data mining algorithms automatically find appropriate points to make such splits, but these points can be obscured by adding noise to the data. Let's assume the data is distributed as shown in Figure 1. On the original data, the natural breaks in the data fall at ages 30 and 50, so a data mining algorithm will pick these as decision points. After adding noise, the data no longer has these obvious points, so a data mining algorithm is likely to pick bad decision points and produce poor results.

In a previous work,[6] Rakesh Agrawal and Ramakrishnan Srikant presented a solution to this problem. Given the distribution of the noise added to the data, as well as the randomized data set, they could reconstruct the data set's distribution (but not actual data values). This enabled a data mining algorithm to construct a much more accurate decision tree than mining the randomized data alone, which approaches the accuracy of a decision tree constructed on the real data.

Agrawal and Srikant define the reconstruction problem as follows: Given a cumulative distribution F_y and the realizations of n random samples $X_1 + Y_1, X_2 + Y_2, \ldots, X_n + Y_n$, estimate F_x. The basic idea is to use Bayes rule to estimate the posterior distribution for a given point. For example,

$$F'_{X_1}(a) = \frac{\int_{-\infty}^{a} f_Y(w_1 - z) f_X(z)dz}{\int_{-\infty}^{\infty} f_Y(w_1 - z) f_X(z)dz}.$$

To estimate the posterior distribution function F'_X given $x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n$, the distribution functions for all of the X_i is averaged.

$$F'_X(a) = \frac{1}{n}\sum_{i=1}^{n} F'_{X_i} = \frac{1}{n}\sum_{i=1}^{n} \frac{\int_{-\infty}^{a} f_Y(w_i - z) f_x(z)dz}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X(z)dz} \quad (1)$$

The corresponding density function, f'_X, can be obtained by differentiating F'_X.

$$f'_X(a) = \frac{1}{n}\sum_{i=1}^{n} \frac{f_Y(w_i - a) f_X(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X(z)dz} \quad (2)$$

Although f_X isn't really known, we can estimate it by using the normal distribution as the initial estimate and iteratively refine this estimate by applying Equation 2.

Similar approaches have also been developed for learning association rules from randomized data, so solutions for other data mining tasks are certainly feasible. Although we won't get the same exact data mining results postrandomization as prerandomization, researchers have experimentally shown the results to be accurate enough in the case of both classification[6] and association rule mining.

One concern with randomization approaches is that the very techniques that let us reconstruct distributions also give us information about the original data values. As a simple example, let's reexamine Figure 1. From the distribution, we don't see any drivers younger than 16. Assume we're told that the randomization was done by adding noise randomly chosen from the range [–15, 15], and based on this and the noisy data set, we reconstruct the original distribution. This doesn't appear to tell us the age of any individual—a driver who is 40 years old is equally likely to have his or her age given as anything from 25 to 55—but what about an individual whose age is shown as 1 in the noisy data? We know (from the reconstructed distribution) that no drivers are younger than 16—so the driver whose age is given as 1 in the noisy data must be 16 years old!

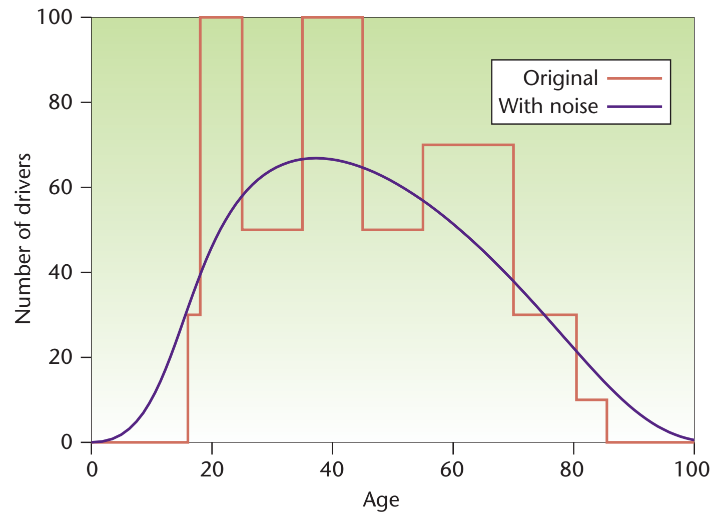
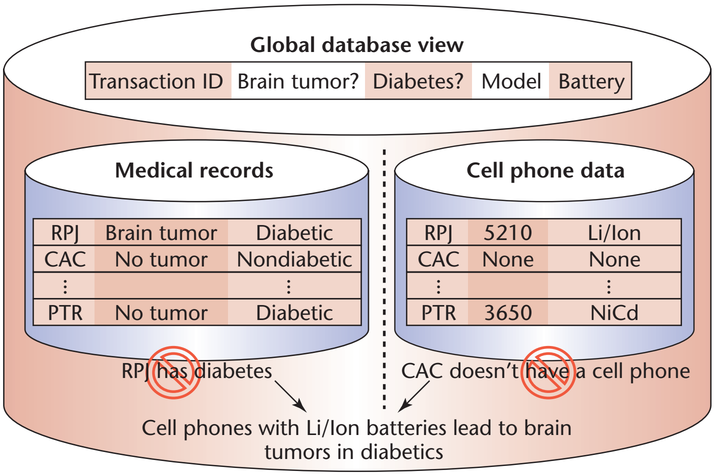


Figure 1. Loss of decision points in randomized data. While the data shows clear groups of people in their 20s, 40s, and 60s, these distinctions disappear when noise is added.

Secure multiparty computation

Let's look at an alternative approach based on the premise that every piece of private information is validly known to one or more parties. Revealing private data to parties such as the data set's owner or the individual to whom the data refers is not a breach of privacy—they already know it. The concern is with revealing the data to other parties. To address this, we use a specialized form of privacy-preserving, distributed data mining. Parties that each know some of the private data participate in a protocol that generates the data mining results, yet that can be proven not to reveal data items to parties that don't already know them. Thus the process of data mining doesn't cause, or even increase the opportunity for, breaches of privacy.

The basic idea is that parties hold their own data, but cooperate to get the final result. In 1986, Andrew Yao[7] proposed the idea of secure two-party computation in which any function with inputs distributed between two

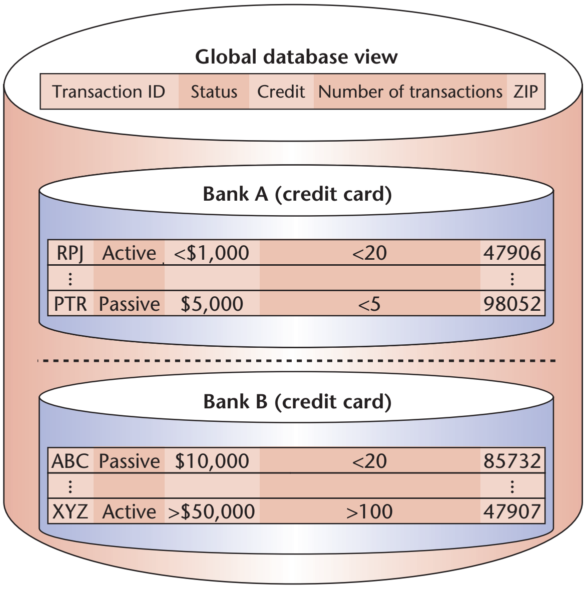

Global database view
Medical records
Cell phone data
RPJ has diabetes
CAC doesn't have a cell phone
Cell phones with Li/Ion batteries lead to brain tumors in diabetics

Transaction ID	Brain tumor?	Diabetes?	Model	Battery

RPJ	Brain tumor	Diabetic
CAC	No tumor	Nondiabetic
⋮		⋮
PTR	No tumor	Diabetic

RPJ	5210	Li/Ion
CAC	None	None
⋮		⋮
PTR	3650	NiCd

Figure 2. Vertically partitioned database. Although mining the global database view could give valuable knowledge, constructing the view violates privacy.



Transaction ID	Status	Credit	Number of transactions	ZIP
RPJ	Active	<$1,000	<20	47906
⋮				⋮
PTR	Passive	$5,000	<5	98052
ABC	Passive	$10,000	<20	85732
⋮				⋮
XYZ	Active	>$50,000	>100	47907

Figure 3. Horizontally partitioned database. The global view is the union of the individual databases.

parties could be securely computed without revealing any information to any party other than its local input and output (or anything inferable from the input and output). Oded Goldreich and colleagues extended and formalized this approach to multiparty computation.[8] The whole class of computation is now known as secure multiparty computation, or secure distributed computation. Matt Franklin and Modi Yung[9] provide an excellent survey of secure distributed computation.

The notion of computational indistinguishability is crucial to SMC. A distribution of numbers is said to be computationally indistinguishable from another distribution of numbers if no polynomial time program can distinguish between the two distributions. As long as the sequence of numbers revealed during a protocol is computationally indistinguishable from numbers drawn from a random distribution, the protocol is assumed to be secure.

Goldreich proved that any problem representable as a circuit can be securely solved, but the computation cost depends on the input's size. This generic method based on circuit evaluation is expensive (even for small inputs) and the computational cost is prohibitive for large inputs, as is the case with data mining. The answer has been to develop practical secure techniques to perform all the tasks of data mining.

The model

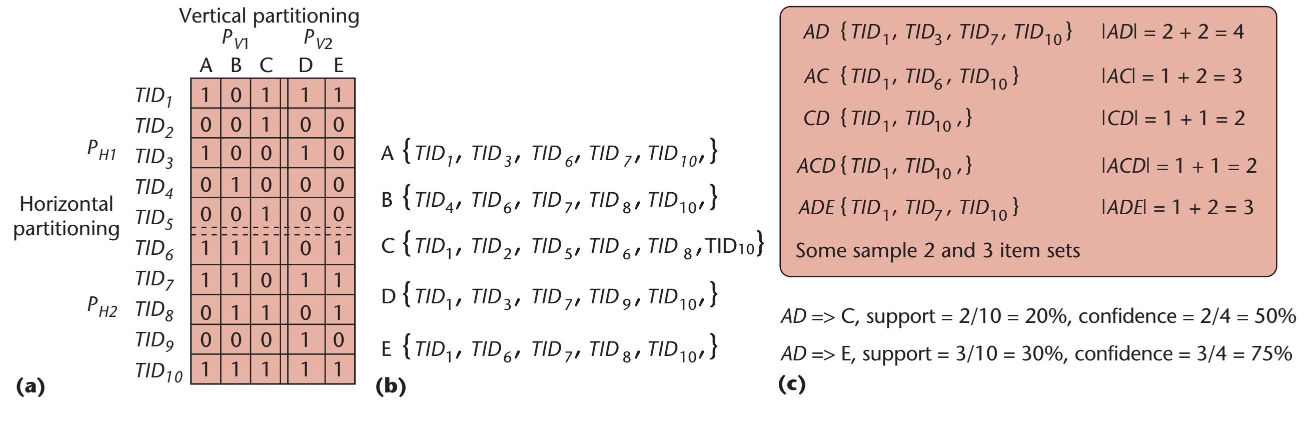
In the SMC model, the question of privacy doesn't arise when all the data is centralized—the data holder computes the function. When the input to a function is distributed among different sources, though, the privacy of each data source comes into question. The way the data is distributed also plays an important role in defining the problem because data can be partitioned into many parts either vertically or horizontally.

Vertical partitioning of data implies that although different sites gather information about the same set of entities, they collect different feature sets. Banks, for example, collect financial transaction information, whereas the IRS collects tax information. Figure 2 illustrates vertical partitioning and the kind of useful knowledge we can extract from it. The figure describes two databases, one containing individual medical records and another containing cell-phone information for the same set of people. Mining the joint global database might reveal such information as cell phones with Li/Ion batteries can lead to brain tumors in diabetics.

In horizontal partitioning, different sites collect the same set of information but about different entities. Different supermarkets, for example, collect the same type of grocery shopping data. Figure 3 illustrates horizontal partitioning and shows the credit-card databases of two different (local) credit unions. Taken together, we might see that fraudulent customers often have similar transaction histories. However, no credit union has sufficient data by itself to discover the patterns of fraudulent behavior.

The methodology

A truly secure SMC protocol doesn't reveal any information other than its input and output or any information polynomially computable from it, but care in performing these tasks is required, as this itself might be a privacy breach. As a trivial example, consider securely computing



	A	B	C	D	E
TID_1	1	0	1	1	1
TID_2	0	0	1	0	0
TID_3	1	0	0	1	0
TID_4	0	1	0	0	0
TID_5	0	0	1	0	0
TID_6	1	1	1	0	1
TID_7	1	1	0	1	1
TID_8	0	1	1	0	1
TID_9	0	0	0	1	0
TID_{10}	1	1	1	1	1

A $\{TID_1, TID_3, TID_6, TID_7, TID_{10},\}$
B $\{TID_4, TID_6, TID_7, TID_8, TID_{10},\}$
C $\{TID_1, TID_2, TID_5, TID_6, TID_8, \text{TID}_{10}\}$
D $\{TID_1, TID_3, TID_7, TID_9, TID_{10},\}$
E $\{TID_1, TID_6, TID_7, TID_8, TID_{10},\}$

$AD\ \{TID_1, TID_3, TID_7, TID_{10}\}$	$\lvert AD\rvert = 2 + 2 = 4$
$AC\ \{TID_1, TID_6, TID_{10}\}$	$\lvert AC\rvert = 1 + 2 = 3$
$CD\ \{TID_1, TID_{10},\}$	$\lvert CD\rvert = 1 + 1 = 2$
$ACD\ \{TID_1, TID_{10},\}$	$\lvert ACD\rvert = 1 + 1 = 2$
$ADE\ \{TID_1, TID_7, TID_{10}\}$	$\lvert ADE\rvert = 1 + 2 = 3$

Some sample 2 and 3 item sets

AD => C, support = 2/10 = 20%, confidence = 2/4 = 50%
AD => E, support = 3/10 = 30%, confidence = 3/4 = 75%

Figure 4. Example database and association rules. For horizontal partitioning, we see the (a) example database, (b) *TID* list representation of the database, and (c) sample association rules.

the sum of local numbers. For two parties, the sum reveals the other party's input—there's no way to avoid that—so it's necessary (even before starting data mining) to analyze whether the input and output taken together could reveal too much information.

Once a protocol is developed, we must prove its security. Because all interaction occurs through the messages sent and received, we simulate the views of all the parties by simulating the corresponding messages. If we can simulate these messages, then we can easily simulate the entire protocol just by running it. The key is that the simulation need not exactly duplicate the messages sent. Instead, we use a notion from cryptography—the same message can be encrypted with different keys to look different, even though they represent the same message. In the context of SMC, multiple executions of a protocol on the same input may exchange different messages, governed by random choices made by the parties (analogous to the choice of keys in cryptography). The protocol is secure if the messages generated over many runs of it are indistinguishable from the messages generated over many runs of the simulator.

Association rule mining

We'll use the common data mining task of association rule mining to demonstrate SMC-based privacy-preserving data mining protocols. Association rules are frequent associations or relationships among items or features. A common real-life example is the frequent co-occurrence of beer and diaper purchases in grocery databases. An important data mining task is to find all the "interesting" association rules in a database. We can formally define the association rule mining problem as follows: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called *items*. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Associated with each transaction is a unique identifier, called its *TID*. We say that a transaction T *contains* X, a set of some items in I, if $X \subseteq T$. (X is also known as an item set.) An association rule is an implication of the form, $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X > Y = \phi$. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if c percent of transactions in D containing X also contain Y. The rule $X \Rightarrow Y$ has support s in the transaction set D if s percent of transactions in D contain $X < Y$. Find all association rules in D with support $s > st$ and confidence $c > ct$, where st and ct are user-defined thresholds for "interesting" rules.

The canonical data mining solution to this problem is the Apriori algorithm. For efficiency, it does filtering first on the basis of support, then on confidence—support filtering can be done simply by finding frequent item sets. All the subsets of a frequent item set are guaranteed to be frequent, thus all frequent item sets are found in a bottom up fashion: If an item set is infrequent, no superset of that set can be frequent and we can restrict further search. Once we have an item set's frequency, it's easy to perform the remaining steps of the algorithm.

As an example, consider the database shown in Figure 4. This database has five attributes, $A, \ldots, E$, and 10 transactions, $TID_1, \ldots, TID_{10}$. If vertically partitioned, assume that the database is split between two parties, P_{V1} and P_{V2}, such that P_{V1} has attributes A, B, and C, and P_{V2} has attributes D and E. If horizontally partitioned, assume that the database is split between the two parties P_{H1} and P_{H2} such that P_{H1} has $TID_1, \ldots, TID_5$, whereas P_{H2} has $TID_6, \ldots, TID_{10}$. Figure 4a shows this database, and Figure 4b shows an alternative representation of the database as a *TID* list. Figure 4c shows samples of association rules along with their confidence and support. If the minimum support threshold were set at 30 percent, we'd still find the rule $AD \rightarrow E$, but the rule $AD \rightarrow C$ would be pruned because the item set $<ACD>$ isn't frequent.

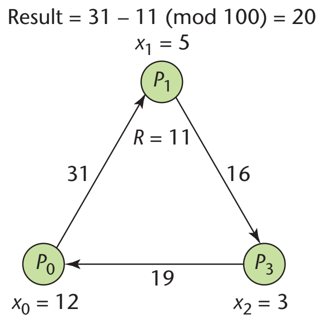


Figure 5. Secure computation of a sum. The addition of the random value *R* hides each site's value from the next site.

Horizontally partitioned data

As noted previously, with horizontal partitioning of data, we assume that different sites have the same attributes, but hold different sets of entities. In this case, an item set's global frequency is the sum of its local frequencies divided by the global number of items. Because our goal is to see if the frequency exceeds a threshold, we can simply sum the number of entities by which each site exceeds or drops below the threshold; if this sum is positive, the threshold is exceeded. We can do this by using a protocol for secure summation. Details on how to efficiently find candidate frequent item sets for which we would evaluate frequency using this technique appear elsewhere.[10]

The *secure sum* protocol is a simple example of an efficient and (information theoretically) secure multiparty computation. Consider the following problem. There exist l parties $P_0 \ldots P_{l-1}$. Each party $P_i (i = 0 \ldots l - 1)$ has a local input x_i. The parties want to securely compute $\sum_{i=0}^{l-1} x_i$, so the number of parties l must be more than two, because for only two parties, the result will reveal the input of the other party, thus eliminating privacy. How can this be done?

Assuming no collusion, the following method securely computes such a sum. Assume that the value $y = \sum_{i=0}^{l-1} x_i$ to be computed is known to lie in the range $[0..n]$. Pick (at random) one site, k, to be the protocol's initiator.

Site k generates a random number R uniformly chosen from $[0..n]$, adds this to its local value x_k, and then sends the sum $R + x_k \pmod{n}$ to site $k + 1 \pmod{l}$. Because the value R is chosen uniformly from $[1..n]$, the number $R + x_k \bmod n$ is also distributed uniformly across this region, so site $k + 1 \pmod{l}$ learns nothing about the actual value of x_k.

For the remaining sites, whenever site z ($z = 0 \ldots l - 1$, $z \neq k$) receives a value from site $z - 1 \pmod{l}$, it adds its local value x_z and then sends the sum on to site $z + 1 \pmod{l}$. Because the value it receives is also uniformly distributed across $[1..n]$ (due to the original random R), site z learns nothing. Finally, site k receives the total sum, $S = R + \sum_{i=0}^{l-1} x_i$, and because it knows R, it can subtract R from S to get the required result.

Any site q can also determine $\sum_{i=0, i \neq q}^{l-1} x_i$ by subtracting x_q. Regardless of how it is computed, site q hasn't learned anything from the global computation. Figure 5 depicts how this method works for three parties.

Obviously, collusion between sites can cause a serious problem. Sites $z - 1$ and $z + 1$ can compare the values they send and receive to determine the exact value for x_z, for example. We can extend the algorithm from earlier to work as long as we have an upper bound on the number of colluding parties. Each site divides its local value into an equal number of shares; this algorithm computes the sum for each share individually. However, the order is changed for each share, such that no site has the same neighbor twice. To compute x_z, z's neighbors from each iteration would have to collude. The number of shares is the security parameter that puts a lower bound on the minimum number of dishonest (colluding) parties required to violate security.

Vertically partitioned data

A solution to finding association rules in vertically partitioned data is to have each site generate a list of the transaction identifiers that contain all the items in the item set about which that site has information, as in Figure 4b. The intersection of these lists is the set of transactions that support the rule. The real goal, however, is simply to find the number of transactions that contain all the items—we don't need to know which transactions these are. Rather, we need to find the size of the intersection without revealing the transaction identifiers in the lists.

To accomplish this, we can use commutative encryption; it has the property that the same data item encrypted with p different keys gives the same cipher text even if the order of encryption differs.

To compute the list intersection's size, each site picks a local key and uses it to encrypt all of its items. It then sends the encrypted list to the next site, which does the same thing. After all the sites have encrypted all the lists, they can intersect them because the encryption is commutative. The resulting intersection set's size gives the required result. None of the sites ever sees an unencrypted item other than its own, and decryption is never performed, so none of the sites learns anything from the other sites.

This description is oversimplified and not truly secure, but it gives a flavor of the solution. We've developed a solution that is both truly secure and more efficient than the simplified version, but the details are beyond this article's scope.[11]

Distributed data mining solutions

Recent research for developing privacy-preserving data mining algorithms focuses almost exclusively on using

SMC. Yehuda Lindell and Benny Pinkas introduced the notion of using SMC techniques for data mining (specifically, classification) by constructing a privacy-preserving ID3 classification algorithm.[12] The fundamental feature of their solution was to develop a secure protocol for computing the logarithm using the Taylor series approximation. Thus, the protocol is fully secure, but it computes an approximate answer that is as precise as required.

Other recent methods include naïve Bayes classification, an ID3 classifier for vertically partitioned data, and clustering on both vertically and horizontally partitioned data.[13]

Advantages and drawbacks

The advantage of an SMC-based solution is that it gives a good idea of exactly what is revealed. In a perfect SMC protocol, nothing is revealed, but in the real world, other more efficient but not completely secure protocols will be used. Nevertheless, the SMC theory enables proofs that clearly delineate what is known and what remains secret.

The most common drawback to using SMC protocols is inefficiency. Generic SMC protocols are impractical for large inputs, which are typically found in data mining. We've implemented the list-intersection-size protocol discussed earlier and used it to construct a secure distributed ID3 classifier for vertically partitioned data. It runs on top of the Weka data mining toolkit; we tested it with University of California, Irvine, machine-learning repository data sets. Although computationally expensive—it took 29 hours to build the 408-node decision tree resulting from the 1,728 item car training set—it's much faster than obtaining approval to view private data, assuming such approval could be obtained. The use of special-purpose encryption hardware (as opposed to a Java implementation) would have given a significant improvement.

It is also necessary to fully understand SMC and the guarantees it gives. Some proofs, for example, depend on the assumption that the result comes from a uniform random distribution over some range, but is this assumption true? If not, is the protocol still secure? We must validate all the assumptions or tailor the protocol to the situation and then reevaluate the proof of protocol's privacy.

Inference from data mining results

One issue that has received little attention in privacy-preserving data mining research has been the implications of inference from the results. Assume two parties collaborate to develop a classifier. *A* holds the target class for each instances and *B* the remaining data (vertical partitioning). A privacy-preserving data mining method would ensure that the training data's classes aren't revealed, but if the resulting classifier is any good, *B* can effectively predict the target class for each instance, negating the privacy-preserving properties.

For some applications, this might be acceptable—US antitrust law, for instance, generally allows the sharing of information if it benefits the consumer. Let's assume that some pharmaceutical companies want to share proprietary information about drug-manufacturing methods to learn association rules about drug reactions. Knowing that the actual use of pharmaceuticals is sometimes affected by cost (sometimes people take less than the recommended dosage of high-cost medications), the companies want to include pricing data in this mix. Although sharing such proprietary information would normally set off antitrust flags, the resulting association rules have a clear benefit to consumers, so sharing the rules would probably be acceptable, even if they can be used to infer proprietary information such as cost or pricing. That said, sharing the data used to learn rules is still questionable; it's where privacy-preserving data mining (such as the methods discussed earlier) shows its value.

In other applications, privacy is paramount. The US Healthcare Information Portability and Accountability Act (HIPAA), for example, doesn't allow companies to compromise privacy just because the results could benefit research (and thus the public). Using personal data, even in a privacy-preserving manner, is likely to be a problem if the results enable someone to infer an individual's private data.

Work related to these issues appears primarily in three areas:

- *Statistical queries.*[14] A statistical database aims to support queries that give summary results, yet prevent the determination of exact values in the database. In statistical queries, the adversary has control over what data is used in the summary; privacy-preserving data mining generally operates over an entire data set.
- *Multilevel secure inference.*[15] The purpose of multilevel secure inference is to ensure that a subset of the database (low data) can't be used to infer exact values of a different subset (high data).
- *Limiting results.*[16] By limiting the results, we prevent the inference of specific rules in the data. Privacy-preserving data mining does the opposite: its goal is to protect the results, not the source data.

Although work in these areas might address the problem of inferring private information from data mining results, none do so directly. Outstanding research challenges abound. First, how do we measure privacy compromise? Most multilevel secure inference work is based on 100 percent accuracy inference, so is privacy compromised if we estimate a value with 90 percent accuracy? What about 10 percent accuracy? Estimation accuracy is not a sufficient metric by itself: for a Boolean value, even estimating with 50 percent accuracy is unlikely to be a privacy compromise. Second, how do we relate data mining models to data compromise? Analyzing an individual association rule, for instance, is straightforward:

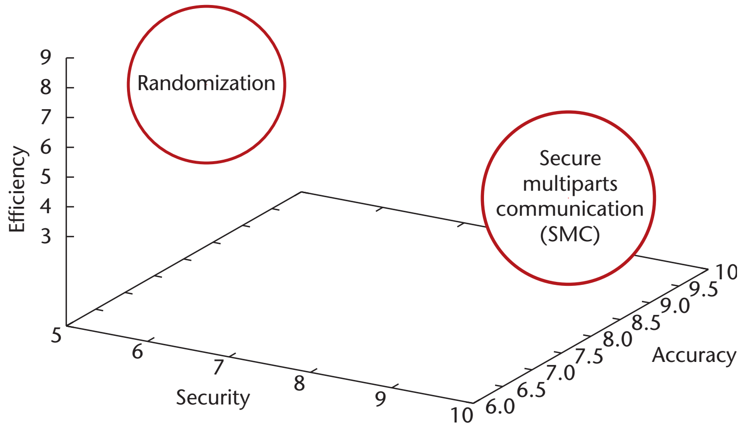


Figure 6. Different approaches to privacy-preserving data mining. Depending on the approach taken, we see trade-offs among efficiency, accuracy, privacy, and security.

given an instance that meets the left-hand side, we have a known confidence that the right-hand side also holds—a simple prediction of the (presumably private) value. Analyzing a set of rules is more difficult, though—what if we know some of the values for an individual, but not all? Some rules apply, others might apply, and others can't, so what does this tell us? Finally, whose privacy are we trying to preserve? Is it just the data used to build the model, or do we impose privacy constraints when the model is used? Although there has been some work in this area, the field is still wide open.

A simple loose notion of privacy is to protect only the actual data values within any transaction—as long as none of the data is known exactly, privacy is preserved. The cryptographic community has formally defined much stronger notions of privacy. Yao,[7] for example, proposes a semantic definition of it: information communicated between parties won't enable one party to compute any polynomial time predicates with greater accuracy than they could without the communication. However, for many functions, coming up with efficient zero-leakage protocols is difficult. Augmenting a function with the information leaked during a candidate protocol makes that protocol fully secure for the modified function, so a minimum knowledge transfer protocol[17] would be a secure protocol for computing a minimally modified version of the original function. This provides a theoretical framework for measuring unintended information disclosure.

Figure 6 shows where the approaches we've described generally fit relative to privacy, result accuracy, and computational efficiency. We can see that the randomization model trades security for efficiency, and that the security definition of the randomization model is much weaker than the one in the SMC model. Specifically, the randomization model aims to protect just the (exact) actual data values. As long as no exact values are learned, privacy is supposedly preserved.

Although the benefits of data mining and analysis are considerable, no matter what the government says, they aren't worth the price of individual privacy. But what if privacy need not be compromised? Data mining and privacy together don't constitute a zero-sum game; ignorance creates problems from both ends of the political spectrum. Neither giving the government complete rights over your data nor banning data mining entirely (as has been proposed) is a wise or widely acceptable solution. We hope this article heralds a new awareness of the possibilities of performing data mining without giving up privacy. Our job as security professionals is to find technical solutions and increase public awareness about them—soon, society can have its cake and eat it too. □

References

1. D. Struck, "Don't Store My Data, Japanese Tell Government," *Int'l Herald Tribune*, 25 Aug. 2002, p. 1.
2. D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.
3. L. Sweeney, *Computational Disclosure Control: A Primer on Data Privacy Protection*, PhD dissertation, Computer Science Dept., Massachusetts Inst. of Technology, 2001.
4. R.A. Moore Jr., *Controlled Data-Swapping Techniques for Masking Public Use Microdata Sets*, Statistical Research Division Report Series RR 96-04, US Bureau of the Census, 1996; www.census.gov/srd/papers/pdf/rr96-4.pdf.
5. V. Estivill-Castro and L. Brankovic, "Data Swapping: Balancing Privacy Against Precision in Mining for Logic Rules," *Proc. 1st Conf. Data Warehousing and Knowledge Discovery* (DaWaK-99), LNCS 1676, Springer-Verlag, 1999, pp. 389–398.
6. R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," *Proc. ACM SIGMOD Conf. Management of Data*, ACM Press, 2000, pp. 439–450; http://doi.acm.org/10.1145/342009.335438.
7. A.C. Yao, "How to Generate and Exchange Secrets," *Proc. 27th IEEE Symp. Foundations of Computer Science*, IEEE CS Press, 1986, pp. 162–167.
8. O. Goldreich, S. Micali, and A. Wigderson, "How to Play any Mental Game: A Completeness Theorem for Protocols with Honest Majority," *Proc. 19th ACM Symp. Theory of Computing*, ACM Press, 1987, pp. 218–229; http://doi.acm.org/10.1145/28395.28420.
9. M. Franklin and M. Yung, "Varieties of Secure Distributed Computing," *Proc. Sequences II, Methods in Communications, Security and Computer Science*, Springer-Verlag, 1991, pp. 392–417.
10. M. Kantarcıoğlu and C. Clifton, "Privacy-Preserving

Distributed Mining of Association Rules on Horizontally Partitioned Data," *IEEE Trans. Knowledge Data Eng.*, vol. 16, no. 9, 2004, pp. 1026–1037; http://csdl.computer.org/comp/trans/tk/2004/09/k1026abs.htm.

11. J. Vaidya and C. Clifton, "Secure Set Intersection Cardinality with Application to Association Rule Mining," to be published in *J. Computer Security*, 2005.
12. Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," *J. Cryptology*, vol. 15, no. 3, 2002, pp. 177–206; http://www.cs.biu.ac.il/~lindell/abstracts/id3_abs.html.
13. C. Clifton et al., "Tools for Privacy Preserving Distributed Data Mining," *SIGKDD Explorations*, vol. 4, no. 2, 2003, pp. 28–34; www.acm.org/sigs/sigkdd/explorations/issue4-2/contents.htm.
14. N.R. Adam and J.C. Wortmann, "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Computing Surveys*, vol. 21, no. 4, 1989, pp. 515–556; http://doi.acm.org/10.1145/76894.76895.
15. T.H. Hinke, H.S. Delugach, and R.P. Wolf, "Protecting Databases from Inference Attacks," *Computers and Security*, vol. 16, no. 8, 1997, pp. 687–708.
16. S.R.M. Oliveira and O.R. Zaiane, "Protecting Sensitive Knowledge by Data Sanitization," *Proc. 3rd IEEE Int'l Conf. Data Mining* (ICDM 03), IEEE CS Press, 2003, pp. 613–616.
17. S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *Proc. 17th Ann. ACM Symp. Theory of Computing* (STOC 85), ACM Press, 1985, pp. 291–304.

Jaideep Vaidya *is an assistant professor of computer information systems at Rutgers University. His research interests include data mining, databases, and privacy/security. He has a PhD in computer science from Purdue University. He is a member of the IEEE Computer Society and the ACM. Contact him at jsvaidya@rbs.rutgers.edu.*

Chris Clifton *is an associate professor of computer science at Purdue University. His research interests include data mining, database support for text, and database security. He has a PhD in computer science from Princeton University. He is a senior member of the IEEE and a member of the IEEE Computer Society and the ACM. Contact him at clifton@cs.purdue.edu.*