Privacy-Preserving Incremental Bayesian Network Learning

Saeed Samet
CHEO Research Institute
Ottawa, Ontario, Canada
ssamet@ehealthinformation.ca

Ali Miri
Ryerson University
Toronto, Ontario, Canada
Ali.Miri@ryerson.ca

Eric Granger
Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA)
*É*cole de technologie supérieure, Montreal, Canada
Eric.Granger@etsmtl.ca

Abstract

Bayesian Networks (BNs) have received significant attention in various academic and industrial applications, such as modeling knowledge in image processing, engineering, medicine and bio-informatics. Preserving the privacy of sensitive data, owned by different parties, is often a critical issue. However, in many practical applications, BNs must train from data that gradually becomes available at different period of times, on which the traditional batch learning algorithms are not suitable or applicable. In this paper, an algorithm based on a new and efficient version of Sufficient Statistics is proposed for incremental learning with BNs. The standard $\mathcal{K}2$ algorithm is also modified to be utilized inside the incremental learning algorithm. Next, some secure building blocks such as secure multi-party multiplication, comparison, and factorial, which are resistant against colluding attacks and could be applied securely over public channels like internet, are presented to be used inside the main protocol. Then a privacy-preserving protocol is proposed for incremental learning of BNs, in which the structure and probabilities are estimated incrementally from homogeneously distributed and gradually available data among two or multi-parties. Finally, security and complexity analyses along with the experimental results are presented to compare with the batch algorithm and to show its performance

Preprint submitted to Applied Soft Computing *June 26, 2011*

The final publication is available at http://dx.doi.org/10.1016/j.asoc.2013.03.011

and applicability in real world applications.[1]

Keywords: Security and Privacy Preserving, Bayesian Networks, Incremental Learning, Data Mining and Machine Learning

1. Introduction

Bayesian Networks are probabilistic graphical models [8] that are trained to represent the relationship between variables from a dataset [6]. Medical diagnosis applications, fraud detection systems, and financial networks widely utilize such networks to create models and make decision according to the probabilistic independencies among the variables of the underlying databases [7]. For instance, Fenton and Neil in [11] show the successful application of Bayesian Networks in risk management.

According to the privacy regulations such as Freedom of Information and Protection of Privacy Act (FIPPA) [34] in Canada, or the Health Insurance Portability and Accountability Act (HIPAA) [35] in the United States, individual's private and sensitive data must be secured when protocols are applied on data used to train BNs. To create the BNs structure and parameters using training data which is securely shared among two or more parties, they cannot simply present their own private data to each other, or even to a third party to run a learning algorithm on the whole data. Therefore, privacy-preserving protocols are needed to apply in these situations.

In many practical applications, BNs must be trained using data that becomes available at different points in time. The traditional techniques for training BNs (e.g. $\mathcal{K}2$ algorithm) are batch in nature, and are not suitable for training on data that arrives incrementally. To obtain a high level of performance, using a batch technique would involve accumulating all training data in memory, and recreating a new BN from scratch using all cumulative data. The time and memory complexity of retraining on all data would be prohibitive in applications with large amounts of training data. For instance, selling records in Walmart, as a chain of large stores, are gradually grow everyday and is not reasonable to store all data and run the data mining and machine learning algorithms on all data every time a block of new data becomes available. This is an ubiquitous scenario in many different fields such as healthcare systems, government applications and so on. Therefore, incremental learning is needed to efficiently update BNs on new data, in terms of data storage and processing time.

In this paper, BNs structure is incrementally constructed each time a block of new training data is available by updating the sufficient statistics of the existing network structure, and a new structure is created accordingly. Note that by updating sufficient statistics, the probability table of each node could also be computed. After reviewing the existing techniques for incremental learning

[1]Some of the material in this paper has been presented at the 2009 IEEE International Conference on Privacy, Security, riSk and Trust (PASSAT-09).

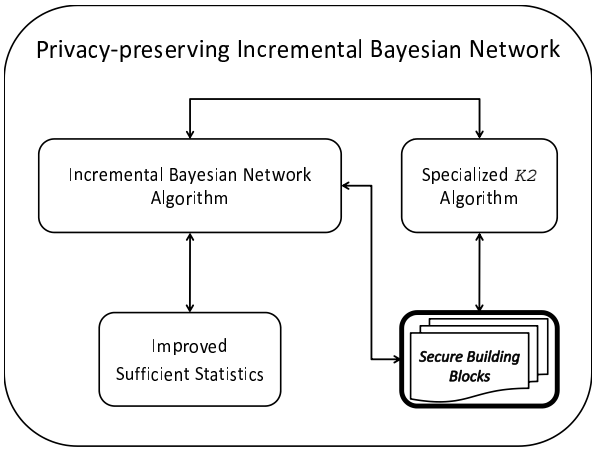

Figure 1: Overview of the Proposed Privacy-Preserving Incremental Bayesian Network Learning.

of BNs, first an improved version of sufficient statistics, in terms of required storage and search time, is proposed followed by a specialized $\mathcal{K}2$ algorithm to be applied in our incremental learning algorithm for BNs. After presenting that algorithm, a privacy-preserving protocol and required secure building blocks, such as secure multi-party multiplication, comparison, and factorial, are proposed along with their security and complexity analyses and experimental results. Figure 1 illustrates the contribution of the paper and the relations of its components. Inside the main protocol of privacy-preserving incremental BNs, the new incremental learning algorithm is used to update the BNs structure by using an improved version of sufficient statistics and calling the specialized $\mathcal{K}2$ algorithm. Also secure building blocks are utilized inside the protocol to maintain the privacy of the parties involved. These building blocks, which are indicated as a bold box in Figure 1, are previously proposed by the same authors in [10].

As a summary, each time a block of new data becomes available, Incremental BNs algorithm is called with the ordered list of nodes, sufficient statistics of the previous data, the set of previous candidate lists of parents, and the new data. Its output is a directed acyclic graph for the BNs and the new updated sufficient statistics. Inside this algorithm, the specialized $\mathcal{K}2$ algorithm is called when needed with the current node, its parents set, sufficient statistics of that node, its predecessors, and the set of candidate parents lists as the inputs. The output of this algorithm will be the new parents set of the node and its updated candidate parents lists. Each time we need to compute the score function in those two algorithms, secure building blocks are used to securely compute this function without revealing private data of each party to the others.

With this privacy-preserving protocol, it is assumed that data is homoge-

nously shared and owned by several parties, while these parties want to keep their sensitive data private. The protocol is also secure against colluding attacks and could be run over public channels. We show the applicability and efficiency of the proposed protocol by testing it against several different datasets, various number of parties involved and reasonable encryption key sizes, 512, 1024 and 2048 bits, to keep the privacy of the protocol strong. Also, the comparison of the final results of the incremental learning with the batch algorithm, in terms of efficiency and accuracy, shows its great promise to be applied in real world applications.

The rest of the paper is organized as follows. In Section 2, the BN is briefly introduced, along with a brief survey of training algorithms and privacy-preserving BNs. An improved version of sufficient statistics, an incremental algorithm for learning BN structure and a modified $\mathcal{K}2$ [3] algorithm which could exploit that algorithm presented in Section 3. A privacy-preserving protocol for Incremental Bayesian Networks is presented in Section 4, followed by the experimental results in Section 5.

2. Techniques for Training BNs

Bayesian Networks, or Belief Networks, are Directed Acyclic Graphs (DAG) encoding probabilistic relations or dependencies among a set of variables. Each node of a graph represents a variable, and an arc from one node to another node shows a conditional dependency between them. Thus, a BNs structure for a set of variables is formally shown by a pair (N_s, N_p), in which $N_s = (V, E)$ is a DAG containing the set of nodes, V, and the set of edges, E. The set of probability distributions, N_p, which is called the parameters of the BNs, is defined as $N_p = \{p(x_i|\pi_i), x_i \in V\}$, where π_i is the set of x_i's parents and $p(x_i|\pi_i)$ is the probability distribution of x_i conditional upon its parents, π_i.

Constructing BNs is NP-hard [2]. There are different batch algorithms for this learning system. CL algorithm proposed by Chow and Liu [16] estimates the underlying n-dimensional discrete probability distribution from a dataset. To approximate the probability distribution, this algorithm generates the product of $n - 1$ second order distributions.

Lam and Bachus [14], and Friedman and Goldszmith [24] use the Minimum Description Length (MDL) principle [23, 22] as their approach to propose their own learning algorithm for BNs. In MDL a database is modeled with the minimum length of encoding. Using this approach, BN is encoded as a model with the minimum bit length.

Bouckaert [25] presents a heuristic algorithm, called B, which uses a hill-climbing search method to generate BN structure, and variables do not need to be sorted at the beginning of the algorithm.

Castelo in [26], and Castelo and Coč̆ka in [27] propose algorithm HCMC, which is similar to algorithm B, because of utilizing a hill-climbing search method on DAGs. However, unlike algorithm B it considers the inclusion order among BNs. In [28], two protocols are proposed for privacy-preserving

Naive Bayes, in both horizontally and vertically partitioned data using secure multi-party computation techniques such as secure sum [32] and secure dot product [33]. Meng et al. in [29] proposed a privacy-preserving estimation for the BN parameters, by assuming that the Bayesian network structure has been already created, and is publicly known to the parties involved. Also, in [30], authors proposed a secure technique to compute the BN parameters on vertically partitioned data using secure multi-party computation sub-protocols, based on their previously presented protocol in [31] which securely developed the BN structure.

$\mathcal{K}2$ algorithm is proposed by Cooper and Herskovits [3]based on a hill-climbing heuristic search to find an optimized BN structure. The $\mathcal{K}2$ algorithm starts with a graph of nodes showing the variables of interest, without any edges. Then, for each node, using a canonical order and a score function, edges by which the score of the graph increases are added as the parents of the current node. This process ends when no more parents can be added or the number of parents reaches a specified threshold, and the next node will be processed in order.

In this paper the $\mathcal{K}2$ algorithm shown in Algorithm 1 is used as a base algorithm for the creation of BNs structure. In this algorithm, π_i is the set of parents of a variable x_i. At each step, score value of the Predecessor nodes of the current node, x_i, is computed and the node, with the maximum score, say z, is selected and compared with the current score of x_i. If it is greater, z will be added to the x_i's parents set, and otherwise **while** loop will end, and the program continues for the next variable. The score function $f(i, \pi_i)$ is defined

Algorithm 1 $\mathcal{K}2$ Algorithm

1. **Input**: A dataset D, a set of m nodes with an assumed order, and an upper bound u on the maximum number of parents of each node
2. **Output**: A directed acyclic graph for the Bayesian network
3. **for** $i = 1$ to m **do**
4. $\pi_i = \emptyset$
5. $Score_{old} = f(i, \pi_i)$
6. OkToProceed = **true**
7. **while** OkToProceed **and** $|\pi_i| < u$ **do**
8. z = the node in Pred(x_i) - π_i that maximize $f(i, \pi_i \bigcup \{z\})$
9. $Score_{new} = f(i, \pi_i \bigcup \{z\})$
10. **if** $Score_{new} > Score_{old}$ **then**
11. $Score_{old} = Score_{new}$
12. $\pi_i = \pi_i \bigcup \{z\}$
13. **else**
14. OkToProceed = **false**
15. **end if**
16. **end while**
17. **end for**

as follows:

$$f(i,\pi_i)=\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}\alpha_{ijk}! \tag{1}$$

where q_i is the number of items in the list of all possible instantiation of π_i in D, and r_i is the number of possible values of x_i, which is two for binary attributes. In 1, α_{ijk} is the number of records in D in which x_i has its kth value and its parents are instantiated with the jth instantiation in the set of Cartesian product of all possible values of π_i, and $N_{ij}=\sum_{k=1}^{r_i}\alpha_{ijk}$.

2.1. Incremental learning techniques for BNs

When new data becomes available, the $\mathcal{K}2$ Algorithm used to train BNs is not efficient since it must be applied on whole dataset, old and new data. Therefore, online and incremental algorithms have been introduced for applications in which new training data arrives at different point in time. The main ideas behind all these incremental learning algorithms are very similar. Storing a set of candidate parents is almost the same as storing a frontier list of candidate networks.

Friedman and Goldszmith in [12] introduce the concept of Sufficient Statistics for the BNs structure, to extract its probability distribution parameters. This concept is defined as follows: suppose $\mathbf{X}$ denotes a vector of variables, and $N_{\mathbf{X}}^{D}(\mathbf{x})$ be the number of records in the dataset D such that $\mathbf{X}=\mathbf{x}$. The vector $\hat{N}_{\mathbf{X}}^{D}(\mathbf{x})$ containing $N_{\mathbf{X}}^{D}$ for all possible values of $\mathbf{X}$, is called the sufficient statistics of $\mathbf{X}$.

Now, by using this concept and decomposability property of the score assigned to a BN structure of a dataset D, we only need to keep the sufficient statistics of each node X_i and its possible parent sets $Pa(X_i)$, $N_{X_i,Pa(X_i)}^{D}$, to learn the parameters and also to compute the score function of each node and its parents, to create the BN structure. Cardinality of sufficient statistics, $|SS(G)|$, have been discussed in some research papers such as [18]. If we denote M as the maximum number of parents a node can have, r as the number of nodes, and A as the average number of possible values a node takes, then for a BN structure G:

$$|SS(G)|=r\sum_{i=0}^{M}\left(\begin{array}{c} r \\ i \end{array}\right)A^i. \tag{2}$$

Friedman and Goldszmidt [12] propose an approach for incremental learning, in which they rely on a frontier set of networks, consisting of all the networks compared in each iteration of the algorithm. The procedure, by maintaining a set of sufficient statistics (explaining in detail in the next section) records, selects the structure with the best score among the frontier set of networks. The frontier set is updated every time algorithm updates the Bayesian network structure.

In [13] a generalized approach of $\mathcal{K}2$ algorithm is first proposed, and then some guidelines are presented to convert that algorithm into an incremental approach. For each node, a set of parents is maintained and is classified as alive, asleep, and dead using three different thresholds. Parents sets are stored in a lattice structure. Two situations are considered for running incremental algorithm. In the first case, there is a short amount of time for updating the BNs and therefore a rapid update is applied in which the posterior probabilities are only updated and parents set are not touched. In the second situation, both structure and parameters are updated according to new data.

Lam and Bachus in[14] extend their own batch algorithm based on Minimum Description Length (MDL) approach to incrementally adapt BNs structures when new data is available. The MDL principle is based on the idea that if the encoding length of a model and its underlying data is minimum then it is the best model of that database. There is an implicit assumption in their approach that the new structure is very similar to the current one. In each iteration of the algorithm, description length of the whole structure is improved by minimizing the description length of the subgraph whose topology is changed by the new data. Thus, a partial structure is learned from the current structure using MDL, and the new data which includes records containing a subset of the nodes of the BNs.

Roure, in [15], presents incremental approaches for some BNs, such as Chow and Liu (CL) [16], $\mathcal{K}2$, and Buntine (B) [25]. Here we only consider the algorithm of $\mathcal{K}2$. Instead of maintaining a set of parents for each node, a set of candidate parents lists is kept, such that in the k-th list some nodes are ordered as the candidates for the k-th parent of that node in decreasing order, and the first item in the list is the current k-th parent of the node. The number of candidates in each list is a parameter of the algorithm as well as the number of variables that compared with the current parent to make sure that the current parent is still the best candidate inside the list. During each iteration of the algorithm, when no revision is needed for the current parents, the algorithm checks for possible new parent for each node.

However, the algorithm proposed in [15] has some issues in terms of candidate parents set. For example, if the current k-th parent, say x_k of a node can no longer be its parent because of the new data, its subsequent parents could not be tested because when the score of these nodes are computed, x_k should be considered as the k-th parent which is not the case. Also, it only searches new parents for each node only if its current parents are correct. However, a node could have new parent(s) by considering new data even the currents parent would be still correct. Another issue is that each parent could be compared with other candidates by considering new parents set and not the current one.

3. A New Incremental $\mathcal{K}2$ Algorithm

In the computation of sufficient statistics, shown in Section 2, all the possible parent sets for all the nodes have been counted. However, by considering some properties of the Bayesian network structure it could be optimized. The first

property is that in some widely used algorithms such as $\mathcal{K}2$, a total order is defined for the existing nodes, reducing the search space to create direct acyclic graph of the network structure.

The second property is that the values of $N^D_{X_i,Pa(X_i)}$ for the nodes which cannot have the maximum number of parents could be computed from the corresponding information of their subsequent nodes. For instance, suppose the number of parties is 5 and the number of maximum parents is 3. Then, vector values of the nodes X_1, X_2, and X_3 could be computed from those of X_4 or X_5. Thus, we do not need to keep those information in the final sufficient statistics.

By using those two properties the cardinality of the sufficient statistics of a BN structure G would be reduced to:

$$|SS(G)| = A^{M+1} \sum_{i=M}^{r-1} \left(\begin{array}{c} i \\ M \end{array} \right). \tag{3}$$

To see the improvement, for example we consider $r = 30$ and $M = 10$. The new sufficient statistics has more than 10 times less items than the one with the previous formula. This significantly decreases the space needed to store and speed up the search of the sufficient statistics set for computing score and parameters of the Bayesian network structure.

Given the Sufficient Statistics of G, an incremental version of the $\mathcal{K}2$ algorithm is proposed to construct BNs. Each time new trained data becomes available the sufficient statistics of the existing network structure is updated to create a new structure and adjust parameters accordingly. This algorithm, receives a node and its correct parent set, and will find other parents using new sufficient statistics and will also update the candidate lists of the parents. Each list $C_{i,l}$, $j+1 \leq l \leq k$, contains the current l-th parent of X_i, which is no longer the correct parent, along with all the candidate parents in decreasing order of their score. The number of nodes in each list could be varied using a threshold value.

Algorithm 2 shows the steps of this procedure. Inside the first loop from step 6 to 17, each candidate list $C_{i,l}$ is checked to find the node with the maximum score to set it as the new l-th parent. Second loop, from step 19 to 29, is almost the same as that in the previous $\mathcal{K}2$ algorithm except that in each iteration if a parent is found, its corresponding candidate list is also created to use in the future run of the algorithms. The $\mathcal{K}2$ algorithm must also be embedded in the incremental algorithm. In the main algorithm, Algorithm 3, for each node, using the previous structure and sufficient statistics, the new data, and previous set of candidate lists of the parents, new network structure and sufficient statistics are created. First, the sufficient statistics of the whole previous and new data are computed. Then, for each node, it is checked whether the current j-th parent is still parent of this node by checking and comparing its score with those of the other candidate nodes. If it can no longer be a parent, then it comes out from the inner loop and new $\mathcal{K}2$ algorithm is called to find the new parents and to update the candidate lists of parents for the current node.

Algorithm 2 Specialized $\mathcal{K}2$ Algorithm

1. **Inputs**:
   X_i,
   $\pi_i = \{X_{i,1}, \cdots, X_{i,j}\}$ (correct parents),
   $SS(X_i)$,
   u (the maximum number of parents for a node),
   set of candidate parents lists, $\{C_{i,j+1}, \cdots, C_{i,k}\}$ (to be checked to find the best parent in each list),
   $Pred(X_i)$ (to be checked for the new parents)
2. **Output**:
   The new parents set for X_i,
   The updated set of candidate parents lists for X_i

---

3. $Score_{old} = f(i, \pi_i)$
4. $l = j + 1$
5. NotFound = **true**
6. **while** NotFound **and** $l \leq k$ **do**
7. $z = arg\ max_{w \in C_{i,l}} f(i, \pi_i \bigcup \{w\})$
8. Reorder $C_{i,l}$ in decreasing order of the nodes' score such that z be the first item of the list.
9. $Score_{new} = f(i, \pi_i \bigcup \{z\})$
10. **if** $Score_{new} > Score_{old}$ **then**
11. $Score_{old} = Score_{new}$
12. $\pi_i = \pi_i \bigcup \{z\}$
13. $l = l + 1$
14. **else**
15. NotFound = **false**
16. **end if**
17. **end while**
18. OkToProceed = **true**
19. **while** OkToProceed **and** $|\pi_i| < u$ **do**
20. $z = arg\ max_{w \in Pred(x_i) - \pi_i} f(i, \pi_i \bigcup \{w\})$
21. $Score_{new} = f(i, \pi_i \bigcup \{z\})$
22. **if** $Score_{new} > Score_{old}$ **then**
23. Create a new list $C_{i,l}$ containing the list of all candidate nodes for the l-th parent in decreasing order such that z be the first item in the list.
24. $Score_{old} = Score_{new}$
25. $\pi_i = \pi_i \bigcup \{z\}$
26. **else**
27. OkToProceed = **false**
28. **end if**
29. **end while**

Algorithm 3 Incremental Bayesian Network Learning Algorithm

1. **Input**:
   $\{X_1, \cdots, X_m\}$ (An ordered list of nodes),
   $SS(D)$ (sufficient statistics of the previous data),
   D' (new dataset),
   u (the maximum number of parents for a node),
   $\{C_{i,1}, \cdots, C_{i,k}\}$ (set of previous candidate lists of parents)
2. **Output**:
   A directed acyclic graph for the Bayesian network according to the previous one and new data,
   The new updated sufficient statistics, SS(D)

---

3. Compute $SS(D')$
4. $SS(D) = SS(D) \bigcup SS(D')$
5. **for** $i = 1$ to m **do**
6. $\quad \pi_i' = \emptyset$
7. $\quad j = 1$
8. $\quad$ OkToProceed = **true**
9. $\quad$ **while** OkToProceed **and** $j < k$ **do**
10. $\quad\quad z =$ the first node in $C_{i,j}$
11. $\quad\quad$ **if** $z = arg\ max_{w \in C_{i,j}} f(X_i, \pi_i' \bigcup \{w\})$ **then**
12. $\quad\quad\quad \pi_i' = \pi_i' \bigcup \{z\}$
13. $\quad\quad\quad j = j + 1$
14. $\quad\quad$ **else**
15. $\quad\quad\quad$ OkToProceed = **false**
16. $\quad\quad$ **end if**
17. $\quad$ **end while**
18. $\quad$ **if** $j < u$ **then**
19. $\quad\quad$ **Call** $\mathcal{K}2(X_i, \pi_i, SS(X_i), u,$
    $\quad\quad\quad\quad \{C_{i,j+1}, \cdots, C_{i,k}\}, Pred(X_i))$
20. $\quad$ **end if**
21. **end for**

4. Privacy-preserving Incremental Learning of Bayesian Networks

In this section, we present our protocol for privacy-preserving incremental learning of BNs when data is horizontally partitioned among several parties. By privacy-preserving BNs, we mean a protocol by which BNs structure and/or parameters are constructed from a securely shared data among two or more parties while each party keeps her own data private. It is assumed, in this protocol, that the parties are semi-honest, i.e. each party follows the protocol and sends correct data to the other parties, however she might use intermediate or final information to compute others' private data.

The private information is the raw data each party owns and sufficient statistics of those data, and the public knowledge at the end of the protocol are the candidate parents lists of each node. In the above algorithms we need to securely compute the score function, securely compare computed scores to find the parents of each node, and construct the candidate lists of parents in decreasing order of their score values.

Some secure building blocks [10] are used inside the main protocol. Secure Multi-party Multiplication, Addition and Factorial, and also Secure Product Comparison are needed in both types of partitioned data. Thus we first discuss the sub-protocols used inside the main protocol and then bring the protocol for homogenously partitioned data.

4.1. Secure Multi-party Multiplication

In this building block, the product of private input shares, x_is, is converted to the summation of private output shares, y_is:

$$\prod_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i. \tag{4}$$

This sub-protocol is widely used in other protocols and also in our building blocks such as secure exponentiation and secure multi-party factorial. A protocol has been proposed for this purpose in [9], which uses additive homomorphic encryption [5]. However that protocol is not secure against collusion attack in the multi-party case. For instance, in the case of three parties, if the first and third parties collude, they can easily deduce the second party's private input. Here, we propose another protocol to prevent that type of attack. The algorithm for the two-party case is the same as that in [9] briefly described in Appendix A, and the multi-party case is presented here.

Suppose E_i is an additive homomorphic encryption established by p_i, with public key e_i and private key d_i. The following are the steps of the algorithm:

1. p_1 and p_2 run a secure two-party multiplication for their inputs such that:

$$x_1 * x_2 = y_{11} + y_{21}.$$

2. Therefore, we have:

$$\begin{aligned} x_1 * x_2 * \cdots * x_n &= (y_{11} + y_{21}) * x_3 * \cdots * x_n \\ &= (y_{11} * x_3 * \cdots * x_n) + \\ &\quad (y_{21} * x_3 * \cdots * x_n) \end{aligned}$$

3. Now, each of

$$(y_{11} * x_3 * \cdots * x_n) \text{ and } (y_{21} * x_3 * \cdots * x_n)$$

are multiplication of $n-1$ items belong to the parties $1, 3, \cdots, n$, and $2, 3, \cdots, n$ respectively. Thus, the algorithm in step one would be repeated until all the parts of the summation are two-party multiplications, and secure two-party multiplication protocol could be applied.

Therefore, if n parties participate in this protocol, $n(n-1)$ encryptions and $\frac{n(n-1)}{2}$ decryptions are needed to complete the protocol. There are also $n(n-1)$ messages needed to be communicated between the parties.

To make it clear, we give a simple example for three parties:

1. p_1 and p_2 run secure two-party multiplication for their inputs, x_1 and x_2, that produces y_{11} and y_{21} such that:

$$x_1 * x_2 = y_{11} + y_{21}.$$

2. Therefore, we have:

$$\begin{aligned} x_1 * x_2 * x_3 &= (y_{11} + y_{21}) * x_3 \\ &= (y_{11} * x_3) + (y_{21} * x_3) \end{aligned}$$

3. p_1 and p_3 run secure two-party multiplication for their inputs, y_{11} and x_3, yielding:

$$y_{11} * x_3 = y_1 + y_{31}.$$

4. p_2 and p_3 do the same for their inputs, y_{21} and x_3, such that:

$$y_{21} * x_3 = y_2 + y_{32}.$$

5. Now, we have:

$$\begin{aligned} x_1 * x_2 * x_3 &= (y_{11} + y_{21}) * x_3 \\ &= (y_{11} * x_3) + (y_{21} * x_3) \\ &= (y_1 + y_{31}) + (y_2 + y_{32}) \\ &= y_1 + y_2 + y_3 \end{aligned}$$

where $y_3 = y_{31} + y_{32}$

4.1.1. Security Analysis

First, we show that the two-party protocol we have used is in fact secure. Through this paper, we utilize following notations in our proof:

- pd : Private data.
- $ppdm$: Privacy-Preserving Data Mining protocol.
- pd_{p_i} : p_i's private data.
- ext_{p_i} : The extra information p_i can obtain through the underlying protocol.
- $gain_{p_i}$: The advantage of p_i to get access to any other party's private data using a protocol.
- $gain_{sec}$: The advantage of p_i to get access to any other party's private data using a protocol by looking at a semantically secure ciphertext, which is negligible in our case of using an RSA type of encryption.
- $pr(pd)$: The probability of disclosing the private data pd without using any privacy-preserving protocol.
- $pr(pd|ppdm)$: The probability of disclosing the private data pd after using a privacy-preserving protocol.
- $|pr(pd|ppdm) - pr(pd)|$: Absolute difference between the probability of disclosing the private data pd with and without using a privacy-preserving protocol.
- ϵ : The level of security.

We have to find an ϵ such that:

$$|pr(pd|ppdm) - pr(pd)| \leq \epsilon$$

Advantages for p_1 and p_2 are:

$$gain_{p_1} = pr(pd_{p_2}|ext_{p_1}, ppdm) - pr(pd_{p_2}|ext_{p_1})$$
$$gain_{p_2} = pr(pd_{p_1}|ext_{p_2}, ppdm) - pr(pd_{p_1}|ext_{p_2})$$

p_2 only receives $E_1(x_1, e_1)$ from p_1 which is semantically secure, and therefore:

$$gain_{p_2} = gain_{sec}$$

which is negligible.

p_1, by decrypting the message received from p_2, knows

$$y_1 = (x_1 * x_2) - y_2$$

which is the final desired result for this party. Thus $gain_{p_1}$ is at most knowing her output share, y_1. We set:

$$\begin{aligned} \epsilon &= max(gain_{p_1}, gain_{p_2}) \\ &= max(gain_{p_1}, gain_{sec}) = gain_{p_1} \end{aligned}$$

Therefore, we have:

$$pr(pd_{p_2}|ext_{p_1}, ppdm) - pr(pd_{p_2}|ext_{p_1}) \leq gain_{p_1}$$

and

$$pr(pd_{p_1}|ext_{p_2}, ppdm) - pr(pd_{p_1}|ext_{p_2}) \leq gain_{p_1}$$

Thus, it proves the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

Now we prove the security of the protocol for three parties. The proof is the same in general multi-party case. Again, an ϵ has to be determined, such that:

$$|pr(pd|ppdm) - pr(pd)| \leq \epsilon$$

Advantages of p_1, p_2 and p_3 are as follows:

$$\begin{aligned} gain_{p_1} &= pr(pd_{p_j}|ext_{p_1}, ppdm) - pr(pd_{p_j}|ext_{p_1}) \\ &\quad (j = 2, 3) \\ gain_{p_2} &= pr(pd_{p_j}|ext_{p_2}, ppdm) - pr(pd_{p_j}|ext_{p_2}) \\ &\quad (j = 1, 3) \\ gain_{p_3} &= pr(pd_{p_j}|ext_{p_3}, ppdm) - pr(pd_{p_j}|ext_{p_3}) \\ &\quad (j = 1, 2) \end{aligned}$$

p_3 only receives $E_1(y_{11}, e_1)$ from p_1, and $E_2(y_{21}, e_2)$ from p_2, which are both semantically secure, and therefore:

$$gain_{p_3} = gain_{sec}$$

which is negligible.

p_2, by decrypting the message receiving from p_3, knows

$$y_2 = (y_{21} + x_3) - y_{32}$$

which is the final desired result for this party. p_2 also receives $E_1(x_1, e_1)$ from p_1, which is semantically secure. Thus, p_2's gain is at most her final private output share, y_2.

p_1, by decrypting the message received from p_3, knows

$$y_1 = (y_{11} + x_3) - y_{31}$$

which is the final desired result for this party. Also, p_1, by decrypting the message receiving from p_2, knows

$$y_{11} = (x_1 * x_2) - y_{21}$$

y_{11} is the partial information p_1 receives through her communication with p_2, but it could not help this party to reach to p_2's private data because of its randomness property, same as y_{21} for p_2.

Therefore, $gain_{p_1} = gain_{p_2}$, and we set:

$$\begin{aligned} \epsilon &= max(gain_{p_1}, gain_{p_2}, gain_{p_3}) \\ &= max(gain_{p_1}, gain_{sec}) = gain_{p_1} \end{aligned}$$

Therefore, we have:

$$\begin{aligned} pr(pd_{p_j}|ext_{p_1}, ppdm) - pr(pd_{p_j}|ext_{p_1}) &\leq gain_{p_1} \\ pr(pd_{p_j}|ext_{p_2}, ppdm) - pr(pd_{p_j}|ext_{p_2}) &\leq gain_{p_1} \\ pr(pd_{p_j}|ext_{p_3}, ppdm) - pr(pd_{p_j}|ext_{p_3}) &\leq gain_{p_1} \end{aligned}$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

To show its security against collusion attack, suppose three parties are involved. We investigate all the scenarios in which two parties collude to access the third one's private data.

- **Collusion of p_1 and p_2:** p_1 receives

  $$E_1(y_{11}, e)^{x_3} * E_1(y_{31}, e)^{-1}$$

  from p_3 and also, because of colluding with p_2, receives

  $$E_2(y_{21}, e)^{x_3} * E_2(y_{32}, e)^{-1}$$

  from p_2. However, these two parties have no information about x_3, y_{31}, and y_{32}. Thus, by decrypting the above information and having the two equations:

  $$\begin{aligned} y_{11} * x_3 - y_{31} &= y_1 \\ y_{21} * x_3 - y_{32} &= y_2 \end{aligned}$$

  and those three unknown values, they are not able to access p_3's private data, except by guessing them.

- **Collusion of p_1 and p_3:** p_1 receives

  $$E_1(x_1, e)^{x_2} * E_1(y_{21}, e)^{-1}$$

  from p_2 and also, because of colluding with p_3, receives $E_2(y_{21}, e)$ from p_2. E_2 is semantically secure and thus p_1 is not able to get any information from that. Also, x_2 and y_{21} could not be disclosed to p_1 from equation, $x_1 * x_2 - y_{21} = y_{11}$ alone.

- **Collusion of p_2 and p_3:** p_2 receives $E_1(x_1, e)$ from p_1, and also $E_1(y_{11}, e)$ from p_3 because of colluding. E_1 is semantically secure, and thus p_2 is not able to get any useful information from the encrypted data.

This building block could be run over the public channels. This is because none of the parties simply sends the encryption of their private data to the other parties separately, except the owner of the private key. Therefore, even this party is not able to obtain others' private data by decrypting the received messages from the public channel.

4.1.2. Complexity Analysis

- **Computation Cost:** By denoting n as the number of parties involved in the protocol, there are $\frac{n(n+1)}{2}$ encryptions and $\frac{n(n-1)}{2}$ decryptions in this building block. By assuming that the computation costs of encryption and decryption are the same, and denoting each of these costs by α:

$$\text{Computation cost} = n^2\alpha.$$

- **Communication Cost:** Communication cost for two parties, as we saw in two party case in Appendix Appendix A, is 2β. By denoting this cost for n parties as $CMM(n)$, $\forall n \geq 3$, and the number of bits exchanged for each message with β:

$$\begin{aligned} CMM(n) &= (CMM(\lfloor\frac{n}{2}\rfloor) + CMM(\lceil\frac{n}{2}\rceil) + \\ &\quad \lfloor\frac{n}{2}\rfloor * (\lceil\frac{n}{2}\rceil + 1))\beta. \end{aligned}$$

  For instance, if $n = 4$, then communication cost of this protocol would be 10β.

4.2. Secure Multi-party Addition

Using this sub-protocol, summation of private input shares, x_is, is converted to the multiplication of private output shares, y_is.

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} y_i. \tag{5}$$

This protocol, with the same lack of security against collusion attack, is proposed in [9]. Here, a protocol is presented to counter collusion attacks. The algorithm for the two-party case is the same as that in [9] and is briefly shown in Appendix B.

Following are the steps of the algorithm for multi-party case:

1. p_n selects $n-1$ numbers $x_{n,1}, x_{n,2}, \cdots, x_{n,n-1}$ such that:

$$x_n = x_{n,1} + x_{n,2} + \cdots + x_{n,n-1}.$$

2. Each party p_i, $1 \leq i \leq n-1$, run a secure two-party addition with p_n for their inputs,x_i and $x_{n,i}$, such that:

$$x_i + x_{n,i} = y_{i,n} * y_n.$$

3. Therefore, we have:

$$\begin{aligned} x_1 + \cdots + x_n &= (y_{1,n} * y_n) + \cdots + (y_{n-1,n} * y_n) \\ &= (y_{1,n} + \cdots + y_{n-1,n}) * y_n \end{aligned}$$

4. Now

$$y_{1,n} + \cdots + y_{n-1,n}$$

is the summation of $n-1$ items belonging to the parties $1, 2, \cdots, n-1$. Thus, the algorithm would be repeated from step one until there is a summation of two parties, p_1 and p_2, on which secure two-party addition could be applied.

The following is an example for the three parties:

1. p_3 selects two numbers $x_{3,1}$ and $x_{3,2}$ given:

$$x_3 = x_{3,1} + x_{3,2}.$$

2. p_1 and p_3 run secure two-party addition for their inputs, x_1 and $x_{3,1}$:

$$x_1 + x_{3,1} = y_{1,3} * y_3.$$

3. p_2 and p_3 do the same for their inputs, x_2 and $x_{3,2}$:

$$x_2 + x_{3,2} = y_{2,3} * y_3.$$

4. Therefore, we have:

$$\begin{aligned} x_1 + x_2 + x_3 &= (y_{1,3} * y_3) + (y_{2,3} * y_3) \\ &= (y_{1,3} + y_{2,3}) * y_3 \end{aligned}$$

5. p_1 and p_2 run secure two-party addition for their inputs,$y_{1,3}$ and $y_{2,3}$, such that:

$$y_{1,3} + y_{2,3} = y_1 * y_2.$$

6. Therefore, we have:

$$x_1 + x_2 + x_3 = (y_{1,3} + y_{2,3}) * y_3 = y_1 * y_2 * y_3$$

The security analysis of this protocol is similar to that of the secure multi-party multiplication.

4.2.1. Complexity Analysis

- **Computation Cost:** The number of encryptions and decryptions in this protocol are $\frac{(n-1)(n+2)}{2}$ and $\frac{n(n-1)}{2}$, respectively. Therefore:

$$\text{Computation cost} = (n-1)(n+1)\alpha.$$

- **Communication Cost:** By considering all the messages sent from the parties, we have:

$$\text{Communication cost} = \frac{(n-1)(n+2)}{2}\beta.$$

4.3. Secure Product Comparison

In the main protocol we need to compare two products of the private input shares. In other words, if each party p_i has two inputs x_i and y_i, we need to compare $\prod_{i=1}^{n} x_i$ and $\prod_{i=1}^{n} y_i$ to figure out which one is greater than the other one. To do this, we test the sign of the expression $\prod_{i=1}^{n} x_i - \prod_{i=1}^{n} y_i$. The steps of our algorithm are as follows:

1. All parties run secure multiplication on both series of their inputs such that:

$$\prod_{i=1}^{n} x_i = \sum_{i=1}^{n} a_i \quad \text{and} \quad \prod_{i=1}^{n} y_i = \sum_{i=1}^{n} b_i.$$

2. Each party p_i, $1 \leq i \leq n$, sets :

$$c_i = a_i - b_i.$$

3. All the parties run secure addition such that:

$$\sum_{i=1}^{n} c_i = \prod_{i=1}^{n} d_i.$$

4. Each party p_i, $2 \leq i \leq n$, sends the sign of her final private output share, d_i, to p_1.
5. p_1 counts the number of negative signs of d_is, and if it is even then:

$$\prod_{i=1}^{n} x_i > \prod_{i=1}^{n} y_i.$$

Otherwise:

$$\prod_{i=1}^{n} x_i < \prod_{i=1}^{n} y_i$$

There are two secure multiplications and one secure addition in this protocol. Therefore, for computational cost, we have $3n(n-1)$ encryptions and $\frac{3n(n-1)}{2}$ decryptions. Also $3n(n-1)$ messages plus $n-1$ signs need to be exchanged among n parties.

4.3.1. Security Analysis

In this sub-protocol we use secure addition and multiplication. The only difference is that p_1 knows the sign of the private output shares of the other parties at the end of the protocol, which will not reveal any useful information to p_1. Thus, we set:

$$\begin{aligned} \epsilon &= max(gain_{p_1}, \cdots, gain_{p_n}) \\ &= max(gain_{p_1}, gain_{sec}) = gain_{p_1}. \end{aligned}$$

This proves that the maximum advantage of each party is not greater than knowing the sign of the final output shares of the secure comparison.

4.3.2. Complexity Analysis

- **Computation Cost:** There are two secure n-party multiplications and one secure n-party addition in this building block. Thus:

$$\text{Computation cost} = (3n^2 - 1)\alpha.$$

- **Communication Cost:** By using the communication costs of the sub-protocols used in this building block, we have:

$$\begin{aligned} \text{Communication cost} &= (\frac{(n-1)(n+2)}{2} + \\ & \quad 2CMM(n))\beta \end{aligned}$$

  in which $CMM(n)$ is the communication cost of the secure n-party multiplication protocol.

4.4. Secure Exponentiation

In this building block, we convert an exponentiation with a positive base, when each part, base and exponent, is privately owned by a different party. Suppose, $x_1 \in p_1$, $x_2 \in p_2$, and $x_1 > 0$. We want to find two private output shares, z_1 and z_2, for these parties such that:

$$x_1^{x_2} = z_1 * z_2. \tag{6}$$

The following are the steps of the protocol.

1. p_1 computes $\ln(x_1)$.
2. p_1 and p_2 run secure multiplication on their inputs, $\ln(x_1)$ and x_2, to find their private output shares, y_1 and y_2, respectively such that $\ln(x_1) * x_2 = y_1 + y_2$.
3. Now we have:

$$\begin{aligned} x_1^{x_2} &= \exp^{\ln(x_1^{x_2})} = \exp^{x_2 * \ln(x_1)} \\ &= \exp^{y_1 + y_2} = \exp^{y_1} * \exp^{y_2}. \end{aligned}$$

   Thus, $\exp^{y_1}$ and $\exp^{y_2}$ are the final private output shares for p_1 and p_2, respectively.

4.4.1. Security Analysis

This sub-protocol is just a version of secure multiplication with two differences. First is the input of the first party which is ln of her input. Second is the final private outputs which are exp of the outputs. Therefore, there is no difference on their security analyses.

4.5. Secure Multi-party Factorial

Using this sub-protocol, multiple parties are able to cooperatively produce their own private output shares, $u_1, \cdots, u_n$ from their private input shares, $x_1, \cdots, x_n$, such that:

$$\left(\sum_{i=1}^{n} x_i\right)! = \prod_{i=1}^{n} u_i. \tag{7}$$

In this protocol, we use Stirling's approximation for factorial and the previously mentioned building blocks, secure addition, multiplication and exponentiation. According to that approximation, for a large number l, we have the following equation:

$$l! \approx \sqrt{2\pi l}(\frac{l}{\exp})^l. \tag{8}$$

Also, in the $\mathcal{K}2$ algorithm we do not need to compute the exact results of the score function. Rather, in each step we just have to compare these results and select the largest one. Therefore, we can use this approximation inside the $\mathcal{K}2$ algorithm.

Without loss of generality and for simplicity of the formula, we show the two-party case of the protocol, which can easily be generalized to the multi-party case by using the multi-party versions of our building blocks. The steps of the protocol are as follows:

1. p_1 and p_2 run secure addition on their inputs, x_1 and x_2, to find their private output shares, y_1 and y_2, respectively, such that:

   $$x_1 + x_2 = y_1 * y_2.$$

   Note that p_2 has to select y_2 as a positive random number. Therefore, y_1 is also positive because x_1 and x_2 are both positive, and we can run secure exponentiation in the next two steps on y_1 and y_2.
2. p_1 and p_2 run secure exponentiation on their inputs, y_1 and x_2, to find their private output shares, s_1 and s_2, respectively, such that:

   $$y_1^{x_2} = \exp^{s_1} * \exp^{s_2}.$$

3. p_1 and p_2 run secure exponentiation on their inputs, x_1 and y_2, to find their private output shares, t_1 and t_2, respectively such that:

   $$y_2^{x_1} = \exp^{t_1} * \exp^{t_2}.$$

4. Both of the values $(\sqrt{2\pi y_1} * y_1^{x_1} * \exp^{s_1+t_1-x_1})$ and $(\sqrt{y_2} * y_2^{x_2} * \exp^{s_2+t_2-x_2})$ are the final output shares for p_1 and p_2, respectively.

We can show the correctness of the algorithm as follows:

$$\begin{aligned}
l! &\approx \sqrt{2\pi l}(\frac{l}{\exp})^l \\
&= \sqrt{2\pi * (x_1 + x_2)} \left(\frac{x_1 + x_2}{\exp}\right)^{x_1+x_2} \\
&= \sqrt{2\pi * (y_1 * y_2)} \left(\frac{y_1 * y_2}{\exp}\right)^{x_1+x_2} \\
&= \sqrt{2\pi y_1 y_2} * \exp^{-x_1} * \exp^{-x_2} * \\
&\quad y_1^{x_1} * y_1^{x_2} * y_2^{x_1} * y_2^{x_2} \\
&= \sqrt{2\pi y_1 y_2} * \exp^{-x_1} * \exp^{-x_2} * \\
&\quad y_1^{x_1} * (\exp^{s_1} * \exp^{s_2}) * (\exp^{t_1} * \exp^{t_2}) * y_2^{x_2} \\
&= (\sqrt{2\pi y_1} * y_1^{x_1} * \exp^{s_1+t_1-x_1}) * \\
&\quad (\sqrt{y_2} * y_2^{x_2} * \exp^{s_2+t_2-x_2}) \\
&= u_1 * u_2
\end{aligned}$$

where u_1 and u_2 are respectively substitutes for

$$(\sqrt{2\pi y_1} * y_1^{x_1} * \exp^{s_1+t_1-x_1})$$

and

$$(\sqrt{y_2} * y_2^{x_2} * \exp^{s_2+t_2-x_2}).$$

Thus, to run secure factorial among n parties $n(n-1)$ secure multiplications and one secure addition will be done.

4.5.1. Security Analysis

First, note that p_1 and p_2 have the following data communications during the algorithm:

$$\begin{aligned}
p_1 &: E(x_1, e) \longrightarrow p_2 \\
p_2 &: (E(x_1, e) * E(x_2, e))^{y_2^{-1}} \longrightarrow p_1 \\
p_2 &: E(x_1, e)^{\ln(y_2)} * E(t_2, e)^{-1} \longrightarrow p_1 \\
p_1 &: E(\ln(y_1), e) \longrightarrow p_2 \\
p_2 &: E(\ln(y_1), e)^{x_2} * E(s_2, e)^{-1} \longrightarrow p_1
\end{aligned}$$

Now, we have to find an ϵ such that:

$$|pr(pd|ppdm) - pr(pd)| \leq \epsilon$$

Advantages of p_1 and p_2 are:

$$\begin{aligned} gain_{p_1} &= pr(pd_{p_2}|ext_{p_1}, ppdm) - pr(pd_{p_2}|ext_{p_1}) \\ gain_{p_2} &= pr(pd_{p_1}|ext_{p_2}, ppdm) - pr(pd_{p_1}|ext_{p_2}). \end{aligned}$$

p_2 receives two encrypted messages from p_1 using homomorphic encryption E_1 which is semantically secure, and therefore:

$$gain_{p_2} = gain_{sec}$$

which is negligible.

p_1, by decrypting the messages receiving from p_2, knows

$$\begin{aligned} y_1 &\quad \text{such that} \quad (x_1 + x_2) = y_1 * y_2 \\ s_1 &\quad \text{such that} \quad y_1^{x_2} = s_1 * s_2 \\ t_1 &\quad \text{such that} \quad y_2^{x_1} = t_1 * t_2 \end{aligned}$$

which are the final desired results for p_1 to create her private output share. Thus $gain_{p_1}$ is at most knowing her output share, u_1. We set:

$$\begin{aligned} \epsilon &= max(gain_{p_1}, gain_{p_2}) \\ &= max(gain_{p_1}, gain_{sec}) = gain_{p_1} \end{aligned}$$

Therefore:

$$\begin{aligned} pr(pd_{p_2}|ext_{p_1}, ppdm) - pr(pd_{p_2}|ext_{p_1}) &\leq gain_{p_1} \\ pr(pd_{p_1}|ext_{p_2}, ppdm) - pr(pd_{p_1}|ext_{p_2}) &\leq gain_{p_1} \end{aligned}$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

4.5.2. Complexity Analysis

- **Computation Cost:** One secure n-party addition and $n(n-1)$ secure two-party multiplications have to be executed in this sub-protocol. Thus:

$$\text{Computation cost} = (n-1)(4n+1)\alpha.$$

- **Communication Cost:** By using the communication costs of the sub-protocols used in this building block, we have:

$$\text{Communication cost} = \frac{(n-1)(5n+2)}{2}\beta.$$

4.6. Protocol for Horizontally Partitioned Data

In this section, a privacy-preserving BNs protocol is presented for horizontally partitioned data. In this configuration each party owns some records of the whole dataset. We use secure factorial and secure product comparison inside

$\mathcal{K}2$ algorithm to preserve the privacy of the parties involved. Since each party has the values for all the variables for some records, she can compute and maintain the sufficient statistics of her own data, and thus the value of each record of the sufficient statistics of the whole dataset would be the summation of the corresponding values from the sufficient statistics owned by all the parties.

In steps 3, 7 and 20 of the new $\mathcal{K}2$ algorithm and step 11 of the incremental algorithm, the score function has to be computed for different sets of items. This function, in the horizontal case, is converted to the following equation:

$$f(i,\pi_i) = \prod_{j=1}^{q_i} \frac{\left(\sum_{k=1}^{n} \alpha_{kij0}\right)! * \left(\sum_{k=1}^{n} \alpha_{kij1}\right)!}{\left(\left(\sum_{k=1}^{n} \alpha_{kij0}\right) + \left(\sum_{k=1}^{n} \alpha_{kij1}\right) + 1\right)!} \tag{9}$$

where, $\alpha_{kijr} \in p_k$ for $r \in \{0,1\}$. After applying secure factorial building block on each part of the Equation (9) as follows:

$$\left(\sum_{k=1}^{n} \alpha_{kij0}\right)! = \prod_{k=1}^{n} u_{kij0}$$

$$\left(\sum_{k=1}^{n} \alpha_{kij1}\right)! = \prod_{k=1}^{n} u_{kij1}$$

$$\left(\left(\sum_{k=1}^{n} \alpha_{kij0}\right) + \left(\sum_{k=1}^{n} \alpha_{kij1}\right) + 1\right)! = \prod_{k=1}^{n} u_{kij2}$$

we have a new function $g(i,\pi_i)$, as an approximation of $f(i,\pi_i)$, given by:

$$\begin{aligned} g(i,\pi_i) &= \prod_{j=1}^{q_i} \frac{\left(\prod_{k=1}^{n} u_{kij0}\right)\left(\prod_{k=1}^{n} u_{kij1}\right)}{\left(\prod_{k=1}^{n} u_{kij2}\right)} \\ &= \prod_{j=1}^{q_i}\prod_{k=1}^{n} \frac{u_{kij0} * u_{kij1}}{u_{kij2}} = \prod_{j=1}^{q_i}\prod_{k=1}^{n} w_{kij} = \prod_{k=1}^{n} z_k \end{aligned}$$

such that $u_{kijl} \in p_k$, and following substitutions:

$$\frac{u_{kij0} * u_{kij1}}{u_{kij2}} = w_{kij} \qquad and \qquad \prod_{j=1}^{q_i} w_{kij} = z_k$$

where z_k, for $1 \leq k \leq n$, is the private output share of p_k.

Now, for each result of the score function we have a multiplication of the private shares owned by the parties. The next step is to compare them and

find the one with the maximum value. For instance for two parties p_1 and p_2, if $x_1, y_1 \in p_1$ and $x_2, y_2 \in p_2$, we have to securely compare $x_1 * x_2$ and $y_1 * y_2$. For this part, the secure product comparison is used, and the item corresponding to the greatest number is added to the set of parents for the current node.

The overhead cost of the proposed privacy-preserving $\mathcal{K}2$ algorithm is due to the use of the secure factorial and the secure product comparison protocols. If we assume that the maximum number of parents for each node is 2, then the maximum number of executions of secure factorial and secure comparison are $m^2 - m + 1$ and $(m-1)^2$, respectively.

4.6.1. Security Analysis

As we see in the $\mathcal{K}2$ algorithm, the parties have data communication in steps 3, 7, and 20 in Algorithm 2 and 11 in Algorithm 3 for computing score functions, and in steps 7, 10, 20, and 22 in Algorithm 2, and 11 in Algorithm 3 for comparisons between the products of the private values. Other steps are publicly executed. According to the composition theorem [4, 1], if the main protocol is partitioned into sub-protocols such that the output of one sub-protocol is the input of the next and all intermediate results are kept private, then the whole protocol would be privacy-preserving. In our main protocol, inputs for computing the score function are private shares of the parties. Output shares, by using secure factorial, are also private and are in turn the inputs for the secure product comparison. This sub-protocol is also secure and at the end, only the node or attribute name with the greater value for the score function is determined. Note that the names of the attributes are available to all parties. Therefore the whole protocol remains privacy-preserving.

5. Experimental Results

Experiments in this section seek to show the performance improvement of incremental learning compared to the batch algorithm. Datasets used in this experiment are ASIA [20], ADULT [21], and ALARM [19]. The first one is an academic model and the next two are real datasets, widely used in various papers as experimental data. Results are produced from 10 independent replications via 10-fold cross-validation, and execution time for different set of configurations are compared between incremental and batch algorithms. First we apply standard batch learning protocol on an online database such that each time algorithm runs on the whole dataset, including existing and new data. Then, incremental algorithm is applied in each step such that new data is only used along with some information maintained from the existing data in the previous step, such as sufficient statistics, and the result is compared with the result of using batch method to investigate the correctness of the extracted Bayesian Networks and the overall spending time on each method.

One important parameter in the incremental algorithm is the number of items in each list of candidate parents of each node. The more items are kept in each list, the faster the incremental algorithm would be. We test our protocol

Table 1: Execution time (in Seconds) of Asia model with different key lengths and methods.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	32.53	94.04	166.50	404.38
	S=2	22.73	65.85	117.15	284.98
	S=3	19.61	56.86	101.40	246.87
	S=4	19.14	55.51	99.00	241.03
1024	Non-Inc	188.61	545.17	969.24	2354.70
	S=2	131.80	381.78	682.23	1660.24
	S=3	113.69	329.68	590.63	1438.56
	S=4	111.00	321.89	576.68	1404.57
2048	Non-Inc	875.27	2529.88	4499.10	10930.51
	S=2	611.63	1771.70	3166.92	7707.10
	S=3	527.59	1529.92	2741.78	6678.14
	S=4	515.13	1493.78	2677.00	6520.35

with different size of those lists to find an optimum value for this parameter. However, depending on the underlying application and dataset, this can be specified by the user to find the desire network structure. Therefore, there is a trade-off between the execution time and the accuracy of the generated network structure. The complexity of techniques is measured in terms of the execution time requires for training the BNs. The quality of the proposed technique is measured in terms of the similarity of BNs learned through incremental learning to those learned through batch learning. It is also measured in terms of the fitness of BNs with respect to independent test data.

In each experiment, underlying dataset is securely shared between 2, 3, 4, and 6 parties. The implementation is done with Java, in which Remote Method Invocation (RMI) technique is used for data communication. We used machines with 2.66 GHZ CPU, 2.98 GB RAM, and Windows XP Professional. The system is tested with key bit lengths of 512, 1024, and 2048.

The first experiment is performed on ASIA [20] model, also called *Chest Clinic*, which is a network of a small medical example, indicating whether a patient has bronchitis, tuberculosis, or lung cancer depending on her/his history for X-ray result, dyspnea, visit-to-Asia and smoking status. Figure 2 shows the Bayesian network, containing 8 nodes, and Table 1 shows the performance results of this experiment. In this table performance results of different methods, batch and incremental with different number of items in the parents candidate lists, S, and for each key length, are shown. Each execution time for batch method is for 11 times of running the $\mathcal{K}2$ algorithm, and for incremental method is for one batch algorithm and 10 running of the incremental protocol each time new data is added to the dataset. Results indicate that the execution time will positively decrease using incremental method, especially with the larger value of S, compare with the batch algorithm. This is because of applying the protocol on a smaller dataset and previously generated sufficient statistics from the original data, and also checking the candidate parent lists first, before calling $\mathcal{K}2$ algorithm which is more time consuming.

The next experiment is for Adult [21] model, also known as *Census Income* dataset, which predicts whether income exceeds \$50K/yr based on census data, and the experiment results are shown in Table 2.

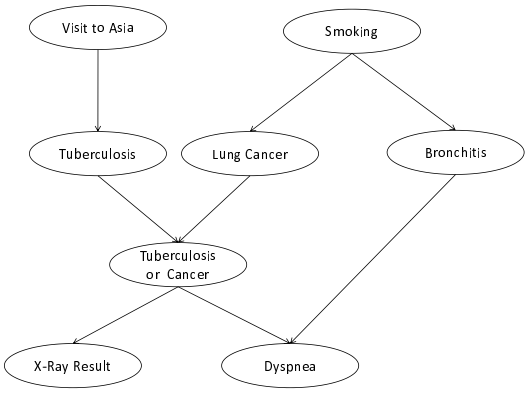

Figure 2: ASIA model.

Table 2: Execution time (in Seconds) of Adult model.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	93.40	269.81	476.80	1157.28
	S=2	49.56	143.48	254.96	619.96
	S=3	44.95	130.18	231.49	563.02
	S=4	41.96	121.59	216.49	526.76
1024	Non-Inc	541.59	1564.11	2775.16	6737.51
	S=2	287.36	831.91	1484.62	3611.33
	S=3	260.64	754.80	1348.03	3279.89
	S=4	243.31	705.03	1260.83	3069.08
2048	Non-Inc	2513.31	7258.31	12881.88	31275.11
	S=2	1333.51	3860.57	6891.63	16764.26
	S=3	1209.52	3502.74	6257.59	15225.75
	S=4	1129.12	3271.77	5852.83	14247.25

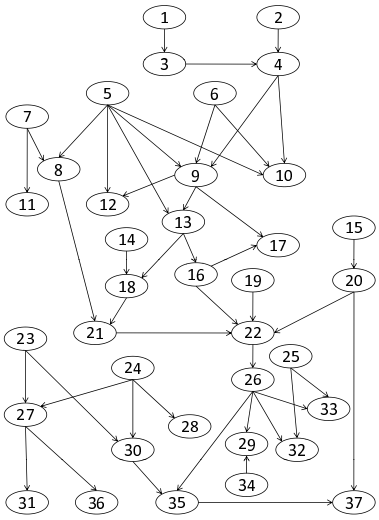

Figure 3: A Logical Alarm Reduction Mechanism (ALARM) model.

Table 3: Execution time (in Minutes) of Alarm model.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	16.01	46.19	81.42	197.44
	S=2	7.29	21.05	37.20	90.31
	S=3	5.79	16.73	29.61	71.90
	S=4	4.11	11.89	21.10	51.30
1024	Non-Inc	92.82	267.74	473.76	1149.15
	S=2	42.25	122.03	216.55	525.75
	S=3	33.56	96.98	172.35	418.66
	S=4	23.82	68.94	122.88	298.78
2048	Non-Inc	430.72	1242.47	2199.10	5334.17
	S=2	196.07	566.29	1005.18	2440.52
	S=3	155.73	450.05	800.04	1943.42
	S=4	110.56	319.91	570.40	1386.97

The last experiment is done for ALARM [19] model, stands for *A Logical Alarm Reduction Mechanism*, which is a medical diagnostic system for patient monitoring. It is a nontrivial belief network with 8 diagnoses, 16 findings and 13 intermediate variables. Figure 3 shows the Bayesian network, and Table 3 contains the performance results of that. Same situation as the previous model is applied.

As it can be seen in the experiment results, by using incremental protocol we can significantly reduce the overall execution time, especially when we are dealing with large number of parties and with large key bit length to strongly preserve the security of the protocol. This time saving is more sensible in large networks, such as Alarm model with 37 nodes in here, after running the protocols during the growth of the underlying dataset. Figure 4 shows the comparison of the execution times of the batch algorithm and incremental protocol with differ-

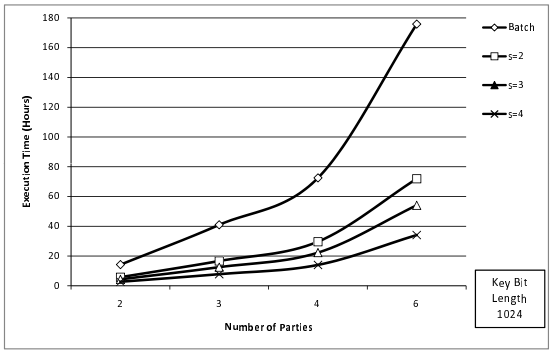

Figure 4: Comparison chart of batch and incremental protocols for ALARM model, with different number of parties, different size of parents candidate list, and key bit length of 1024.

ent parameters after 101 runs of the protocols on data for the Alarm model. For batch algorithm we run the protocol 101 times and for the incremental methods we run the batch algorithm at the first time and then run the incremental protocol for 100 times, each time new data is added to the dataset.

To compare the results of our incremental $\mathcal{K}2$ algorithm with the result of the batch algorithm, we run the protocol on 16 testing data chunks. At the end of each step the total number of edges and also the fitness of the created edges in the incremental approach are compared with those using batch algorithm. Figures 5 and 6 illustrates the comparison charts for the last experiment, ALARM model. As it is shown in Figure 5, there is a relatively big difference between the number of edges between the incremental and batch approaches in the first couple of steps. This means that incremental algorithm is not very suitable at first, especially with the large value of S, the number of items in the parents candidate lists. This could be seen in any real world applications. Suppose a batch algorithm is applied on a large dataset, and after a period of time a block of data, which is very small compare with the original dataset, becomes available. Now the result of applying batch algorithm, which runs on both original and new data, could be far different from the one from incremental method. This might happen because of a very different pattern in the new data from the original one. However, after some duration this bias will be low or even disappeared by coming more data and running incremental algorithm in long term. This fact could be also observed from the comparison chart for fitness of the edges inside constructed BNs structures using batch and incremental methods in Figure 6.

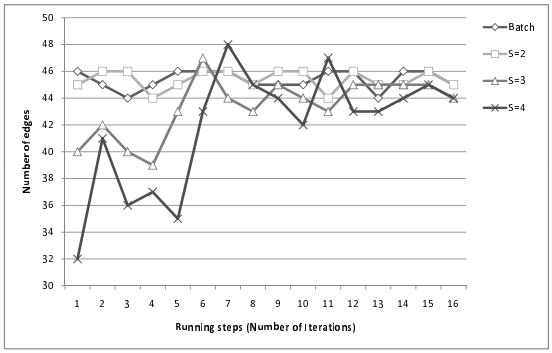

Figure 5: Comparison chart of edges inside created BNs structures using batch and incremental $\mathcal{K}2$ algorithm for ALARM model.

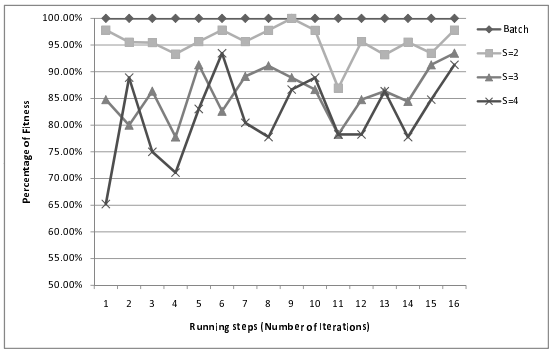

Figure 6: Comparison chart for fitness of the edges inside constructed BNs structures using batch and our incremental algorithm for ALARM model.

6. Conclusions and Future Work

Incremental learning algorithm for stream and online data to construct Bayesian Networks is considered in this paper when data is privately and horizontally shared among two or more parties, by improvement on computation of Sufficient Statistics and providing an incremental and efficient version of $\mathcal{K}2$ algorithm. Secure building blocks, such as secure multi-party multiplication, product comparison and factorial, which are resistant to colluding attacks and are secure in dedicated as well as public channels are also proposed to utilize in the main protocol to preserve the privacy of the parties involved. Experimental results indicate that the efficiency will significantly increase by using the new version of incremental $\mathcal{K}2$ algorithm an improved computation of Sufficient Statistics. Applying the proposed algorithms and protocols on real-world applications, in which privacy and efficiency are crucial factors and data is gradually growing, would be suitable to show the applicability of these methods and to find obstacles and new directions to improve these types of privacy-preserving protocols.

References

[1] R. Canetti. *Security and Composition of Multiparty Cryptographic Protocols*. Journal of Cryptology: the journal of the International Association for Cryptologic Research, 13(1):143–202, 2000.

[2] D. Chickering. *Learning Bayesian Networks is Np-complete*. pages 121–130, 1996.

[3] G. F. Cooper and E. Herskovits. *A Bayesian Method for the Induction of Probabilistic Networks from Data*. Machine Learning, 9(4):309–347, 1992.

[4] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[5] P. Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.* In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), pages 223–238, Prague, Czech Republic, 1999.

[6] J. Pearl. *Bayesian networks: A model of self-activated memory for evidential reasoning*. In Proceeding of the 7th Conference of the Cognitive Science Society, University of California, Irvine, USA, 1985.

[7] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[8] Steffen L. Lauritzen. *Graphical models*. Oxford University Press, 1996.

[9] S. Samet and A. Miri. *Privacy Preserving ID3 Using Gini Index over Horizontally Partitioned Data*. In Proceeding of The 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), pages 645–651, Doha, Qatar, 2008.

[10] S. Samet and A. Miri. *Privacy-Preserving Bayesian Network for Horizontally Partitioned Data*. To appear in the Proceedings of the 2009 IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2009), Vancouver, Canada, August 29-31, 2009.

[11] Norman Fenton and Martin Neil. *Managing Risk in the Modern World, Applications of Bayesian Networks*. A Knowledge Transfer Report from the London Mathematical Society and the Knowledge Transfer Network for Industrial Mathematics, London Mathematical Society De Morgan House, 57/58 Russell Square London WC1B 4HS, 2007.

[12] Nir Friedman and Moises Goldszmidt. *Sequential Update of Bayesian Network Structure*, In Proceeding of the 13th Conference on Uncertainty in Artificial Intelligence (UAI97), pages 165–174, Providence, Rhode Island, USA, 1997.

[13] Wray Buntine. *Theory Refinement on Bayesian Networks*, In Proceeding of the seventh conference on Uncertainty in artificial intelligence, pages 52–60, Los Angeles, CA, USA, 1991.

[14] Wai Lam and Fahiem Bacchus. *Using New Data to Refine a Bayesian Network*, In Proceeding of the Tenth Conference on Uncertainty in Artificial Intelligence, pages 383–390, Seattle, Washington, USA, 1994.

[15] Josep Roure. *Incremental Methods for Bayesian Network Structure Learnig*, PhD Thesis, Software department of the Technical University of Catalonia, 2004.

[16] C.K. Chow and C.N. Liu. *Approximating discrete probability distributions with dependence trees*, IEEE Transactions on Information Teory, 14:462–467, 1968.

[17] R.R. Bouckaert. *Bayesian belief networks: from inference to construction*, PhD Thesis, Faculteit Wiskunde en Informatica, Utrecht University, 1995.

[18] Aulius Pacekajus and Snorri Karlsson. *Refining Bayesian Network Structure When New Data Has Different Set of Attributes*, 2006.

[19] Beinlich, Ingo, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. *The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks*, In Proceeding of the Second European Conf. on Artificial Intelligence in Medicine (London, Aug.) Vol. 38, pages 247–256, 1989.

[20] Lauritzen, Steffen L. and David J. Spiegelhalter. *Local computations with probabilities on graphical structures and their application to expert systems*, In J. Royal Statistics Society B Vol. 50(2), pages 157–194, 1988.

[21] Ron Kohavi. *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*, In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD) pages 202–207, 1996.

[22] Peter D. Grnwald. *The Minimum Description Length Principle*. The MIT Press, 2007.

[23] W. Lam and F. Bacchus. *Learning Bayesian Belief Networks: An Approach Based on the MDL Principle*. Computational Intelligence, 10(4):269-293, 1994.

[24] N. Friedman and M. Goldszmidt. *Learning Bayesian networks with local structure*. In Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, Portland, Oregon, USA, 1996.

[25] R.R. Bouckaert. *Bayesian belief networks: from inference to construction*. PhD thesis, Faculteit Wiskunde en Informatica, Utrecht University, 1995.

[26] Robert Castelo. *A Discrete Acyclic Digraph Markov Model in Data Mining*. PhD thesis, Faculteit Wiskunde en Informatica, Univeriteit Utrecht, 2002.

[27] Robert Castelo and Tomás Kočka. *On inclusion-driven learning of Bayesian networks*. Journal of Machine Learning Research, (4), September 2003.

[28] Vaidya, Jaideep and Kantarcioúglu, Murat and Clifton, Chris. *Privacy-preserving Nave Bayes classification*. The International Journal on Very Large Data Bases, Volume 17 Issue 4, July 2008.

[29] D. Meng and K. Sivakumar. *Privacy-sensitive bayesian network parameter learning*. In Proceedings of the Fourth IEEE International Conference on Data Mining, 2004, pages 487–490.

[30] Yang, Zhiqiang and Wright, Rebecca N. *Improved Privacy-Preserving Bayesian Network Parameter Learning on Vertically Partitioned Data*. Proceedings of the 21st International Conference on Data Engineering Workshops, 2005. Washington, DC, USA.

[31] Wright, Rebecca and Yang, Zhiqiang. *Privacy-preserving Bayesian network structure computation on distributed heterogeneous data*. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, pages 713–718. Seattle, WA, USA.

[32] Kantarcioúglu, Murat and Clifton. *Privacy-preserving distributed mining of association rules on horizontally partitioned data*. The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 2002, pages 24–31. Madison, Wisconsin, USA.

[33] Bart Goethals and Sven Laur and Helger Lipmaa and Taneli Mielikainen. *On Private Scalar Product Computation for Privacy-Preserving Data Mining.* In Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC), 2004, pages 104–120. Seoul, Korea.

[34] *Freedom of Information and Protection of Privacy Act.* http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_90f31_e.htm. Last amendment: May 2011.

[35] *The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy and Security Rules.* http://www.hhs.gov/ocr/privacy/. Last Access: June 2011.

Appendix A. Secure Two-party Multiplication

1. p_1 selects an additive homomorphic encryption E, keeps the private key d and sends the public key e to p_2.
2. p_1 encrypts her input x_1, $E(x_1, e)$, and sends it to p_2.
3. p_2 powers the received value to her input x_2, $E(x_1, e)^{x_2}$, and randomly selects her, nonzero, output share y_2
4. p_2 computes $iy_2 = E(y_2, e)^{-1}$, and sends back $E(x_1, e)^{x_2} * iy_2$ to p_1.
5. p_1 decrypts the received value from p_2 and sets it as her output share y_1.

Appendix A.1. Correctness Analysis

$$\begin{aligned} y_1 &= D(E(x_1, e)^{x_2} * E(y_2, e)^{-1}, d) \\ &= (x_1 * x_2) - y_2 \\ \Rightarrow\ & y_1 + y_2 = x_1 * x_2 \end{aligned}$$

Appendix A.2. Complexity Analysis

- **Computation Cost:** There are two encryptions and one decryption in this building block. Therefore:

$$\text{Computation cost} = 3\alpha.$$

- **Communication Cost:** Considering the previous notations:

$$\text{Communication cost} = 2\beta.$$

The notations we employed in this section are also used in the similar sections in the rest of this thesis.

Appendix B. Secure Two-party Addition

1. p_1 selects an additive homomorphic encryption E, keeps the private key d and sends the public key e to p_2.
2. p_1 encrypts her input x_1, $E(x_1, e)$, and sends it to p_2.
3. p_2 encrypts her input x_2, $E(x_2, e)$, and randomly selects her nonzero output share y_2
4. p_2 computes $iy_2 = y_2^{-1}$, and sends back $(E(x_1, e) * E(x_2, e))^{iy_2}$ to p_1.
5. p_1 decrypts the received value from p_2 and sets it as her output share y_1.

Appendix B.1. Correctness Analysis

$$\begin{aligned} y_1 &= D((E(x_1, e) * E(x_2, e))^{y_2^{-1}}, d) \\ &= (x_1 + x_2) * y_2^{-1} \\ \Rightarrow\ & y_1 * y_2 = x_1 + x_2 \end{aligned}$$

Appendix B.2. Complexity Analysis

- **Computation Cost:** Similar to the secure two-party multiplication, there are two encryptions and one decryption in this protocol. Therefore:

$$\text{Computation cost} = 3\alpha.$$

- **Communication Cost:** Its communication cost is also the same as that in protocol 1:

$$\text{Communication cost} = 2\beta.$$