# Privacy Preserving Naive Bayes Classifier for Horizontally Partitioned Data

Murat Kantarcıoğlu[*]
Department of Computer Sciences
Purdue University
kanmurat@cs.purdue.edu

Jaideep Vaidya[†]
Department of Computer Sciences
Purdue University
jsvaidya@cs.purdue.edu

## Abstract

*The problem of secure distributed classification is an important one. In many situations, data is split between multiple organizations. These organizations may want to utilize all of the data to create more accurate predictive models while revealing neither their training data / databases nor the instances to be classified. The Naive Bayes Classifier is a simple but efficient baseline classifier. In this paper, we present a privacy preserving Naive Bayes Classifier for horizontally partitioned data.*

## 1 Introduction

Classification is a predictive modeling task with the specific aim of predicting the value of a single nominal variable based on the known values of other variables. There are many practical situations in which classification is of immense use. Examples include: providing a diagnosis for a medical patient based on a set of test results, estimating the probability of purchase of a given item given the other items purchased, and others.

Though there are some organizations which single handedly collect a lot of data on their own, often large (possibly) correlated data is collected over many sites. It is possible that several organizations collect similar data about different people (a.k.a. horizontal partitioning of data). Examples include banks collecting credit card information for their customers or supermarkets collecting transaction information for their clients. On the other hand different organizations may collect different information about the same set of people (a.k.a vertical partitioning of data). Examples of this include hospitals and insurance companies collecting information or producer / consumer industrial concerns collecting data which can be jointly linked. In all of these cases, mining on the local data is simply not as accurate as mining on the global data. It may lead to inaccurate, even improper results. Thus all corporations would like to leverage their data to get useful knowledge.

Privacy concerns restrict the free flow of information. Organizations do not want to reveal their private databases for various legal and commercial reasons. Neither do individuals want their data to be revealed to parties other than those they give permission to. This implies that revealing an instance to be classified may be tantamount to revealing secret information. However, the idea of leveraging other parties' data to have more accurate private classification is quite appealing.

The Naive Bayes classifier is a simple but efficient baseline classifier. It is the de facto classifier used for text classification. Naive Bayes is based on a bayesian formulation of the classification problem which uses the simplifying assumption of attribute independence. It is simple to implement and use while giving surprisingly good results. Thus, preliminary evaluation is carried out using the Naive Bayes classifier to serve both as a baseline and to decide whether more sophisticated solutions are required.

The problem of secure distributed classification is an important one. The goal is to have a simple, efficient and privacy-preserving classifier. The ideal would be for all parties to decide on a model. Jointly select/discover the appropriate parameters for the model and then use the model locally as and when necessary. We discuss the specifics in the context of the Naive Bayes classifier later.

We assume in this work that data is horizontally partitioned. This means that many parties collect the same set of information about different entities. Parties want to improve classification accuracy as much as possible by leveraging other parties data. They do not want to reveal their own instances or the instance to be classified. Thus, what we have is a collaboration for their own advantage. One way to solve this is to decide on a model. The model parameters are generated jointly from the local data. Classification is performed individually without involving the other parties. Thus, the parties decide on sharing the model, but not the training set nor the instance to be classified. This

[*]Equal contribution by both authors. Author names in alphabetical order.

[†]Contact author.

is quite realistic. For example, consider banks which decide to leverage all data to identify fraudulent credit card usage, or insurance companies which jointly try to identify high-risk customers. In this paper, we use / extend several existing cryptographic techniques to create a privacy preserving Naive Bayes Classifier for horizontally partitioned data.

The organization of the paper is as follows: Section 2 briefly describes the related work in this area. Section 3 presents the Naive Bayes classifier. Section 4 gives more detail on secure multi-party computation as well as several definitions required in this work. Section 5 presents the actual privacy-preserving Naive Bayes classifier under a relaxed privacy assumption as well as a proof of security. Section 6 presents an algorithm satisfying more stringent privacy requirements. Section 7 concludes the paper and gives future directions for research.

## 2 Related Work

We now give some of the related work in this area. Previous work in privacy-preserving data mining has addressed two issues. In one, the aim is preserving customer privacy by distorting the data values [2]. The idea is that the distorted data does not reveal private information, and thus is "safe" to use for mining. The key result is that the distorted data, and information on the distribution of the random data used to distort the data, can be used to generate an approximation to the original data *distribution*, without revealing the original data *values*. Since then, there has been work improving this approach in various ways (e.g. [1]).

The data distortion approach addresses a different problem from our work. The assumption with distortion is that values must be kept private from the data mining party. We instead assume that *some* parties are allowed to see *some* of the data, while no one is allowed to see *all* the data. In return, *exact* results can be obtained.

The other approach uses cryptographic tools to build decision trees[10]. In this work, the goal is to securely build an ID3 decision tree where the training set is distributed between two parties. An ID3 classifier on vertically partitioned data is presented in [5]. There has also been other work in privacy preserving data mining on horizontally partitioned data (e.g. [8, 9]) and vertically partitioned data (e.g. [12, 13]). Details and references to more work can be found at [3].

## 3 Naive Bayes Classifier

The *naive Bayes classifier* is a highly practical Bayesian learning method. The following description is based on the discussion of the Naive Bayes classifier in Mitchell[11]. The naive Bayes classifier applies to learning tasks where each instance $x$ is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set $V$. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< a_1, a_2, \ldots, a_n >$. The learner is asked to predict the target value, or classification, for this new instance.

The Bayesian approach to classifying the new instance is to assign the most probable target value, $v_{MAP}$, given the attribute values $< a_1, a_2, \ldots, a_n >$ that describe the instance.

$$v_{MAP} = \underset{v_j \in V}{argmax} \left( P(v_j | a_1, a_2, \ldots, a_n) \right) \tag{1}$$

Using Bayes theorem,

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{argmax} \left( \frac{P(a_1, a_2, \ldots, a_n) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \right) \\ &= \underset{v_j \in V}{argmax} \left( P(a_1, a_2, \ldots, a_n) P(v_j) \right) \end{aligned} \tag{2}$$

The Naive Bayes classifier makes the further simplifying assumption that the attribute values are conditionally independent given the target value. Therefore,

$$v_{NB} = \underset{v_j \in V}{argmax} \left( P(v_j) \prod_i P(a_i | v_j) \right) \tag{3}$$

where $v_{NB}$ denotes the target value output by the naive Bayes classifier.

The conditional probabilities $P(a_i|v_j)$ need to be estimated from the training set. The prior probabilities $P(v_j)$ also need to be fixed in some fashion (typically by simply counting the frequencies from the training set). The probabilities for differing hypotheses (classes) can also be computed by normalizing the values received for each hypothesis (class). Probabilities are computed differently for nominal and numeric attributes.

### 3.1 Training and Classification

The model parameters (viz the probabilities) are computed from the training data. The procedure for computing the probabilities is different for nominal and numeric attributes.

For a nominal attribute $X$, with $r$ possible attributes values $x_1, \ldots, x_r$ the probability $P(X = x_k|v_j) = \frac{n_j}{n}$ where $n$ is the total number of training examples for which $V = v_j$, and $n_j$ is the number of those training examples which also have $X = x_k$.

For a numeric attribute, in the simplest case, the attribute is assumed to have a "normal" or "Gaussian" probability distribution, $\mathcal{N}(\mu, \sigma^2)$. The mean $\mu$ and variance $\sigma^2$ are calculated for each class and each numeric attribute from the training set. Now the required probability that the instance is of the class $v_j$, $P(X = x'|v_j)$, can be estimated by substituting $x = x'$ in the probability density equation.

An instance is classified as per equation 3. Thus the conditional probability of a class given the instance is calculated for all classes, and the class with the highest relative probability is chosen as the class of the instance.

## 4 Secure Multi-party Computation

Substantial work has been done on secure multi-party computation. The key result is that a wide class of computations can be computed securely under reasonable assumptions. We give a brief overview of this work, concentrating on material that is used later in the paper. The definitions given here are from Goldreich[6]. For simplicity, we concentrate on the two-party case. Extending the definitions to the multi-party case is straightforward.

### 4.1 Security in semi-honest model

A semi-honest party follows the rules of the protocol using its correct input, but is free to later use what it sees during execution of the protocol to compromise security. This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. Also a protocol that is buried in large, complex software can not be easily altered.

The formal definition of private two-party computation in the semi-honest model is given in [6]. Computing a function privately is equivalent to computing it securely. The formal proof of this can also be found in [6].

The formal definition essentially says that a computation is secure if the view of each party during the execution of the protocol can be effectively simulated by the input and the output of the party. This is not quite the same as saying that private information is protected. For example, if two parties use a secure protocol to learn a naive bayes classifier, each party will learn the probability $P(a_i|v_j)$ for an attribute value $a_i$ and class label $v_j$. Let us assume that the first party has $10$ instances with label $v_j$ and $5$ of them have $a_i$ as the value of the $i^{th}$ attribute. If $P(a_i|v_j) = 0.5$, from this information, the first site will learn that half of the instances having the label $v_j$ have the value $a_i$ in the $i^{th}$ attribute at the second site. A site can deduce this information by solely looking at its local information and the learned probabilities. Information such as this cannot be protected since it is inferable from the local input and the output. On the other hand, there is no way to deduce the exact number of instances that have the class $v_j$.

In summary, a secure multi-party protocol will not reveal more information to a particular party than the information that can be induced by looking at that party's input and the output. A key result which is also used in this work is the composition theorem. We state it for the semi-honest model.

**Theorem 4.1** *(Composition Theorem for the semi-honest model): Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then there exists a protocol for privately computing g.*

The composition theorem states if a protocol consists of several sub-protocols, and can be shown to be secure other than the invocations of the sub-protocols, if the sub-protocols are themselves secure, then the protocol itself is also secure. A detailed discussion of this theorem, as well as the proof, can be found in [6].

### 4.2 General multi-party secure function evaluation

In 1986, Yao suggested a general secure two-party function evaluation technique [14]. Goldreich et al. extended this to any multi-party function [7]. The generic method is based on expressing the function $f(x_1, \ldots, x_n)$ as a circuit and encrypting the gates for secure evaluation[7]. With this protocol any multi-party function can be evaluated securely in semi-honest model. However, for efficient evaluation, the function must have a small circuit representation. We will not give details of this generic method. In any case, direct application of this method to data mining algorithms is not feasible due to the large size of the inputs.

## 5 Privacy Preserving Naive Bayes

In order to see how a privacy-preserving Naive Bayesian classifier is constructed, we need to address two issues: How to select the model parameters and how to classify a new instance. The following subsections provide details on both issues. The protocols presented below are very efficient. However, they compromise a little on security. At the end of the protocol, all parties learn the total number of instances. In effect, they learn the numerator and denominator for all the fractions computed. For multiple parties, this may not be a serious privacy concern. However, we also present a technical solution to this problem. Thus, in section 6, we present methods which do not reveal anything except the final result.

### 5.1 Computing Required Values / Setup

The procedures for calculating the parameters are different for nominal attributes and numeric attributes. They are described in the subsections below.

### 5.1.1 Nominal attributes

For a nominal attribute, the conditional probability that an instance belongs to a certain class $c$ given that the instance has an attribute value $A = a$, $P(C = c|A = a)$ is given by

$$P(C = c|A = a) \quad = \quad \frac{P(C = c \cap A = a)}{P(A = a)} = \frac{n_{ac}}{n_a} \quad (4)$$

where $n_{ac}$ is the number of instances in the (global) training set which have the class value $c$ and an attribute value of $a$, while $n_a$ is the (global) number of instances which simply have an attribute value of $a$. Thus, the necessary parameters are simply the counts of instances, $n_{ac}$ and $n_a$. Due to horizontal partitioning of data, each party has partial information about every attribute. Each party can locally compute the local count of instances. The global count is given by the sum of the local counts. We use the secure sum protocol (see section 5.2) to secure compute the global count. Assuming that the total number of instances is public, the required probability can simply be computed by dividing the appropriate global sums. Protocol 1 formally defines the protocol.

For an attribute $a$ with $l$ different attribute values, and a total of $r$ distinct classes, $l * r$ different counts need to be computed for each combination of attribute value and class value. For each attribute value a total instance count also needs to be computed, which gives $l$ additional counts.

**Protocol 1** Nominal Attributes

```
Require: $k$ parties, $r$ class values, $l$ attribute values
 1: {$c^x_{yz}$ represents #instances with party $P_x$ having class $y$ and attribute value $z$}
 2: {$n^x_y$ represents #instances with party $P_x$ having class $y$}
 3: {$p_{yz}$ represents the probability of an instance having class $y$ and attribute value $z$}
 4: for all class values $y$ do
 5:   for $i = 1 \ldots k$ do
 6:     $\forall z$, Party $P_i$ locally computes $c^i_{yz}$
 7:     Party $P_i$ locally computes $n^i_y$
 8:   end for
 9: end for
10: $\forall (y, z)$, All parties calculate using the secure sum protocol (see section 5.2), $c_{yz} = \sum_{i=1}^{k} c^i_{yz}$
11: $\forall y$, All parties calculate using secure sum protocol, $n_y = \sum_{i=1}^{k} n^i_y$
12: All parties calculate $p_{yz} = c_{yz}/n_y$
```

### 5.1.2 Numeric attributes

For a numeric attribute, the necessary parameters are the mean $\mu$ and variance $\sigma^2$ for all the different classes. Again, the necessary information is split between the parties. In order to compute the mean, each party needs to sum the attribute values of the appropriate instances having the same class value. These local sums are added together and divided by the total number of instances having that same class to get the mean for that class value. Once all of the means $\mu_y$ are known, it is quite easy to compute the variance $\sigma^2_y$, for all class values. Since each party knows the classification of the training instances it has, it can subtract the appropriate mean $\mu_y$ from an instance having class value $y$, square the value, and sum all such values together. The global sum divided by the global number of instances having the same class $y$ gives the required variance $\sigma^2_y$. Protocol 2 formally describes the protocol.

**Protocol 2** Numeric Attributes

```
 1: {$x_{iyj}$ represents the value of instance $j$ from party $i$ having class value $y$}
 2: {$s^i_y$ represents the sum of instances from party $i$ having class value $y$}
 3: {$n^i_y$ represents #instances with party $P_i$ having class value $y$}
 4: for all class values $y$ do
 5:   for $i = 1 \ldots k$ do
 6:     Party $P_i$ locally computes $s^i_y = \sum_j x_{iyj}$
 7:     Party $P_i$ locally computes $n^i_y$
 8:   end for
 9:   All parties calculate using secure sum protocol (see section 5.2), $s_y = \sum_{i=1}^{k} s^i_y$
10:   All parties calculate using secure sum protocol, $n_y = \sum_{i=1}^{k} n^i_y$
11:   All parties calculate $\mu_y = s_y/n_y$
12: end for
13: {Create $\vec{V} = (\vec{X} - \mu)^2$}
14: for $i = 1 \ldots k$ do
15:   $\forall j, v_{iyj} = x_{iyj} - \mu_y$
16:   $\forall j, v_{iy} = \sum_j (v^2_{iyj})$
17: end for
18: $\forall y$, All parties calculate using secure sum protocol, $v_y = \sum_{i=1}^{k} v_{iy}$
19: All parties calculate $\sigma^2_y = \frac{1}{n_y - 1} * v_y$
```

## 5.2 Secure Sum

The above methods frequently need to calculate the sum of values from individual sites. Assuming three or more parties and no collusion, the following method (example usage in privacy preserving data mining from [8]) securely computes such a sum.

Assume that the value $v = \sum_{i=1}^{k} v_i$ to be computed is known to lie in the range $[0..n]$ where $v_i$ denotes the share of the $i^{\text{th}}$.

One site is designated the *master* site, numbered 1. The remaining sites are numbered $2..k$. Site 1 generates a random number $R$, uniformly chosen from $[0..n]$. Site 1 adds this to its local value $v_1$, and sends the sum $R + v_1 \bmod n$ to site 2. Since the value $R$ is chosen uniformly from $[1..n]$, the number $R + v_1 \bmod n$ is also distributed uniformly across this region, so site 2 learns nothing about the actual value of $v_1$.

For the remaining sites $i = 2..k-1$, the algorithm is as follows. Site $i$ receives

$$V = R + \sum_{j=1}^{i-1} v_j \bmod n.$$

Since this value is uniformly distributed across $[1..n]$, $i$ learns nothing. Site $i$ then computes

$$R + \sum_{j=1}^{i} v_i \bmod n = (v_i + V) \bmod n$$

and passes it to site $i + 1$.

Site $k$ performs the above step, and sends the result to site 1. Site 1, knowing $R$, can subtract $R$ to get the actual result. Note that site 1 can also determine $\sum_{i=2}^{k} v_i$ by subtracting $v_1$. This is possible from the global result *regardless of how it is computed*, so site 1 has not learned anything from the computation.

This method faces an obvious problem if sites collude. Sites $i-1$ and $i+1$ can compare the values they send/receive to determine the exact value for $v_i$. The method can be extended to work for an honest majority. Each site divides $v_i$ into shares. The sum for each share is computed individually. However, the path used is permuted for each share, such that no site has the same neighbor twice. To compute $v_i$, the neighbors of $i$ from each iteration would have to collude. Varying the number of shares varies the number of dishonest (colluding) parties required to violate security.

## 5.3 Evaluation

Since all the model parameters are completely present with all the parties, evaluation is no problem at all. The party which wants to evaluate an instance simply uses the Naive Bayes evaluation procedure locally to classify the instance. The other parties have no interaction in the process. Thus, there is no question of privacy being compromised.

## 5.4 Proof of Security

To prove that the protocols are secure, we utilize the definitions given in the earlier section on Secure Multi-party Computation.

**Theorem 5.1** *Protocol 1 securely computes the probabilities $p_{yz}$ without revealing anything except the probability $p_{yz}$, the global count $c_{yz}$ or the global number of instances $n_y$.*

**Proof:**
The only communication taking place is at steps 11 and 12. At steps 11 and 12, the secure sum algorithm is invoked to compute the global counts $c_{yz}$ and $n_y$. Thus, we simply need to apply the composition theorem stated in Theorem 4.1, with $g$ being the nominal attribute computation algorithm and $f$ being the secure sum algorithm.

**Theorem 5.2** *Protocol 2 securely computes the means $\mu_y$ and variance $\sigma_y^2$ without revealing anything except $\mu_y, \sigma_y^2$, the global sum of instance values for each class $s_y$ and the global number of instances $n_y$, as also the sum $v_y$.*

**Proof:**
The only communication taking place is at steps 9, 10 and 18. At all three of these steps the secure sum algorithm is invoked to compute $s_y, n_y$ and $v_y$. Thus, again, we simply need to apply the composition theorem stated in Theorem 4.1, with $g$ being the numeric attribute computation algorithm and $f$ being the secure sum algorithm.

## 6 Enhancing Security

The protocols given above are not completely secure in the sense that something more than just the model parameters are revealed. The true numerators and the denominators making up the actual parameter values are revealed. For three or more parties, this allows upper bounds on the number of instances with a party and upper bounds on the composition of those instances (i.e. upper bound on the number belonging to a particular class etc.). In any case, privacy of individual instances is always preserved though. With increasing number of parties, it is more difficult to get accurate estimates of the remaining parties. However, with just two parties, this does reveal quite a bit of extra information. In general, the problem is to calculate the value of the fraction without knowing the shared numerator and/or shared denominator. For two parties, Du and Atallah solve exactly this problem under the term of the Division protocol[4]. This is based on a secure scalar product protocol for 2 parties. The protocol is easily extendible to the general case of multiple parties assuming that a general scalar product protocol for multiple parties is available. However, no such protocol has yet been developed.

Note that the amount of information revealed for multiple parties is not much more than what the parameters themselves reveal. However technical solutions (even with increased cost) are more satisfying as they allow an individual decision of whether to trade off security for efficiency.

In the following subsection, we now present a secure protocol based on computing the logarithm securely.

## 6.1 Secure logarithm based Approach

As mentioned above, in order to make our algorithm fully secure (i.e. reveal nothing), we need to evaluate $(\sum_{i=1}^{k} c_i / \sum_{i=1}^{k} n_i)$ securely. Here evaluating the division becomes the main problem. In order to overcome this problem, we can rewrite the above expression as follows:

$$exp\left[ln(\sum_{i=1}^{k} c_i) - ln(\sum_{i=1}^{k} n_i)\right]$$

Therefore evaluating the $ln(\sum_{i=1}^{k} c_i) - ln(\sum_{i=1}^{k} n_i)$ securely, will be enough for our purposes. Clearly, this requires secure evaluation of $ln(\sum_{i=1}^{k} x_i)$ function. In our work, we will use the secure $ln(x)$ evaluation method given in [10]. The one important restriction of their method is that it only works for two parties. In our case, it is easy to reduce the $k$ party problem to the two-party case. Note that the last step in the semi-honest version of the secure summation protocol has the first party subtracting the random number from the result. So just before this subtraction occurs, no party has the summation and nothing is revealed. At this point, instead of subtracting the random number, both parties can use the secure approximate $ln(x_1 + x_2)$ protocol given in [10]. Using their protocol, it is easy to get random $v_1, v_2$ such that $v_1 + v_2 = C.ln(x_1 + x_2) \bmod p$

One important fact to notice about the secure $ln(x)$ evaluation algorithm is that there is a public constant $C$ used to make all the elements used in the protocols integer. The exact value of this $C$ is given in [10]. Also operations are executed in a field with size $p$ that is large enough to fit the actual results multiplied by the constant. Our reduction also requires us to slightly change their protocol, in their protocol the $x_1 + x_2$ is directly added using a small addition circuit, in our case we need a modular addition.(This does not change the asymptotic performance of the method) After using secure logarithm, it is easy to evaluate our desired function securely. The protocol 3 describes how these ideas can be applied to our problem. Here, we only give the protocol for nominal attributes, the similar version can be written for real valued attributes but it is omitted here for space reasons.

**Theorem 6.1** *Protocol 3 securely evaluates $p_{yz}$ in semi-honest model.*

**Proof:**
In order to show that the above protocol is secure in semi-honest model, we will show that each party's view of the protocol can be simulated based on its input and its output.

**Protocol 3** Fully secure approach for nominal attributes

```
Require: $k$ parties, $r$ class values, $l$ attribute values
1:  $\{c^x_{yz}, n^x_y, p_{yz}$ are defined as in Protocol 1$\}$
2:  for all class values $y$ do
3:      for $i = 1 \ldots k$ do
4:          $\forall z$, Party $P_i$ locally computes $c^i_{yz}$
5:          Party $P_i$ locally computes $n^i_y$
6:      end for
7:  end for
8:  $\forall(y,z)$, All parties, use secure sum protocol (see
    section 5.2) until last step for finding
    $c_{yz} = \sum_{i=1}^{k} c^i_{yz}$ and $n_y = \sum_{i=1}^{k} n^i_y$
9:  Let party 1 has $R_c$ and $R_n$
10: Let party $k$ has $R_c + c_{yz} \bmod p$ and $R_n + n_y \bmod p$
11: {Note that last step of the summation has not been executed}
12: Using secure $ln(x)$ protocol, party 1 and $k$ gets random
    $v_1$,$v_k$ s.t,
    $v_1 + v_k = C.ln(R_c + c_{yz} - R_c \bmod p) \bmod p$
13: Using secure $ln(x)$ protocol, party 1 and $k$ gets random
    $u_1$,$u_k$ s.t,
    $u_1 + u_k = C.ln(R_n + n_y - R_n \bmod p) \bmod p$
14: Party $k$ calculates $s_k = v_k - u_k \bmod p$ and sends it to
    party 1
15: Party 1 calculates the $s_1 = s_k + v_1 - u_1 \bmod p$
16: All parties calculate $p_{yz} = exp(s_1/C)$
```

Again, we will use the secure composition theorem to prove the entire protocol secure since our method securely reduces to the logarithm function.

Parties $2, \ldots, k-2$ only see a summation added to some random number. Therefore, as earlier, the simulator for these parties will be a uniform number generator. Note that the probability that they will see some number $x$ during the execution is $\frac{1}{p}$. The simulator will generate the number with the same probability.

For parties,1 and $k$, there is the additional step of computing the logarithm. We have to show that this does not reveal anything either. Let us assume that logarithm protocol returns the random shares as intended.

Now let us define the simulator for the party $k$. Clearly, before the logarithm protocol has started, party $k$ has $R_n + n_y \bmod p$ and $R_c + c_{yz} \bmod p$. These are indistinguishable from a random number drawn from an uniform distribution. The execution of the logarithm protocol can be simulated by using the simulator for the logarithm protocol. The details for this simulator can be found in [10]. After the protocol, party $k$ only sees $u_2, v_2$ which are also indistinguishable from uniform distribution. Therefore all the inputs it sees during the protocol can be easily generated by an uniform random number generator.

If we look at the messages received by the party 1, one set of messages come from the execution of logarithm, then it receives random shares of $u_1, v_1$. Also it receives $(u_2 - v_2) \bmod p$. We can define the simulator for $k$ as follows: First it runs the simulator of the logarithm function, then it generates three random numbers uniformly chosen between $0$ and $p-1$. Please note that $u_2, v_2$ are independent and $u_2 - v_2 \bmod p$ is also uniformly distributed because,

$$
\begin{aligned}
& Pr(u_2 - v_2 = k \bmod p) \\
= & \sum_{v=0}^{p-1} Pr(u_2 = k + v \bmod p | v_2 = v).Pr(v_2 = v) \\
= & \sum_{v=0}^{p-1} Pr(u_2 = k + v \bmod p).Pr(v_2 = v) \\
= & \sum_{v=0}^{p-1} \frac{1}{p^2} = \frac{1}{p}
\end{aligned}
$$

This concludes our proof.

#### 6.1.1 Communication and Computation Cost

Security is not free. In order to evaluate the secure logarithm, we need to make $O(log(p))$ oblivious transfer and total of $O(log(p).t)$ bits must be transferred. ($p$ is the size of the field used and depends on the range of the variables and the precision required in calculating the logarithm; $t$is the security parameter.) Therefore, total number of bits transferred will be $O(log(p).(t + k))$. (k is the number of parties). Since oblivious transfer is much more expensive then addition, $O(log(p))$ oblivious transfers will dominate the computation cost.

## 7 Conclusion

In this work, we show that using secure summation and logarithm, we can learn distributed naive bayes classifier securely. The result of this paper also supports the view that having a few useful secure protocols enables the secure implementation of many distributed data mining algorithms. We are currently developing such a toolkit. As part of future work, we plan on conducting experiments to validate the feasibility and scalability of the approach. Although generic techniques exist to extend any secure algorithm in the semi-honest model (inclusive of the one proposed in this paper) to resist some degree of collusion, specific efficient solutions can be devised for our purposes. We plan on further exploring this direction in the future. With increasing privacy concerns, more data mining algorithms need to be adapted to be privacy preserving.

## References

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 247–255, Santa Barbara, California, USA, May 21-23 2001. ACM.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, TX, May 14-19 2000. ACM.

[3] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, Jan. 2003.

[4] W. Du and M. J. Atallah. Privacy-preserving statistical analysis. In *Proceeding of the 17th Annual Computer Security Applications Conference*, New Orleans, Louisiana, USA, December 10-14 2001.

[5] W. Du and Z. Zhan. Building decision tree classifier on private data. In C. Clifton and V. Estivill-Castro, editors, *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, volume 14, pages 1–8, Maebashi City, Japan, Dec. 9 2002. Australian Computer Society.

[6] O. Goldreich. Secure multi-party computation, Sept. 1998. (working draft).

[7] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[8] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, pages 24–31, Madison, Wisconsin, June 2 2002.

[9] M. Kantarcıoğlu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

[10] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, Aug. 20-24 2000.

[11] T. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1st edition, 1997.

[12] J. Vaidya and C. Clifton. Privacy-preserving $k$-means clustering over vertically partitioned data. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, Aug. 24-27 2003.

[13] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *ACM Transactions on Information Systems Security*, submitted.

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