Scalable Privacy-Preserving Data Mining with Asynchronously Partitioned Datasets

Hiroaki Kikuchi[1], Daisuke Kagawa[1], Anirban Basu[1], Kazuhiko Ishii[2], Masayuki Terada[2], and Sadayuki Hongo[3]

[1] Graduate School of Engineering, Tokai University,
1117, Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan
kikn@tokai.ac.jp, {nagomin03, abasu}@cs.dm.u-tokai.ac.jp
[2] NTT DoCoMo Inc.
3-5 Hikarinooka, Yokosuka-shi, Kanagawa, 239-8536, Japan
{ishiikaz, teradama}@nttdocomo.co.jp
[3] Department of Frontier Information Engineering, Faculty of Advanced Engineering, Hokkaido Institute of Technology 7-15 Maeda, Teine-ku, Sapporo, Hokkaido, Japan, 006-8585

Abstract. In the Naïve Bayes classification problem using a vertically partitioned dataset, the conventional scheme to preserve privacy of each partition uses a secure scalar product and is based on the assumption that the data is synchronised amongst common unique identities. In this paper, we attempt to discard this assumption in order to develop a more efficient and secure scheme to perform classification with minimal disclosure of private data. Our proposed scheme is based on the work by Vaidya and Clifton[1], which uses commutative encryption to perform secure set intersection so that the parties with access to the individual partitions have no knowledge of the intersection. The evaluations presented in this paper are based on experimental results, which show that our proposed protocol scales well with large sparse datasets.

1 Introduction

Privacy-preserving data mining aims to allow computation of useful aggregate statistics over the entire dataset without compromising the privacy of individual data. The parties collaborating to obtain aggregate results may not fully trust each other, such as a Sybil attack[2] resistant recommendation system [3] or the Naïve Bayes classifier [1]. Such parties may also be competitors in the same field, for example companies which may have privacy policies restricting access to each other's customer datasets.

Vertically partitioned data is an important data distribution model often found in real life. For example, Table 1 illustrates two datasets partitioned vertically where attributes A_1 and A_2 are owned by Alice (A); and Bob (B)[4] owns the attribute A_3 and a target class C, which indicates whether or not to play tennis

[4] From this point forward, we use Alice, A, and party A interchangeably; and the same for Bob, B or party B.

Table 1. Synchronously (vertically) partitioned dataset

Alice			Bob	
day	A_1	A_2	A_3	C
1	sunny	hot	high	no
2	sunny	hot	low	yes
3	rainy	hot	high	yes
4	rainy	cool	low	yes

Table 2. Asynchronously partitioned dataset

Alice			Bob		
ID	A_1	A_2	ID	A_3	C
1	sunny	hot	1	high	no
2	sunny	hot	-	-	-
3	rain	hot	3	low	yes
4	rain	cool	4	low	yes
-	-	-	5	high	yes
-	-	-	3	high	yes

on the day. Alice and Bob separately collect the different features, e.g. temperature, humidity, etc. for each day. Collaboratively performing Naïve Bayes classification allows them to accurately predict the decision to play or not, i.e. predict C given A_1, A_2 and A_3, although they can not share each other's datasets.

Syncronous and Asynchronous Partitions: Vaidya and Clifton presented, in [1], a secure protocol for Naïve Bayes classification for vertically partitioned datasets without revealing the individual partitions. Their protocol combines homomorphic public-key encryption algorithm to compute scalar product of two vectors, with the secure function evaluation [4] for comparison of class $c \in C$ in terms of conditional attributes, i.e. $Pr(C = c|a_1, a_3)$.

Their protocol assumes that the input vectors are of the same dimensions. Hence, the partitioned datasets are synchronous with the days (in our example in Table 1) when the attributes are observed. However, datasets may not always be synchronous. For example, the dataset in Table 2 is vertically but asynchronously partitioned, where attributes are stored with common IDs. This type of asynchronous partitions are of frequent occurrences in our daily lives. Examples include some content service providers with common user IDs, while hospitals and pharmacies may share some common patient identities.

Before delving further, we define few terms that we use in this paper:

asynchronous partitions are vertical partitions of a dataset which are more generalised cases of synchronous partitions. Asynchronous partitions do not necessarily exhibit a coherent sequence of data between the partitions, for example, by having missing and duplicate instances, or not being indexed by the same identity column.

index set is a set, denoted by ID, of values for identities of all instances in a dataset or its partition.

The simplest solution to the problem of asynchronously partitioned datasets is to sort instances by IDs so as to make the two datasets consistent by IDs, and then perform secure scalar product protocols on the vectors. However, it is not so easy. Attributes may be missing for certain IDs, e.g. for $id = 2$ in Bob's partition. One ID may have multiple instances, e.g. the 3rd and the 6th instances conflict for the common $id = 3$ in Bob's partition. The most significant issue in asynchronous partitions is scalability. The conventional vector-based approaches are not efficient for datasets in which most instances are empty, i.e. sparse datasets and this leads us to what is often called the *sparsity problem*, e.g. [5].

missing values Datasets may consist of missing values for some IDs, which contributes to low density of data. For example, the density is only 0.03 in "EachMovie" dataset[5] [6]!

duplicate assignment Datasets may assign the same ID to distinct instances. The arbitrary assignment of IDs can make datasets inconsistent.

scalability Vector-based approach requires processing for each element of the input vector even if most elements are empty. Such schemes do not scale well as the computational costs increase dramatically with increases in dataset size in sparse datasets.

Our goal and approaches: The goal of this paper is to construct a privacy-preserving protocol for applying the Naïve Bayes classifier to vertically and asynchronously partitioned datasets, which deals with the issues of (1) missing values, (2) duplicate assignments, and (3) scalablity.

In order to address these issues in asynchronous partitions, we introduce the secure set intersection scheme presented by Agrawal et al. in [7], which uses commutative public-key encryption. The particular advantage of the scheme is that it works only on non-zero elements and is, therefore, appropriate for sparse datasets.

The contributions of this paper are: (1) first work of its kind, to our knowledge, to consider asynchronously partitioned datasets; (2) a secure privacy preserving scheme which scales well with the size of data and works efficiently with sparse datasets; (3) a performance based evaluation of an experimental implementation of the proposed scheme; and (4) an analytical evaluation of the scheme in terms of the how much information is revealed.

Organisation: This paper is organised as follows. In Section 2, we review some of the fundamental concepts and the existing work in privacy-preserving data mining. In section 3, we present our proposed schemeIn Section 4, we evaluate our scheme based on an experimental implementation.

[5] EachMovie dataset in the Grouplens project: http://www.grouplens.org/node/76

2 Building Blocks

2.1 Naïve Bayes Classifier

Naïve Bayes is a widely used classifier based on the Bayes theorem, where the class with the highest likelihood is chosen. Since the algorithm is simple but efficient to implement, it is widely used for many purposes including email spam filtering, prediction of credit scoring, and so on.

Given attributes a_1, a_2, the most probable class variable, c_{MAP}, is determined by

$$\begin{aligned} c_{MAP} &= \underset{c_i \in C}{\text{argmax}} = \text{argmax}_{c_i \in C} Pr(c_i | a_1,\, a_2) \\ &= \underset{c_i \in C}{\text{argmax}} \frac{Pr(a_1,\, a_2 | c_i) Pr(c_i)}{Pr(a_1,\, a_2)}. \end{aligned}$$

Computing c_{MAP}, however, requires computation of the conditional probability for *every combination of* a_1 *and* a_2. This is not realistic because it implies an exhaustive observation of the data instances. Instead, the Naïve Bayes classifier makes a naïve assumption that all attributes are independent, i.e. $Pr(a_i,\, a_j) = Pr(a_i)Pr(a_j)$. The assumption allows us to predict the most likely class variable without requiring the exhaustive combinations of attributes. We can do this as follows:

$$\begin{aligned} c_{NB} &= \underset{c_i \in C}{\text{argmax}}\, Pr(a_1 | c_i) Pr(a_2 | c_i) Pr(c_i) \\ &= \underset{c_i \in C}{\text{argmax}}\, Pr(c_i) \prod_j Pr(a_j | c_i). \end{aligned}$$

2.2 Secure Scalar Product Based Scheme

Vaidya and Clifton proposed a privacy-preserving scheme for the Naïve Bayes classifier in [1]. The method allows two parties, each having access to only one partition of a vertically partitioned dataset to predict the most likely target class for any given instance without revealing data from each other's partitions.

Predicting the most likely class requires evaluation of conditional probabilities of attributes, e.g. a given c_i, i.e. $Pr(a|c_i)$, through collaborative computation by Alice with attribute $a \in A_1$ and by Bob with class $c_i \in C$. By denoting a binary vector $\boldsymbol{a}$ corresponding to the attribute $a \in A_1$ with 1 for which $a =$ 'sunny' in Table 1 and 0 otherwise, i.e. $\boldsymbol{a} = (1, 1, 0, 0)$, we have the conditional probability represented by the scalar product of $\boldsymbol{a}$ and $\boldsymbol{c}$ as:

$$Pr(a|c_i) = \frac{Pr(a,\, c_i)}{Pr(c_i)} = \frac{\boldsymbol{a} \cdot \boldsymbol{c}}{N} \frac{N}{|\boldsymbol{c}|}, \tag{1}$$

The two partitions of the datasets are *synchronous*. Bob computes $Pr(c_i)$ without the help of Alice and both parties securely and jointly compute the scalar product of $\boldsymbol{a}$ and $\boldsymbol{c}$ in illustrated in Algorithm 1[8].

Algorithm 1 Secure Scalar Product

Input: Alice has n-dimensional vector $\boldsymbol{x} = (x_1, \ldots, x_n)$. Bob has n-dimensional vector $\boldsymbol{y} = (y_1, \ldots, y_n)$.
Output: Alice has s_A and Bob has s_B such that $s_A + s_B = \boldsymbol{x} \cdot \boldsymbol{y}$.

1. Alice generates a homomorphic public-key pair and sends the public key to Bob.
2. Alice sends to Bob n ciphertexts $E(x_1), \ldots, E(x_n)$.
3. Bob chooses s_B at random, computes
$$c = E(x_1)^{y_1} \cdots E(x_n)^{y_n} / E(s_B)$$
and send c to Alice.
4. Alice decrypts c to get $s_A = D(c) = x_1 y_1 + \cdots + x_n y_n - s_B$.

After evaluating conditional probabilities for every attribute in every target class, both parties apply the secure logarithm protocol proposed in [9]. Finally, a secure addition and comparison circuit[4] is used to determine the highest value for the target class. Since Yao's protocol[4] is known to be functionally complete at the cost of heavy computational overhead, we cannot directly apply it to a function taking shares (i.e. shares x_i belongs to Alice while shares y_i belongs to Bob) of the scalar products for every attribute to determine the class without the secure logarithm protocol.

The drawback of the protocol is the strong assumption of a *synchronous partition*, i.e. (1) vectors $\boldsymbol{a}$ and $\boldsymbol{c}$ have the same dimension, (2) elements correspond to each other for two vectors. The secure scalar product based methods, hence, can not simply be applied to asynchronously partitioned datasets such as Table 2.

2.3 Naïve Solution to Asynchronous Partition: *Sort-and-Match*

The simplest (but naïve) solution to the problem of applying the aforementioned secure scalar product scheme[1] to asynchronously partitioned dataset is to sort both sides with a unique common identity and then apply the scheme. Ordered by such unique identities, the partitions of the dataset are transformed into constant-dimension binary vectors with 0 for missing instances. Candidates for such common unique identities include transaction IDs, cellphone IDs, customer IDs, amongst others.

Sorting asynchronously partitioned dataset is not deterministic because some distinct entries share the same identities. For example, recall the example in Table 2, where the instance for $id = 2$ is missing while $id = 3$ has two inconsistent instances: one is "low" and the other is "high" for attribute A_3. These incompletenesses and inconsistencies can be addressed by assigning values distributed in the observed statistics, e.g. class variable c for $id = 2$ (missing) is given as:

$$c = \begin{cases} 0 \text{ with probability} & 1/5, \\ 1 \text{ with probability} & 4/5. \end{cases}$$

Similarly, the inconsistent values, such as $id = 3$, can be replaced by *high* or *low* with even chances. Note that these assignments decrease the accuracy of prediction!

Dataset partitions do not always have common unique identities. For instance, certain medical data are maintained with common names and postal addresses. Alternatively, a hash value of some private information such as common name may be used as a pseudo identity. Since the range of secure hash algorithm is sometimes too large to apply the secure scalar product protocol, it may be possible to use only a small portion of hash value for the pseudo identity.

This simple scheme is, however, not scalable with respect to the dimension of vectors. The secure scalar product protocol requires a number of encryptions and modular exponentiations linearly related to the dimensions of the vectors. In order to ensure privacy, every element of the vectors is taken into account even if it is empty, thus resulting in a waste of computational resources for sparse datasets.

3 Proposed scheme

3.1 Idea

In order to perform Naïve Bayes in a scalable way, we introduce a secure set intersection protocol, which allows Alice and Bob with subsets X and Y, respectively, to compute $X \cap Y$ without revealing X or Y. Intersection is an useful primitive for many data mining algorithms and hence has been studied so far in [10, 11]. The scheme presented in [10] uses oblivious polynomial evaluation that suffers from the linear relation between computational cost and the order of the polynomial. It is, therefore, not appropriate for our purpose. For our study, we focus on the scheme presented by Agrawal, et. al. in [7], which uses commutative public-key encryption, which is performed only for active (i.e. not missing) elements and therefore is more appropriate for sparse datasets.

The aforementioned intersection protocol, however, reveals intermediate results to get the final prediction for the class variable, because one party must learn how many elements belong to both Alice and Bob in order to proceed with the protocol. On the other hand, the existing secure scalar product preserves the secrecy about the size of intersection $|X \cap Y|$ through an additional random number (s_B at Step 3 in Algorithm 1). The revealed information is critical to privacy preservation. Therefore, we propose a new secure protocol based on [7] in order to improve both privacy and scalability for privacy-preserving data mining.

3.2 Secure Set Intersection Protocol

Agrawal, et. al. proposed, in [7], a secure intersection protocol using a public-key encryption algorithm that is commutative, i.e. $f(g(x)) = g(f(x))$ and proved its security under assumption of semi-honest model and random-oracle model.

For concrete discussion, we illustrate the scheme in Algorithm 2 using a power function $f_e(x) = x^e \bmod p$ defined under Decisional Diffie-Hellman hypothesis as commutative encryption[6].

Algorithm 2 Secure Intersection Protocol

Input: Alice has subset $X = \{x_1, \ldots, x_{n_A}\}$, Bob has subset $Y = \{y_1, \ldots, y_{n_B}\}$.
Output: Intersection $|X \cap Y|$.

Let Z_q be a multiplicative group with prime order q and H be a secure hash function that maps into range G.

1. Alice chooses random $u \in Z_q$ and send to Bob $H(x_1)^u, \ldots, H(x_{n_A})^u$ in random order.
2. Bob chooses random $v \in Z_q$ and send to Alice $H(y_1)^v, \ldots, H(y_{n_B})^v$ and $(H(x_1)^u)^v, \ldots, (H(x_{n_A})^u)^v$ as well.
3. Alice computes $(H(y_1)^v)^u, \ldots, (H(y_{n_B})^v)^u$ and selects pairs (x_j, y_i) such that $H(y_i)^{vu} = H(x_j)^{uv}$; the number of pairs being the size of intersection $= |X \cap Y|$.

Algorithm 2 with an input of set of n elements requires n hash value evaluation, $2n$ modular exponentiations for each party[7], and n-element set comparison, which runs in $n \log n$ time with any appropriate algorithm. So, total complexity is $O(n) + O(n \log n) = O(n \log n)$, but the most significant cost is that for modular exponentiation. Supposing t_e be a processing time for exponentiations, the cost of $2n$ exponentiations is $2nt_e$.

While the polynomial interpolation based algorithm in [10], known as popular intersection protocol, requires $O(n \log \log n)$ modular exponentiations for oblivious polynomial evaluation, we will show, later in this paper, that the computational cost is considerably large and hence the commutative encryption is proper for large-scale data mining.

3.3 Proposed Protocol: Distorted Intersection

The goal of our protocol is to compute a conditional probability of $c \in C$ given $a \in A_j$, $Pr(c|a)$, where party A has the attribute A_j and B has the target class C. We denote as index sets, X and Y, defined over the ranges of A_j and C, as

$$X_{a,A_j} = \{id \in ID | A_j(id) = a\},$$
$$Y_c = \{id \in ID | C(id) = c\}.$$

For instance, the datasets in Table 2 define the corresponding index sets for $a = sunny$ and $c = yes$ as $X_{sunny,A_1} = \{1, 2\}$ and $Y_{yes} = \{3, 4, 5\}$ respectively.

[6] For easy understanding, we describe a simplified protocol where only Alice learns the results.

[7] $n = max(n_A, n_B) + \epsilon$ where ϵ is a positive integral constant; thus n is bigger than both n_A and n_B

In order to hide the size of intersection from Alice (A), Bob (B) wishes to add random noise to his secret input. However, B does not know which elements belong to the intersection prior to the execution of the protocol. Hence, he makes his own input distorted by discarding some elements with random probability $p = s_B/n_B$ so that A cannot learn the exact size of the intersection without knowledge of the random probability distribution.

With the randomisation step, the resulting size of the intersection is skewed with p as $s_A = |X \cap Y| s_B/n_B$, which is known to A who does not know p; while, B knows p but does not know s_A. Therefore, both parties participate in Yao's secure multi-party protocol to compute the multiplication

$$s_A \cdot \frac{n_B}{s_B} = |X \cap Y|,$$

which gives the conditional probability

$$Pr(X|Y) = \frac{|X \cap Y|}{|C|}.$$

Finally, the prediction of target class for a given instance, c_{NB}, is obtained from Equation 1. Yao's protocol allows them to compare several candidates of the class without revealing any partial intermediate information.

Our proposed protocol is described in Algorithm 3.

Algorithm 3 Distorted Intersection

Input: Alice has subset $X = \{x_1, \ldots, x_{n_A}\}$, Bob has subset $Y = \{y_1, \ldots, y_{n_B}\}$.
Output: shares of intersection, such that $s_A \cdot s_B = |X \cap Y|$.

1. Alice chooses random $u \in Z_q$, computes $H(x_1)^u, \ldots, H(x_{n_A})^u$ and send to Bob in random order. Alternatively, she can sort these values in numerical order.
2. Bob chooses random $v \in Z_q$, computes $H(x_1)^{uv}, \ldots, H(x_{n_A})^{uv}$ and send to Bob in random order.
3. Bob chooses random $s_B (< n_B)$ and for $i = 1, \ldots, n_B$, compute

$$w_i = \begin{cases} H(y_i)^v & \text{with probability} = s_B/n_B, \\ r_i & \text{otherwise,} \end{cases}$$

   where r_i is randomly chosen from Z_q except $H(y_i)^v$ for every i. Then Bob sends $w_1, \ldots, w_{n_B}$ in random order to Alice.
4. Alice finds pairs x_j, y_i such that $H(y_i)^{vu} = H(x_j)^{uv}$, and where s_A is the number of pairs, i.e. $s_A (= |X \cap Y|(s_B/n_B))$.

4 Evaluation

The purpose of our evaluation is to answer the following questions:

Table 3. Processing time for cryptographical primitives (2048 bit Paillier)

primitives	time (sec)
encryption	$t_E = 1.1$
decryption	$t_D = 1.6$
exponentiation	$t_P = 0.15$

- Is the proposed scheme more efficient than [1]?
- What scalability in terms of dimension does the proposed scheme achieve?
- How secure is the proposed scheme for privacy-preservation?

4.1 Performance evaluation

In order to evaluate performance improvement of the proposed scheme in comparison with others, we implement the following schemes for the secure Naïve Bayes classifier:

1. Scalar product based scheme, Vaidya and Clifton [1], which requires a homomorphic encryption and a secure function evaluation of comparison of additively shared value (SFE_1),
2. Set intersection schemes, proposed by Freedman, Nissim and Pinkas [10], requiring a secure polynomial evaluation, and
3. Commutative encryption scheme, proposed in this paper, which requires a homomorphic encryption and a secure function evaluation of comparison of multiplicatively shared value (SFE_2).

Test implementation Our trial experimental system is implemented using Java (SDK 1.6.0) running on Intel Core2 Duo CPU 2.53 GHz, 2GB, Windows 7 (32 bit). We use the Paillier encryption with $|n^2| = 2048$ bit modulus for additive homomorphic property, with a proprietary public key format. Table 3 shows the average processing time of our trial implementation for encryption, decryption and modular exponentiation, denoted by t_E, t_D and t_P, respectively. Note that the cost of decryption is higher than that of encryption because of property of Paillier encryption [12].

Secure Scalar Product The secure scalar product based scheme requires N encryptions and one decryption plus secure function evaluation of comparison of shared sum (SFE_1), i.e.

$$T_1 = Nt_E + t_D + SFE_1$$

where N is the dimension of the vectors. Note that mostly $N \gg n$, where n is the number of active IDs in the asynchronously partitioned dataset. It is also well known that matrix of items and users is sparse for many datasets[6], [13]. We will estimate performance of SFE in a subsequent section. Figure 1 shows processing time for the secure scalar product (without SFE), which bears a linear relation to N, the size of dimension of the vectors.

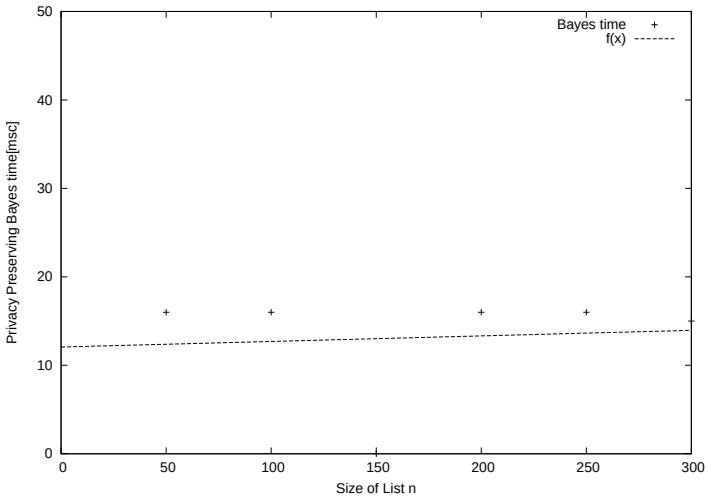

Fig. 1. Processing time for secure scalar product

Secure Set Intersection The secure set intersection protocol[10] takes a large computational cost to evaluate n-degree polynomial, as $T_3 = n^2$ where n is the size of subset of active IDs. Figure 2 illustrates the processing time evaluated in our trial implementation, with 1024 bit modulus, the block size parameter $b = 1, 2, 3$, described in [10]. The protocol is free from the dimension size N, but the squared complexity does not scale well for a practical problem.

Secure Function Evaluation (SFE) We use the generic two-party secure function evaluation evaluation system, Fairplay[14]. Fairplay consists of a compiler of a high level procedural definition language, SFDL, into a one-pass Boolean circuit in a language called SHDL.

With Fairplay, we can perform secure function without revealing inputs. Figure 3 is the source code 'SharedCmp' to securely test $s_{A0} + s_{B0} > s_{A1} + s_{B1}$ where s_{A0}, s_{A1} are owned by Alice and values s_0 and s_1, additively shared as $s_0 = s_{A0} + s_{B0}$ are compared. The bit size is 16.

The example shows that Fairplay allows us to code arbitrary functions easily. However, due to the processing cost, multiplication and division are not provided as primitive operations[14]. We have to code those as programmed functions to perform comparison for multiplicatively shared values as $s_{A0} \cdot s_{B0} > s_{A1} \cdot s_{B1}$. (In our trial implementation, we omit the division since we can replace it by multiplication with some constant).

Table 4 shows the average processing time measured by Alice and Bob for several classes. Both parties have almost the same overhead to jointly evaluate comparison. In this experiment, we use 16-bit integers.

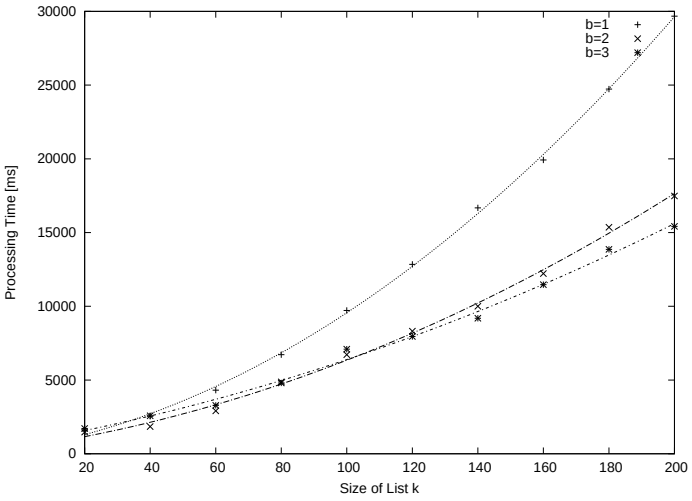

Fig. 2. Processing time for oblivious polynomial evaluation[10]

Table 4. Processing Time SFE_1 (Shared Comparison)

	Alice	Bob
Mean processing time (sec)	0.80	0.82

Table 5 gives our estimation of performance for SFE_1 (addition) and SFE_2 (multiplication) based on the experimental measurement. Based on the runtime complexity, the curve fitting polynomial in terms of size of input x and the processing time when $x = 10bits(= 1024)$ are given. Figure 4 illustrates the estimation. We observe that the cost for secure multiplication increases with respect to the input size, and hence the our proposed scheme has a considerable large constant time overhead.

Scalability of Proposed Scheme Our proposed Algorithm 3 requires as many encryptions as the number of active users, n, and runs in time

$$T_2 = 2t_P n + t_c n \log n + SFE_2,$$

where t_c is the cost of comparison of n size lists. We may omit the overhead for comparison because $t_c \ll t_e, t_d$. Since $n \ll N$, it runs faster than the secure

Table 5. Processing Time for SFE_1 (addition) and SFE_2 (multiplication) with size of 10 bit (= 1024) interger

circuit	fitting	Time (sec)
SFE_1 (addition)	$0.97 + 0.106x$	2.03
SFE_2 (multiplication)	$1.77 + 0.003e^{0.89x}$	23.76

```
program SharedCmp {
  const size = 20;   type int = Int<16>;
  type AliceInput = int[size]; type BobInput = int[size];
  type AliceOutput = int;  type BobOutput = int;
  type Output = struct {AliceOutput alice,   BobOutput bob};
  type Input = struct {AliceInput alice,   BobInput bob};
  function Output output(Input input) {
    if(input.alice[0] + input.bob[0] > input.alice[1] + input.bob[1]){
      output.alice = input.alice[0];
      output.bob = input.bob[0];
    }else{
      output.alice = input.alice[1];
      output.bob = input.bob[1];
    }
  }
}
```

Fig. 3. Fairplay program (in SFDL) 'SharedCmp': shared integer comparison $s_{A0} + s_{B0} > s_{A1} + s_{B1}$

scalar product based protocol (T_1). However, the constant overhead for secure function evaluation of multiplicatively shared values (SFE_2) is higher than that of additive shared values (SFE_1). The proposed protocol is scalable in terms of the entire size of dataset, N, but suffers the constant overhead of SFE.

Therefore, we conclude that the proposed scheme is efficient only for large sparse datasets such that

$$N^* \geq \frac{SFE_2 - SFE_1 - t_D}{t_E - 2t_P\alpha},$$

where $\alpha = n/N$ assuming n is proportional to the entire size of dataset. We illustrate the scalability of our proposed scheme in Figure 5, where the proposed one is more efficient than the secure scalar product based scheme [1] when N is large. Most of asynchronously partitioned datasets are considered as ones with small fractions of intersection. Consequently, we can say that the proposed scheme improves the performance for large scale sparse datasets.

4.2 Security

In [7], assuming the random oracle model and no hash collisions, and in semi-honest model, there is no polynomial-time algorithm that can distinguish between a random value and $H(x)^u$ given x. This means that Algorithm 2 preserves the privacy of input subsets X and Y. With zero-knowledge proof, the security in the random oracle model can even be extended to a malicious model where parties behave arbitrarily.

Algorithm 1 is also proved as secure even after one party (Alice) learns the result of the protocol, $s_A = x_1y_1 + \cdots + x_ny_n - s_B$, which is randomised with s_B chosen uniformly by the other party (Bob). More formally, learning partial

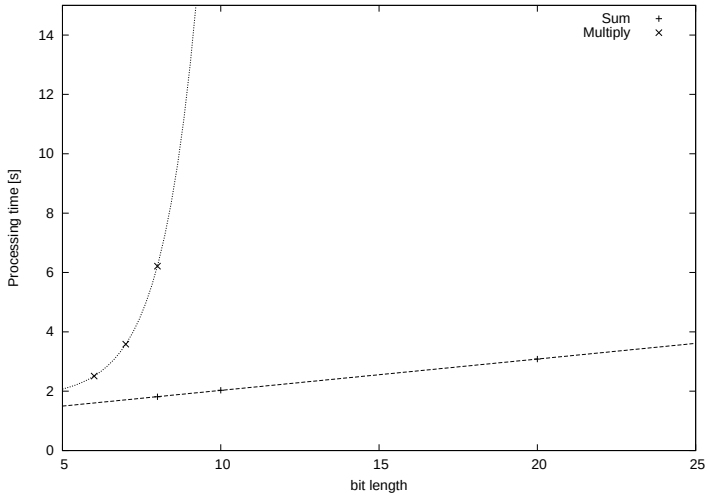

Fig. 4. Processing Time in Yao's SFE (Secure Function Evaluation) for sum and multiplication

result s_A reveals nothing about the distribution of $s_A + s_B$, which is distributed uniformly over group Z_q of order q, that is, the conditional probability of the sum given s_A is identical to apriori probability, i.e. $Pr(s_A + s_B|s_A) = Pr(s_A + s_B) = 1/q$. SFE also preserves the secrecy of shared inputs under the assumptions of semantically secure public key algorithm.

However, the distortion in Algorithm 3 is not uniform. Let z be the size of intersection $|X \cap Y|$, and p be a probability to apply commutative encryption in the algorithm, defined as $p = s_B/n_B$. The conditional probability of the algorithm outputs s_A given z is computed with the binomial distribution as:

$$Pr(s_A|z) = \begin{cases} 0 & \text{if } s_A > z, \\ \begin{pmatrix} z \\ s_A \end{pmatrix} p^{s_A}(1-p)^{z-s_A} & \text{otherwise.} \end{cases} \tag{2}$$

Then, what can Alice guess about z after she learns the output of the algorithm, s_A?

Bayes theorem gives to her an useful hint, i.e. the probability distribution of z as

$$\begin{aligned} Pr(z|s_A) &= \frac{Pr(s_A|z)Pr(z)}{Pr(s_A)} \\ &= \frac{Pr(s_A|z)1/(n+1))}{\sum_z Pr(s_A|z)Pr(z)}, \end{aligned}$$

where we assumes $Pr(z) = 1/(n+1)$. Figure 6 illustrates the skewed distribution of z given s_A, for $s_A = 0, 3, 6$ and $n = 10$, $p = 0.5$. We observe that probability

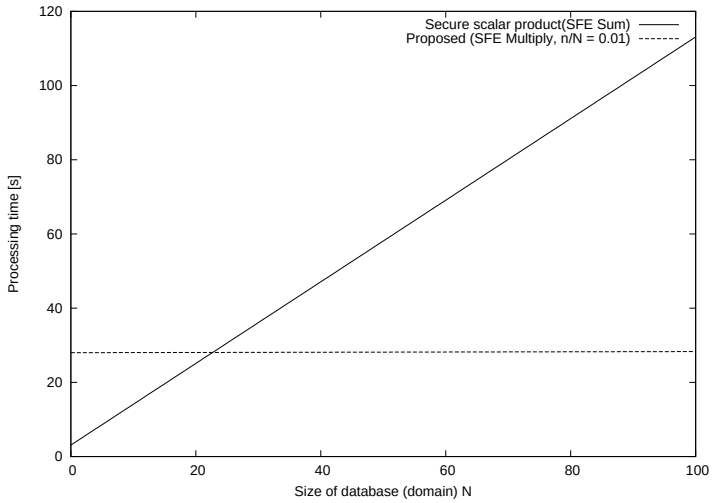

Fig. 5. Scalability in terms of processing time for the size of database (set of indexes) N

distribution of z is dependent on the value of s_A. The entropy of z is reduced if we have extreme values, while the entropy is preserved for more moderate values, e.g. $s_A = 3$.

The analysis assumes Alice already knows the probability $p = s_B/n_B$ chosen by Bob. Since Alice has no idea about s_B, the probability is uniformly distributed over $[0, 1]$ and hence $Pr(p) = 1/n_B$. For example, the distribution of z given $s_A = 3$ and $p = 1/2$ is distributed with the most likely value of $L(z) = 6 = s_A/p$ in Figure 6. If $p = 1/3$, the distribution will be shifted to more to the right, with the most likely value $L(z) = 9 = 3s_A$. Accordingly, the distribution of z becomes flat with many possible value p and therefore the overall entropy is preserved in our scheme.

5 Conclusion

We have proposed a scalable privacy-preserving Naïve Bayes classifier for asynchronously partitioned datasets. Our proposed scheme is based on the work presented in [7] using a public-key encryption algorithm that satisfies commutative property. The performance of our proposed protocol is shown to be better than the scheme based on the secure scalar product [1] when the matrix is sparse, i.e. most entries are missing and the fraction of active data is small, that is $n \ll N$, which frequently happens in asynchronously partitioned datasets. Table 6 gives the summary of the features of our proposed protocol.

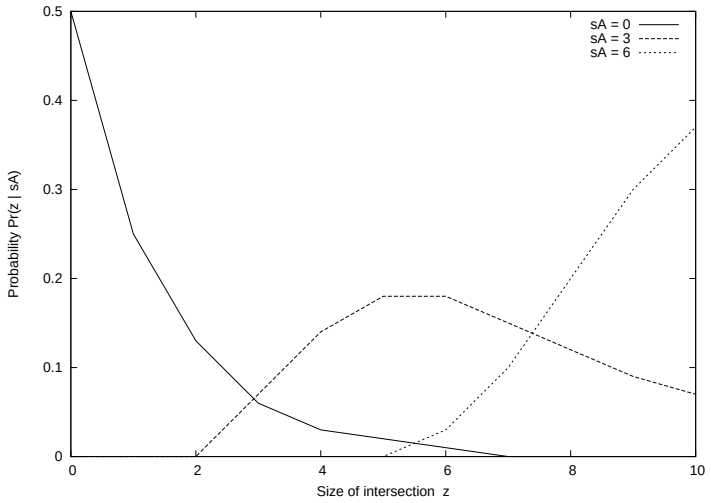

Fig. 6. Probability distribution of size of intersection z given s_A ($n = 10$, $p = s_B/n_B = 0.5$, uniform apriori probability $Pr(z) = 1/(n+1)$)

Table 6. Summary of proposed scheme

scheme:	Vaidya & Clifton [1]	Proposed
based on:	Secure Scalar Product[8]	Commutative encryption [7]
input	N-dimension binary vectors	integer subset of size n
computation cost	$T_1 = t_E N + t_D + SFE_1$	$T_3 = t_P n + n \log n + SFE_2$
accuracy	accurate	accurate with probability s_B/n_B
security	$Pr(z\|X.Y) = 1/N$	$Pr(z\|s_A) > 1/n$

References

1. Vaidya, J., Clifton, C.: Privacy Preserving Naïve Bayes Classifier for Vertically Partitioned Data. In: SIAM International Conference on Data Mining, Lake Buena Vista, Florida, Society of Industrial and Applied Mathematics (2004) 522–526
2. Douceur, J.: The Sybil attack. Peer-to-peer Systems (2002) 251–260
3. Yu, H., Shi, C., Kaminsky, M., Gibbons, P.B., Xiao, F.: DSybil: Optimal Sybil-resistance for Recommendation Systems. In: 30th IEEE Symposium on Security and Privacy, IEEE (2009) 283–298
4. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, IEEE (1986) 162–167
5. Zhou, J., Luo, T.: A novel approach to solve the sparsity problem in collaborative filtering. In: International Conference on Networking, Sensing and Control (ICNSC), IEEE (2010) 165–170
6. GroupLens: GroupLens Research. http://www.grouplens.org/ (2010)
7. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: The ACM SIGMOD International Conference on Management of Data, ACM (2003) 86–97

8. Du, W., Atallah, M.J.: Privacy-preserving cooperative statistical analysis. In: 17th Annual Computer Security Applications Conference, ACSAC., IEEE (2001) 102–110
9. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology **15**(3) (2008) 177–206
10. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Advances in Cryptology – EUROCRYPT'04, Springer (2004) 1–19
11. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security **13**(4) (2005) 593–622
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology – EUROCRYPT'99, Springer (1999) 223–238
13. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. (1998) 43–52
14. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – a secure two-party computation system. In: The 13th USENIX Conference on Security Symposium, USENIX Association (2004) 20
15. Kikuchi, H., Kizawa, H., Tada, M.: Privacy-Preserving Collaborative Filtering Schemes. In: International Conference on Availability, Reliability and Security, ARES'09., IEEE (2009) 911–916
16. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: The 10th international conference on World Wide Web, ACM (2001) 285–295